



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 run IOS on a simulator?
8. How to generate app store uploadable files?
9. How to Update for Android?
10. How to configure social login?
11. How to configure push notification?
12. How to configure google maps ?
13. How to configure the default language for mobile apps?
14. How to configure multiple languages for mobile apps?
15. 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)

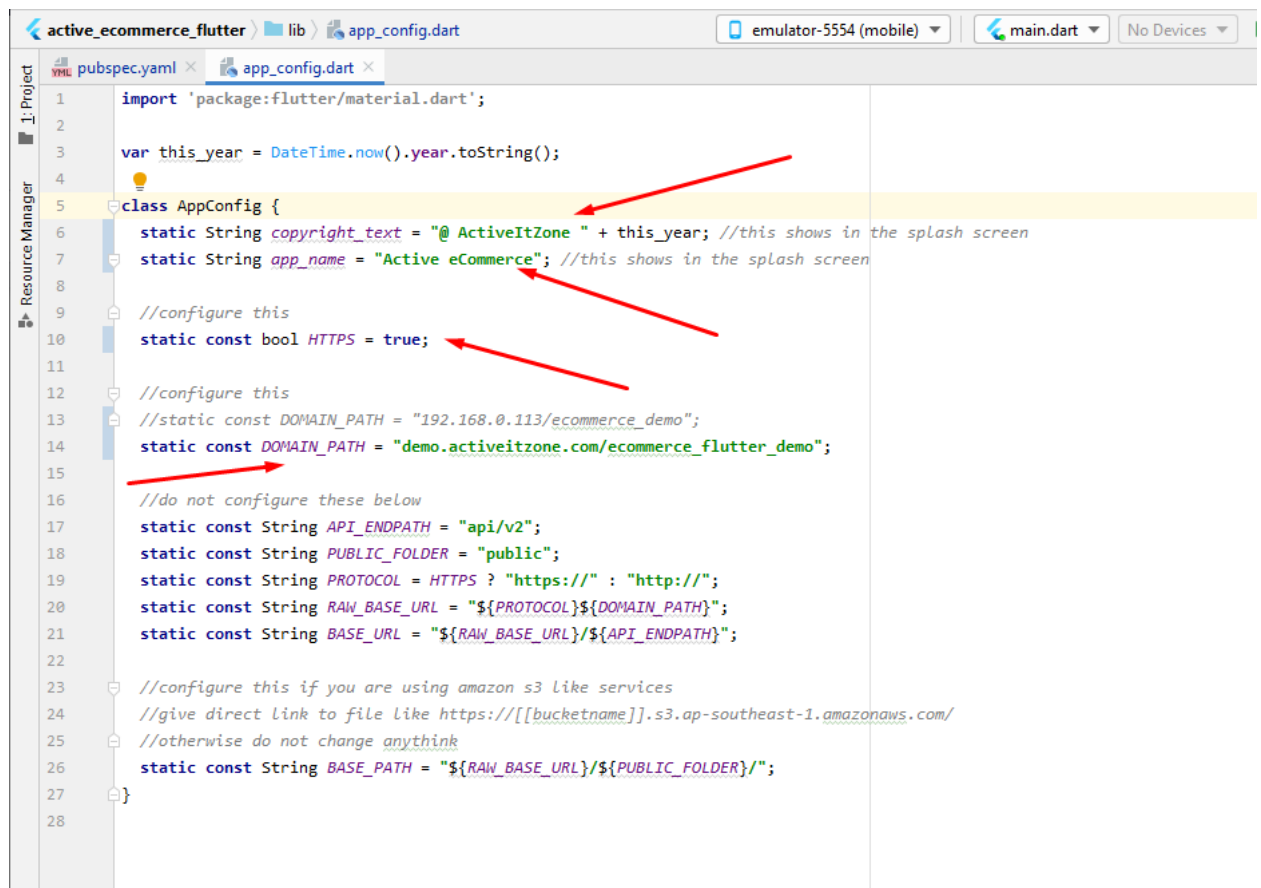
If you are using localhost , `DOMAIN_PATH` should be `"your_ip_address/your_project"`;

**** "localhost/your_project" will not 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/";`



```
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 String DOMAIN_PATH = "192.168.0.113/eCommerce_demo";
14    static const String DOMAIN_PATH = "demo.activeitzone.com/eCommerce_flutter_demo";
15
16    //do not configure these below
17    static const String API_ENDPATH = "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_ENDPATH}";
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
```

The screenshot shows the `app_config.dart` file in an IDE. Red arrows point to the following lines: line 6 (comment), line 7 (comment), line 10 (`HTTPS = true`), line 13 (`DOMAIN_PATH`), and line 14 (`DOMAIN_PATH`).

B. Theme Config:

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

Open lib/my_theme.dart

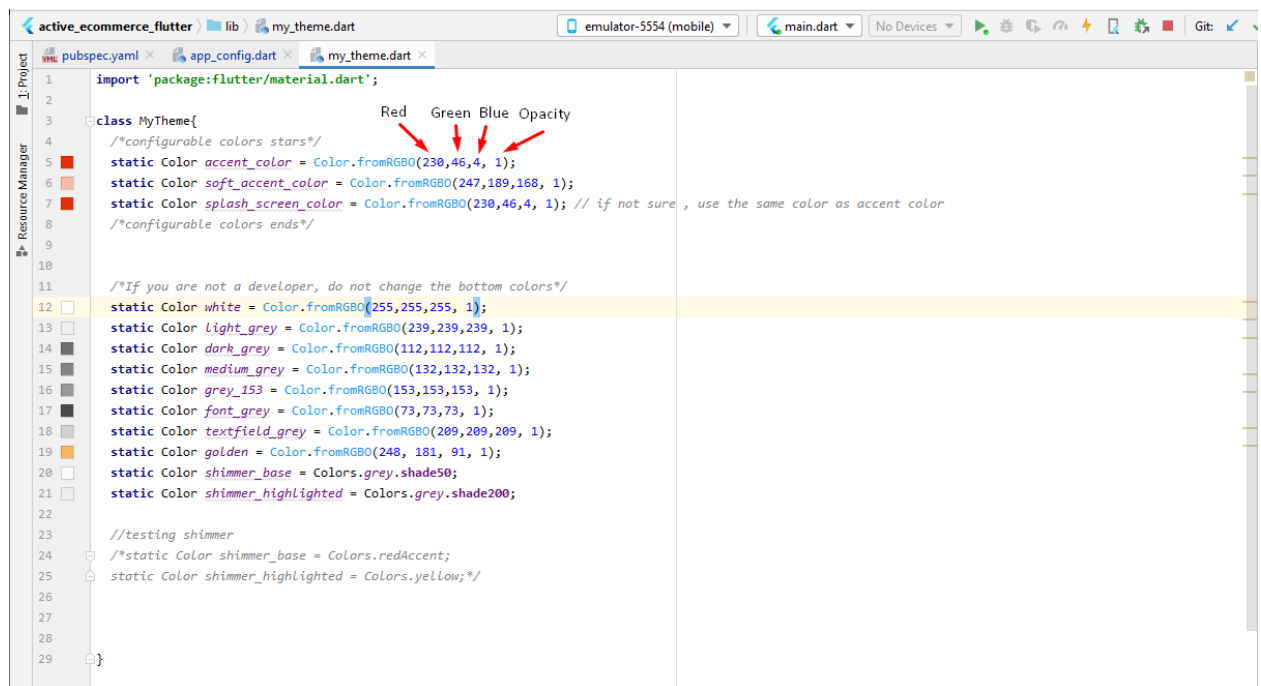
You can change the `accent_color`, `soft_accent_color`, `splash_screen_color` variable.

