



 ACTIVE ECOMMERCE CMS
LARAVEL VERSION OF ACTIVE SUPER SHOP

Documentation

Author : Active IT zone

Software Framework : Flutter

Addon For: Active eCommerce CMS

Provided by : codecanyon



Documentation

- 1.** What are the prerequisites?
- 2.** How to run Flutter Application in Android Studio?
- 3.** How to configure the App according to your setup?
- 4.** How to change the package name?
- 5.** How to build the App for testing (build and apk) ?
- 6.** How to generate play store uploadable files for release?
- 7.** How to generate app store uploadable files? (This section will be available soon)
- 8.** How to Update for Android?
- 9.** How to configure social login?
- 10.** How to configure push notification?
- 11.** How to configure google map ?
- 12.** How to configure default language for mobile app?
- 13.** How to configure multiple languages for mobile app?
- 14.** My data is not changing. How to clear cache?

1. What are the prerequisites?

This Flutter app can be hosted into Google Play Store + Apple Appstore as your branded eCommerce CMS app. The app will communicate with your hosted eCommerce CMS web application through APIs. That means the prerequisite to publish the eCommerce Mobile application is to have the eCommerce CMS Web application in the latest version always.

Flutter version must be : **Flutter 3.0.2 • channel stable**

Dart version must be : **Dart 2.17.3**

Make sure your flutter and dart versions are correct. Follow the flutter documentation from <https://flutter.dev/docs/get-started/install> to install the given version of flutter in your pc/mac.

2. How to run Android Application in Android Studio?

- Install Android Studio from <https://developer.android.com/studio>
- Extract the source_code.zip. You will find this inside the main zip.
- Open the folder in your android studio.
- Even if you are building an app for ios, use android studio for the build.
- Then in your android studio terminal run:
`flutter pub get` **** You need this to get all 3rd party packages from pub.dev**

3. How to configure the App according to your setup?

A. App Config:

This helps you connect your app to your server.

Open lib/app_config.dart

You can change the `copyright_text`, `app_name`, `purchase_code`, `HTTPS_DOMAIN_PATH` variable.

Do not change the other variables.

Make sure that `purchase_code` is given. Otherwise your app will not work properly.

If your site does not have https or your are using a local machine as server (localhost) the make `HTTPS = false;`

Your `DOMAIN_PATH` is your site url without any protocol. (see screenshot below)

If you are using localhost , `DOMAIN_PATH` should be "[your_ip_address/your_project](#)";

**** "localhost/your_project" will not work ****

Normally you do not have to change the `BASE_PATH`. Keep it as given.

But if you are using s3 for image uploading your `BASE_PATH` should be :

`BASE_PATH = "https://\[\[bucketname\]\].s3.\[\[regeion\]\].amazonaws.com/";`

```
active_ecommerce_flutter > lib > app_config.dart
pubspec.yaml app_config.dart
Project Resource Manager
1:1-Project
1 import 'package:flutter/material.dart';
2
3 var this_year = DateTime.now().year.toString();
4
5 class AppConfig {
6     static String copyright_text = "@ ActiveItZone " + this_year; //this shows in the splash screen
7     static String app_name = "Active eCommerce"; //this shows in the splash screen
8
9     //configure this
10    static const bool HTTPS = true;
11
12    //configure this
13    //static const DOMAIN_PATH = "192.168.0.113/ecomerce_demo";
14    static const DOMAIN_PATH = "demo.activeitzone.com/ecomerce_flutter_demo";
15
16    //do not configure these below
17    static const String API_ENDPOINT = "api/v2";
18    static const String PUBLIC_FOLDER = "public";
19    static const String PROTOCOL = HTTPS ? "https://" : "http://";
20    static const String RAW_BASE_URL = "${PROTOCOL}${DOMAIN_PATH}";
21    static const String BASE_URL = "${RAW_BASE_URL}/${PUBLIC_FOLDER}";
22
23    //configure this if you are using amazon s3 like services
24    //give direct link to file like https://[[bucketname]].s3.ap-southeast-1.amazonaws.com/
25    //otherwise do not change anything
26    static const String BASE_PATH = "${RAW_BASE_URL}/${PUBLIC_FOLDER}/";
27 }
28
```

B. Theme Config:

This helps you change your app's colors according to your theme/branding

Open lib/my_theme.dart

You can change the `accent_color`, `soft_accent_color`, `splash_screen_color` variable.

Flutter by default does not support hex color. Do not change the other variables.

Use <https://www.rapidtables.com/convert/color/hex-to-rgb.html> To get the RGB value if you do not already know your theme's RGB color.

You should keep the Opacity value 1 (Opacity can be 0, 0.1, 0.2, ,0.9 ,1)

See the screenshot below.

```
active_ecommerce_flutter lib my_theme.dart
pubspec.yaml app_config.dart my_theme.dart
1:1 Project
1 import 'package:flutter/material.dart';
2
3 class MyTheme{
4   /*configurable colors starts*/
5   static Color accent_color = Color.fromRGBO(230,46,4, 1);
6   static Color soft_accent_color = Color.fromRGBO(247,189,168, 1);
7   static Color splash_screen_color = Color.fromRGBO(230,46,4, 1); // if not sure , use the same color as accent color
8   /*configurable colors ends*/
9
10  /*If you are not a developer, do not change the bottom colors*/
11  static Color white = Color.fromRGBO(255,255,255, 1);
12  static Color light_grey = Color.fromRGBO(239,239,239, 1);
13  static Color dark_grey = Color.fromRGBO(112,112,112, 1);
14  static Color medium_grey = Color.fromRGBO(132,132,132, 1);
15  static Color grey_153 = Color.fromRGBO(153,153,153, 1);
16  static Color font_grey = Color.fromRGBO(73,73,73, 1);
17  static Color textfield_grey = Color.fromRGBO(209,209,209, 1);
18  static Color golden = Color.fromRGBO(248, 181, 91, 1);
19  static Color shimmer_base = Colors.grey.shade50;
20  static Color shimmer_highlighted = Colors.grey.shade200;
21
22  //testing shimmer
23  /*static Color shimmer_base = Colors.redAccent;
24  static Color shimmer_highlighted = Colors.yellow; */
25
26
27
28
29 }
```

C. Configure the launcher icon:

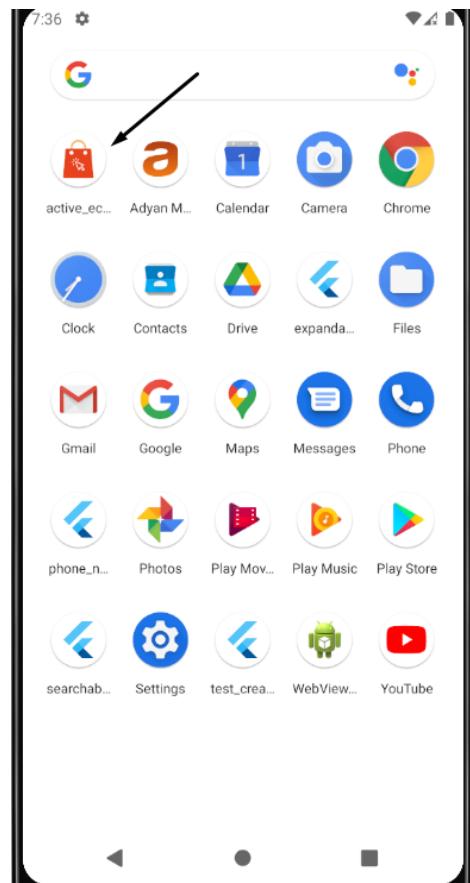
This helps you change your app's launcher icon.

Change the `app_logo.png` in `assets` folder with your own logo. Your file name should also be `app_logo.png` and it should be a `512x512 png` image and the image format should be the same.

After replacing the file, **uninstall** your app from your emulator. Otherwise the logo will not be changed.

Then in your android studio terminal run:

`flutter pub get`



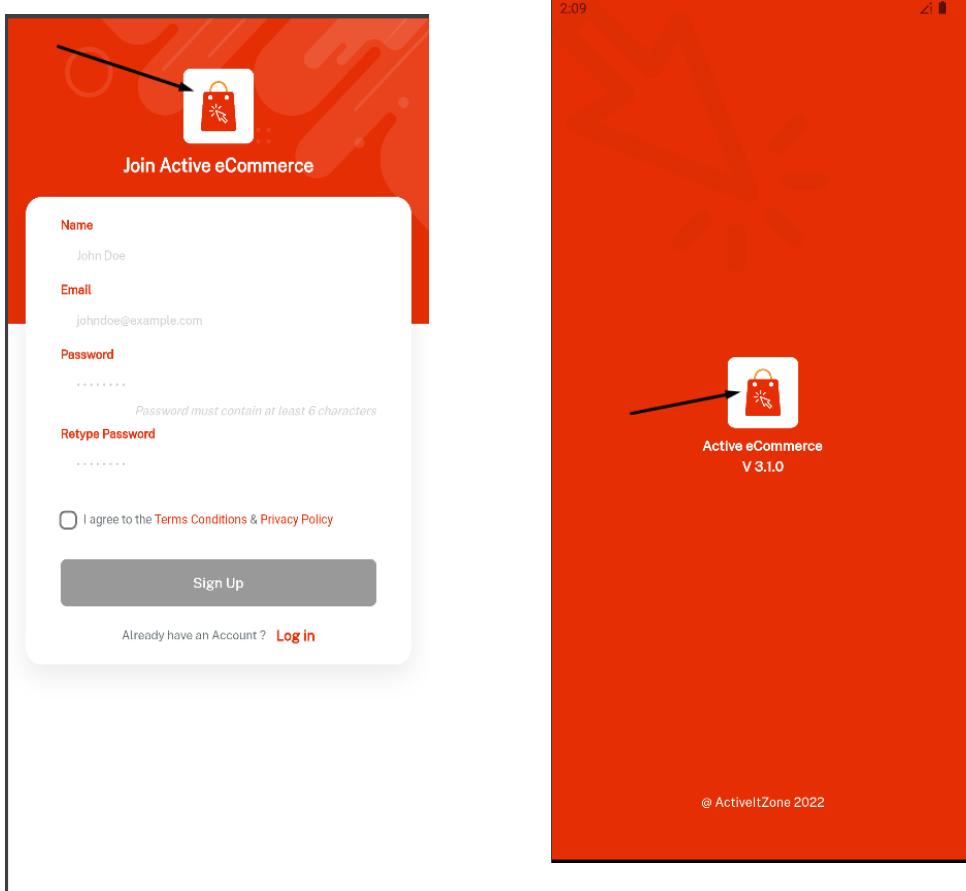
Then run :

flutter pub run flutter_launcher_icons:main

Then run your app (shift +10). The app will be installed again with your given launcher icon.

D. Configure other logos:

In the asset folders we have other logos that you may want to change according to your branding.



This logos will be found in :

[assets/login_registration_form_logo.png \(512x512\)](#)

[assets/splash_screen_logo.png \(512x512\)](#)

Change this logo with your own logo. File name , image format and size should be the same for each logo.

Then in your android studio terminal run:

`flutter pub get`

Then restart your app (shift +10). You should see your own logo in these places.

4. **How to change the package name ?**

This is very important. Your app cannot have the same package name as other app. If it does, the playstore will not accept it as an unique application. So rename your app according to your business/brand name.Try to write an unique package name.

Naming convention :

<https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html>

For example

Let's say your package is : `com.onatcipli.networkUpp`

And your app name is "`Network Upp`"

Then ,

Run this command inside your flutter project root.

Run the command in android studio terminal :

`flutter pub run rename --bundleId com.onatcipli.networkUpp`

`flutter pub run rename --appname "Network Upp"`

Try uninstalling the app from the emulator , then run the commands and then restart the app.

If it does not work, first uninstall, then restart the app then run the commands.

**In case the above do not work:

In Android

for **package name** just change in build build.gradle only (anddroid/app/build.gradle)

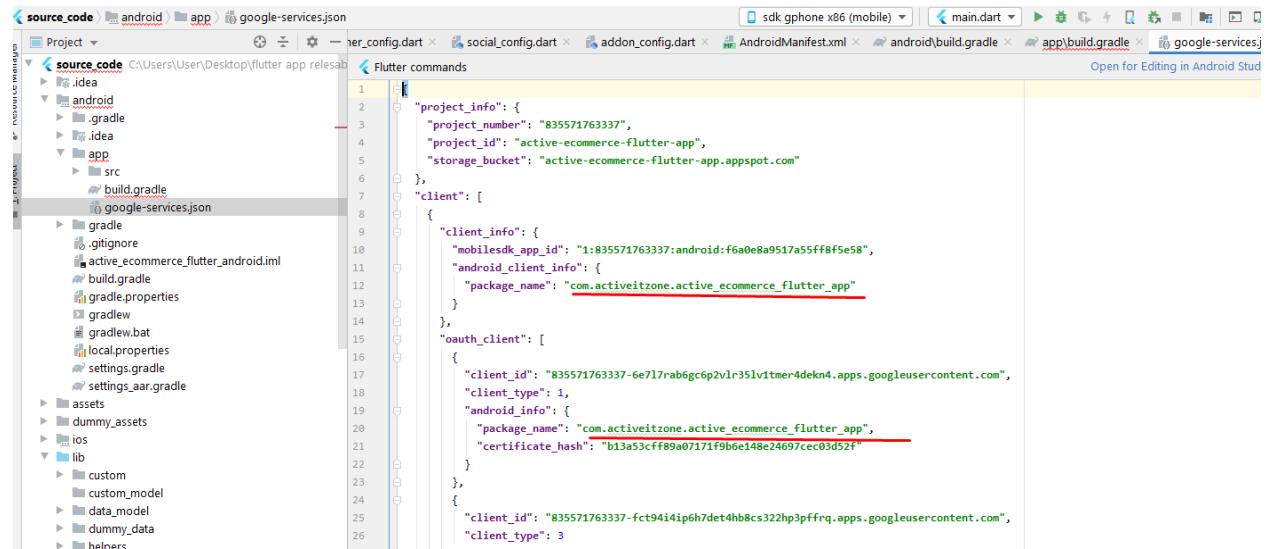
```
defaultConfig {  
    applicationId "bundleId com.onatcipli.networkUpp"  
    .....
```

```
}
```

**However in one place you must have to manually change your package name.

Open android/app/google-services.json , and change your package name manually.

