



 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 open the project on 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 run IOS on a simulator?
- 8.** How to activate an app?
- 9.** How to config the Stipe payment gateway?
- 10.** How to generate app store uploadable files?
- 11.** How to Update for Android?
- 12.** How to configure social login?
- 13.** How to configure push notification?
- 14.** How to configure google maps ?
- 15.** How to configure the default language for mobile apps?
- 16.** How to configure multiple languages for mobile apps?
- 17.** My data is not changing. How to clear cache?

1. What are the prerequisites?

Answer:

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.

- a. Flutter version must be : **Flutter 3.10.04 • channel stable**
- b. **Android Studio Flamingo | 2022.2.1 Patch 1**
<https://developer.android.com/studio/archive>
- c. **Java version OpenJDK Runtime Environment (build 17.0.6+0-b2043.56-9586694)**
- d. 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 on your PC/mac.

2. How to open the project on Android Studio?

Answer:

Extract the **sOURCE_CODE.zip**. You will find this inside the **main zip**.

- a. Open the folder in your **Android studio**.
- b. Even if you are building an app for iOS, use Android Studio for the build.
- c. 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?

Answer:

a. App Config:

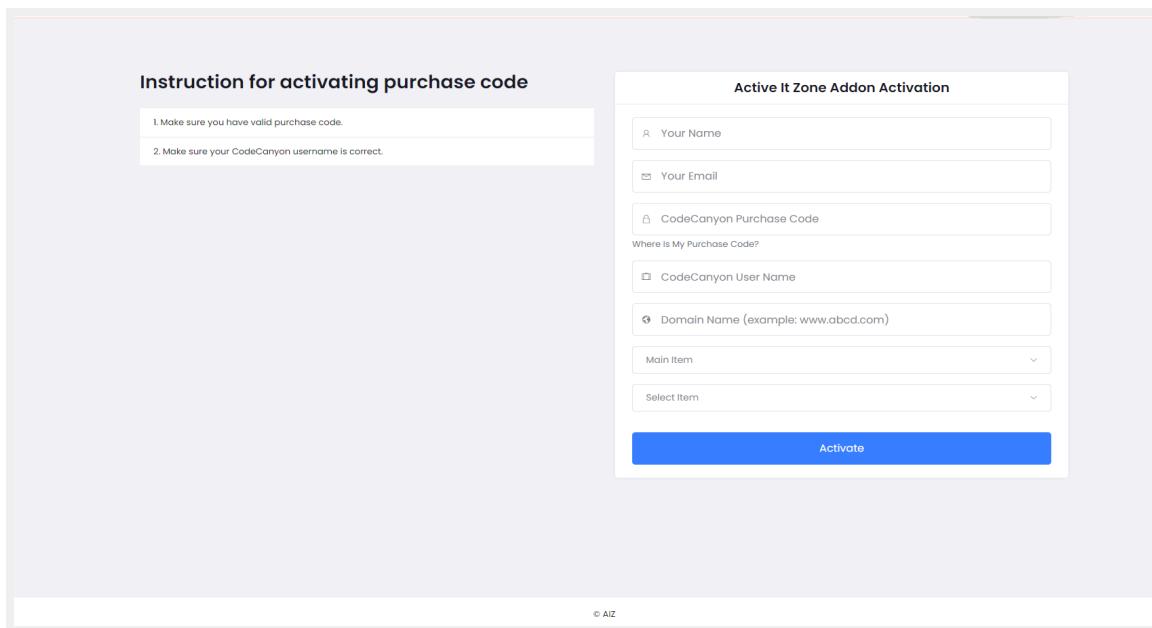
- This helps you connect your app to your server.

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 **system_key** and **purchase_code** are given. Otherwise, your app will not work properly.

b. Activating the Flutter app:

First, **activate** your Flutter app from here: <https://activeitzone.com/activation/addon>.



c. Procedure for getting the system key:

i. Please make sure that your **Active e-commerce CMS** system is **activated** from this site <https://activeitzone.com/activation> and that you have collected a **System key**. For more information please check the ecommerce CMS system **documentation**.

ii. Now you can follow the manual process to **set the system key**.

a. Use the system key in your **app_config.dart file**. Find the **app_config.dart** in your **project->lib** folder.

Here is the screenshot for a better understanding.

```

active_ecommerce_flutter D:\StudioProjects\active_ecommerce_flutter
> .dart_tool
> .idea
> android
> assets
> build
> dummy_assets
> ios
> lib
>   custom
>   data_model
>   dummy_data
>   helpers
>   l10n
>   middlewares
>   presenter
>   providers
>   repositories
>   screens
>   services
>   ui_elements
>   ui_sections
>     app_config.dart
>     lang_config.dart
>     main.dart
>     my_theme.dart
>     other_config.dart
>     social_config.dart
> test
>   flutter-plugins
>   flutter-plugins-dependencies
>   .gitattributes
>   .gitignore
>   .metadata
>   FlutterCommerceAPI.postman_collection.json
>   how_to_access_from_android_device_as_emulator.txt
>   l10n.yaml
>   pubspec.lock
>   pubspec.yaml
>   README.md
1  var this_year = DateTime.now().year.toString();
2
3  class AppConfig {
4    static String copyright_text =
5      "@ ActiveITZone " + this_year; //this shows in the splash screen
6    static String app_name = "Active eCommerce"; //this shows in the splash screen
7
8    static String purchase_code =
9      "bkash"; //enter your purchase code for the app from codecanyon
10   static String system_key =
11      "\$2\$1\$0H..."; //enter your purchase
12
13   //Default language config
14   static String default_language = "en";
15   static String mobile_app_code = "en";
16   static bool app_language_rtl = false;
17
18   //configure this
19   static const bool HTTPS = false;
20
21   static const DOMAIN_PATH = "domain.com"; //localhost
22
23   //do not configure these below
24   static const String API_ENDPOINT = "api/v2";
25   static const String PROTOCOL = HTTPS ? "https://" : "http://";
26   static const String RAW_BASE_URL = "${PROTOCOL}${DOMAIN_PATH}";
27   static const String BASE_URL = "${RAW_BASE_URL}${API_ENDPOINT}";
28
29   @override
30   String toString() {
31     // TODO: implement toString
32     return super.toString();
33   }
34
35 }

```

****Make sure that your system uses the same system key otherwise it will not work,
check your .env file****

```

APP_NAME="Active eCommerce CMS"
APP_ENV=local
APP_KEY=base64:WrvqXJ+ilWOLItqI7C2N2R2dQkqFAHFYzTrQBBJqtso=
APP_DEBUG=true
APP_URL="http://localhost/e-commerce"
APP_TTIMEZONE="UTC"
SYSTEM_KEY=""

DEMO_MODE="Off"

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST="localhost"
DB_PORT="3306"
DB_DATABASE="db_ecommerce"
DB_USERNAME="root"
DB_PASSWORD=""

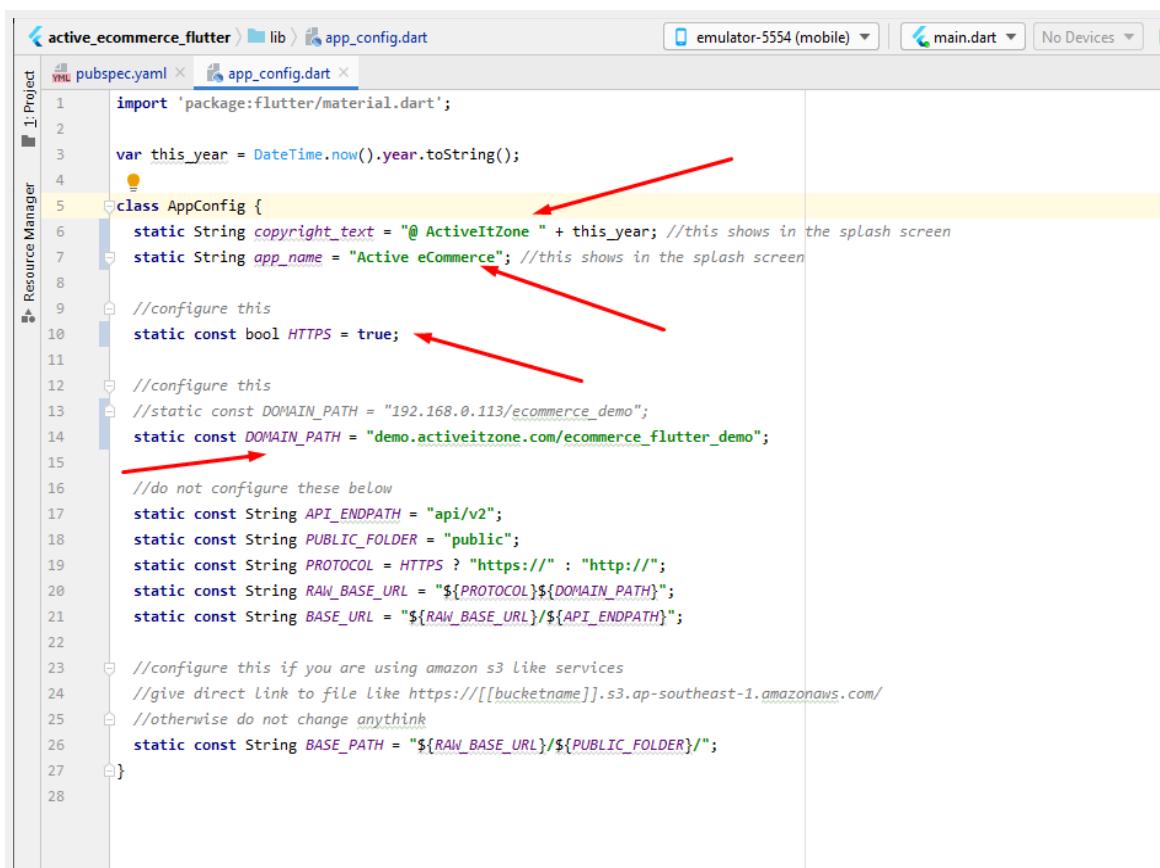
```

If your site does not have https or you 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 ****



```
active_ecommerce_flutter lib app_config.dart
emulator-5554 (mobile) main.dart No Devices
Project pubspec.yaml app_config.dart
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 = "${RAW_BASE_URL}/${API_ENDPOINT}";
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
```

d. 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
Project
1 import 'package:flutter/material.dart';
2
3 class MyTheme{ Red Green Blue Opacity
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 }

```

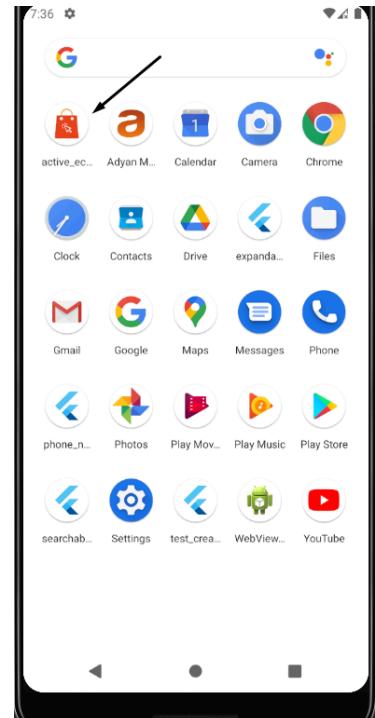
e. Configure the launcher icon:

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:

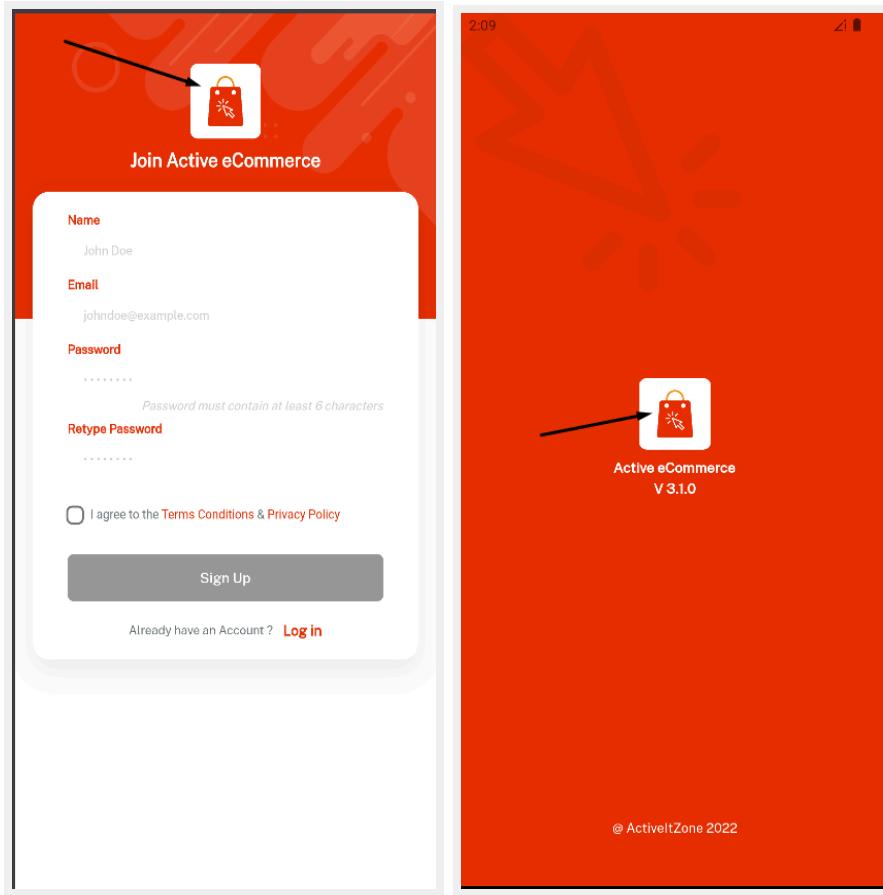
`flutter pub run flutter_launcher_icons:main`

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

f. Configure other logos:

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

branding.



This logo 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?

Answer:

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