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.

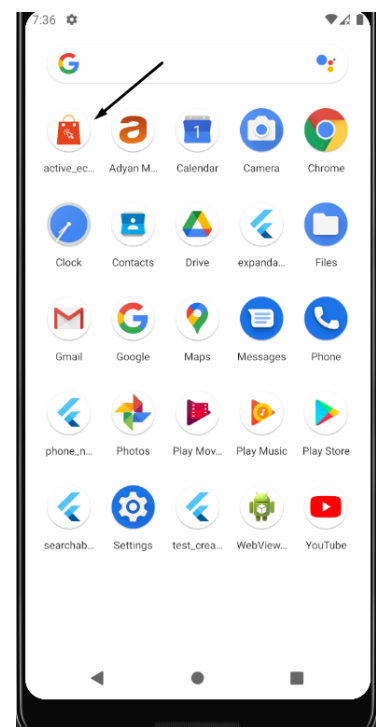


C. Configure the launcher icon:

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
```

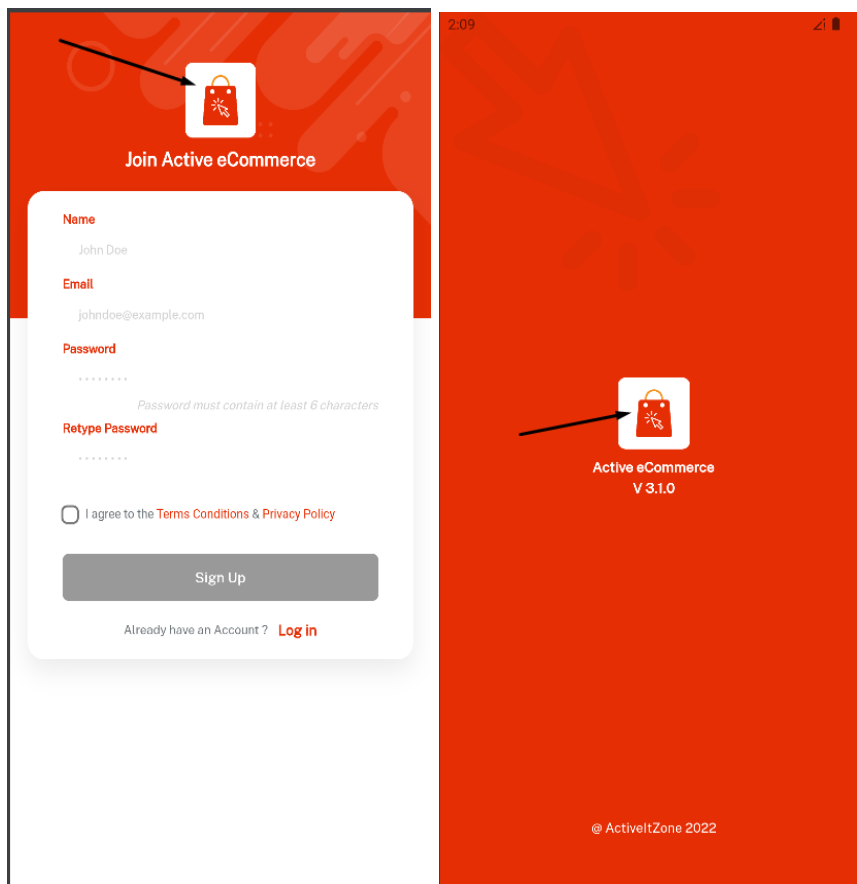
Then run :

```
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

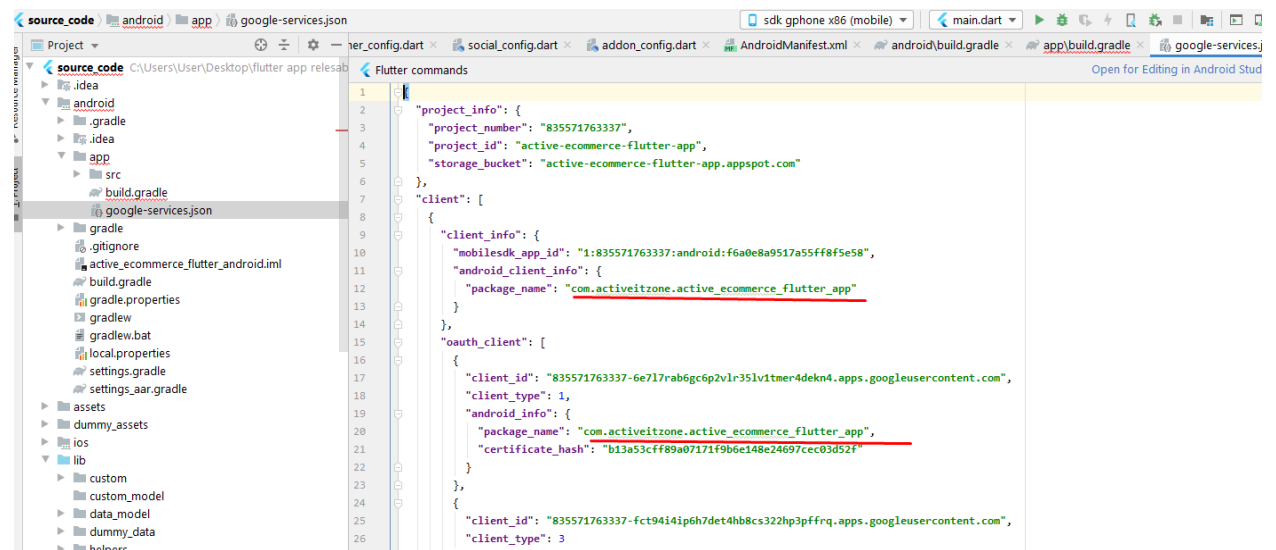
for **package name** just change in build build.gradle only (android/app/build.gradle)

```
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



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 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 running 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

[V] 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

[V] 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)

[V] 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

[V] 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

[V] 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:
-certreq          Generates a certificate request
-changealias      Changes an entry's alias
-delete          Deletes an entry
-exportcert       Exports certificate
-genkeypair       Generates a key pair
-genseckey        Generates a secret key
-gencert          Generates certificate from a certificate request
-importcert       Imports a certificate or a certificate chain
```