Otherwise you will get a build error , even on emulator on debug build



```
1 "project_info": {
2     "project_number": "835571763337",
3     "project_id": "active-ecommerce-flutter-app",
4     "storage_bucket": "active-ecommerce-flutter-app.appspot.com"
5 },
6 "client": [
7     {
8         "client_info": {
9             "mobilesdk_app_id": "1:835571763337:android:f6a0e8a9517a59ff8f5e58",
10            "android_client_info": {
11                "package_name": "com.activeitzone.active_ecommerce_flutter_app"
12            }
13        },
14        "oauth_client": [
15            {
16                "client_id": "835571763337-6e7l7rab6gc6p2vlr35lv1tmer4dekn4.apps.googleusercontent.com",
17                "client_type": 1,
18                "android_info": {
19                    "package_name": "com.activeitzone.active_ecommerce_flutter_app",
20                    "certificate_hash": "b13a53cff89a07171f9b6e148e24697ccc03d52f"
21                }
22            },
23            {
24                "client_id": "835571763337-fct94i4ip6h7det4hb8cs322hp3pffrq.apps.googleusercontent.com",
25                "client_type": 3
26            }
27        ]
28    }
29 }
```

For iOS:

Change the bundle identifier from your Info.plist file inside your ios/Runner directory.

```
<key>CFBundleIdentifier</key>
<string>bundleId com.onatcipli.networkUpp</string>
```

If you face issues consult a flutter developer.

5. How to Build the app for testing (build an apk) ?

<https://flutter.dev/docs/deployment/android> see the doc for reference

In terminal run : `flutter build apk`

It will build an apk and show the folder. You can then install it in your phone to test, or share to multiple users for testing .

6. How to generate play store uploadable files for release?

<https://flutter.dev/docs/deployment/android> see the doc for reference

Signing the app:

To publish on the Play Store, you need to give your app a digital signature. Use the following instructions to sign your app.

Go through the screenshots below carefully to understand how to generate key and and use it for the released signed app:

Note:

- The `keytool` command might not be in your path—it's part of Java, which is installed as part of Android Studio. For the concrete path, run `flutter doctor -v` and locate the path printed after 'Java binary at:'. Then use that fully qualified path replacing `java` (at the end) with `keytool`. If your path includes space-separated names, such as `Program Files`, use platform-appropriate notation for the names. For example, on Mac/Linux use `Program\ Files`, and on Windows use "`Program Files`".
- The `-storetype JKS` tag is only required for Java 9 or newer. As of the Java 9 release, the keystore type defaults to `PKS12`.

Setup environment for keytool:

1. Run this command in your system terminal: `flutter doctor -v`
2. Find the Java binary at: copy this path except the "java" folder.
3. Change your terminal path by run this command : `cd "past here your copied path"` then press enter. Example: `cd C:\Program Files\Android\Android Studio\jre\bin`
4. Run the `keytool.exe` file. Example: `C:\Program Files\Android\Android Studio\jre\bin>keytool.exe`

```

C:\Users\User>flutter doctor -v
[✓] Flutter (Channel stable, 3.0.2, on Microsoft Windows [Version 10.0.19044.1889], locale en-US)
  • Flutter version 3.0.2 at C:\flutter_sdk\flutter
  • Upstream repository https://github.com/flutter/flutter.git
  • Framework revision cd41fdd495 (3 months ago), 2022-06-08 09:52:13 -0700
  • Engine revision f15f824b57
  • Dart version 2.17.3
  • DevTools version 2.12.2

[!] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)
  • Android SDK at C:\Users\User\AppData\Local\Android\sdk
  • Platform android-32, build-tools 32.1.0-rc1
  • Java binary at: C:\Program Files\Android\Android Studio\jre\bin\java (Copy this path)
  • Java version OpenJDK Runtime Environment (build 11.0.12+7-b1504.28-7817840)
  ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses

[✓] Chrome - develop for the web
  • Chrome at C:\Program Files\Google\Chrome\Application\chrome.exe

[X] Visual Studio - develop for Windows
  X Visual Studio not installed; this is necessary for Windows development.
    Download at https://visualstudio.microsoft.com/downloads/.
    Please install the "Desktop development with C++" workload, including all of its default components

[✓] Android Studio (version 2021.2)
  • Android Studio at C:\Program Files\Android\Android Studio
  • Flutter plugin can be installed from:
    https://plugins.jetbrains.com/plugin/9212-flutter
  • Dart plugin can be installed from:
    https://plugins.jetbrains.com/plugin/6351-dart
  • Java version OpenJDK Runtime Environment (build 11.0.12+7-b1504.28-7817840)

[✓] VS Code (version 1.70.2)
  • VS Code at C:\Users\User\AppData\Local\Programs\Microsoft VS Code
  • Flutter extension can be installed from:
    https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter

[✓] Connected device (4 available)
  • sdk gphone64 x86_64 (mobile)   • emulator-5554   • android-x64     • Android 12 (API 31) (emulator)
  • Windows (desktop)           • windows        • windows-x64   • Microsoft Windows [Version 10.0.19044.1889]
  • Chrome (web)                • chrome         • web-javascript • Google Chrome 104.0.5112.102
  • Edge (web)                  • edge           • web-javascript • Microsoft Edge 104.0.1293.54

[✓] HTTP Host Availability
  • All required HTTP hosts are available

! Doctor found issues in 2 categories.

C:\Users\User>cd C:\Program Files\Android\Android Studio\jre\bin
C:\Program Files\Android\Android Studio\jre\bin>keytool.exe
Key and Certificate Management Tool

Commands:

-centreq      Generates a certificate request
-changealias  Changes an entry's alias
-delete       Deletes an entry
-exportcert   Exports certificate
-genkeypair   Generates a key pair
-genseckeypair Generates a secret key
-gencert      Generates certificate from a certificate request
-importcert   Imports a certificate or a certificate chain

```

Create jks file for uploading app on playstore:

1. Open your project path and run this command: **keytool -genkey -v -keystore android/app/key.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias key**
2. Provide all information.

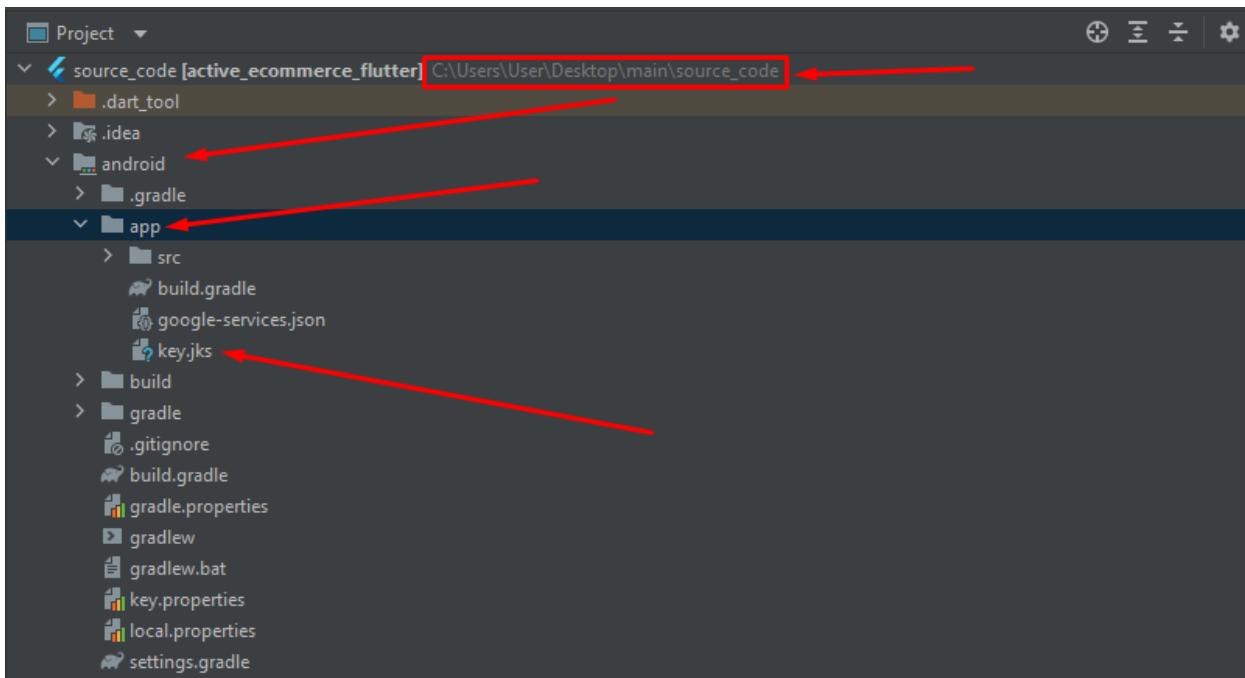
```

PS C:\Users\User\Desktop\main\source_code> keytool -genkey -v -keystore android/app/key.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias key
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: Test
What is the name of your organizational unit?
[Unknown]: Test
What is the name of your organization?
[Unknown]: Test
What is the name of your City or Locality?
[Unknown]: Test
What is the name of your State or Province?
[Unknown]: Test
What is the two-letter country code for this unit?
[Unknown]: TS
Is CN=Test, OU=Test, O=Test, L=Test, ST=Test, C=TS correct?
[no]: y

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10,000 days
    for: CN=Test, OU=Test, O=Test, L=Test, ST=Test, C=TS
Enter key password for <key>

```

Your created JKS file is located your project->android->app folder



Reference the keystore from the app

Create a file named `<your app dir>/android/key.properties` that contains a reference to your keystore:

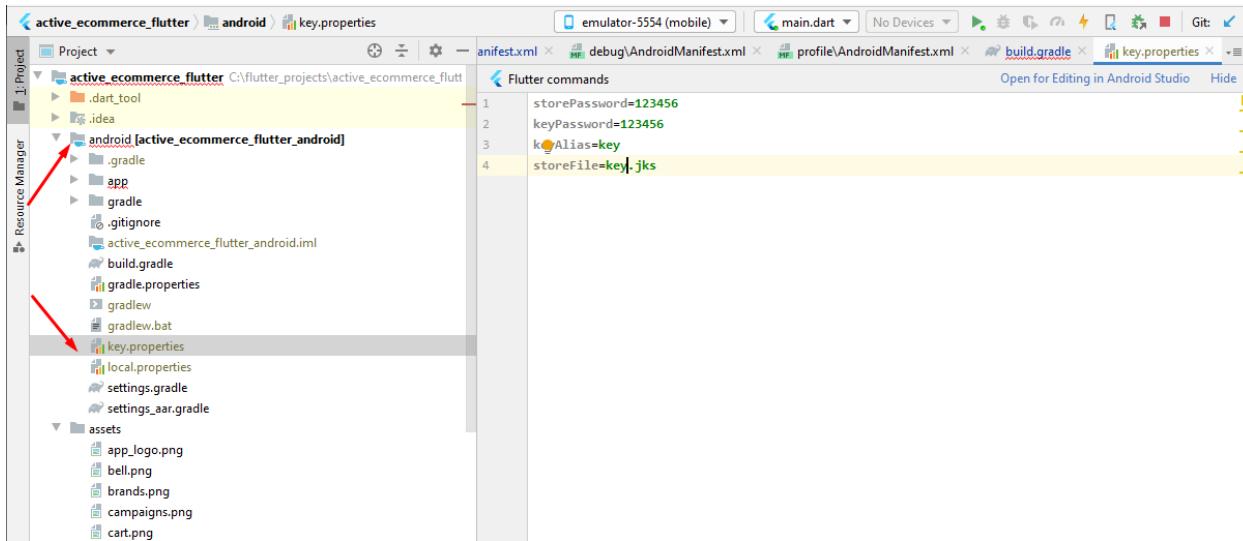
```

storePassword=<password from previous step>
keyPassword=<password from previous step>
keyAlias=key
storeFile=<location of the key store file, such as /Users/<user name>/key.jks>

```

**** If you lose the jks file , you will not be able to release a new update your app in playstore****

Create new file `key.properties` in android folder . Enter the information



Read this

Configure signing in gradle

Configure signing for your app by editing the `<your app dir>/android/app/build.gradle` file.