```

{
  "project_info": {
    "project_number": "835571763337",
    "project_id": "active-ecommerce-flutter-app",
    "storage_bucket": "active-ecommerce-flutter-app.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:835571763337:android:f6a0e8a9517a55ff8f5e58",
        "android_client_info": {
          "package_name": "com.activeitzone.active_ecommerce_flutter_app"
        }
      },
      "oauth_client": [
        {
          "client_id": "835571763337-6e717rab6gc6p2vlr35lv1tmer4dekn4.apps.googleusercontent.com",
          "client_type": 1,
          "android_info": {
            "package_name": "com.activeitzone.active_ecommerce_flutter_app",
            "certificate_hash": "b13a53cff89a07171f0b6e148e24697cec03d52f"
          }
        },
        {
          "client_id": "835571763337-fct94i4ip6h7det4hb8cs322hp3pffrq.apps.googleusercontent.com",
          "client_type": 3
        }
      ]
    }
  ]
}

```

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)?

Answer:

<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?

Answer:

<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 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:

- a. Run this command in your system terminal: `flutter doctor -v`
- b. Find the Java binary at: copy this path except the "java" folder.
- c. Change your terminal path by running this command : `cd "past here your copied path"` then press enter. Example: `cd C:\Program Files\Android\Android Studio\jre\bin`
- d. 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

[✗] 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:

-certrq      Generates a certificate request
-changealias Changes an entry's alias
-delete      Deletes an entry
-exportcert  Exports certificate
-genkeypair  Generates a key pair
-genseckeyp  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:

- Open your project path and run this command: **keytool -genkey -v -keystore android/app/key.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 100000 -alias key**
- Provide all information.

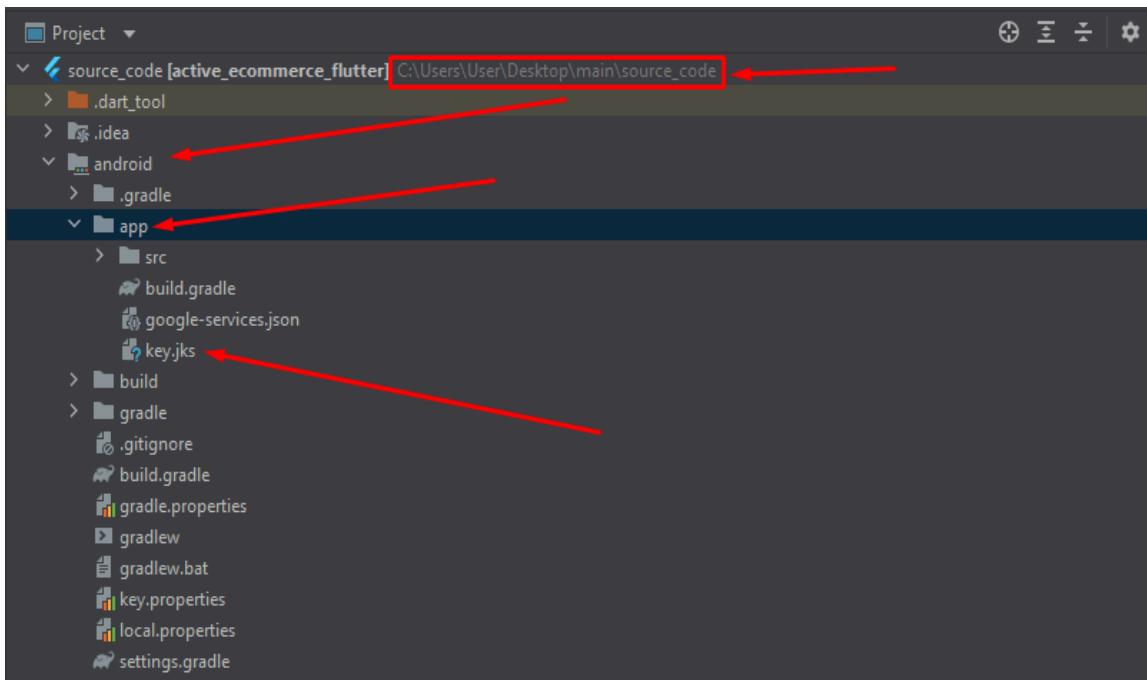
```

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:

```

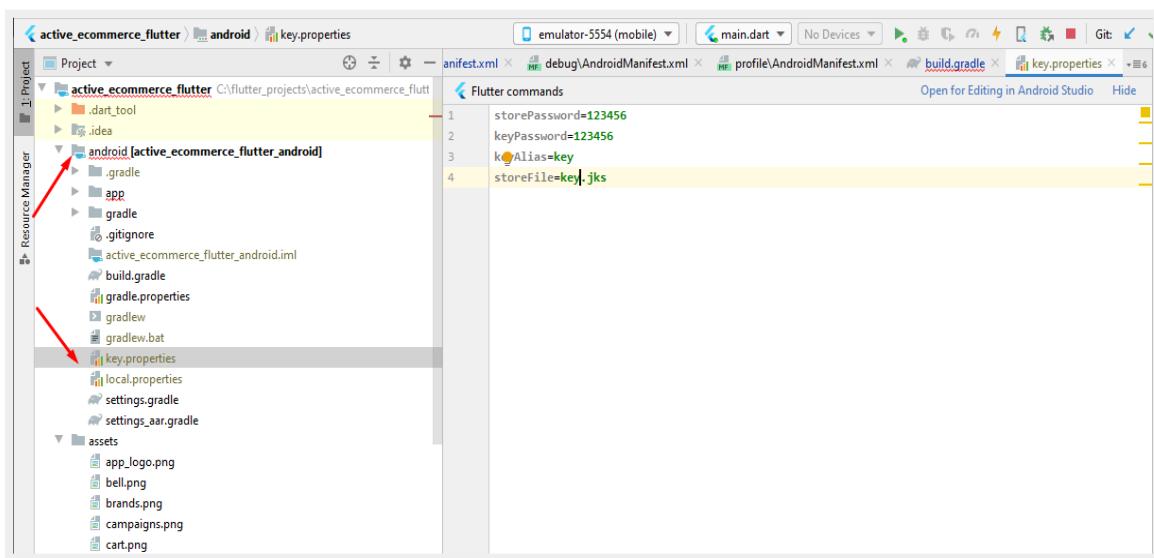
storePassword=<password from previous step>
keyPassword=<password from previous step>
keyAlias=key
storeFile=<location of the key store file, such as /Users/<user name>/key.jks>

```

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

Enter the information in the file **key.properties** in the android folder.

- a. storePassword.
- b. keyPassword.
- c. keyAlise.
- d. storeFile



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 `signingConfig` block in your module's `build.gradle` file.

Release builds of your app will now be signed automatically.

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`

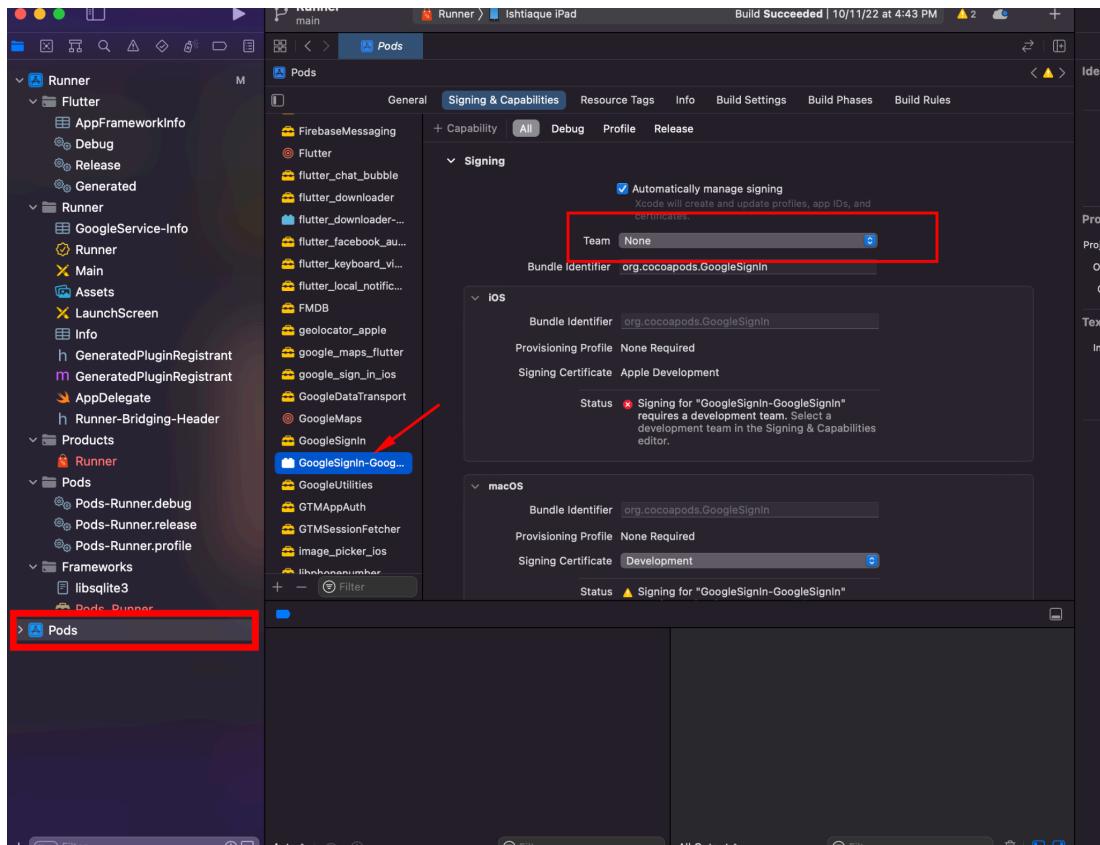
The release bundle for your app is created at <your app dir>/build/app/outputs/bundle/release/app.aab.