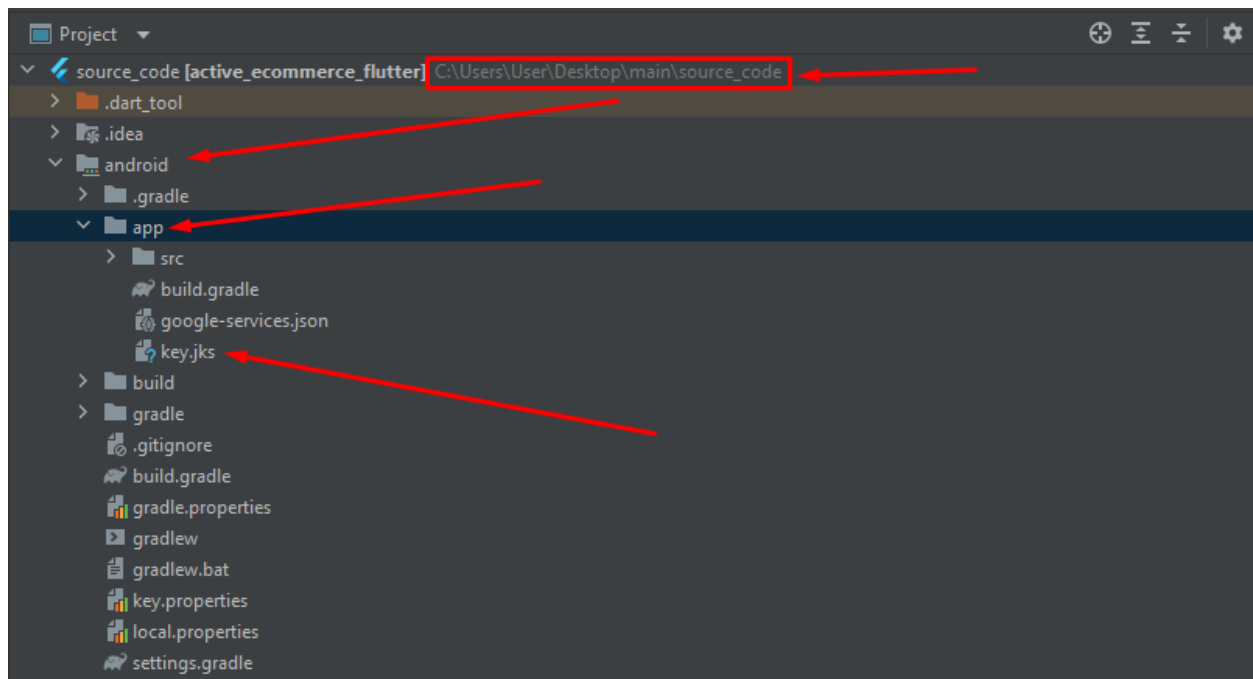
Create jks file for uploading app on playstore:

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



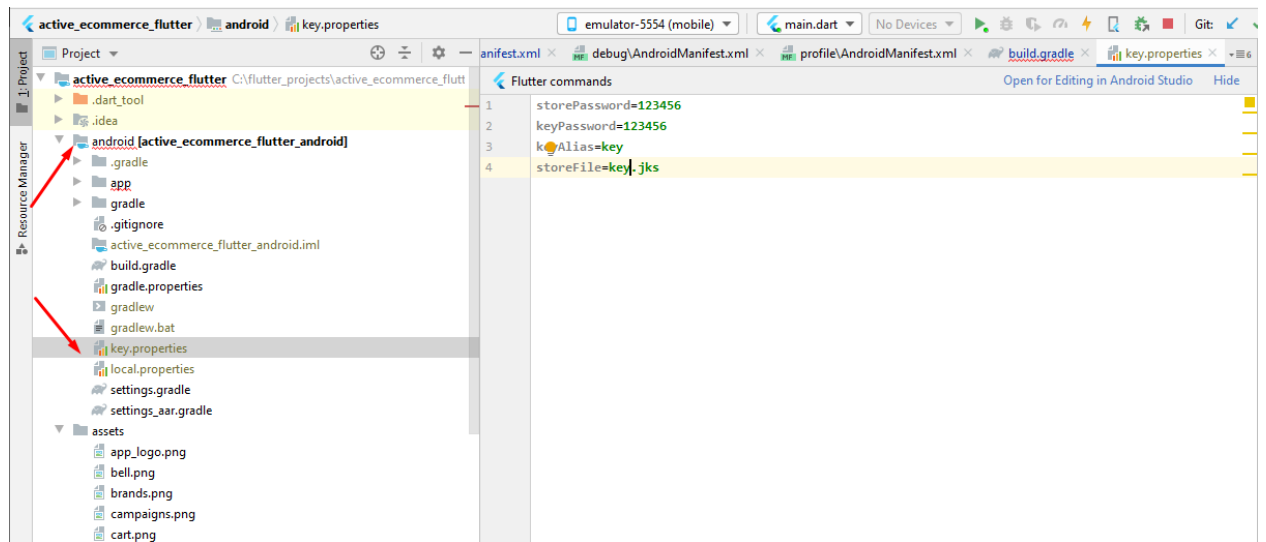
Reference the keystore from the app

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 the 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  
    }  
}
```

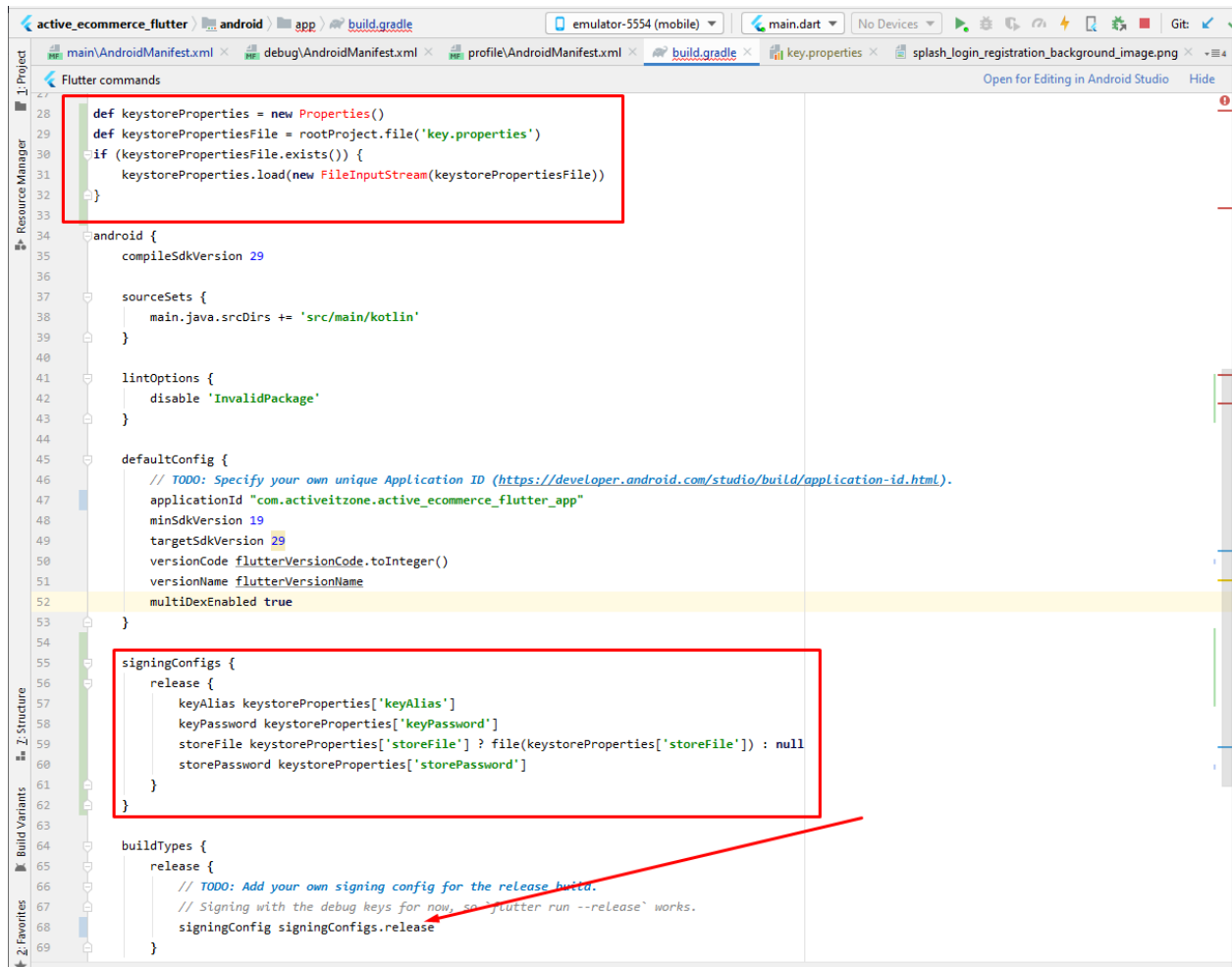
With the signing configuration info:

```
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.

Release builds of your app will now be signed automatically.

in app/build.gradle do necessary changes



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

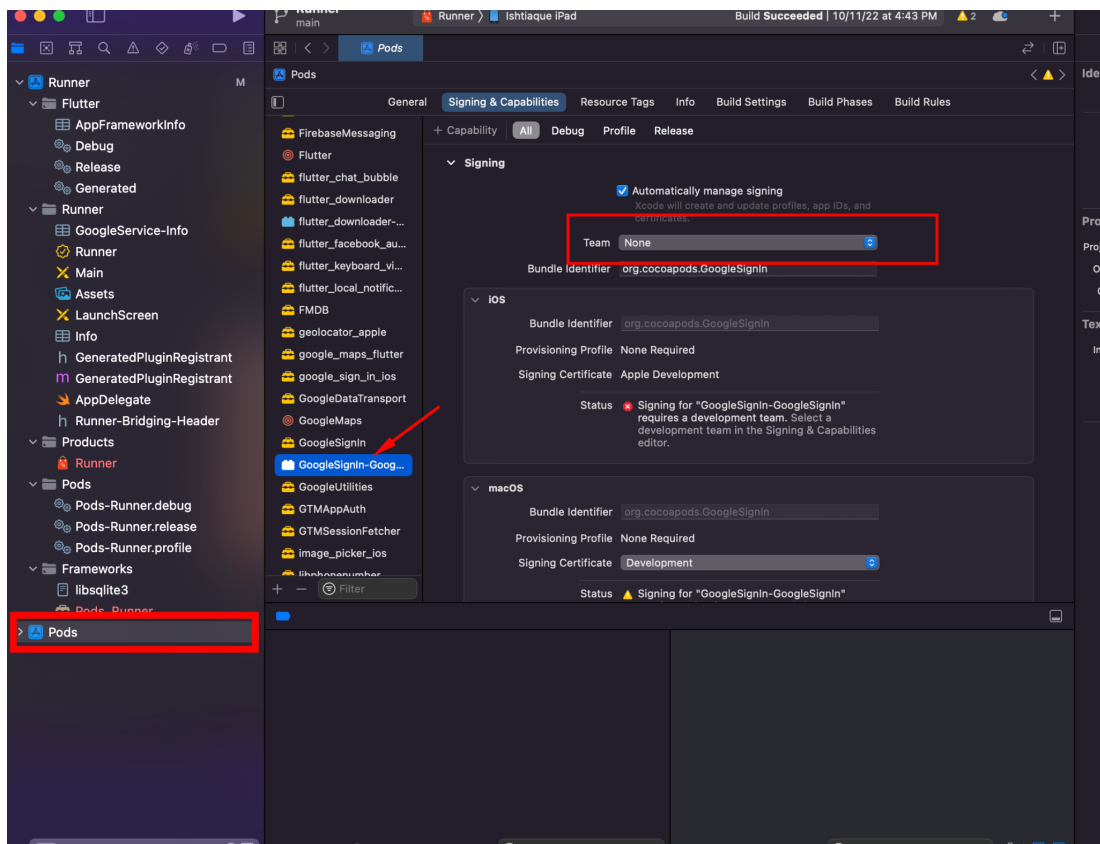
In your terminal run : `flutter build appbundle`