1. Add code before `android` block:

```
    android {
        ...
    }
```

With the keystore information from your properties file:

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
}

android {
    ...
}
```

Load the `key.properties` file into the `keystoreProperties` object.

2. Add code before `buildTypes` block:

```
buildTypes {
    release {
        // TODO: Add your own signing config for the release build.
        // Signing with the debug keys for now,
        // so 'flutter run --release' works.
        signingConfig signingConfigs.debug
    }
}
```

With the signing configuration info:

```
signingConfigs {
    release {
        keyAlias keystoreProperties['keyAlias']
        keyPassword keystoreProperties['keyPassword']
        storeFile keystoreProperties['storeFile'] ? file(keystoreProperties['storeFile']) : null
        storePassword keystoreProperties['storePassword']
    }
}
buildTypes {
    release {
        signingConfig signingConfigs.release
    }
}
```

Configure the `signingConfigs` block in your module's `build.gradle` file.

in app/build.gradle do necessary changes

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))

    android {
        compileSdkVersion 29

        sourceSets {
            main.java.srcDirs += 'src/main/kotlin'
        }

        lintOptions {
            disable 'InvalidPackage'
        }

        defaultConfig {
            // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
            applicationId "com.activeitzone.active_ecommerce_flutter_app"
            minSdkVersion 19
            targetSdkVersion 29
            versionCode flutterVersionCode.toInt()
            versionName flutterVersionName
            multiDexEnabled true
        }

        signingConfigs {
            release {
                keyAlias keystoreProperties['keyAlias']
                keyPassword keystoreProperties['keyPassword']
                storeFile keystoreProperties['storeFile'] ? file(keystoreProperties['storeFile']) : null
                storePassword keystoreProperties['storePassword']
            }
        }

        buildTypes {
            release {
                // TODO: Add your own signing config for the release build.
                // Signing with the debug keys for now, so `flutter run --release` works.
                signingConfig signingConfigs.release
            }
        }
    }
}
```

Note: You may need to run `flutter clean` after changing the gradle file. This prevents cached builds from affecting the signing process.

Now you are almost done

In your terminal run : `flutter build appbundle`

The release bundle for your app is created at [`<your app dir>/build/app/outputs/bundle/release/app.aab`](#).

Upload this app.aab file to your google play console

8. How to update for android? ****Read all the points carefully before doing anything**

- This section will help you if you are here for the update and have already generated the signed release apk/appbundle the last time and already have the keytool and the manifest file ready in your old project folder.
- If you are installing and building the release file for the first time this section is not for you.

- Extract the scource_code.zip. You will find this inside the main zip.
- Open the folder in your android studio.
- ****Remember to open this in a separate folder than your old project.**
- Even if you are building an app for ios, use android studio for the build.
- Then in your android studio terminal run:
`flutter pub get`
- This will fetch all the necessary packages

- If you are updating, you must have build the key.jks previously
- Copy the key.jks , key.properties, and the manifest file from your old project and paste in the correct locations
- See the previous screenshots for the file locations
- If you are missing your old project, you have to configure key.properties, and the manifest file like described in the installation.
- As our source code is made ready for the fresh installation , you will have to do all your configuration (like domain path, app color, package name etc) shown in the previous steps.
- But do not create a new key.jks, you have to update your app with the existing key
- If you have somehow lost your previous key , you have to release a totally new app to the play store. You will not be able to release an update.

- In your terminal run : `flutter build appbundle`
- The release bundle for your app is created at `<your app dir>/build/app/outputs/bundle/release/app.aab.`
- Upload this app.aab file to your google play console.

9. How to configure social login?

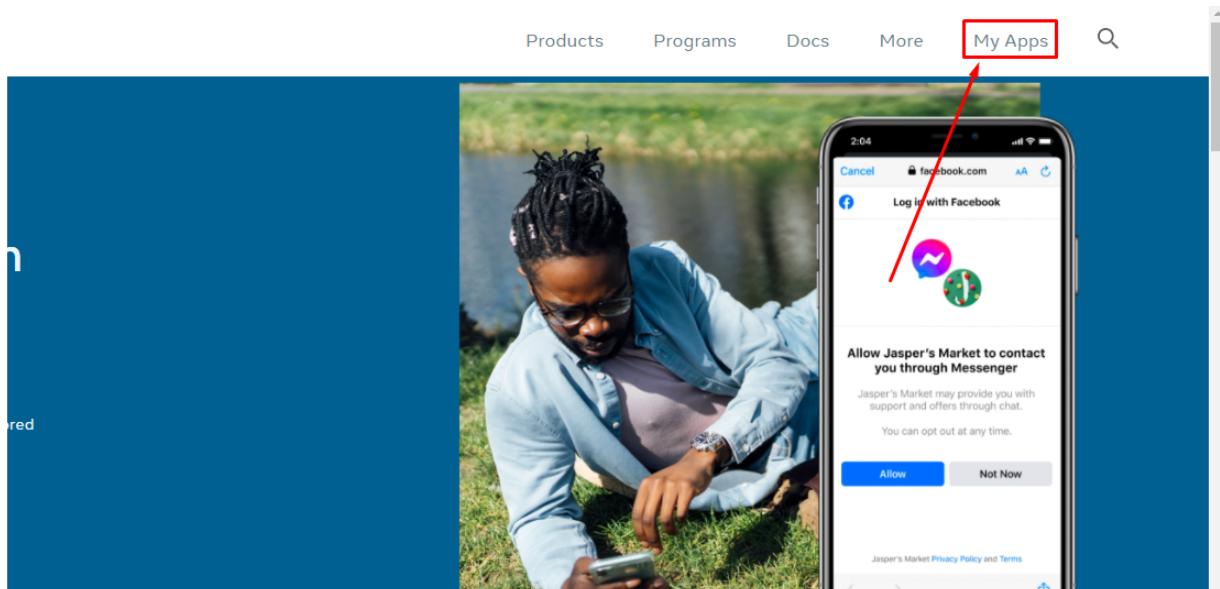
I) Facebook: Package Used

https://pub.dev/packages/flutter_facebook_auth

See its documentation and steps

How to create a facebook app ?

- A) First you have to login facebook in your browser.
- B) Goto <https://developers.facebook.com/>
- C) Goto My Apps.
- D)



The screenshot shows the Meta developer portal's main dashboard. At the top right, there is a green 'Create App' button with a white outline, which is highlighted with a red rectangular box. Below the header, there is a search bar labeled 'Search by App Name or App ID'. To the right of the search bar is a dropdown menu labeled 'Recently Used'. On the left side, there is a sidebar titled 'Apps' with a 'Filter by' section containing 'All Apps (2)' and 'Archived'. The main content area contains sections for 'Products', 'Programs', 'News', 'Support', 'Terms and Policies', and 'Activate Windows'. The footer features the Meta logo and social media links.

5. Select App Type None.

The screenshot shows the 'Create an App' wizard. On the left, there is a sidebar with 'Type' selected (highlighted with a blue box) and 'Details' available. The main area is titled 'Select an app type' with a sub-instruction: 'The app type can't be changed after your app is created. [Learn more](#)'. Below this, there are five options: 'Business' (with a brief description), 'Consumer' (with a brief description), 'Games' (with a brief description), 'Gaming' (with a brief description), and 'Workplace' (with a brief description). At the bottom, the 'None' option is selected (indicated by a blue dot) and highlighted with a red box. A brief description for 'None' is provided: 'Create an app with combinations of consumer and business permissions and products.' At the bottom right, there is a 'Next' button.

F) Add your app details

- App Name.
- App Contact Email.
- Business Account. (Don't Change it).

Create an App

X Cancel

Type Details

Add details

Display name **App Name**
This is the app name associated with your app ID. You can change this later.

App Contact Email **Email**
This email address is used to contact you about potential policy violations, app restrictions or steps to recover the app if it's been deleted or compromised.

Business Account · Optional
To access certain permissions or features, apps need to be connected to a Business Account.
No Business Account selected

Don't change this option

By proceeding, you agree to the [Facebook Platform Terms and Developer Policies](#).

Previous **Create App**

G) Add product "Facebook Login".

- ❖ Click the "Set Up" button.

Add a Product

Facebook Login

The world's number one social login product.

Read Docs **Set Up**

Audience Network

Monetize your app and grow revenue with ads from Facebook advertisers.

Read Docs **Set Up**

App Events

Understand how people engage with your business across apps, devices, platforms and websites.

Read Docs **Set Up**

Messenger

Customize the way you interact with people on Messenger.

Read Docs **Set Up**

Webhooks

Subscribe to changes and receive updates in real time without calling the API.

Read Docs **Set Up**

Games

Create a cross-platform HTML5 game hosted on Facebook.

Read Docs **Set Up**

H) Click the “Basic” Option.

The screenshot shows the Facebook App Settings interface. On the left, there's a sidebar with options like Dashboard, Settings (with 'Basic' highlighted by a red arrow), Roles, Alerts, App Review, Products, Facebook Login, and Activity Log. The main area is titled 'Basic' and contains fields for App ID, Display Name (set to 'testapp'), Namespace, App Domains, Contact Email (set to 'mdtusaraddmob@gmail.com'), Privacy Policy URL (set to 'Privacy policy for Login dialog and App Details'), Terms of Service URL (set to 'Terms of Service for Login dialog and App Details'), App Icon (with a placeholder image), and Category (set to 'Choose a Category').

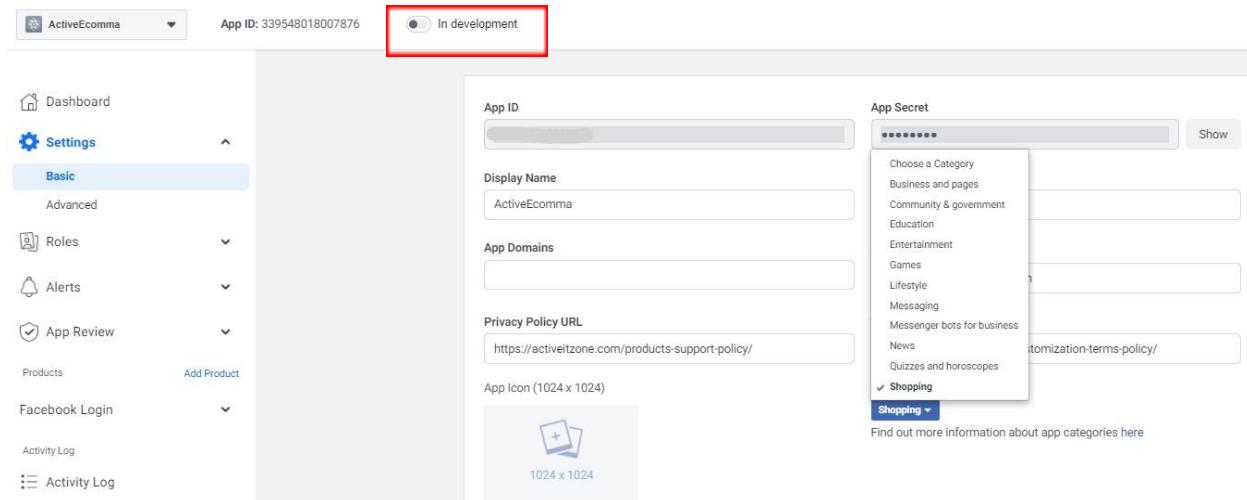
I) Add some information and save it.

- I. This app privacy policy url.
- II. This app terms of service url.
- III. This app icon (Icon size also 1024 X 1024 Or 512 X 512).
- IV. This app category.

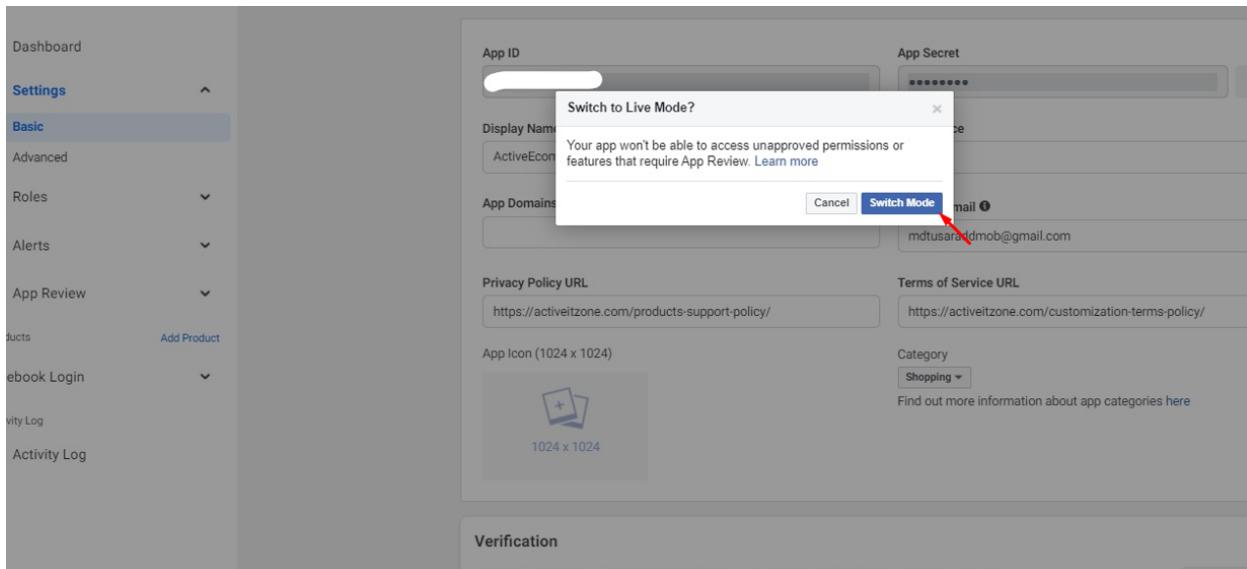
This screenshot shows the same Facebook App Settings page as above, but with several fields highlighted with red arrows:

- Privacy Policy URL**: Points to the 'Privacy policy for Login dialog and App Details' input field.
- Terms of Service URL**: Points to the 'Terms of Service for Login dialog and App Details' input field.
- App icon**: Points to the 'App Icon (1024 x 1024)' section, which includes a placeholder icon and the text '1024 x 1024'.
- Choose your app category**: Points to the 'Category' dropdown menu labeled 'Choose a Category'.