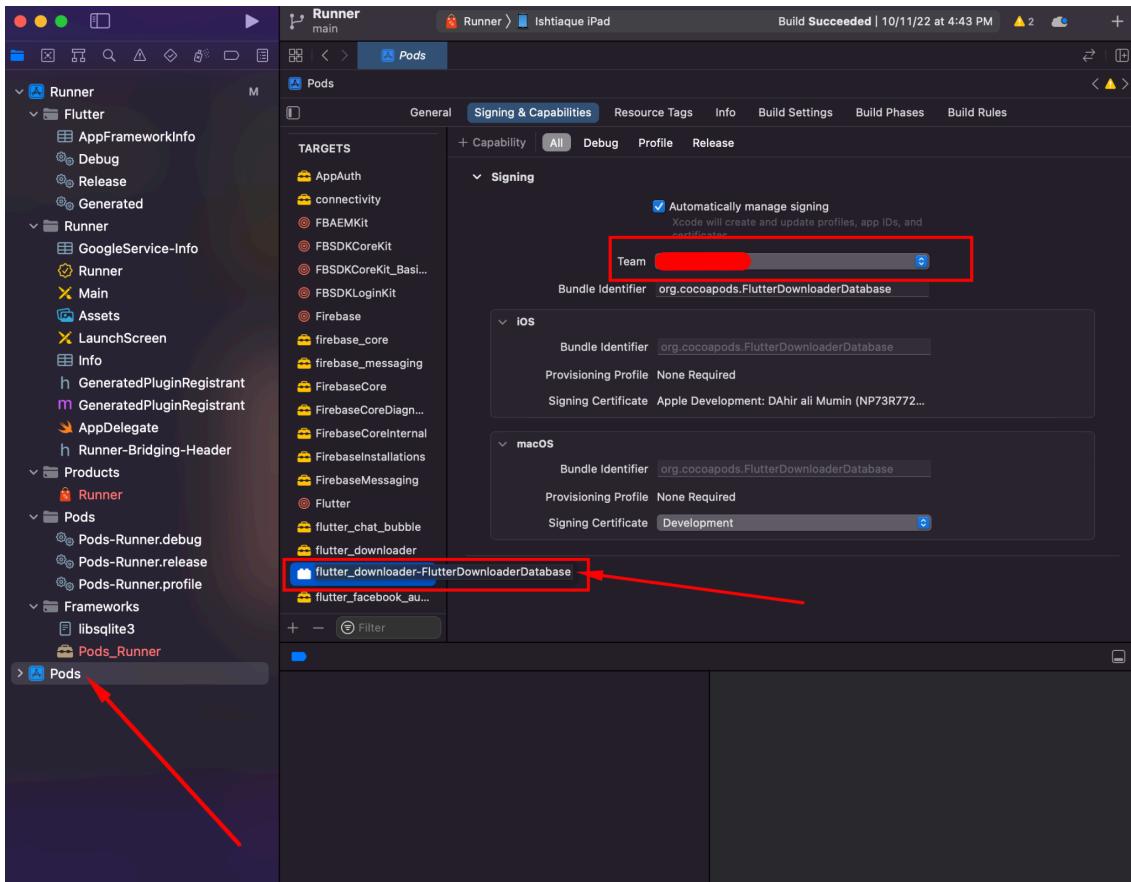
Upload this app.aab file to your google play console

7. How to run IOS? **Read all the points carefully before doing anything

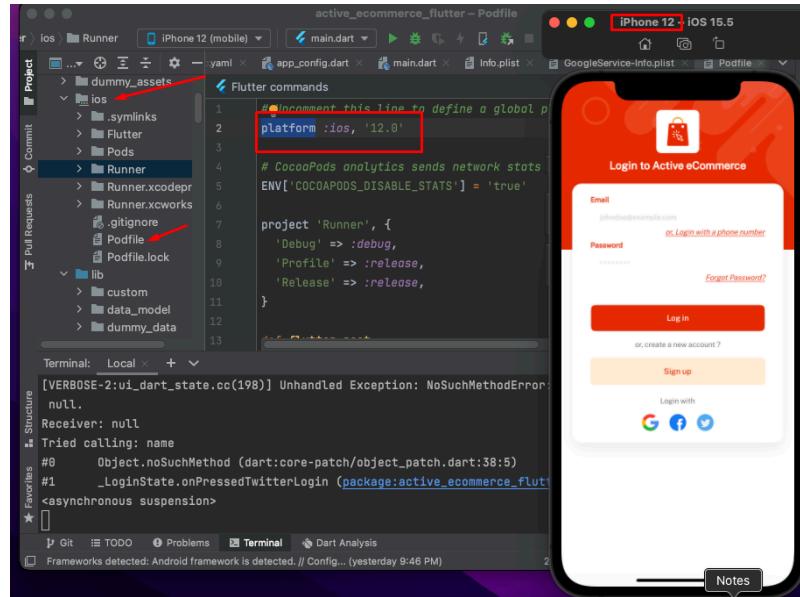
Answer:

- a. Install Xcode on your mac.
- b. Open your ios folder in Xcode and select your developer team from Signin and capabilities.
- c. Choose your team from GoogleSignin-GoogleSignin and flutter_downloader-FlutterDownloaderDatabase from Pod->GoogleSignin-GoogleSignin and Pod->flutter_downloader-FlutterDownloaderDatabase





- d. You need to open a simulator. Run this command for open a simulator `open -a simulator`
- e. Change the platform version according to your simulator version.
 - i) Goto your source_code->ios->Podfile add your ios simulator version.



- Run this command **flutter run**.

More info <https://docs.flutter.dev/get-started/install/macos>

8. How to activate an app?

Answer:

Before running the app you need to activate your Flutter app otherwise the app will show an inactivated screen.

- Go to this link <https://activeitzone.com/activation/addon>
- Put your name in the name field.
- Put your email in the email field.
- Put your purchase code in the purchase field.
- Put your codecanyon user name in the codecanyon user name field.
- Put your domain name in the domain field.
- Select **Active Ecommerce** option from main item.
- Select the **eCommerce Flutter App** option from the item.
- Click the **Activate** button After filling all the fields and activate your app.

9. How to config the Stipe payment gateway?

Answer:

Please follow the below procedure:

Just put www before your domain name

Also make sure your server is configured with www

10. How to generate app store uploadable files?

Answer:

Archive and upload your app using Xcode

- Before you can submit your app for review through App Store Connect, you need to upload the build through Xcode.
- In Xcode, select Generic iOS Device as the deployment target.
- Choose Product from the top menu and click on Archive.
- The Xcode Organizer will launch, displaying any archives you've created.
- Make sure the current build is selected and click on Upload to App Store in the right-hand panel.
- Select your credentials and click Choose.
- In the next window that appears, click on Upload in the bottom right-hand corner.

A success message will appear when the upload has been completed. Click Done.

11. How to update for Android? **Read all the points carefully before doing anything

Answer:

- a. 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.
- b. If you are installing and building the release file for the first time this section is not for you.
- c. Extract the source_code.zip. You will find this inside the main zip.
- d. Open the folder in your Android studio.
- e. **Remember to open this in a separate folder than your old project.

- f. Even if you are building an app for iOS, use Android Studio for the build.
- g. Then in your Android studio terminal run:
`flutter pub get`
- h. This will fetch all the necessary packages
- i. If you are updating, you must have built the key.jks previously
- j. Copy the key.jks , key.properties, and the manifest file from your old project and paste in the correct locations
- k. See the previous screenshots for the file locations
- l. If you are missing your old project, you have to configure key.properties, and the manifest file like described in the installation.
- m. 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.
- n. But do not create a new key.jks, you have to update your app with the existing key
- o. 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.
- p. In your terminal run: `flutter build appbundle`
- q. The release bundle for your app is created at `<your app dir>/build/app/outputs/bundle/release/app.aab`.
- r. Upload this app.aab file to your Google Play console.

12. How to configure social login?

Answer:

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 information:

- 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.

```
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:B0:37:CC:DA:0E:44:77:00:42:4C
SHA1: 82:53:85: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 with several red arrows pointing to specific parts of the code and terminal output.

Project Structure:

- Red arrow 1 points to the `android` folder.
- Red arrow 2 points to the `app` folder.
- Red arrow 3 points to the `keyks` file.
- Red arrow 4 points to the `gradle` folder.
- Red arrow 5 points to the `build.gradle` file.

Code Editor (app/config.dart):

```
//configure this
static const DOMAIN_PATH = "192.168.1.112/ecommerce"; //localhost
static const DOMAIN_PATH = "demo.activelivezone.com/ecommerce_flutter_demo"; //inside a folder
static const DOMAIN_PATH = "mydomain.com"; // directly inside the public folder

//do not configure these below
static const String API_ENDPOINT = "api/v2";
static const String PROTOCOL = HTTPS : "https://" : "http://";
static const String RAW_BASE_URL = "${PROTOCOL}${DOMAIN_PATH}";
static const String BASE_URL = "${RAW_BASE_URL}${API_ENDPOINT}";
```

Terminal:

Local +

Use "keytool -help" for all available commands