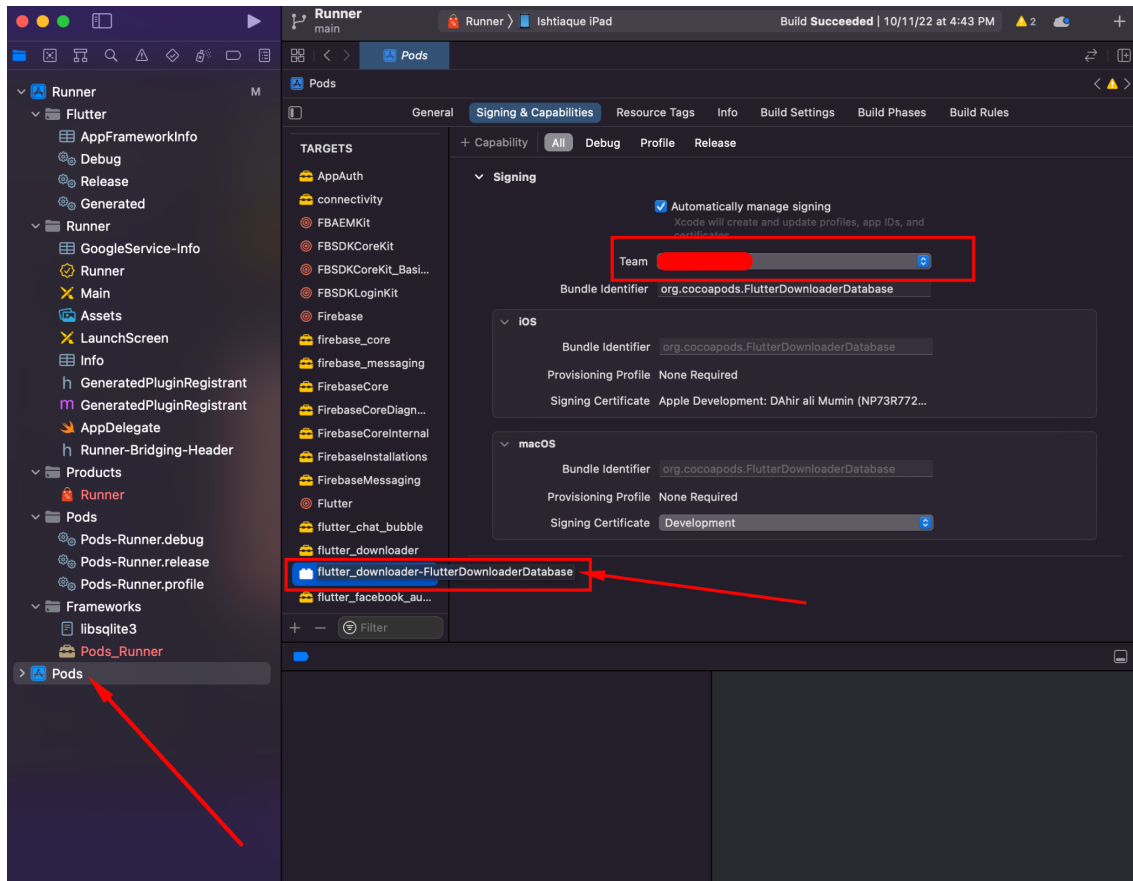
The release bundle for your app is created at `<your app dir>/build/app/outputs/bundle/release/app.aab`.

Upload this app.aab file to your google play console

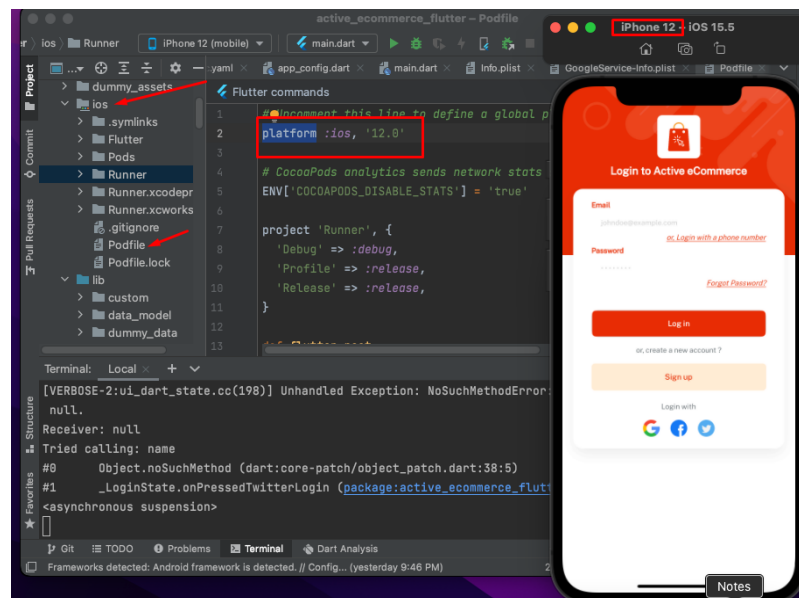
7. How to run IOS? ****Read all the points carefully before doing anything**

- 1) Install Xcode on your mac.
- 2) Open your ios folder in Xcode and select your developer team from Signin and capabilities.
- 3) Choose your team from GoogleSignin-GoogleSignin and flutter_downloader-FlutterDownloaderDatabase from Pod->GoogleSignin-GoogleSignin and Pod->flutter_downloader-FlutterDownloaderDatabase





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



- 6) Run this command **flutter run**.