J) Now activate your app.



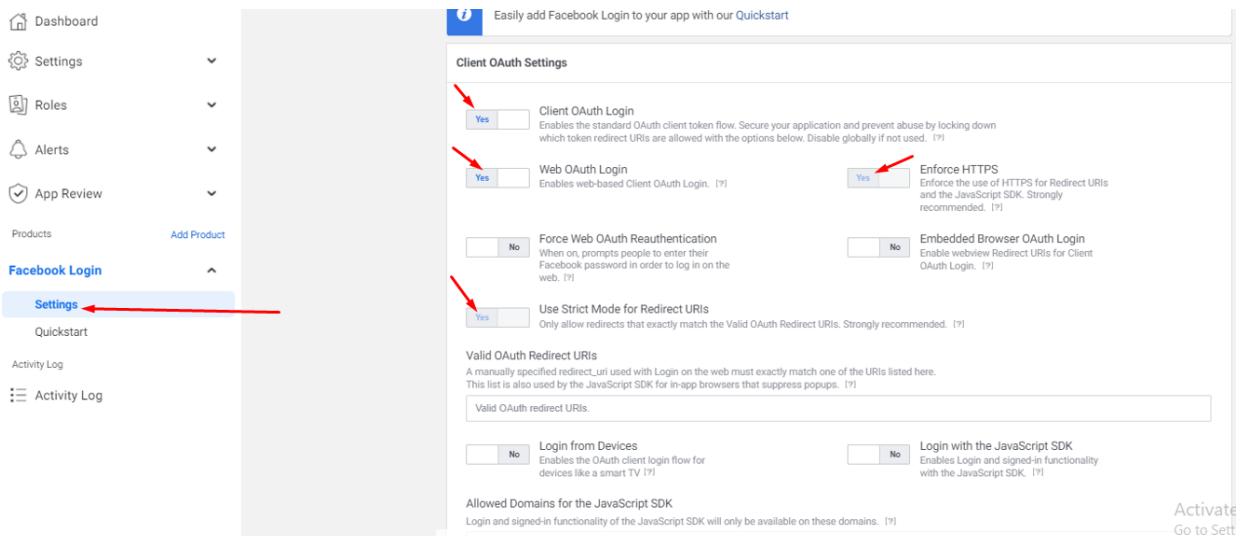
K) Click the “Switch Mode” Button.



L) Make sure your app facebook login settings are on these options.

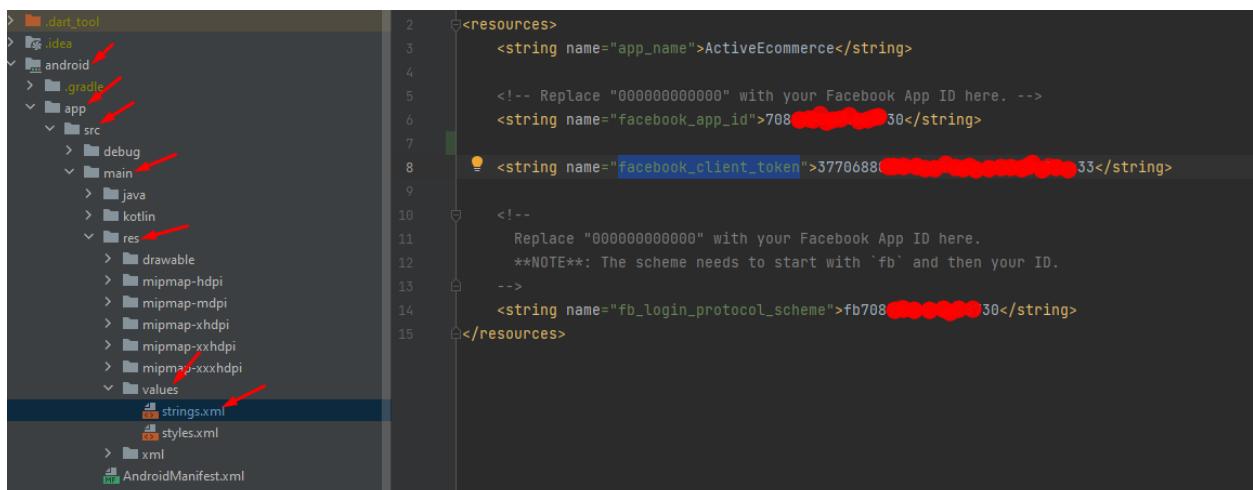
- A) Client OAuth Login.
- B) Web OAuth Login.
- C) Enforce HTTPS

D) Use Strict Mode for Redirect URIs



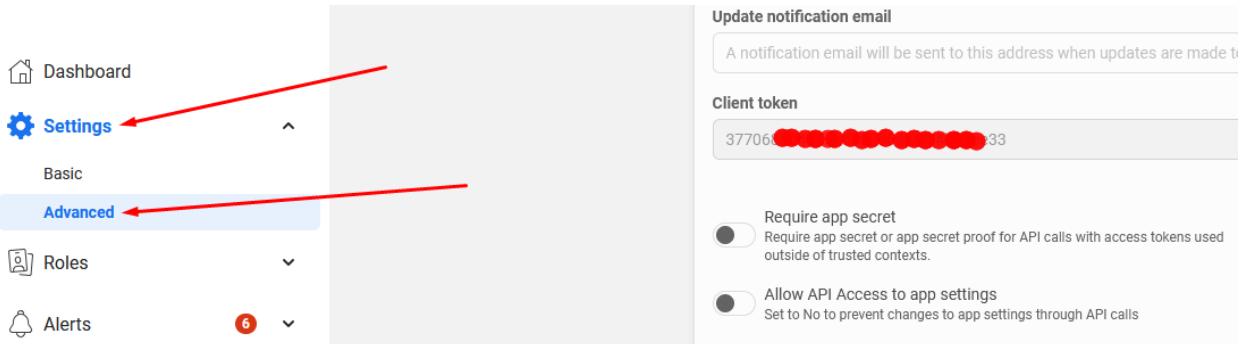
For Android:

M) Add your facebook app id, app name, `facebook_client_token` and fb login protocol scheme (***NOTE***: The scheme needs to start with `fb` and then your ID. example: `fb123456789`) into your `project->android->app->src->main->res->values->string.xml` file.



Client Token: Where do you get the client token?

Ans: Goto your facebook app setting->advance finde the Client token



For IOS: Add your facebook app id, app name, `facebook_client_token` and fb login protocol

scheme (**NOTE**: The scheme needs to start with `fb` and then your ID. example:
`fb123456789`)

into your project->ios->Runner->info.plist file.

The screenshot shows a file browser with the path `source_code [active_ecommerce_flutter] C:\Users\User\Desktop\main\source_code`. The 'ios' folder is expanded, showing subfolders like 'Flutter', 'Runner', and 'Info.plist'. Red arrows point from the 'ios' folder and the 'Info.plist' file to the code snippet below. The code is a portion of an info.plist file with several key-value pairs highlighted with red arrows:

```
<key>NSLocationAlwaysUsageDescription</key>
<string>User can set address using phone location</string>

<key>NSLocationWhenInUseUsageDescription</key>
<string>location when in use</string>

<key>NSBluetoothPeripheralUsageDescription</key>
<string>This app uses Bluetooth to connect with wearables.</string>

<key>NSBluetoothAlwaysUsageDescription</key>
<string>This app uses Bluetooth to connect with wearables.</string>

<!-- Facebook Login configuration -->
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>fbxxxxxxxxxx</string>
        </array>
    </dict>
</array>
<key>FacebookAppID</key>
<string>xxxxxxxxxx</string>
<key>FacebookClientToken</key>
<string>xxxxxxxxxxxxxxxxxx</string>
<key>FacebookDisplayName</key>
<string>Your App Name</string>

<key>LSApplicationQueriesSchemes</key>
<array>
    <string>fbapi</string>
    <string>fb-messenger-share-api</string>
    <string>fbauth2</string>
    <string>fbshareextension</string>
</array>
</dict>
</plist>
```

Annotations in the code:

- `fbxxxxxxxxxx` is labeled **App Id**.
- `xxxxxxxxxxxxxxxxxx` is labeled **Client Token**.
- `Your App Name` is labeled **App Name**.

Google: Package Used

https://pub.dev/packages/google_sign_in

See its documentation and steps from the link.

1. Create a firebase project.

- A) First you need to have a google account and login in your browser.
- B) Goto this link: <https://console.firebaseio.google.com/>
- C) Click the “Add Project” Button.
- D) Add your project name and continue.
- E) Off the “disable google analytics” and Click the “Continue” Button.
- F) Click the “Continue” Button.

Add Android app on your Firebase project

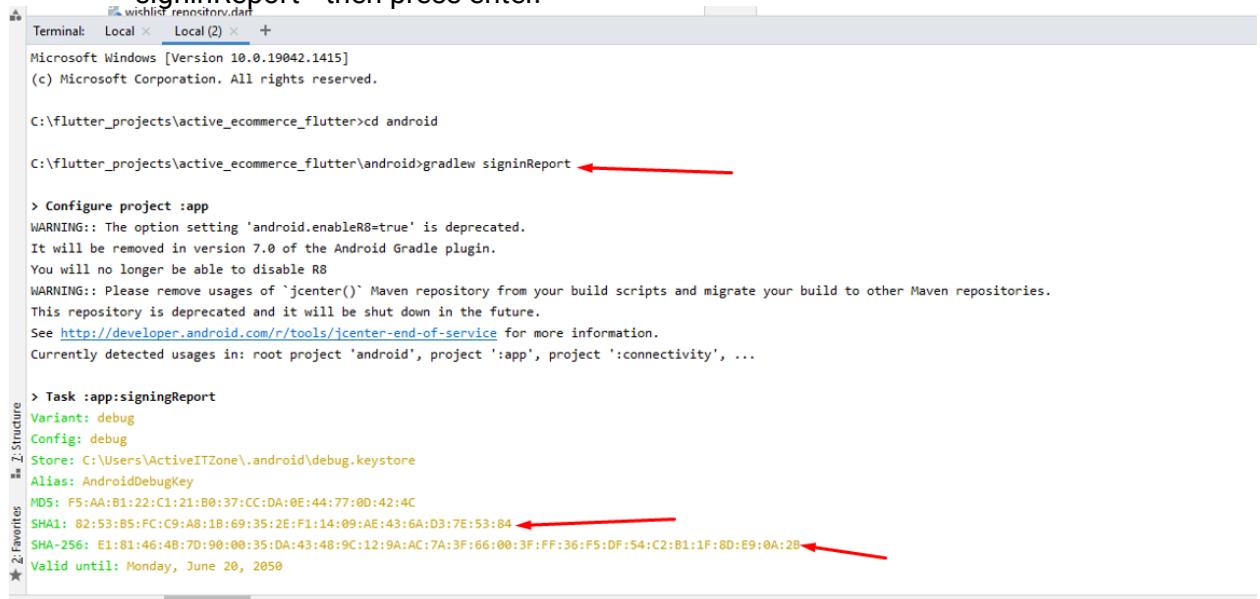
1. Click the android icon.

Add some informations:

- A) Android package name
- B) App nickname
- C) Add signing certificate SHA-1 and SHA-256

How to get SHA-1 code in your project ?

- A. Goto android studio and open your project then open terminal and ensure that it shows your project directory in the terminal.
- B. Then drive into the android folder(cd android).
- C. i)For debug mode: write this command on your android studio terminal “gradlew signinReport” then press enter.