```
C:\Flutter_projects\active_ecommerce_flutter\android>keytool -list -v -keystore 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.docCommands(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: Private key entry
Certificate chain length: 1
Certificate[1]:
Owner: CN=IIS, O=Activelivezone, L=Delhi, S=Delhi, C=IN
Issuer: CN=Activelivezone, O=Activelivezone, L=Delhi, S=Delhi, C=IN
Serial number: 7dd5263u
Valid from: 1/5/2022 6:23:41 AM until: 1/5/2023 6:23:41 AM
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:05:74:45:0F:58
```

Activate Windows
Go to Settings to activate Windows.

Event Log

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

Google Play Console

Search Play Console

All apps

Dashboard

Inbox 2

Statistics

Publishing overview

Release

Releases overview

Production

Testing

Reach and devices

Overview

Device catalog

App bundle explorer

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

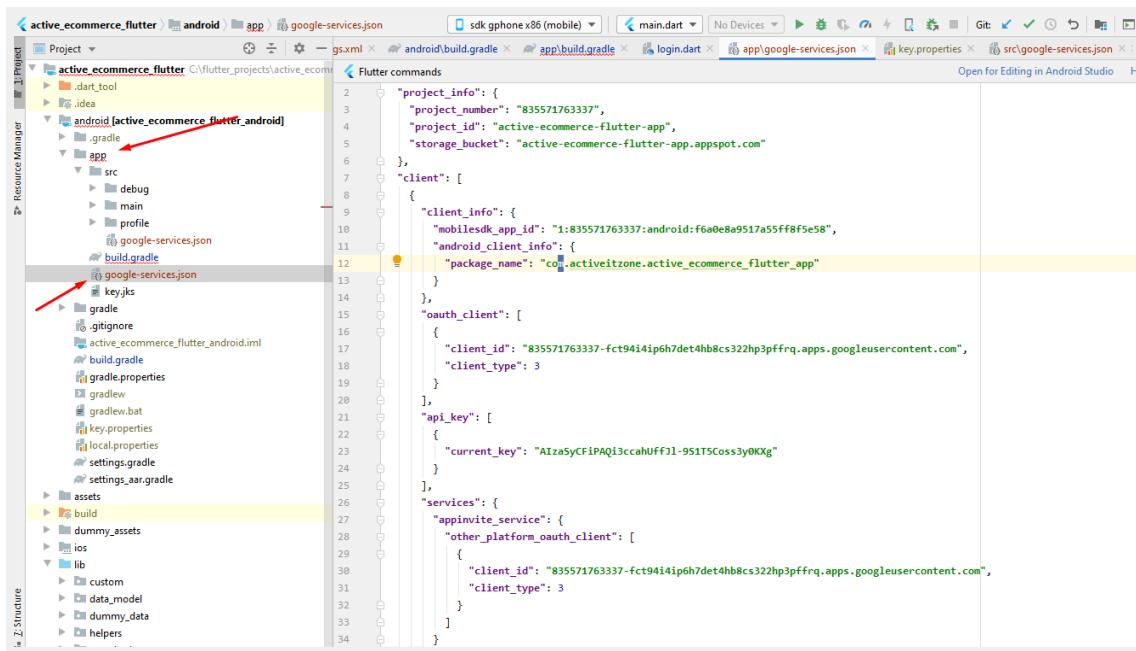
SHA-1 certificate fingerprint

SHA-256 certificate fingerprint

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.

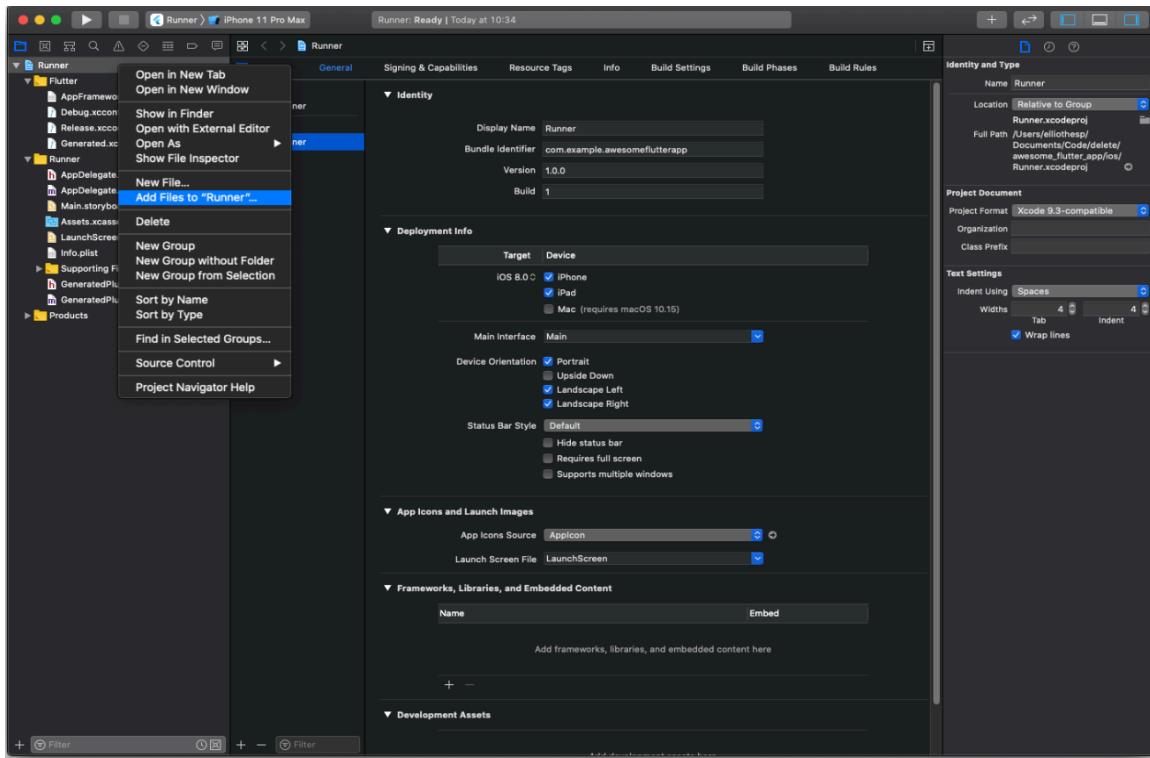
You must generate your own google-services.json. Do not use ours - it will not work for you.



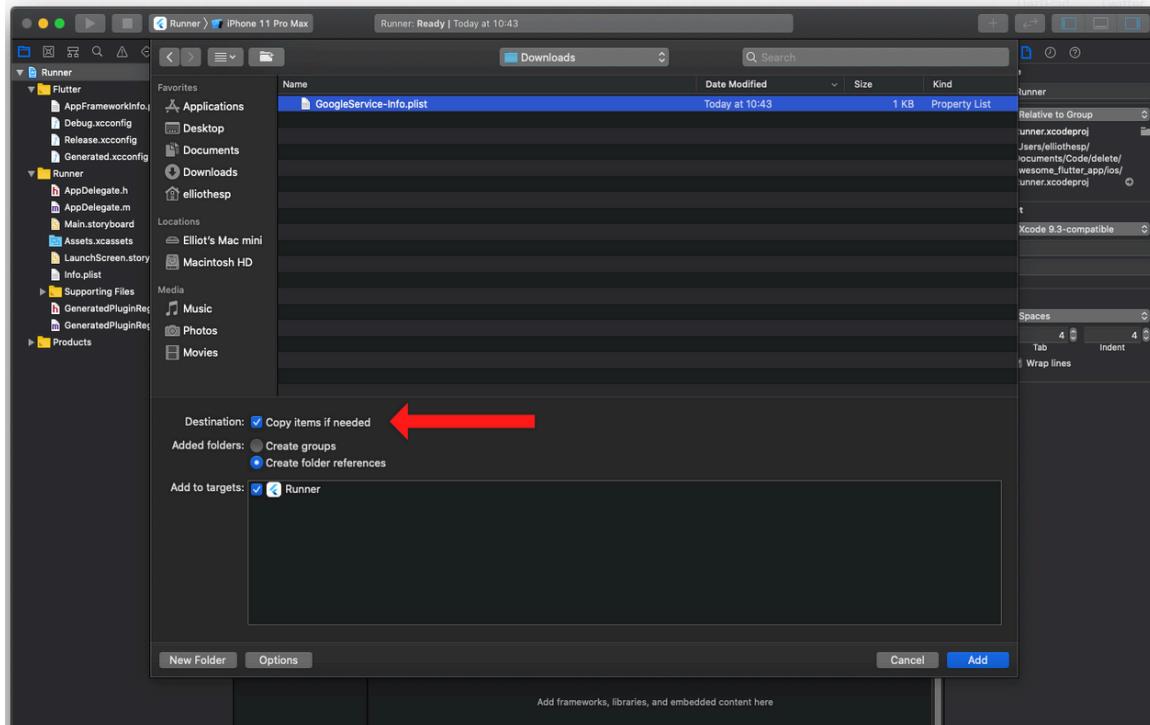
Add the iOS app on your Firebase project

1. Goto your Firebase project setting and click **add app** button.
2. Click **iOS** icon
3. Add some information.
 - i) Apple bundle ID: Insert your app bundle ID.
 - ii) App nickname (optional).
 - iii) App Store ID (optional).
4. Click the **Register app** button.
5. Download **GoogleService-Info.plist** file

Next you must add the file to the project using Xcode (adding manually via the filesystem won't link the file to the project). Using Xcode, open the project's `ios/{projectName}.xcworkspace` file. Right click `Runner` from the left-hand side project navigation within Xcode and select "Add files", as seen below:

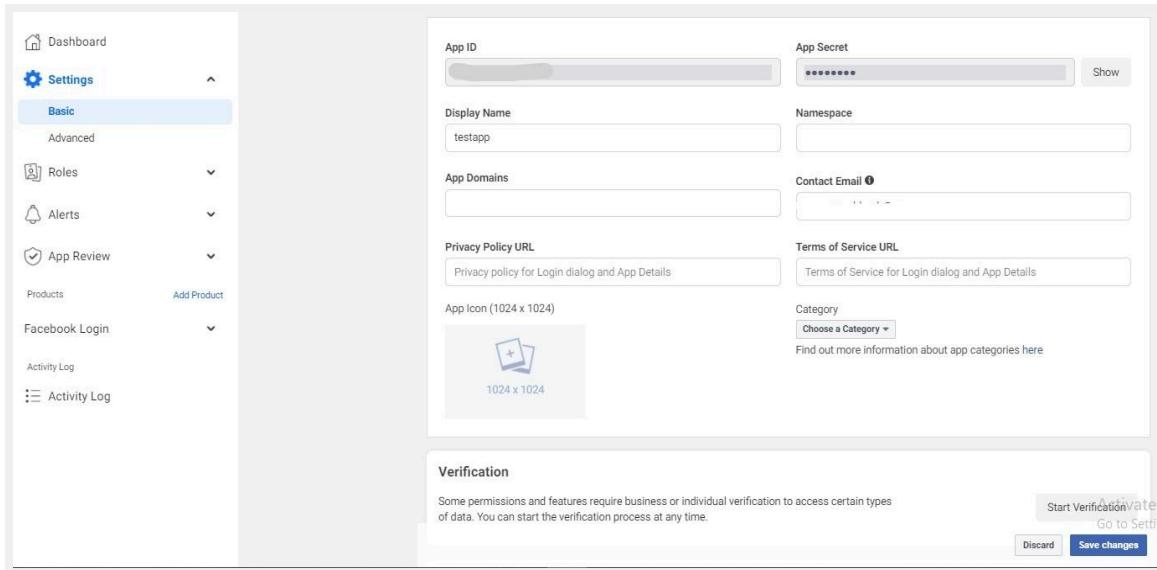


Select the GoogleService-Info.plist file you downloaded, and ensure the "Copy items if needed" checkbox is enabled:



Firebase Authentication :

1. Goto signin method and then enable Google, facebook and twitter and apple.
2. For facebook you need a facebook app id and app secret, we will create the next step.
Add these documents.



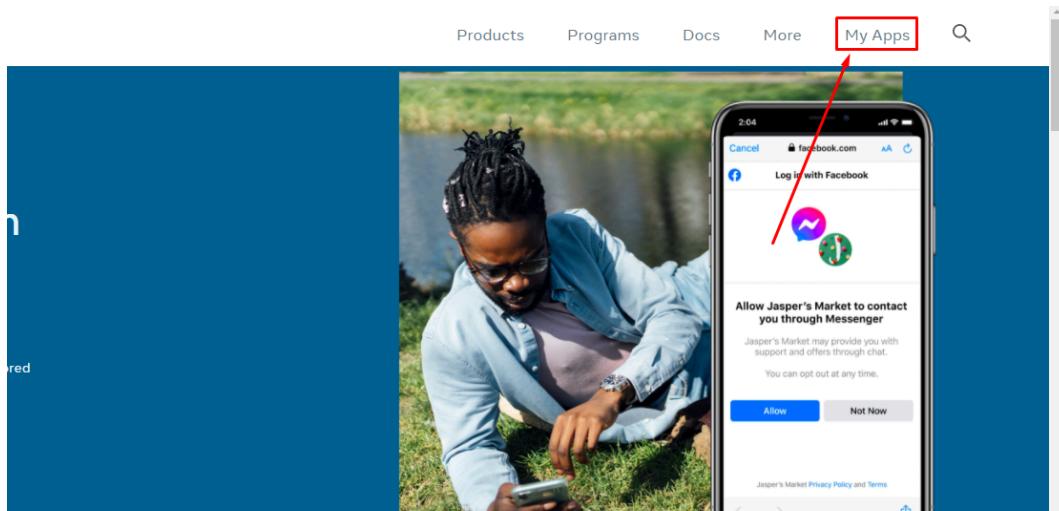
I) Facebook: Package Used

https://pub.dev/packages/flutter_facebook_auth

See its documentation and steps

How to create a Facebook app?

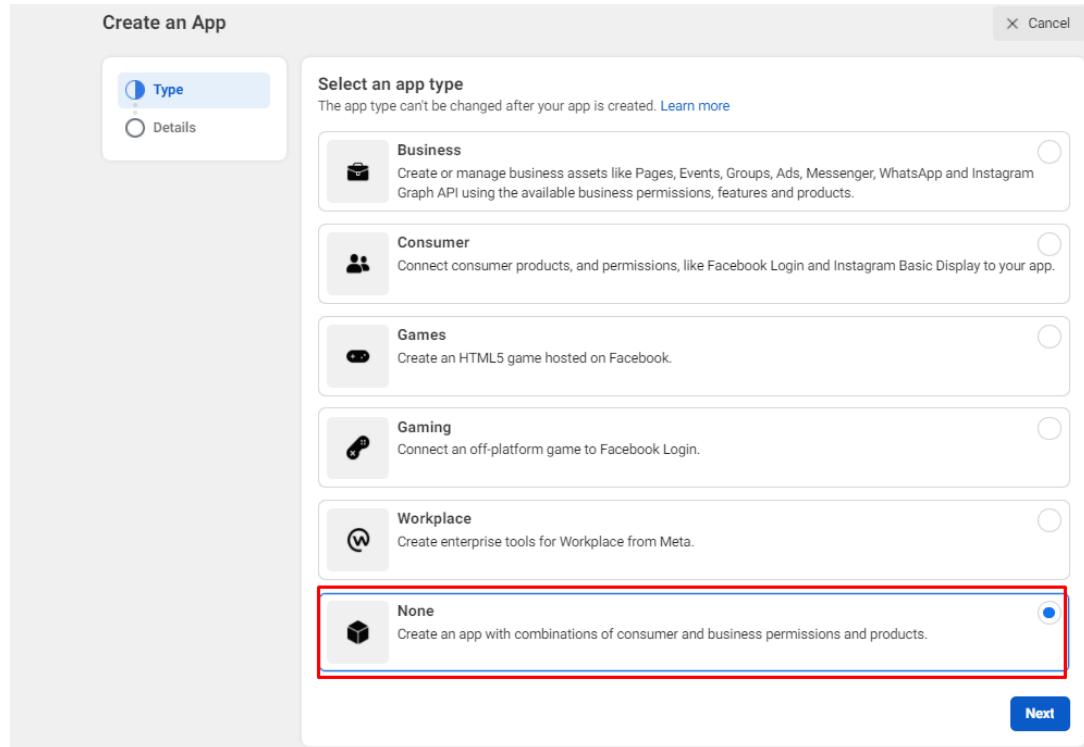
- A) First you have to log in to facebook in your browser.
- B) Goto <https://developers.facebook.com/>
- C) Goto My Apps.



D) Create an app.

A screenshot of the Meta developer portal's 'Apps' section. At the top, there is a search bar with the placeholder 'Search by App Name or App ID' and a 'Create App' button, which is highlighted with a red box and has a red arrow pointing to it from the left. Below the search bar is a 'Filter by' dropdown set to 'All Apps (2)'. The main area shows a list of apps, with one app card visible. At the bottom of the page, there is a footer with links to various Meta products like Artificial Intelligence, AR/VR, Business Tools, etc., and sections for News, Programs, Support, and Terms and Policies. There are also links to activate Windows and social media follow buttons.

Select App Type None.



E) Add your app details

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

Add details

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

App Name

App Contact 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.

Email

Business Account · Optional