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

8. How to generate app store uploadable files?

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 create.
- 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.

9. 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 source_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.

10. How to configure social login?

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.firebase.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.

```
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:44:77:00:42:4C
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
```

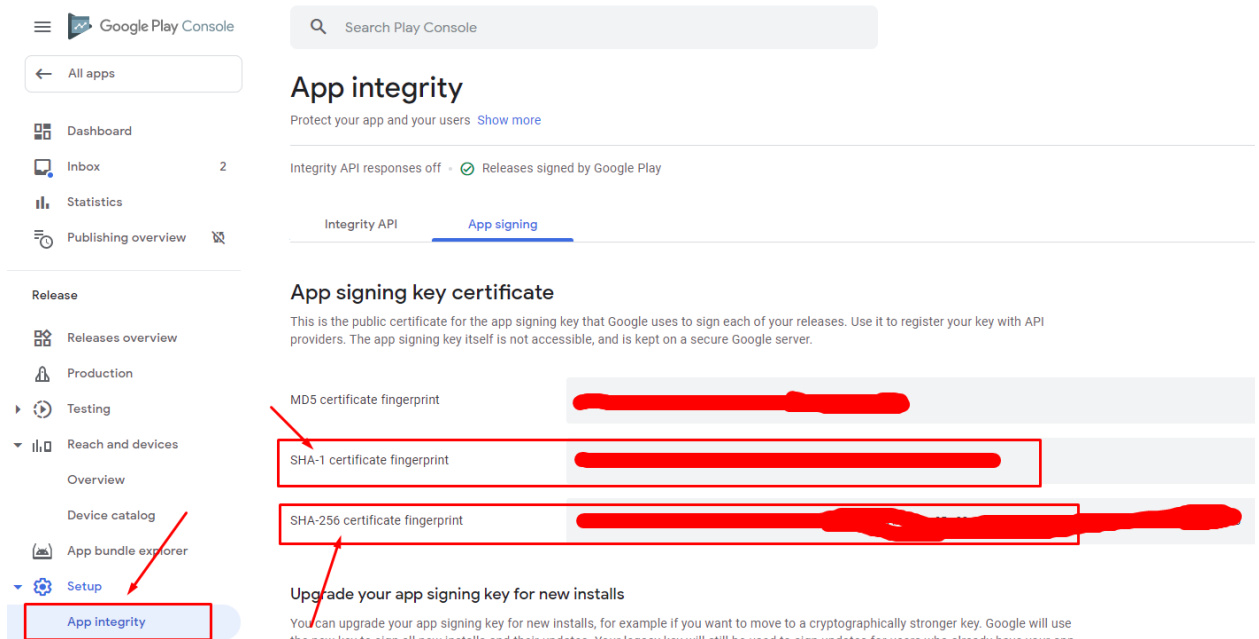
- ii) For release mode: write this command on your android studio terminal “keytool -list -v -keystore YOUR_PROJECT_DIRECTORY\key.jks -alias key” then press enter.

```
Project
├── android
│   ├── build.gradle
│   ├── google-services.json
│   ├── key.jks
│   └── ...
└── ...

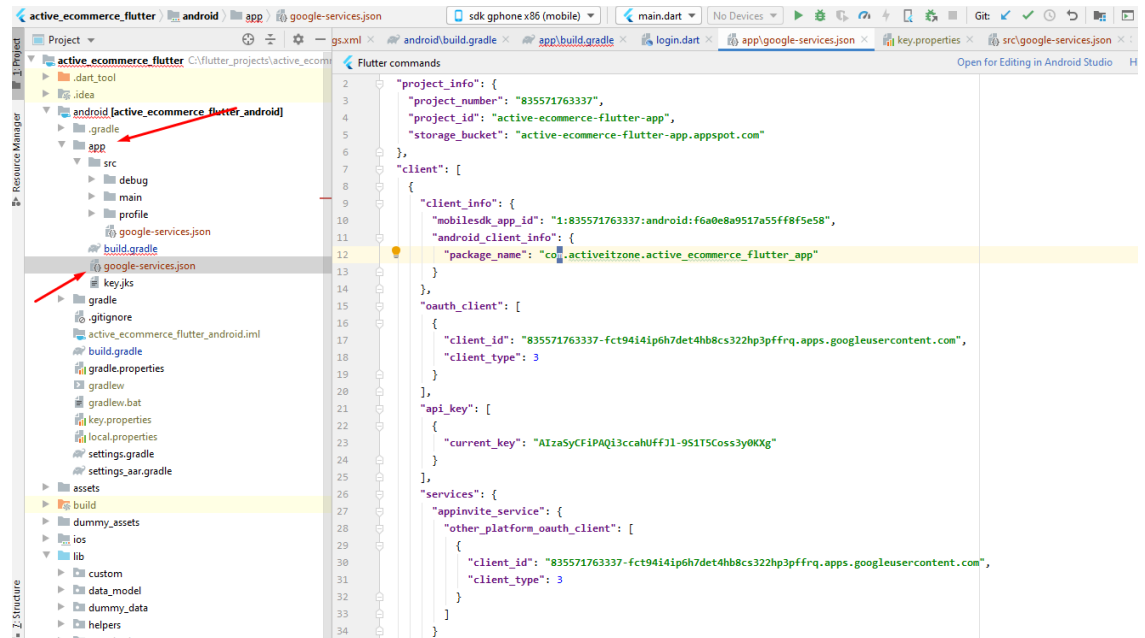
Terminal
C:\flutter_projects\active_ecommerce_flutter\android>keytool -list -v -keystore key.jks -alias key
keytool error: java.lang.Exception: Keystore file does not exist: key.jks
java.lang.Exception: Keystore file does not exist: key.jks
    at sun.security.tools.keytool.Main.doCommands(Unknown Source)
    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: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=ActiveIT, OU=ActiveIT, O=ActiveIT, L=Shame, ST=Bangladesh, C=BD
Issuer: CN=ActiveIT, OU=ActiveIT, O=ActiveIT, L=Shame, ST=Bangladesh, C=BD
Serial number: 7dd9263u
Valid from: Thu Jan 06 12:23:00 EST '22 until: Sun May 22 06:23:41 EST '28
Certificate fingerprints:
    MD5: D6:A0:00:A2:1D:66:53:87:FA:CD:47:56:56:56:D0:43
    SHA1: 8B:2A:32:AC:06:3C:89:1A:6E:7F:94:2A:07:14:0B:D5:74:45:0F:58
```