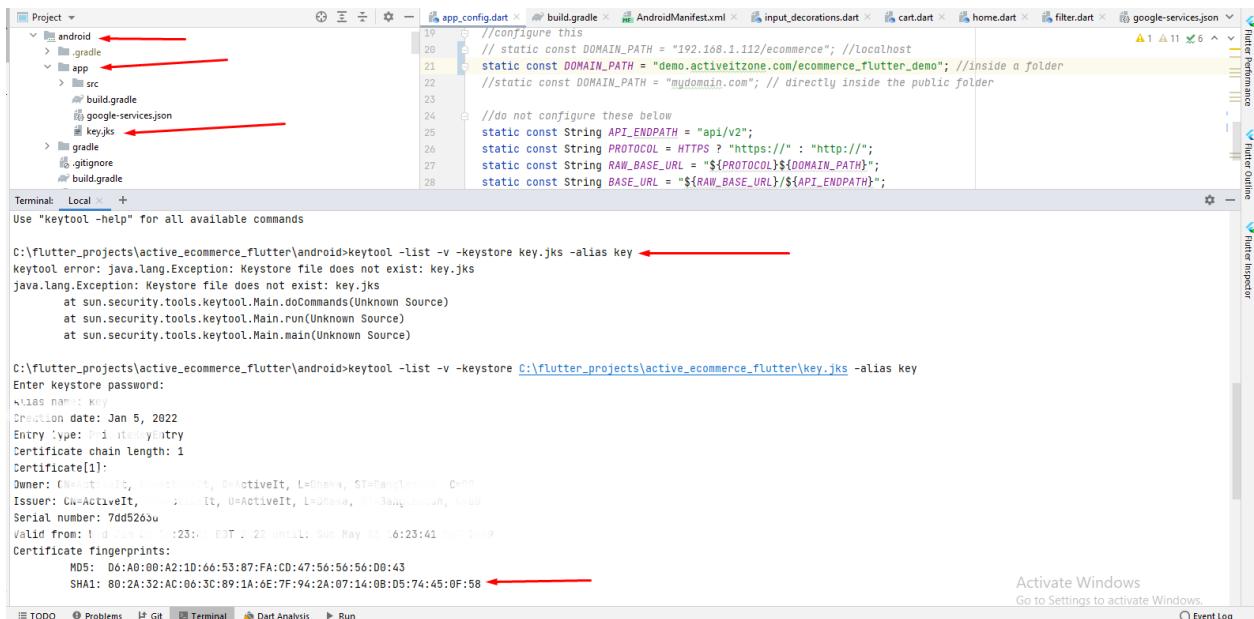
```
wishlist:renontrv.dat
Terminal Local Local (2) +
Microsoft Windows [Version 10.0.19042.1415]
(c) Microsoft Corporation. All rights reserved.

C:\flutter_projects\active_ecommerce_flutter>cd android
C:\flutter_projects\active_ecommerce_flutter\android>gradlew signinReport ←

> Configure project :app
WARNING: The option setting 'android.enableR8=true' is deprecated.
It will be removed in version 7.0 of the Android Gradle plugin.
You will no longer be able to disable R8
WARNING: Please remove usages of `jcenter()` Maven repository from your build scripts and migrate your build to other Maven repositories.
This repository is deprecated and it will be shut down in the future.
See http://developer.android.com/r/tools/jcenter-end-of-service for more information.
Currently detected usages in: root project 'android', project ':app', project ':connectivity', ...

> Task :app:signingReport
Variant: debug
Config: debug
Store: C:\Users\ActiveITZone\.android\debug.keystore
Alias: AndroidDebugKey
MD5: F5:AA:B1:22:C1:21:80:37:CC:DA:0E:44:77:0D:42:4C
SHA1: 82:53:B5:FC:C9:A8:1B:69:35:2E:F1:14:09:AE:43:6A:D3:7E:53:84 ←
SHA-256: E1:81:46:4B:7D:90:00:35:DA:43:48:9C:12:9A:AC:7A:3F:66:00:3F:FF:36:F5:DF:54:C2:B1:1F:8D:E9:0A:2B ←
Valid until: Monday, June 20, 2050
```

ii) For release mode: write this command on your android studio terminal
 "keytool -list -v -keystore YOUR_PROJECT_DIRECTORY\key.jks -alias key" then press enter.



The screenshot shows the Android Studio interface. The project structure on the left has red arrows pointing to the 'android' and 'app' folders, and another arrow pointing to the 'key.jks' file within the 'app' folder. The terminal window at the bottom shows the command being run:

```
C:\flutter_projects\active_ecommerce_flutter\android>keytool -list -v -keystore C:\flutter_projects\active_ecommerce_flutter\key.jks -alias key
keytool error: java.lang.Exception: Keystore file does not exist: key.jks
java.lang.Exception: Keystore file does not exist: key.jks
  at sun.security.tools.keytool.Main.doCommands(Unknown Source)
  at sun.security.tools.keytool.Main.run(Unknown Source)
  at sun.security.tools.keytool.Main.main(Unknown Source)

C:\flutter_projects\active_ecommerce_flutter\android>keytool -list -v -keystore C:\flutter_projects\active_ecommerce_flutter\key.jks -alias key
Enter keystore password:
Alias name: key
Creation date: Jan 5, 2022
Entry type: 1 trusted entry
Certificate chain length: 1
Certificate[1]:
Owner: CN=ActiveIt, O=ActiveIt, L=Delhi, ST=Delhi, C=IN
Issuer: CN=ActiveIt, O=ActiveIt, L=Delhi, ST=Delhi, C=IN
Serial number: 7dd5260u
Valid from: 1 Jan 2022 06:23:41 -05:00 to: 1 Jan 2029 06:23:41 -05:00
Certificate fingerprints:
  MD5: D6:A0:00:A2:1D:66:53:87:FA:CD:47:56:56:56:00:43
  SHA1: 80:2A:32:AC:06:3C:89:1A:6E:7F:94:2A:07:14:0B:D5:74:45:0F:58
```

Red arrows highlight the command line, the keystore path, and the SHA-1 fingerprint.

2. Click the “Register App” button
3. Download config file and add this file into your project->android->app folder
4. Add firebase SDK
5. Click the “Continue to console” button
- 6) After upload your app on playstore you need to add signing certificate SHA-1 and SHA-256 in firebase project setting : Open your google play console->Your App->Setup->App integrity

App integrity

Protect your app and your users [Show more](#)

Integrity API responses off [Releases signed by Google Play](#)

Integrity API App signing

App signing key certificate

This is the public certificate for the app signing key that Google uses to sign each of your releases. Use it to register your key with API providers. The app signing key itself is not accessible, and is kept on a secure Google server.

MD5 certificate fingerprint [REDACTED]

SHA-1 certificate fingerprint [REDACTED]

SHA-256 certificate fingerprint [REDACTED]

Upgrade your app signing key for new installs

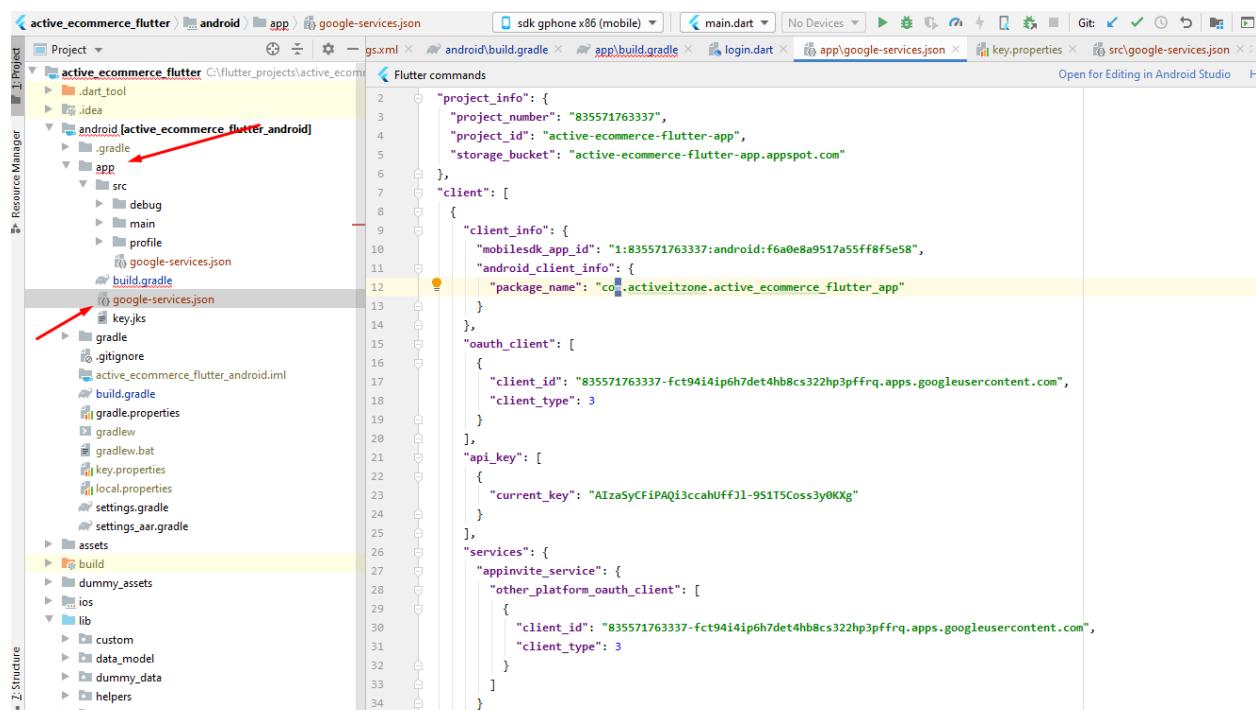
You can upgrade your app signing key for new installs, for example if you want to move to a cryptographically stronger key. Google will use the new key to sign all new installs and their updates. Your legacy key will still be used to sign updates for users who already have your app installed. You can request a new key once per app, and this works for all Android API levels. There are some important things to consider before requesting. [Learn more](#)

[Request key upgrade](#)

Upload key certificate

This is the public certificate for your private upload key. Use your upload key to sign each release so that Google knows updates are from you. Use the certificate below to register your upload key with API providers.

You will need to generate your own google-services.json. Do not use ours - it will not work for you.



Firebase Authentication :

1. Goto signin method and then enable Google and facebook.
2. For facebook you need a facebook app id and app secret, previously we created.
Add these documents.

The screenshot shows the Firebase console's Settings page for a new application. The left sidebar has 'Basic' selected under 'Settings'. The main area contains the following fields:

- App ID:** [REDACTED]
- App Secret:** [REDACTED] [Show](#)
- Display Name:** testapp
- App Domains:** [Empty]
- Privacy Policy URL:** [Empty]
- Terms of Service URL:** [Empty]
- App Icon (1024 x 1024):** [Placeholder icon]
- Category:** Choose a Category [▼](#)

A red arrow points to the 'Display Name' field. Below the form is a 'Verification' section with the message: "Some permissions and features require business or individual verification to access certain types of data. You can start the verification process at any time." It includes buttons for 'Start Verification', 'Go to Settings', 'Discard', and 'Save changes'.

<https://console.firebaseio.google.com/u/0/>

Follow the guideline from here https://pub.dev/packages/google_sign_in

Twitter : package used

https://pub.dev/packages/twitter_login

How to create a Twitter app.

- 1.1. First you have to login twitter in your browser.
- 1.2. Go to <https://developer.twitter.com/>
- 1.3. Go to [Developer Portal](#).
- 1.4. Go to Project & Apps->Overview create a new project.

D

Developer Portal

Dashboard

Projects & Apps

Overview

Products NEW

Account

Elevated

NO PROJECTS HERE **+ New Project**

Standalone Apps

V1.1 ACCESS

1.5. Fill up some information for your project. Project Name, Use case, Project description.

1

2

3

Project name

Your Project helps you organize your work and monitor your usage with the Twitter API.

Your project name

Use case

This is how you intend to use the Twitter developer platform.

Building consumer tools (professional)

Select a use case

- Doing academic research
- Making a bot
- Build customized solutions in-house
- Embedding Tweets in a website
- Exploring the API
- Doing something else
- Publishing ads programmatically
- Student
- Teaching
- Building tools for businesses
- Building consumer tools (hobbyist)
- Building consumer tools (professional)**

Project description

Required

2. Setup your app

- 2.1. Select your app environment.
- 2.2. Enter your app name.
- 2.3. Collect your **API Key** and **API Key Secret**.(you will have to collect it.)

The image consists of three side-by-side screenshots from a mobile application interface:

- Screenshot 1 (App Environment):** Shows three environment options: Development (selected), Staging, and Production. The Development section is highlighted with a red box and has a red number "1" below it.
- Screenshot 2 (App name):** Shows the "App name" field, which is required (indicated by a red asterisk) and has a red number "2" below it.
- Screenshot 3 (Keys & Tokens):** Shows the "API Key" and "API Key Secret" fields. Both fields have red arrows pointing to them, and each has a "Copy" button with a red border. A red number "3" is located below the "API Key Secret" field.