To access certain permissions or features, apps need to be connected to a Business Account.

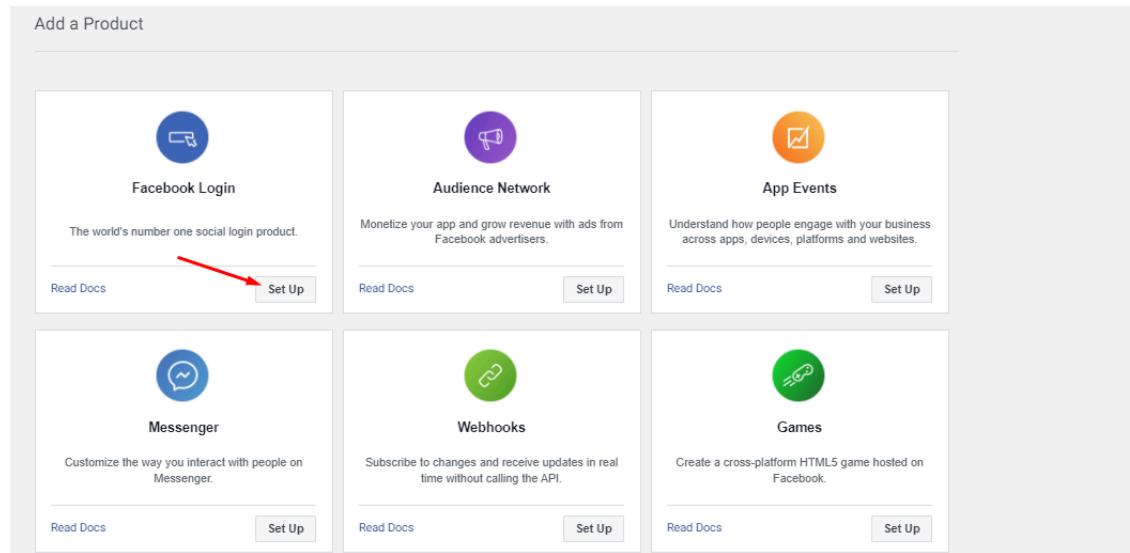
No Business Account selected

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

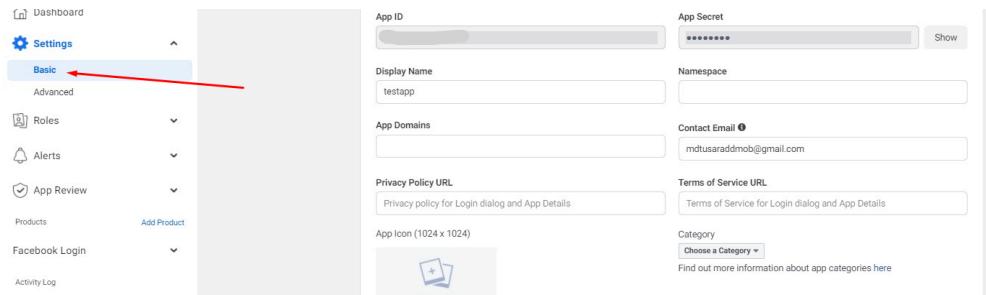
Previous **Create App** Next

F) Add product “Facebook Login”.

- ❖ Click the “Set Up” button.



D) Click the “Basic” Option.



E) 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.

Display Name: testapp

Namespace:

App Domains:

Contact Email: mdtusaraddmob@gmail.com

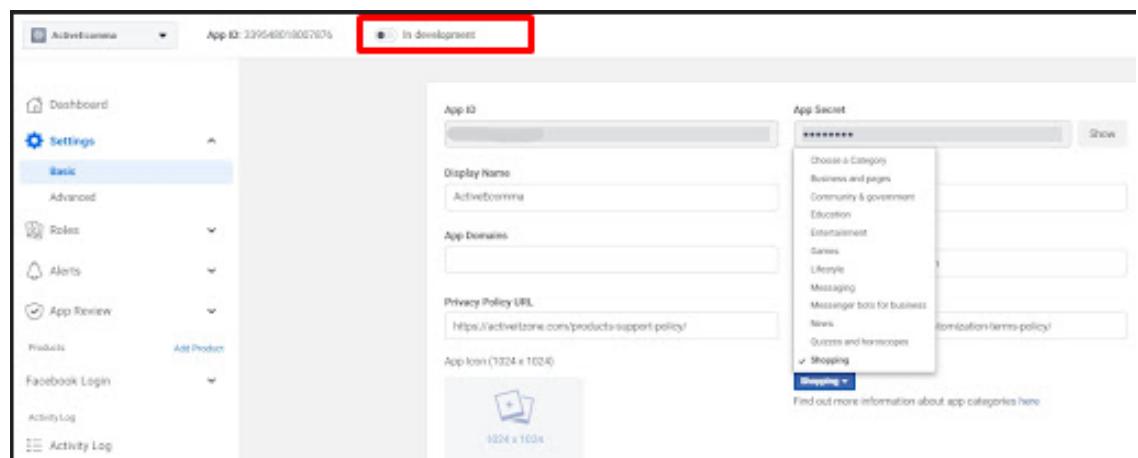
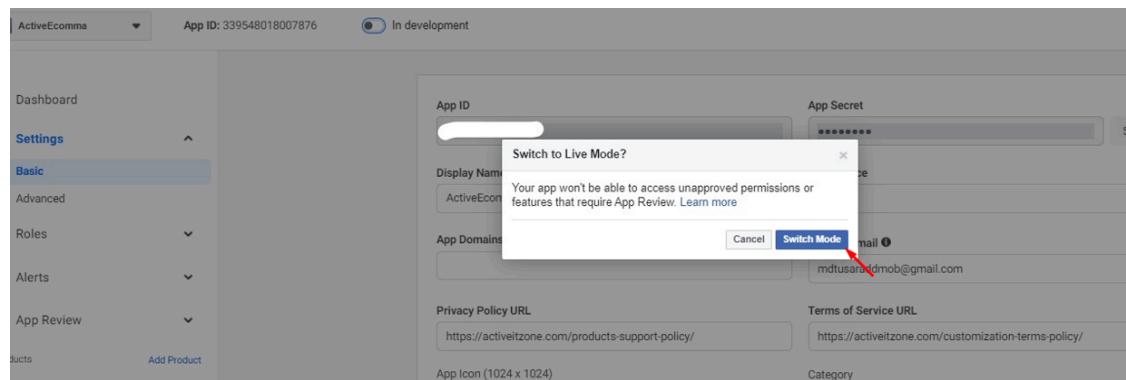
Privacy Policy URL: Privacy policy for Login dialog and App Details

Terms of Service URL: Terms of Service for Login dialog and App Details

App Icon (1024 x 1024):

Category: 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

Client OAuth Settings

- Client OAuth Login**: Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. Yes No
- Web OAuth Login**: Enables web-based Client OAuth Login. Yes No
- Enforce HTTPS**: Enforce the use of HTTPS for Redirect URIs and the JavaScript SDK. Strongly recommended. Yes No
- Force Web OAuth Reauthentication**: When on, prompts people to enter their Facebook password in order to log in on the web. No
- Embedded Browser OAuth Login**: Enable webview Redirect URIs for Client OAuth Login. No
- Use Strict Mode for Redirect URIs**: Only allow redirects that exactly match the Valid OAuth Redirect URIs. Strongly recommended. Yes No
- Valid OAuth redirect URLs**: A manually specified redirect_uri used with Login on the web must exactly match one of the URLs listed here. This list is also used by the JavaScript SDK for in-app browsers that suppress popups. No
- Login from Devices**: Enables the OAuth client login flow for devices like a smart TV. Yes No
- Login with the JavaScript SDK**: Enables Login and signed-in functionality with the JavaScript SDK. No Yes
- Allowed Domains for the JavaScript SDK**: Login and signed-in functionality of the JavaScript SDK will only be available on these domains. No

Activate Go to Sett

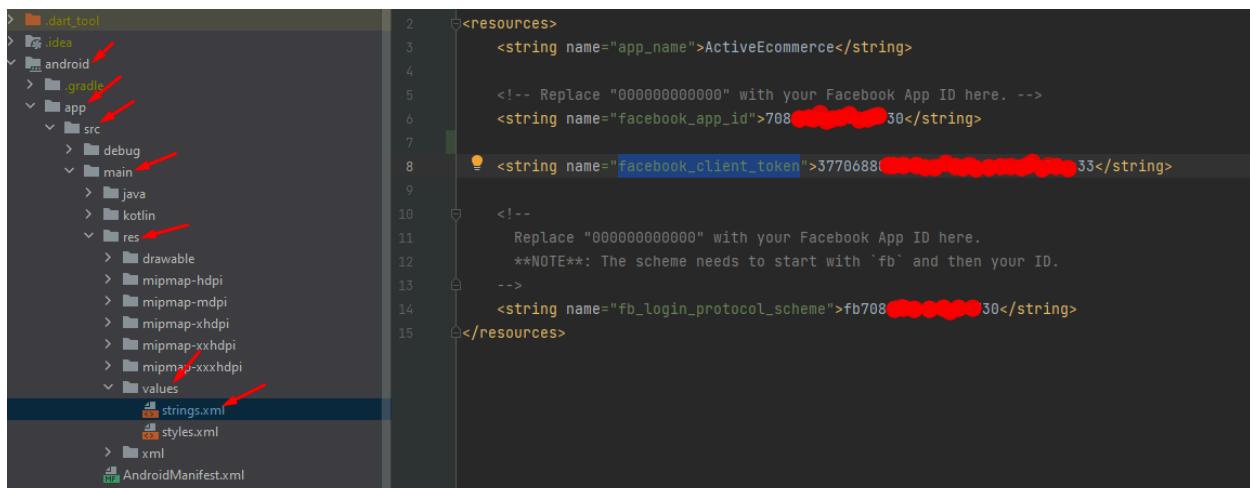
13.

How to configure social login?

Integration

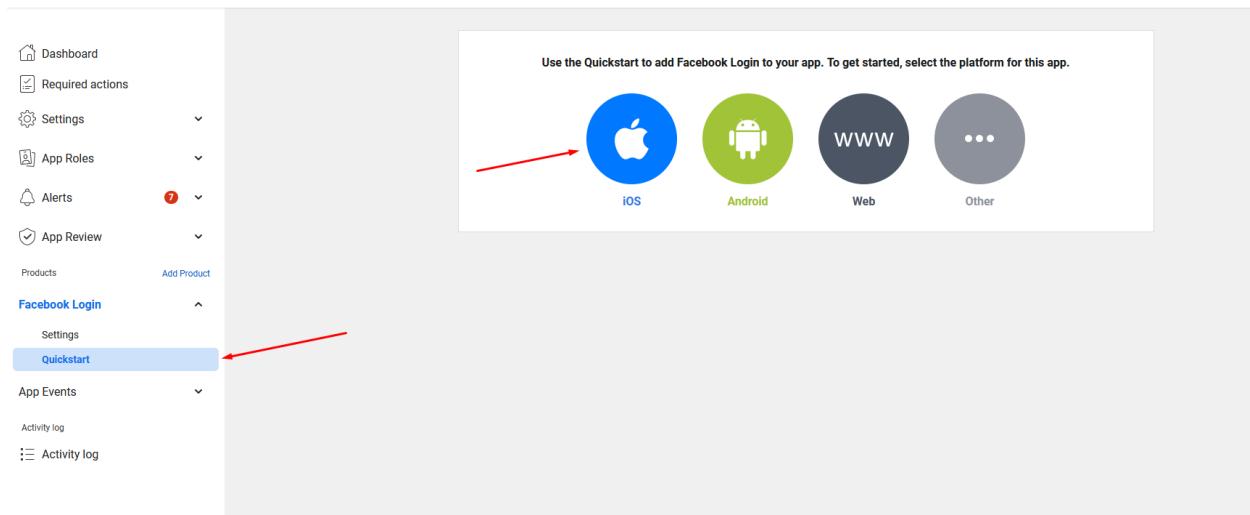
For Android:

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.



For iOS:

Goto Facebook Login->Quickstart then select iOS platform and just follow 1 and 2 steps.



Then choose your environment SDK: Cocoapods

1. Set up Your Development Environment

To make your app compatible with iOS 11, be sure to use the latest Facebook SDKs for iOS.
If you link to the SDKs with CocoaPods, you must update your pods for the SDKs your app uses and recompile your app. You can also download the latest version of the Facebook iOS SDK, integrate it into your app, and recompile.

Set up your development environment before using Facebook Login for iOS.

SDK: Cocoapods ▾

Then add your Bundle ID then save it and exit from the quick start page.

1. Set up Your Development Environment

2. Add your Bundle Identifier

The bundle identifier (Bundle ID) should appear in the box below. If the box is empty, find your bundle identifier in your Xcode Project's iOS Application Target and paste it into the box below.

Bundle ID

You can change your bundle identifier in the future via the iOS section on the settings page.

com.facebook.samples.applaunch

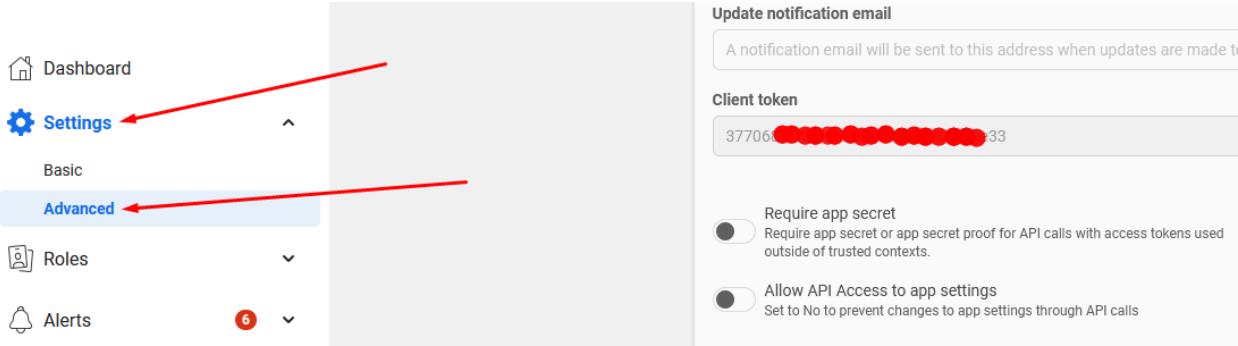
Save

Back Continue

Do not need to follow the others step in the quick start page.

Client Token: Where do you get the client token?

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



In your project's ios folder: 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.