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



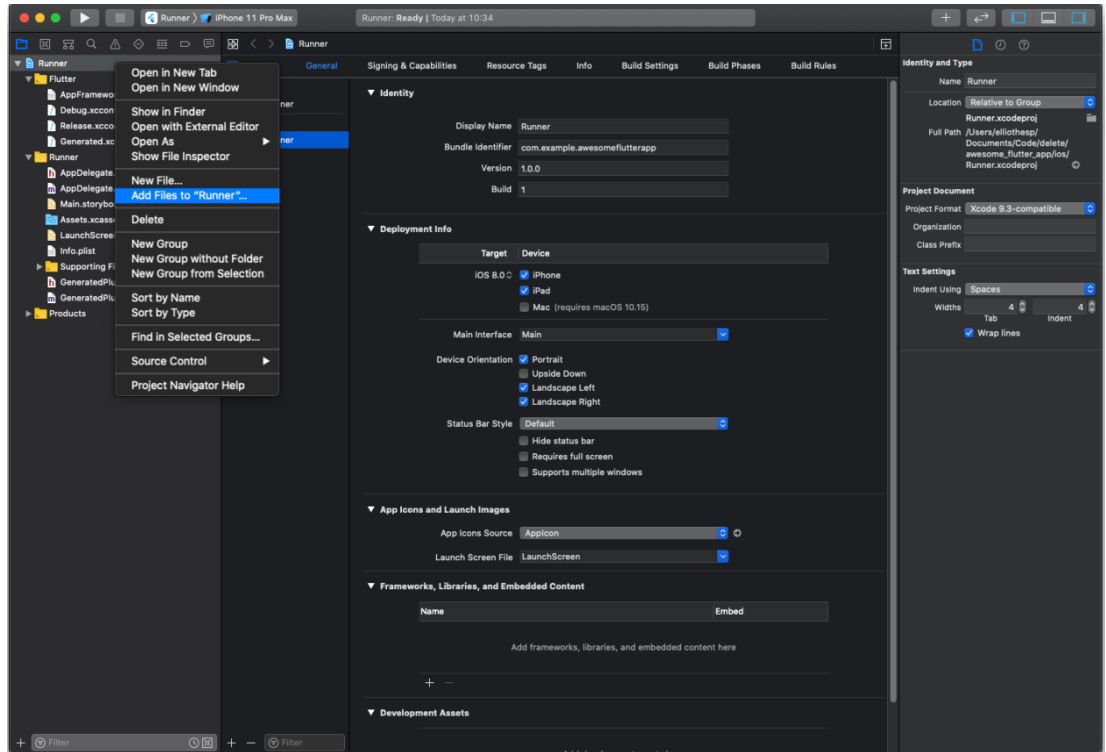
You must generate your own google-services.json. Do not use ours - it will not work for you.