3. Setup user authentication setting.

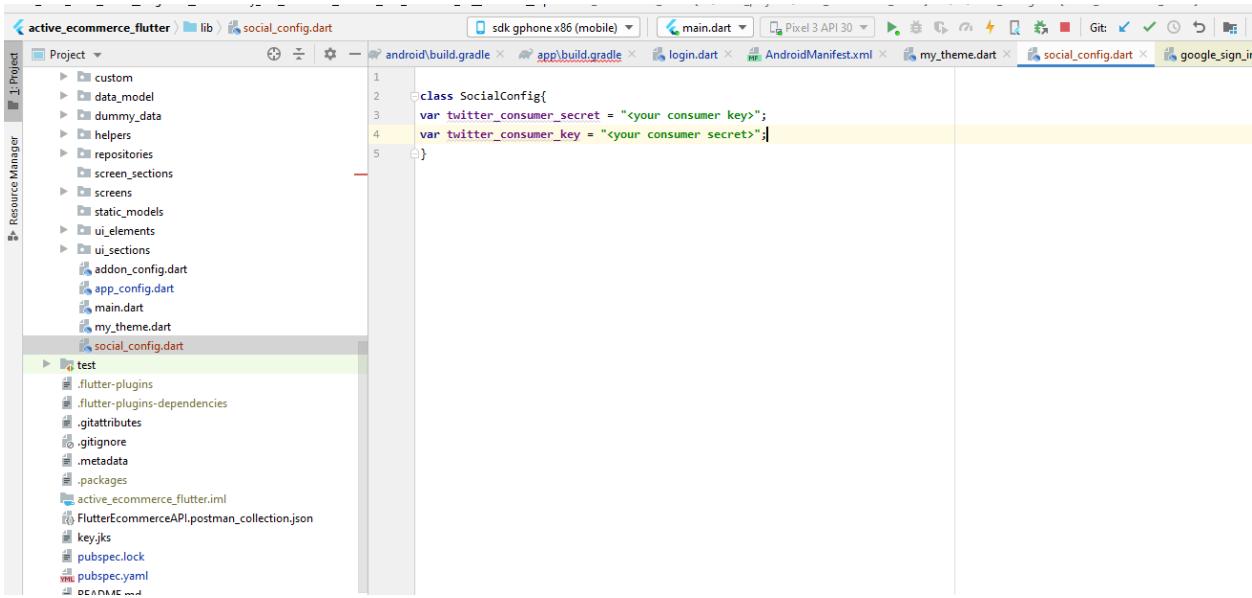
- 3.1. Go to Project & Apps-> select your project and then select your created app.
- 3.2. Find **User authentication settings**.
- 3.3. Click the **Setup** button.
- 3.4. Enable OAuth 2.0
- 3.5. Select **Type of App.(Native app)**
- 3.6. Callback URI.(**activeecommerceflutterapp://**).
- 3.7. Web site URL.(It's your web URL).
- 3.8. Save it.

The image consists of four side-by-side screenshots from a mobile application interface:

- Screenshot 3.3:** Shows the "User authentication settings" screen with a note about OAuth 2.0 and OAuth 1.0a. A red box highlights the "Set up" button, and a red number "3.3" is below it.
- Screenshot 3.4:** Shows the "OAuth 2.0" section with a green "NEW" badge. A red box highlights the toggle switch, and a red number "3.4" is below it.
- Screenshot 3.5:** Shows the "OAUTH 2.0 SETTINGS" screen where "Type of App" is set to "Native App". A red box highlights the dropdown menu, and a red number "3.5" is below it.
- Screenshot 3.6:** Shows the "GENERAL AUTHENTICATION SETTINGS" screen with the "Callback URI / Redirect URL" field containing "activeecommerceflutterapp://". A red box highlights the field, and a red number "3.6" is below it.
- Screenshot 3.7:** Shows the same "GENERAL AUTHENTICATION SETTINGS" screen with the "Website URL" field containing "https://activeitzone.com/". A red box highlights the field, and a red number "3.7" is below it.

4. Setup in your flutter e-commerce code.

- 4.1. Go to your flutter project->lib->social_config.dart
- 4.2. Enter your `twitter_consumer_secret` and `twitter_consumer_key`.



```
active_ecommerce_flutter lib social_config.dart
Project social_config.dart
1
2 class SocialConfig{
3     var twitter_consumer_secret = "<your consumer key>";
4     var twitter_consumer_key = "<your consumer secret>";
5 }
```

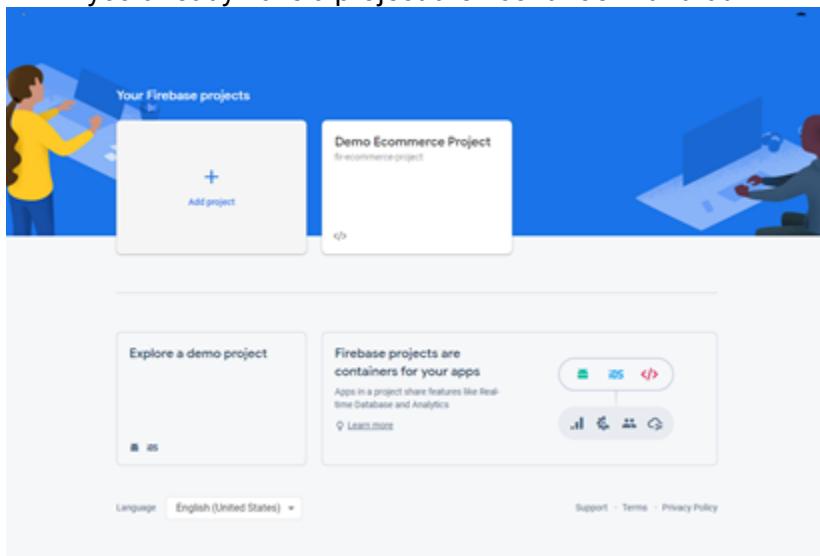
The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the Project Manager, which lists various files and folders under the 'lib' directory, including 'social_config.dart'. The main editor area displays the code for 'social_config.dart', which contains a single class definition for 'SocialConfig' with two static variables: 'twitter_consumer_secret' and 'twitter_consumer_key', both of which are currently set to placeholder strings ('<your consumer key>' and '<your consumer secret>').

10. How to configure push notification?

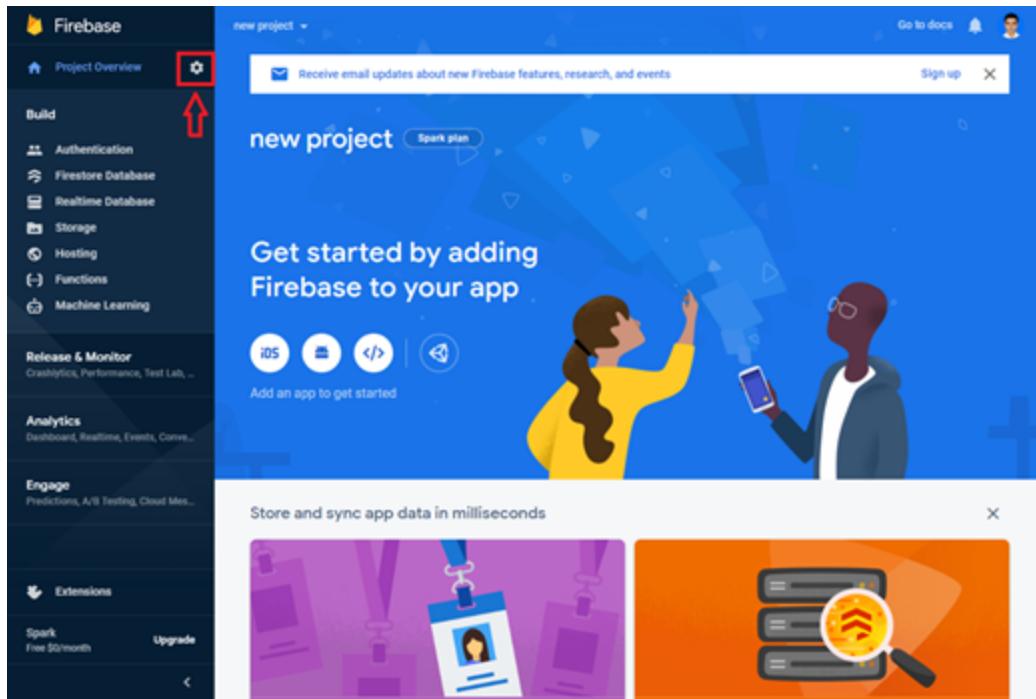
To use firebase follow the procedure which are mentioned below

1. Go to this URL to create project <https://console.firebaseio.google.com/u/0/>

If you already have a project then continue with that.



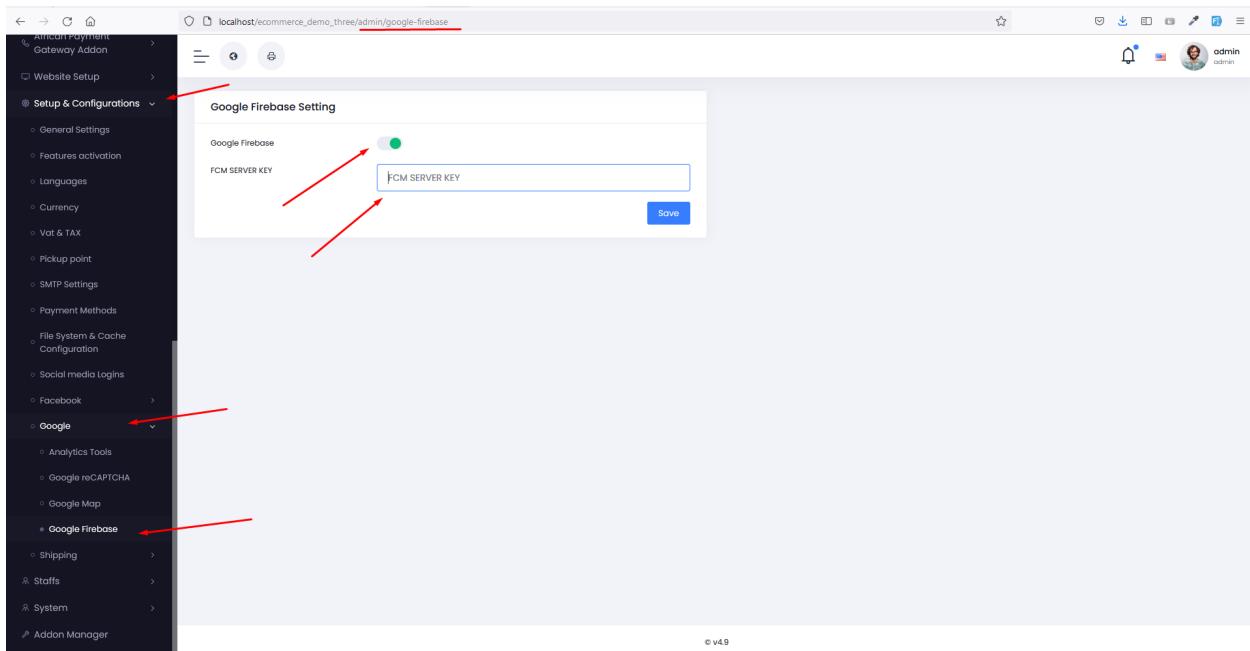
2. Now go to project settings to get server key



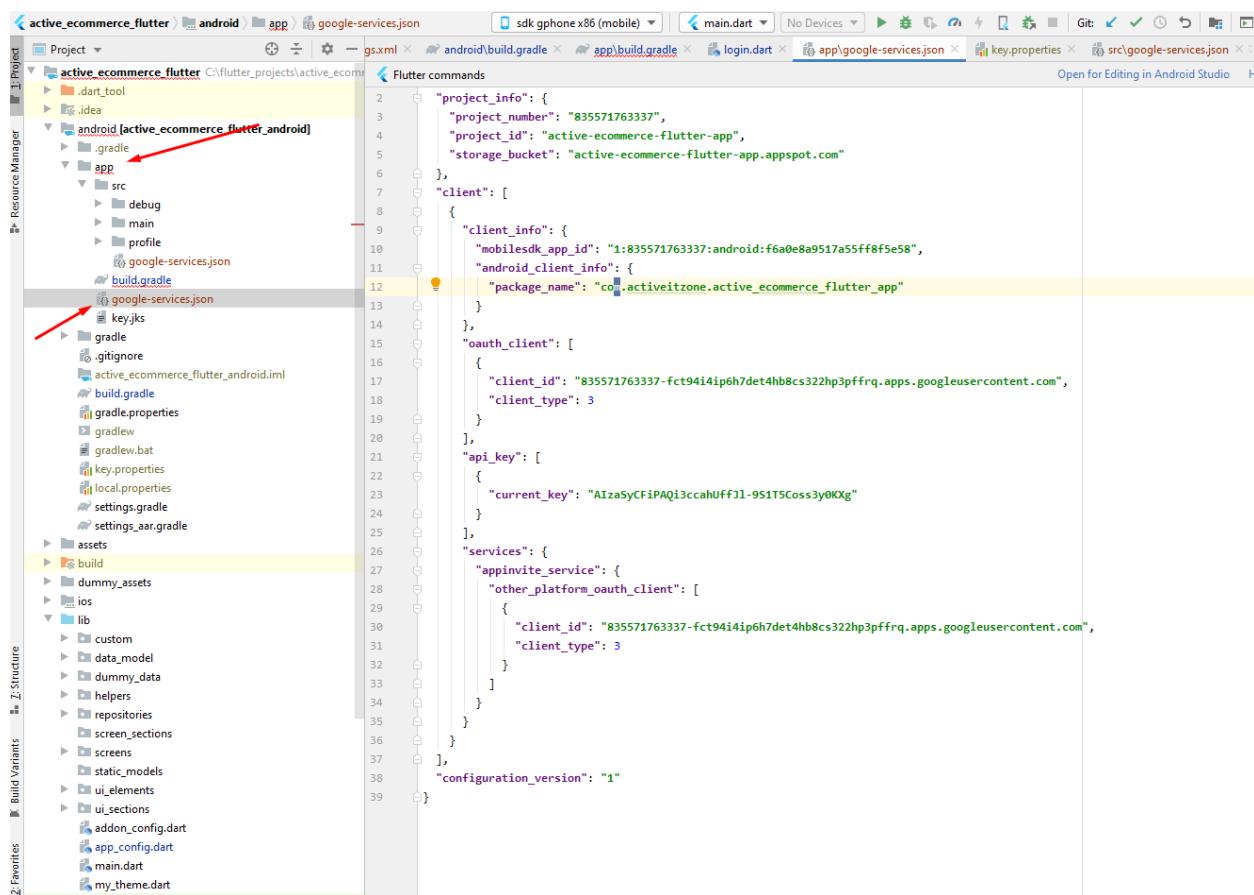
3. To get server key click on Cloud Messaging option

A screenshot of the Project settings page, specifically the Cloud Messaging tab. The sidebar on the left remains the same. The main content area has a tab bar at the top with "General", "Cloud Messaging" (which is highlighted in blue), "Integrations", "Service accounts", "Data privacy", "Users and permissions", and "App Check". Under the "Cloud Messaging" tab, there's a "Project credentials" section with a table. The first row contains "Key" and "Token". The second row contains "Server key" and a large input field which is also highlighted with a red box and a red arrow pointing towards it from below. There's also a "Sender ID" input field below it. Further down, there are sections for "iOS app configuration" and "Web configuration", each with its own sub-sections like "Web Push certificates".