```

<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>fbxxxxxxxx</string>
    </array>
  </dict>
</array>
<key>FacebookAppID</key>
<string>xxxxxxxx</string> App Id
<key>FacebookClientToken</key>
<string>xxxxxxxxxxxxxxxx</string> Client Token
<key>FacebookDisplayName</key>
<string>Your App Name</string> App Name

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

```

Google: Package Used

https://pub.dev/packages/google_sign_in

Integration

For IOS:

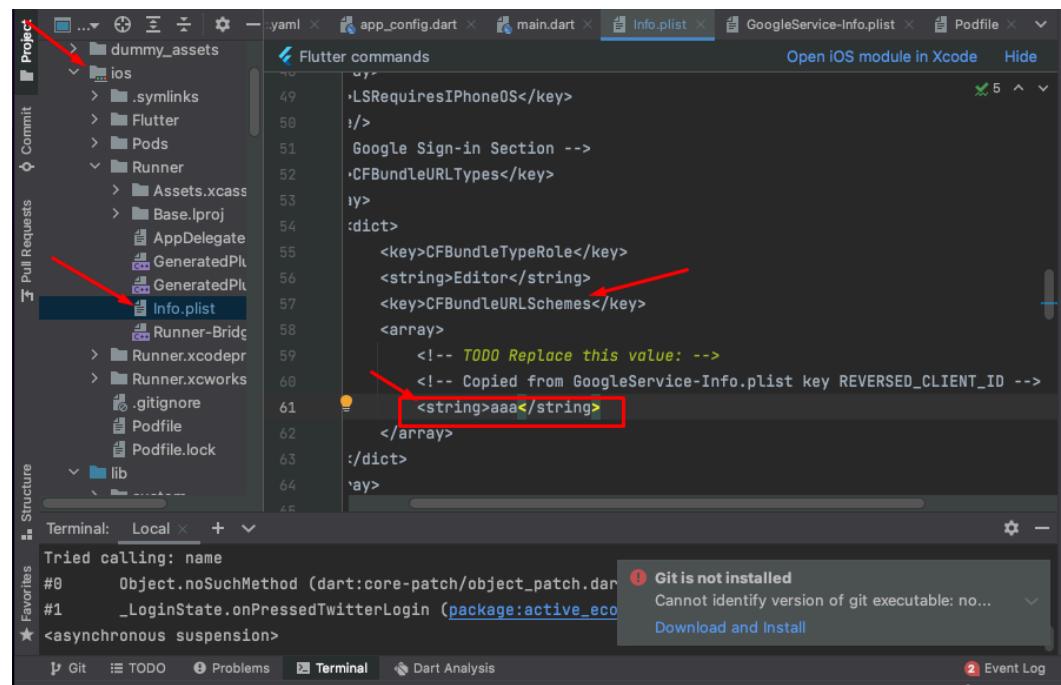
1. [First register your application.](#)
2. Make sure the file you download in step 1 is named GoogleService-Info.plist.
3. Move or copy GoogleService-Info.plist into the [my_project]/ios/Runner directory.
4. Open Xcode, then right-click on Runner directory and select Add Files to "Runner".
5. Select GoogleService-Info.plist from the file manager.
6. A dialog will show up and ask you to select the targets, select the Runner target.
7. Then add the CFBundleURLTypes attributes below into the [my_project]/ios/Runner/Info.plist file.

Open your GoogleService-Info.plist file and copy **REVERSED_CLIENT_ID**

```
yaml × app_config.dart × main.dart × Info.plist × GoogleService-Info.plist × Podfile × ✓ 12 ▲ ▼
5 <dict>
6   <key>CLIENT_ID</key>
7   <string>620463100599-ddb91en0ain1g4lim7ub25ced6lm6flr.apps.googleusercontent.com</string>
8   <key>REVERSED_CLIENT_ID</key>
9   <string>com.googleusercontent.apps.620463100599-[REDACTED]</string>
10  <key>ANDROID_CLIENT_ID</key>
11  <string>620463100599-6ugm9kpt4scqtr7hl4j35c0loe3mga6c.apps.googleusercontent.com</string>
12  <key>API_KEY</key>
13  <string>AIzaSyBbegHlQdo_R_S59Z2sLzGyJ73Tp-dbbNk</string>
14  <key>GCM_SENDER_ID</key>
15  <string>620463100599</string>
16  <key>PLIST_VERSION</key>
17  <string>1</string>
18  <key>BUNDLE_ID</key>
19  <string>com.activeitzone.activeecommerceflutterapp</string>
20  <key>PROJECT_ID</key>
21  <string>testproject-43</string>
```

Put **REVERSED_CLIENT_ID** in your source_code->ios->info.plist files

CFBundleURLSchemes-> array-> string



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.

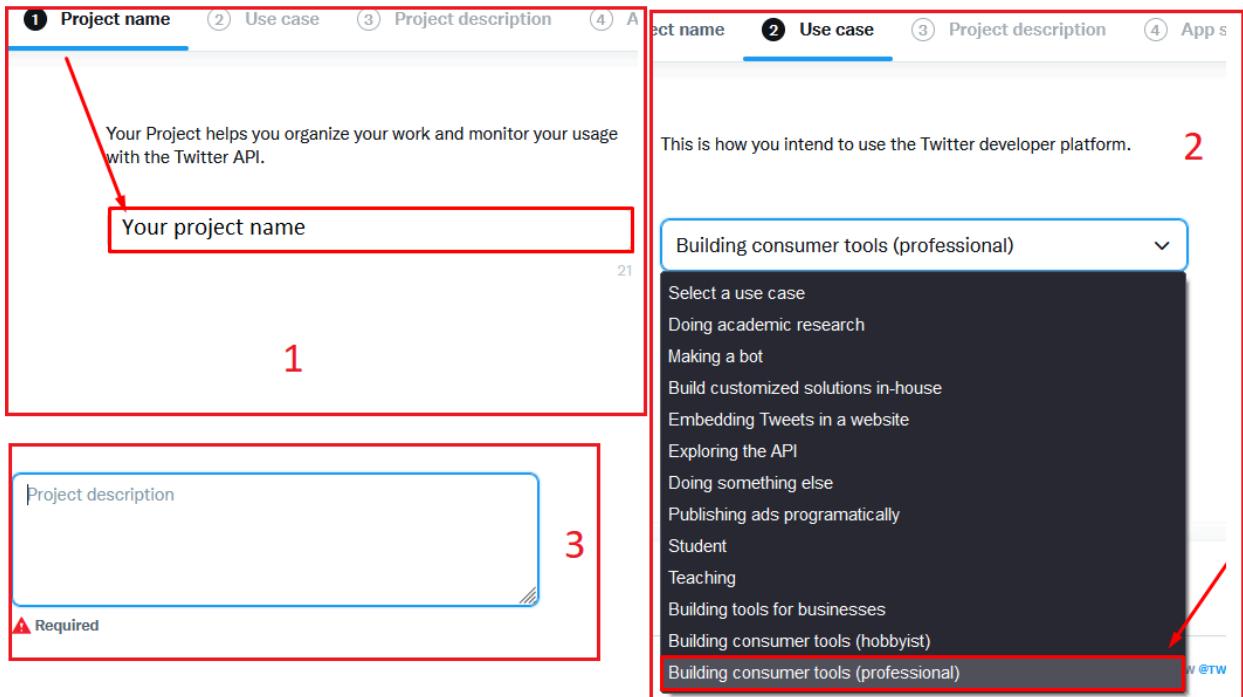
- 1.1. First you have to login twitter in your browser.
- 1.2. Go to <https://developer.twitter.com/>
- 1.3. Go to [Developer Portal](#).
- 1.4. Go to Project & Apps->Overview create a new project.

The screenshot shows the Twitter Developer Portal's Overview page. On the left, there is a dark sidebar with the following menu items:

- Dashboard
- Projects & Apps** (highlighted with a red box)
- Overview (highlighted with a red box)
- Products (NEW)
- Account

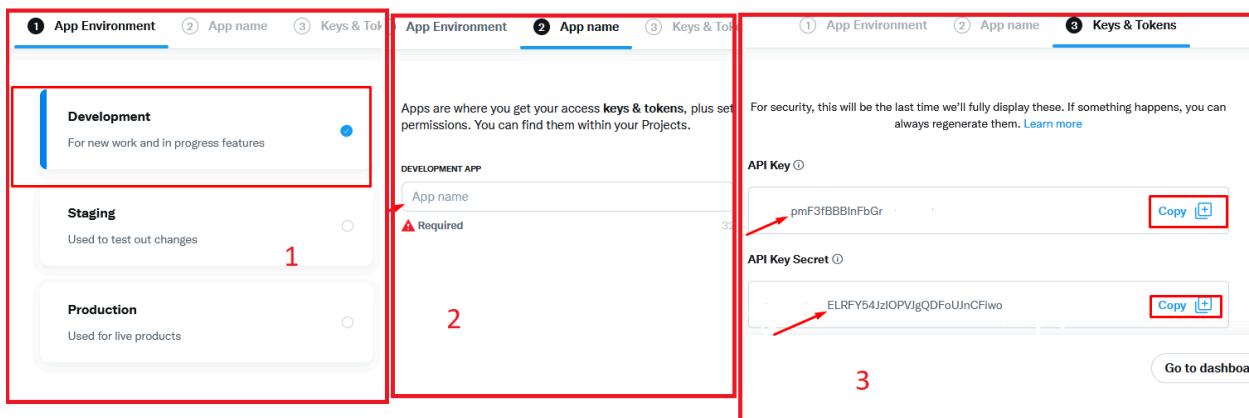
The main content area has a header "Overview" and a sub-section titled "Elevated". It displays a message "NO PROJECTS HERE" and a prominent red-bordered button labeled "+ New Project". Below this, there is another section titled "Standalone Apps" with a "V1.1 ACCESS" badge.

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



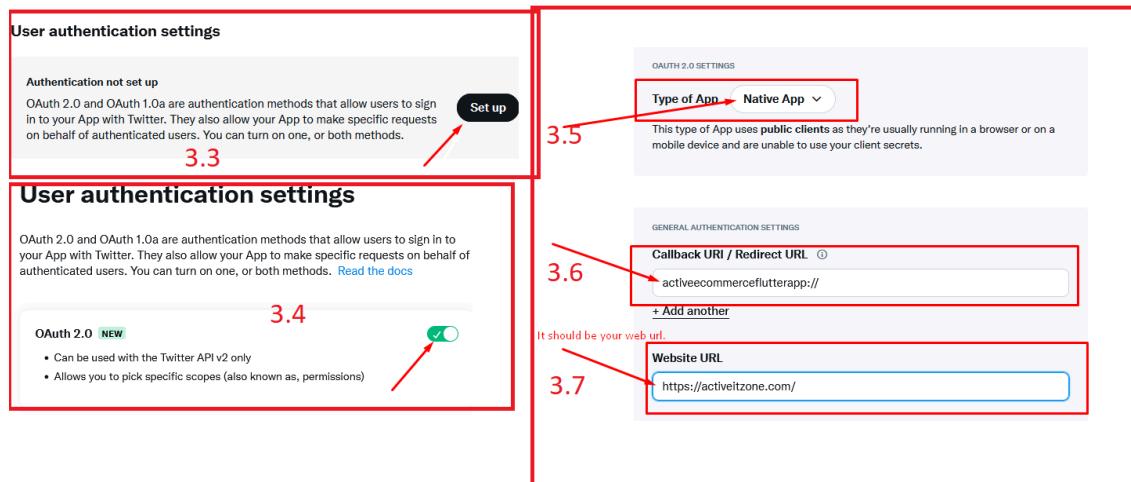
2. Setup your app

- 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.)



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.



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  android.build.gradle  app.build.gradle  login.dart  AndroidManifest.xml  my_theme.dart  social_config.dart  google_sign_in
1: class SocialConfig{
2:   var twitter_consumer_secret = "<your consumer key>";
3:   var twitter_consumer_key = "<your consumer secret>";}
4:
5:

```

Apple SignIn: package used

https://pub.dev/packages/sign_in_with_apple

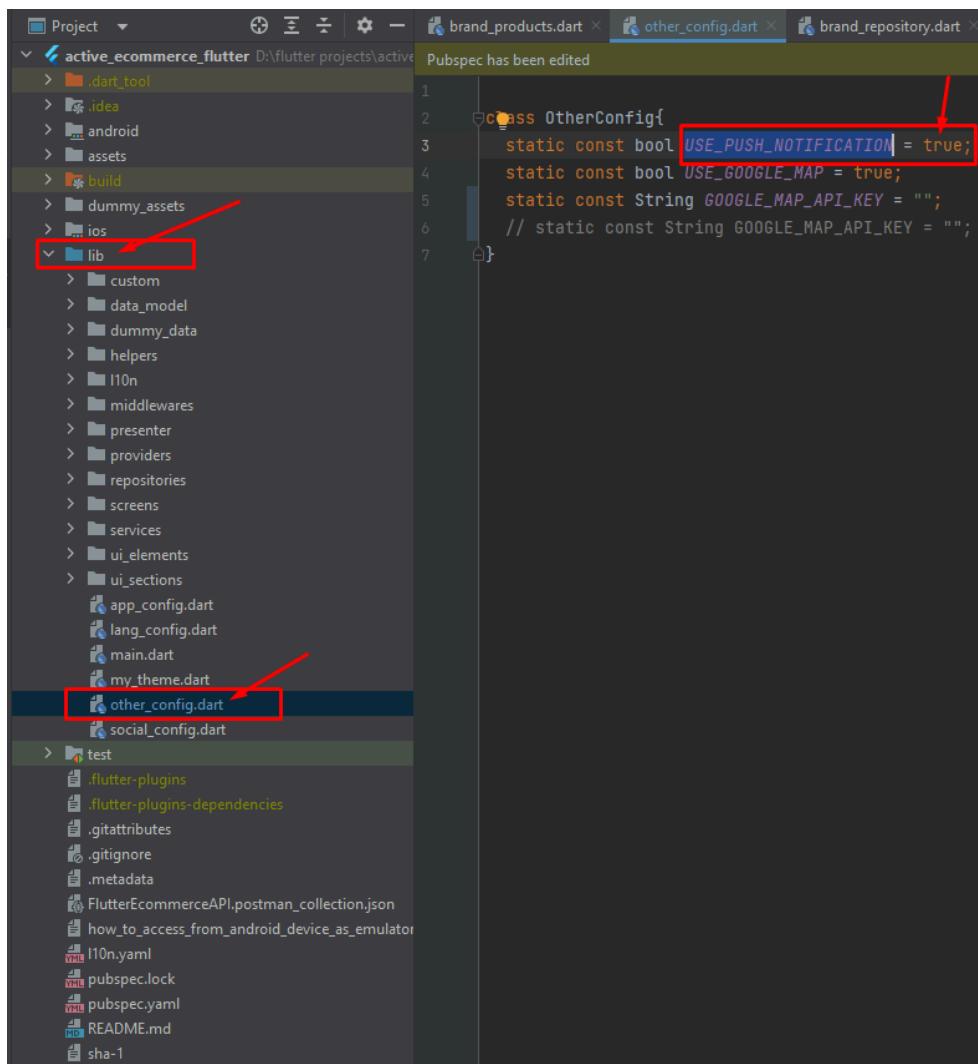
Configure properly as we mention in our ecommerce cms documentation. You don't need extra configuration for the app.

Note: Use the same Bundle ID name for IOS bundle id as you created an apple developer account for web apple signUp.

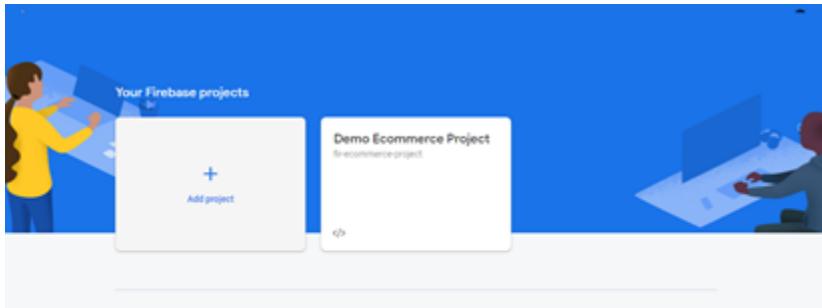
12. How to configure push notification?

To use firebase follow the procedure which are mentioned below

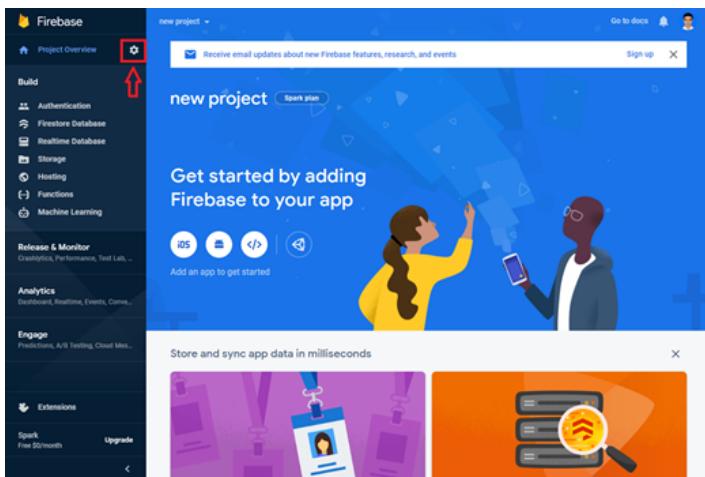
1. Open other_config.dart file from your project_file->lib->other_config.dart file and put the true in variable USE_PUSH_NOTIFICATION = true;



2. 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

The screenshot shows the Firebase Project settings interface. The left sidebar includes sections for Authentication, Firestore Database, Realtime Database, Storage, Hosting, Functions, Machine Learning, Release & Monitor, and Analytics. The main area is titled 'Project settings' with tabs for General, Cloud Messaging, Integrations, Service accounts, Data privacy, Users and permissions, and App Check (HTTP). The 'Cloud Messaging' tab is active. Under 'Project credentials', there is a table with columns 'Key' and 'Token'. The 'Server key' row has a red box around its input field. Below it is a 'Sender ID' field with a red box around its input field. A blue button labeled 'Add server key' is visible.

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

The screenshot shows the Admin Panel interface with a URL of 'localhost/ecommerce_demo_three/admin/google-firebase'. The left sidebar lists various configuration sections like Payment Methods, Social media Logins, Google, and Google Firebase. The 'Google Firebase' section is expanded, indicated by a red arrow. The main content area is titled 'Google Firebase Setting' and contains a 'Google Firebase' toggle switch (which is turned on, indicated by a green dot) and a 'FCM SERVER KEY' input field. A red box highlights the 'FCM SERVER KEY' field, and a green checkmark is placed above it. A blue 'Save' button is at the bottom right. Red arrows point from the sidebar's 'Google Firebase' link to the main content area's 'Google Firebase' setting.

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

The screenshot shows the Android Studio interface with the project 'active_ecommerce_flutter' open. The left sidebar displays the project structure, including the 'app' module which contains the 'google-services.json' file. A red arrow points from the 'app' folder in the structure to the 'google-services.json' file in the code editor. The code editor shows the JSON configuration for the app's Firebase setup, including the project info, client info, OAuth client, API key, and services sections.

```

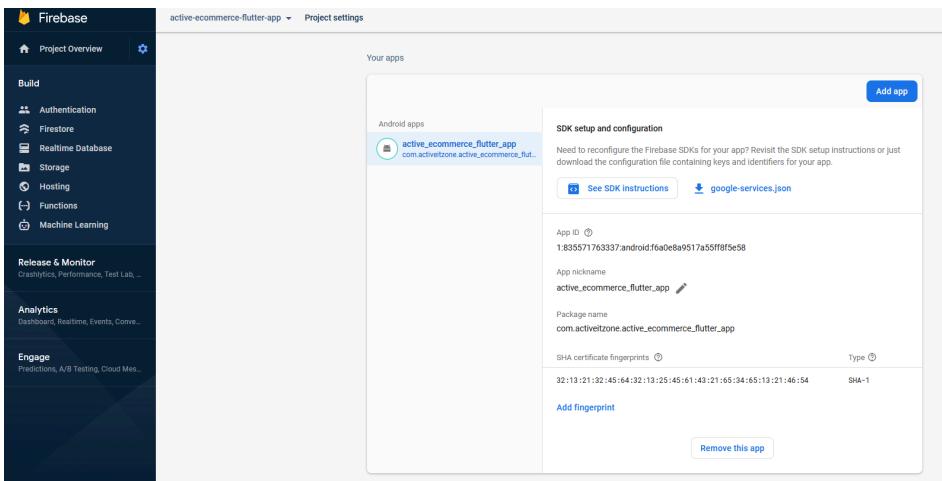
{
  "project_info": {
    "project_number": "835571763337",
    "project_id": "active-ecommerce-flutter-app",
    "storage_bucket": "active-ecommerce-flutter-app.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:835571763337:android:f6a0e8a9517a55ff8f5e58",
        "android_client_info": {
          "package_name": "com.activeitzone.active_ecommerce_flutter_app"
        }
      }
    }
  ],
  "oauth_client": [
    {
      "client_id": "835571763337-fct9414ip6h7det4hb8cs322hp3pffrq.apps.googleusercontent.com",
      "client_type": 3
    }
  ],
  "api_key": [
    {
      "current_key": "AIzaSyCfPAQj3ccahuFFJ1-9S1T5Coss3yOKXg"
    }
  ],
  "services": {
    "appinvite_service": {
      "other_platform_oauth_client": [
        {
          "client_id": "835571763337-fct9414ip6h7det4hb8cs322hp3pffrq.apps.googleusercontent.com",
          "client_type": 3
        }
      ]
    }
  },
  "configuration_version": "1"
}

```

Firebase console:

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

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



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, ST=Test, C=us
Issuer: CN=Test, OU=Test, O=Test, L=Test, ST=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:D0:B8
    SHA1: B1:3A:53:C:F8:9A:07:17:1F:9B:6E:14:8E:24:69:7C:EC:83:D5:2F
    SHA256: 92:69:F0:BF:56:F9:14: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.

Apple Integration

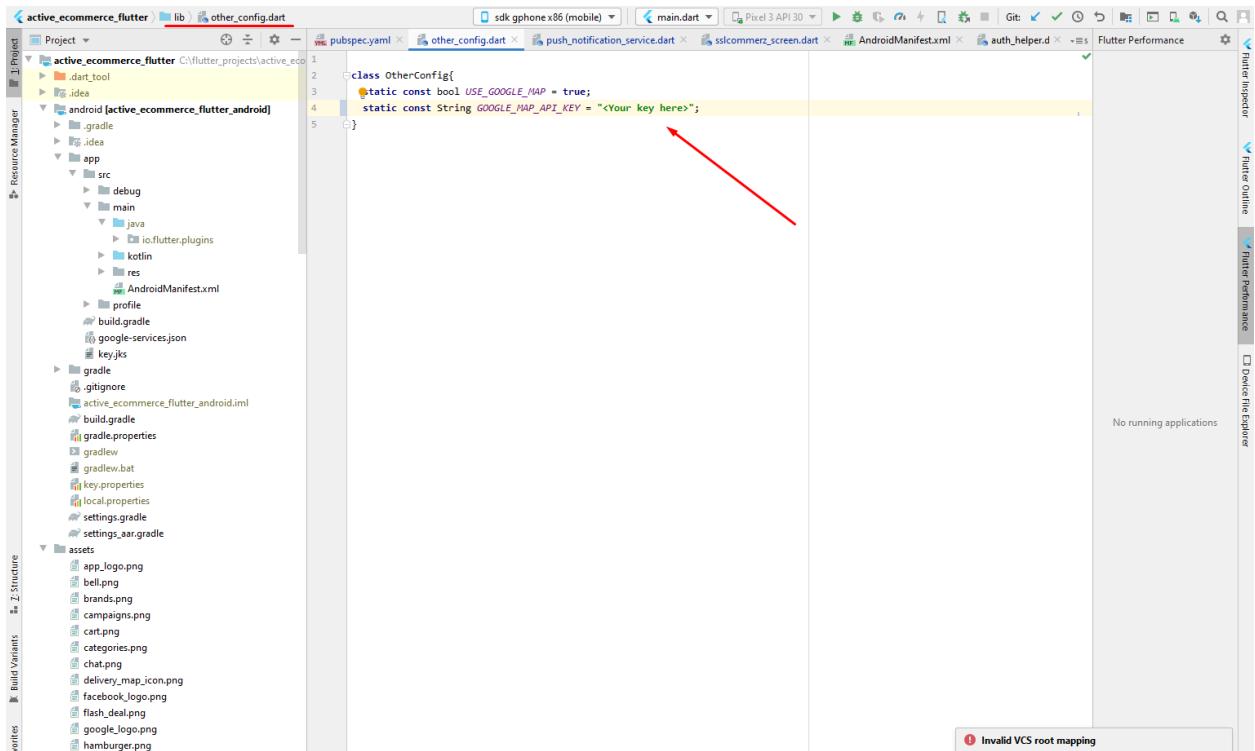
Please Follow this URL's documentation

<https://firebase.flutter.dev/docs/messaging/apple-integration/>

13. 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

."/>

```

<activity>
    <!-- Don't delete the meta-data below.
        This is used by the Flutter tool to generate GeneratedPluginRegistrant.java -->
    <meta-data
        android:name="FlutterEmbedding"
        android:value="2" />

    <meta-data android:name="com.google.android.geo.API_KEY"
        android:value="Your map API key" />

    <!-- Facebook Login configuration -->
    <meta-data android:name="com.facebook.sdk.ApplicationId"
        android:value="@string/facebook_app_id"/>

    <activity android:name=".com.facebook.FacebookActivity"
        android:configChanges=
            "keyboard|keyboardHidden|screenLayout|screenSize|orientation"
        android:label="@string/app_name"
        android:exported="true"
        />
    <activity
        android:name=".com.facebook.CustomTabActivity"
        android:exported="true"
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />
            <category android:name="android.intent.category.BROWSABLE" />
        </intent-filter>
    </activity>
</application>

```

No running applications

manifest > application > meta-data

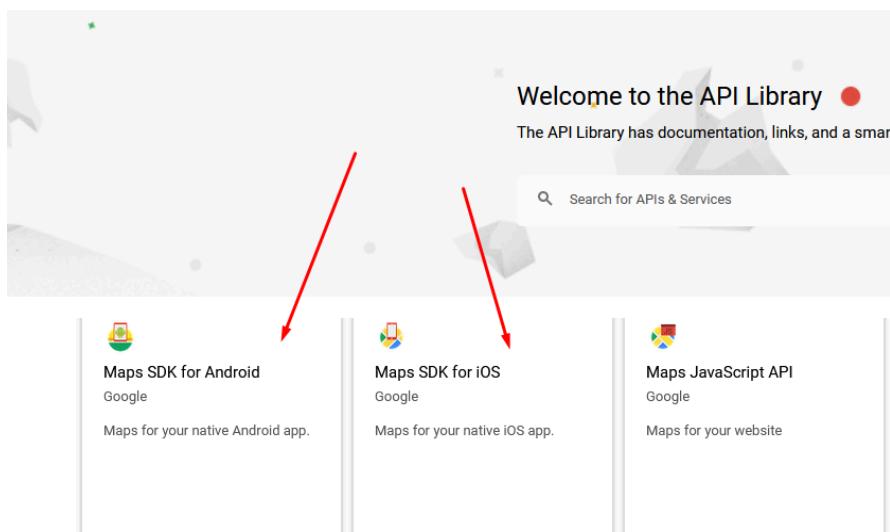
Invalid VCS root mapping
The directory <Proj_active_ecommerce_flutter is...
Configure...

3. For ios follow this

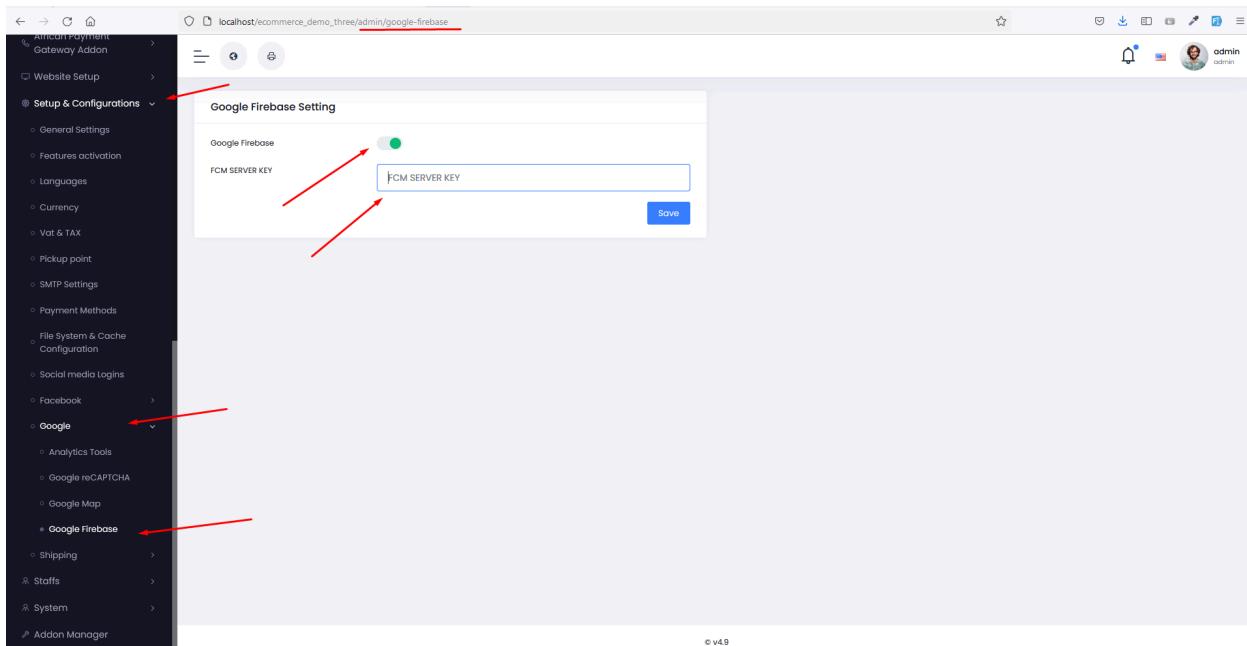
<https://blog.logrocket.com/adding-google-maps-to-a-flutter-app/#addinggooglemapstoflutterios>

S

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



Machine learning



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

14. 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;
}

```

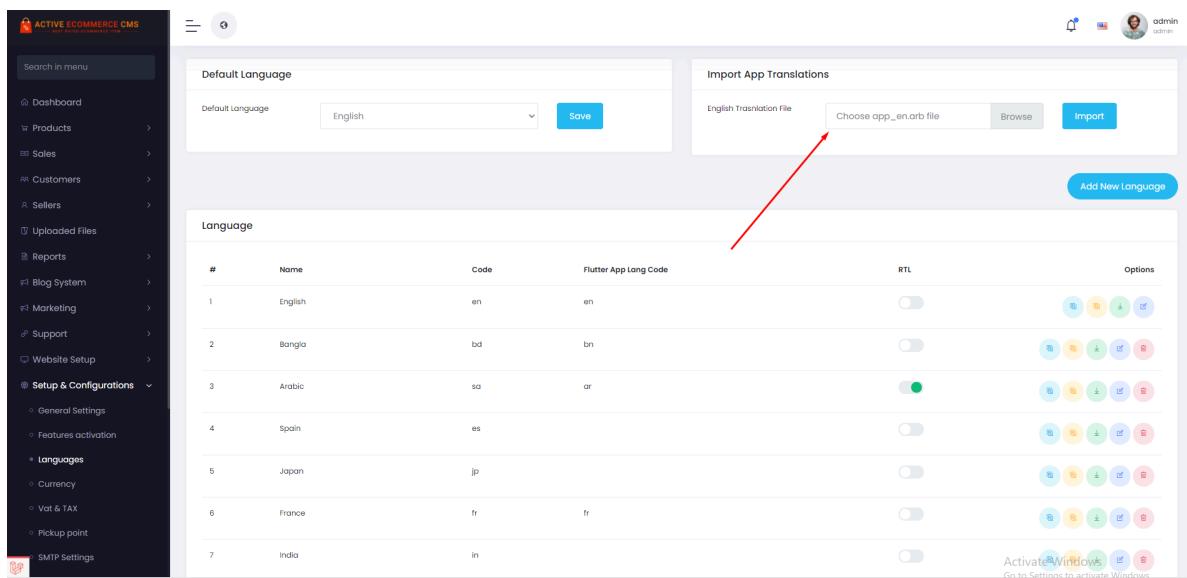
This value you can find in your admin panel. Go to your admin panel->setup & configurations->languages.

The screenshot shows the 'Language' section of the Admin Panel. It includes a 'Default Language' dropdown set to 'English', a 'Save' button, and an 'Import App Translations' section with a file input field and 'Import' button. Below these are three columns: 'Language', 'Code', and 'Options'. The 'Language' column lists 'English', 'Bangla', and 'Arabic'. The 'Code' column lists 'en', 'bd', and 'ar'. The 'Options' column contains icons for each language, with the 'RTL' switch turned on for Arabic.

Language	Code	Options
1 English	en	
2 Bangla	bd	
3 Arabic	ar	

15. 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 Admin Panel interface for managing languages. On the left, there's a sidebar with various menu items like Dashboard, Products, Sales, Customers, Sellers, Uploaded Files, Reports, Blog System, Marketing, Support, Website Setup, Setup & Configurations (with sub-options for General Settings, Features activation, Languages, Currency, VAT & TAX, Pickup point, and SMTP Settings), and Windows Activation. The main content area has two sections: 'Default Language' (set to English) and 'Import App Translations' (with a 'Choose app_en.arb file' input field and 'Import' button). 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 has a green 'RTL' switch (except for English) and a set of circular icons representing different language variants. At the bottom right of the table, there's a note: 'Activate Windows 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 639-1 format. Without a valid code , you will not see a translated output in the 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"/>	

1 2 3 >

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.

16. 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.