4. Turn on the switch and put the server key in admin panel



5. You will need to generate your own google-services.json. Do not use ours - it will not work for you



Firebase console:

<https://console.firebaseio.google.com/u/0/>

Follow the guideline from here https://pub.dev/packages/google_sign_in

You need to provide your fingerprints here (sha1 and sha 256)

The screenshot shows the Firebase console interface. On the left, there's a sidebar with various services like Authentication, Firestore, and Functions. The main area is titled 'active-commerce-flutter-app' and shows 'Project settings'. Under 'Your apps', there's a card for an 'Android app' named 'active_commerce_flutter_app' with the package name 'com.activeitzone.active_ecommerce_flutter_app'. Below the app card, under 'SDK setup and configuration', it shows the App ID (1:83557176337:android:f6a0e8a9517a55ff8f5e58), App nickname ('active_commerce_flutter_app'), Package name ('com.activeitzone.active_ecommerce_flutter_app'), SHA certificate fingerprints ('32:13:21:32:45:64:32:13:25:45:61:43:21:65:34:65:13:21:46:54'), and Type ('SHA-1'). There's also a 'See SDK instructions' button and a 'google-services.json' download link. At the bottom of the configuration panel, there's a blue 'Add fingerprint' button.

You will find your signature/fingerprints from here (Provided that you already have generated the key). You will also need the path of your key.jks. You may have already kept it in the root folder.

```
C:\Program Files\Android\Android Studio\jre\bin>keytool -list -v -keystore C:\flutter_projects\active_ecommerce_flutter\key.jks -alias key -storepass 123456 -keypass 123456
Alias name: key
Creation date: Apr 1, 2021
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Test, OU=Test, O=Test, L=Test, S=Test, C=us
Issuer: CN=Test, OU=Test, O=Test, L=Test, S=Test, C=us
Serial number: 656ab3f5
Valid from: Thu Apr 01 21:56:13 BDT 2021 until: Mon Aug 17 21:56:13 BDT 2048
Certificate fingerprints:
    MD5: B4:6B:55:48:8F:D9:E1:1B:43:2D:76:3D:99:1A:D9:B8
    SHA1: B1:3A:53:CF:F8:9A:07:17:1F:9B:6E:14:8E:24:69:7C:EC:03:D5:2F
    SHA256: 92:09:F0:BF:56:F8:14:AB:AD:C8:C6:43:1D:79:FA:3F:66:2E:D8:2D:66:FD:5F:BE:B0:10:88:06:FA:37:46:A1
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:
#1 ObjectId: 2.5.29.14 Criticality=False
```

7. Although most of the configuration for android is done you can check guidelines from here.

<https://firebase.google.com/docs/cloud-messaging/android/client>

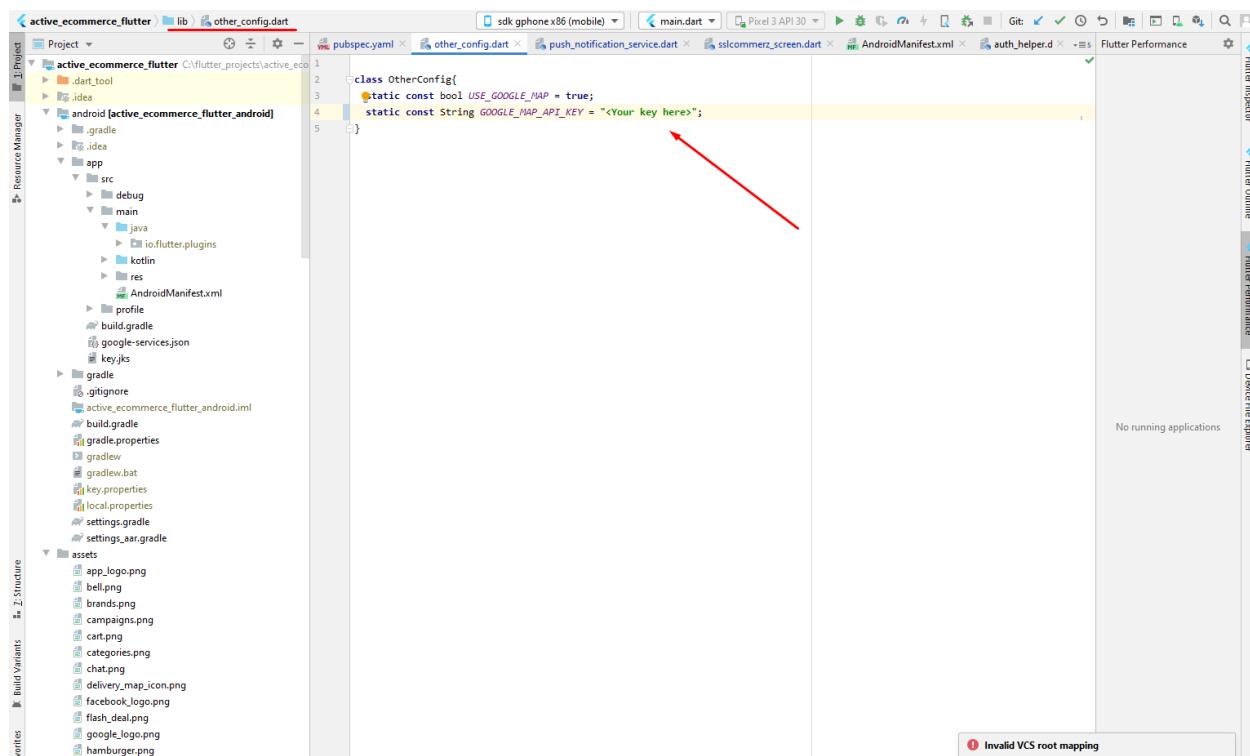
8. For ios follow this <https://firebase.google.com/docs/cloud-messaging/ios/client>

9. Push notification is a little bit tricky , so follow the guidelines properly. Learn more about how a firebase application connects with your mobile app from google searching if needed.

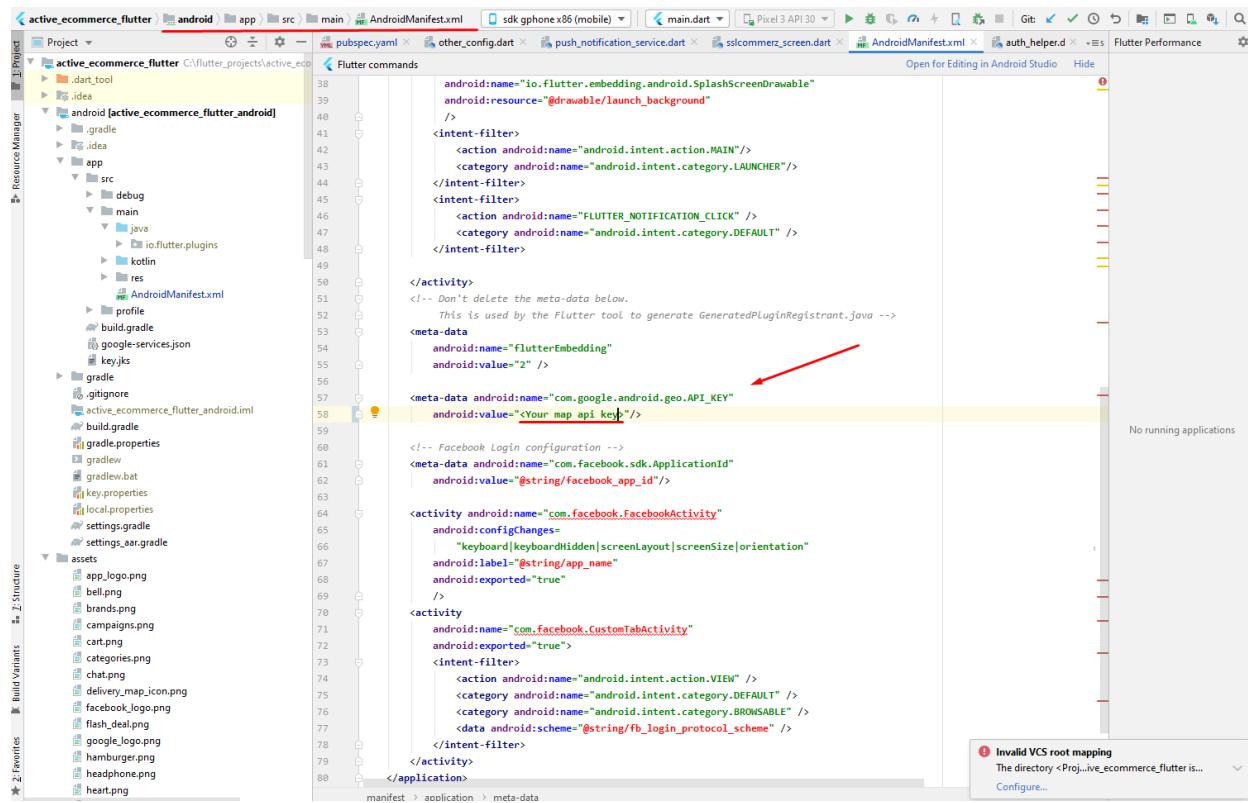
11. How to configure google map? (Read the whole thing before implementing)

1. Go to <https://console.developers.google.com/> and generate api keys separately for ios and android. No restrictions are needed

1. In lib/other_config.dart make, use google map = true and put google map api key



2. In main AndroidManifest.xml put the map api key

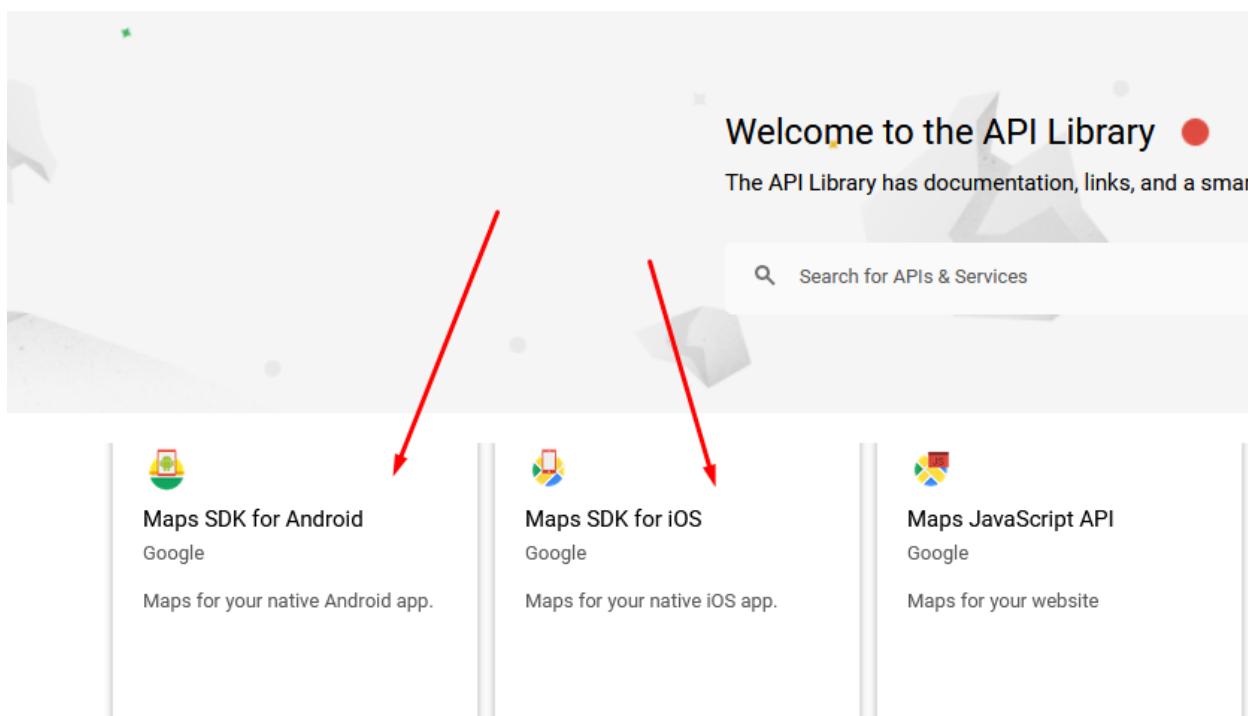


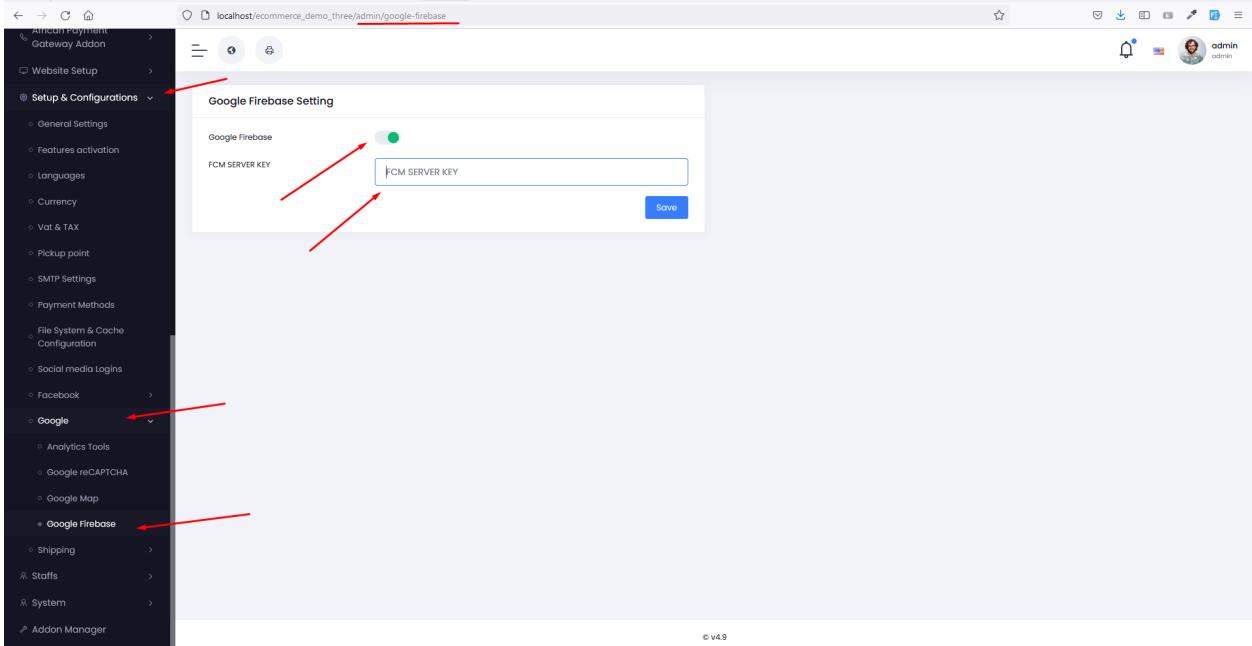
```
38
39
40    android:name="io.flutter.embedding.android.SplashScreenDrawable"
41    android:resource="@drawable/launch_background"
42  />
43  <intent-filter>
44    <action android:name="android.intent.action.MAIN"/>
45    <category android:name="android.intent.category.LAUNCHER"/>
46  </intent-filter>
47  <intent-filter>
48    <action android:name="FLUTTER_NOTIFICATION_CLICK" />
49    <category android:name="android.intent.category.DEFAULT" />
50  </intent-filter>
51
52  </activity>
53  <!-- Don't delete the meta-data below.
54      This is used by the Flutter tool to generate GeneratedPluginRegistrant.java -->
55  <meta-data
56    android:name="flutterEmbedding"
57    android:value="2" />
58
59  <meta-data android:name="com.google.android.geo.API_KEY"
60    android:value="
```

3. For ios follow this

<https://blog.logrocket.com/adding-google-maps-to-a-flutter-app/#addinggooglemapstoflutterios>

4. Enable android and ios api. These are free.





5. In the customer app we are searching location via text .And while setting pin to location taking information from the location. For these we would need these apis enabled. Unfortunately these api are not free, you will need to add card.If you do not want to spend money you cannot use google map in the customer app



Geocoding API

Google Enterprise API

Convert between addresses and geographic coordinates.

[MANAGE](#)

API Enabled



Places API

Google Enterprise API

Get detailed information about 100 million places

[MANAGE](#)

API Enabled

12. How to configure the default language for mobile apps?

Go to your flutter project->lib->app_config.dart

Change variables value //Default language config

```
static String default_language ="en";
static String mobile_app_code ="en";
static bool app_language_rtl =false;
```

```

import 'package:flutter/material.dart';

var this_year = DateTime.now().year.toString();

class AppConfig {
    static String copyright_text = "@ ActiveItZone " + this_year; //this shows in the splash
    static String app_name = "Active eCommerce"; //this shows in the splash screen
    static String purchase_code = ""; //enter your purchase code for the app from codecanyon
    //static String purchase_code = ""; //enter your purchase code for the app from codecanyon

    //Default language config
    static String default_language ="en";
    static String mobile_app_code ="en";
    static bool app_language_rtl =false;

    //configure this
    static const bool HTTPS = false;
}

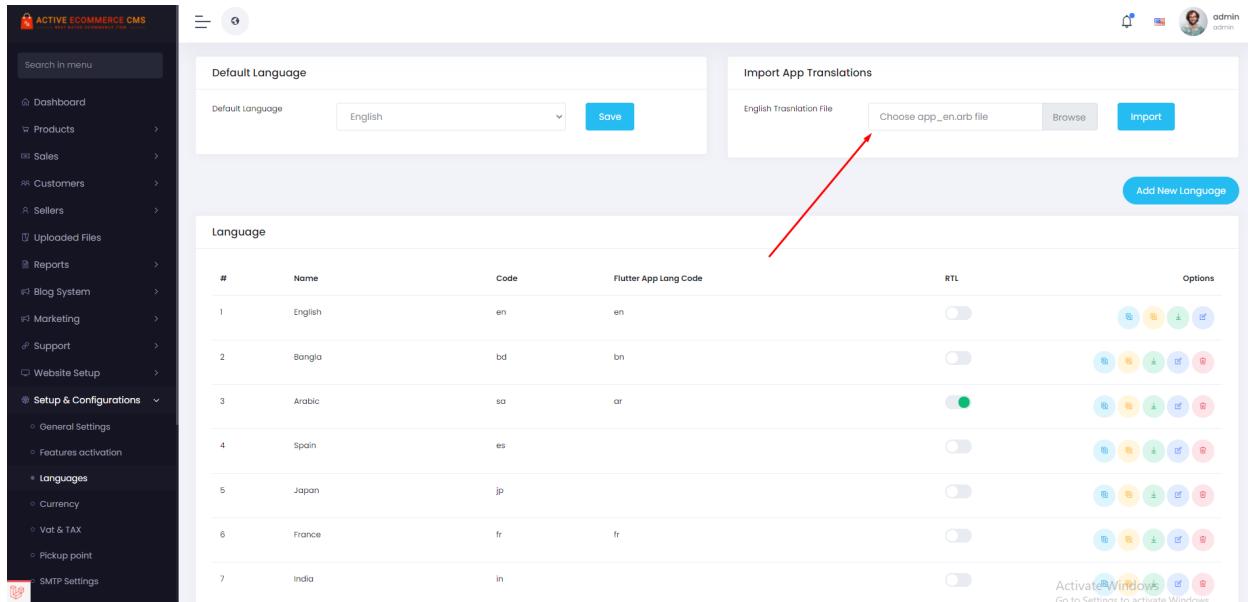
```

This value you can find in your admin panel. Go to your admin panel->setup & configurations->languages.

| # | Name | default_language Code | mobile_app_code Flutter App Lang Code | app_language_rtl RTL | Options |
|---|---------|--------------------------|--|-------------------------------------|-----------------|
| 1 | English | en | en | <input type="checkbox"/> | EN |
| 2 | Bangla | bd | bn | <input type="checkbox"/> | BN |
| 3 | Arabic | sa | ar | <input checked="" type="checkbox"/> | AR |

13. How to configure multiple languages for mobile app? (Read the whole thing before implementing)

1. In your **lib/l10n** folder you will see an **app_en.arb** file. This is your main translation + interpretation file. Never delete this.NEVER.
2. If you want another language file you can copy the app_en.arb file and make another language file like app_fr.arb and so on. But we will suggest that you use our translation generator from admin panel.
3. Always make sure your language code is valid.
https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes Use iso 639-1 codes. By default flutter localization uses 78 major language codes from here.
4. Upload **app_en.arb** in the admin panel. It will fetch strings from the file and uploads to your database.



The screenshot shows the Active Ecommerce CMS admin panel interface. On the left, there's a sidebar with various menu items like Dashboard, Products, Sales, Customers, Sellers, Reports, Blog System, Marketing, Support, Website Setup, and Setup & Configurations (which is expanded to show General Settings, Features activation, Languages, Currency, Vat & TAX, and Pickup point). At the bottom of the sidebar, there's a link to SMTP Settings. The main content area has two main sections: 'Default Language' (set to English) and 'Import App Translations' (with a 'Choose app_en.arb file' button highlighted by a red arrow). Below these is a table titled 'Language' with columns for #, Name, Code, Flutter App Lang Code, RTL, and Options. The table lists seven languages: English, Bangla, Arabic, Spain, Japan, France, and India. Each row in the table has a set of colored circular icons in the 'Options' column. At the bottom right of the main area, there's a message: 'Activate Windows' and 'Go to Settings to activate Windows.'

5. Make sure while adding/editing a language , your flutter app language code exists. The code must be in iso 639-1format. Without a valid code , you will not see a translated output in app.

| # | Name | Code | Flutter App Lang Code | RTL | Options |
|---|-------------|------|-----------------------|-------------------------------------|---------|
| 1 | English | en | en | <input type="checkbox"/> | |
| 2 | Bangla | bd | bn | <input type="checkbox"/> | |
| 3 | Arabic | sa | ar | <input checked="" type="checkbox"/> | |
| 4 | Spain | es | | <input type="checkbox"/> | |
| 5 | Japan | jp | | <input type="checkbox"/> | |
| 6 | France | fr | fr | <input type="checkbox"/> | |
| 7 | India | in | | <input type="checkbox"/> | |
| 8 | Netherland | nl | | <input type="checkbox"/> | |
| 9 | Afghanistan | af | | <input type="checkbox"/> | |

6. Then translate your app strings like you did for your web. You can use google chrome's translation extension and the copy button for a faster output. See, our documentation on translation is provided with the cms. Remember the translations for web and app are kept separate, so even if you did create the translation, for the web , you have to create it for the mobile app too.

7. Once all the strings are converted for a particular language , say for example french, you can download the `app_fr.arb` file from the panel and put this arb file in your flutter apps `lib/l10n` folder along with your `app_en.arb` file. You can also change the main `app_en.arb` file this way but we encourage you **not to do it** . If you face any error due to `app_en.arb` file changes , we will not provide you any support.

Make sure the file you pasted in the **lib/l10n** is not empty. If you provide an empty file you will get errors.

| # | Name | Code | Flutter App Lang Code | RTL | Options |
|----|-------------|------|-----------------------|-------------------------------------|-----------------|
| 1 | English | en | en | <input type="checkbox"/> | |
| 2 | Bangla | bd | bn | <input type="checkbox"/> | |
| 3 | Arabic | sa | ar | <input checked="" type="checkbox"/> | |
| 4 | Spain | es | | <input type="checkbox"/> | |
| 5 | Japan | jp | | <input type="checkbox"/> | |
| 6 | France | fr | fr | <input type="checkbox"/> | arb File Export |
| 7 | India | in | | <input type="checkbox"/> | |
| 8 | Netherland | nl | | <input type="checkbox"/> | |
| 9 | Afghanistan | af | | <input type="checkbox"/> | |
| 10 | Egypt | eg | | <input type="checkbox"/> | |

8. For the same language , your language code for app and web can be different. This is not an issue. But you have to make sure the code for the app is in 639-1 format.

9. The language list to the app is shown from the backend api, so if you are using a lot of languages , make sure you provide translation for all of them.If you don't , by default the text from app_en.arb will be shown.

14. How to remove cache data.

To enrich user experience we have cached (Mostly for a day) a lot of api responses. If you think your app data is not changing even after your data has been changed from the backend, try clearing cache from the admin panel. There is a big red button on the top navbar in the admin panel to clear cache.