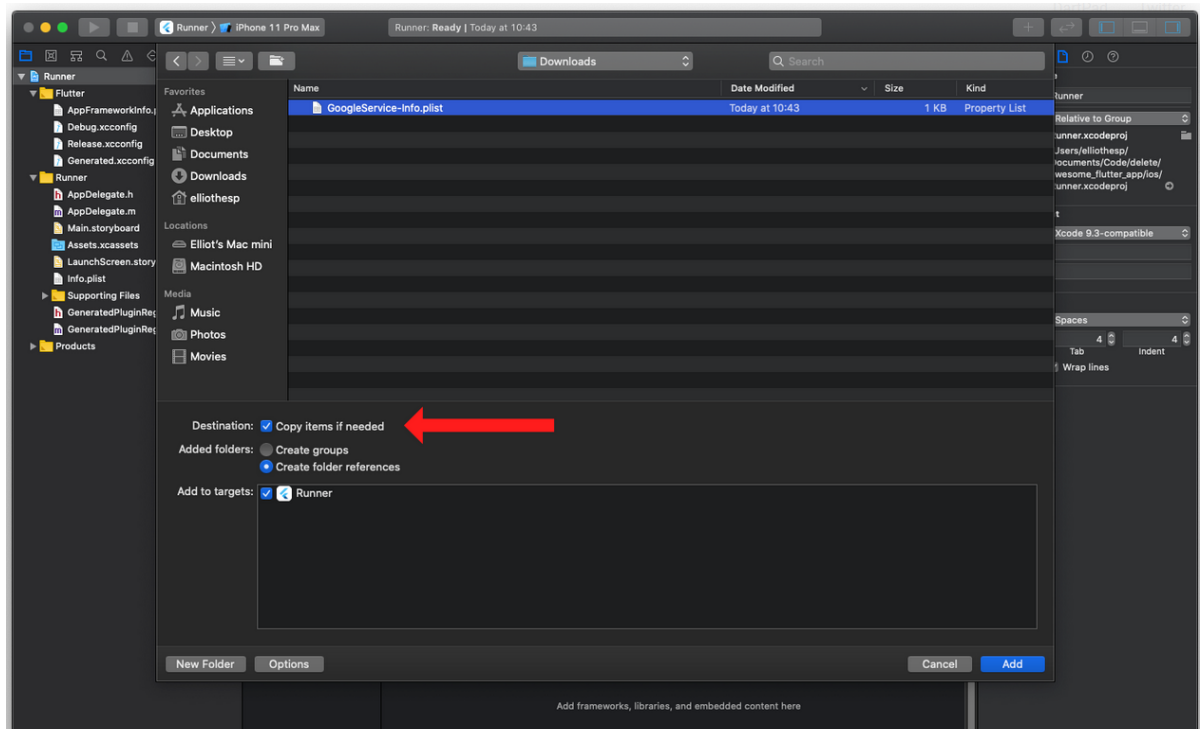
Add IOS app on your Firebase project

1. Goto your firebase project setting 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:



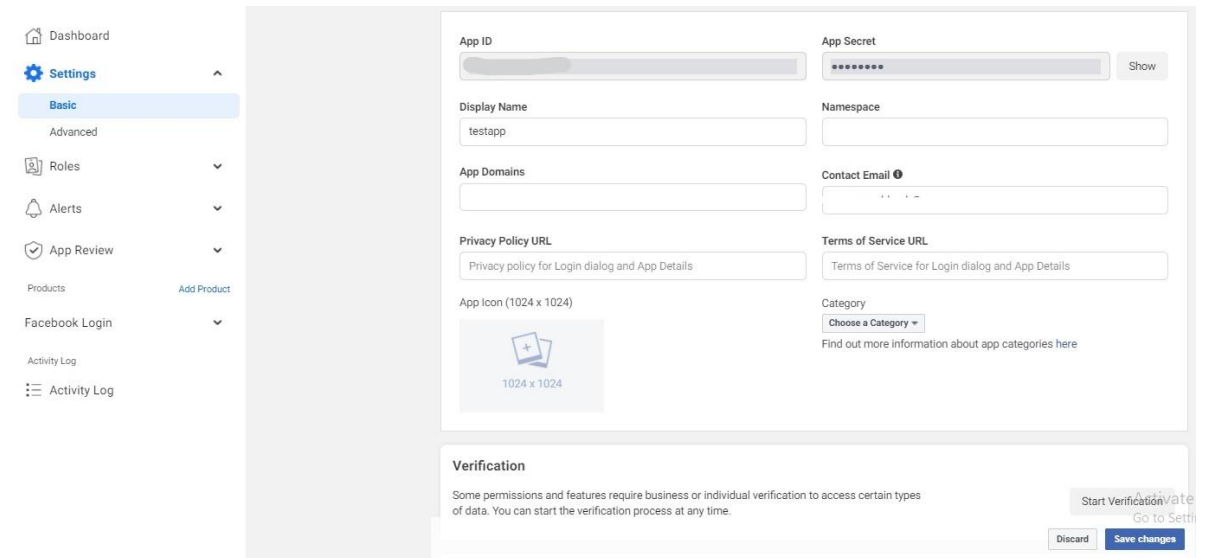
Select the GoogleService-Info.plist file you downloaded, and ensure the "Copy items if needed" checkbox is enabled:



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.



The screenshot shows the 'Settings' page for a Firebase app. The left sidebar contains navigation links: Dashboard, Settings (selected), Basic, Advanced, Roles, Alerts, App Review, Products, Facebook Login, Activity Log, and Activity Log. The main content area is divided into two columns. The left column contains fields for App ID, Display Name (testapp), App Domains, Privacy Policy URL, and App Icon (1024 x 1024). The right column contains fields for App Secret, Namespace, Contact Email, Terms of Service URL, and Category. At the bottom, there is a 'Verification' section with a 'Start Verification' button and a 'Go to Settings' link. A 'Save changes' button is located at the bottom right.

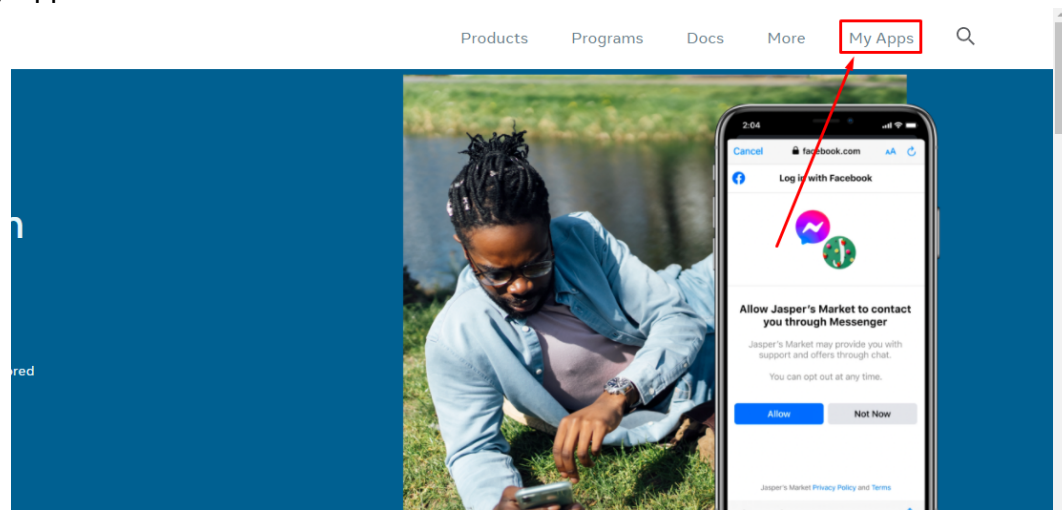
I) Facebook: Package Used

https://pub.dev/packages/flutter_facebook_auth

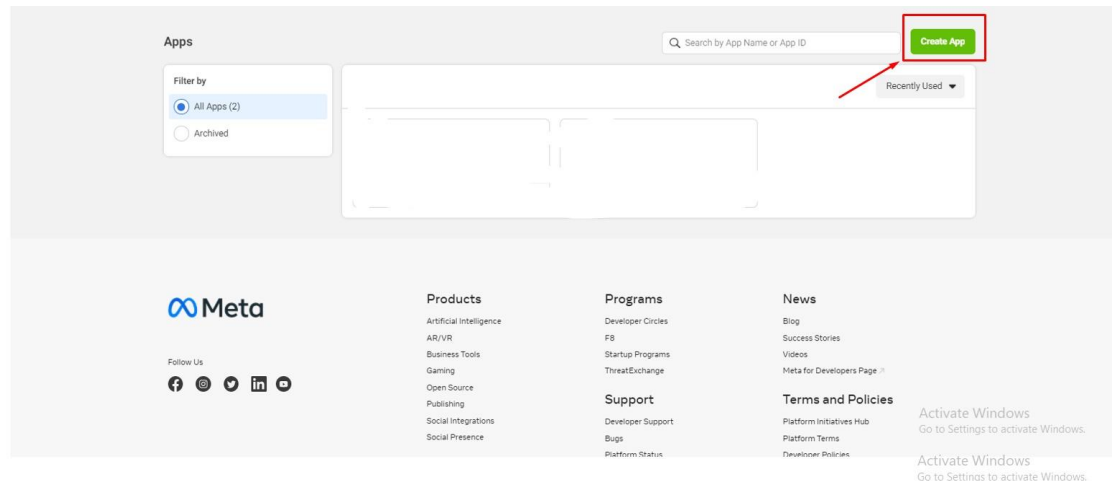
See its documentation and steps

How to create a facebook app ?

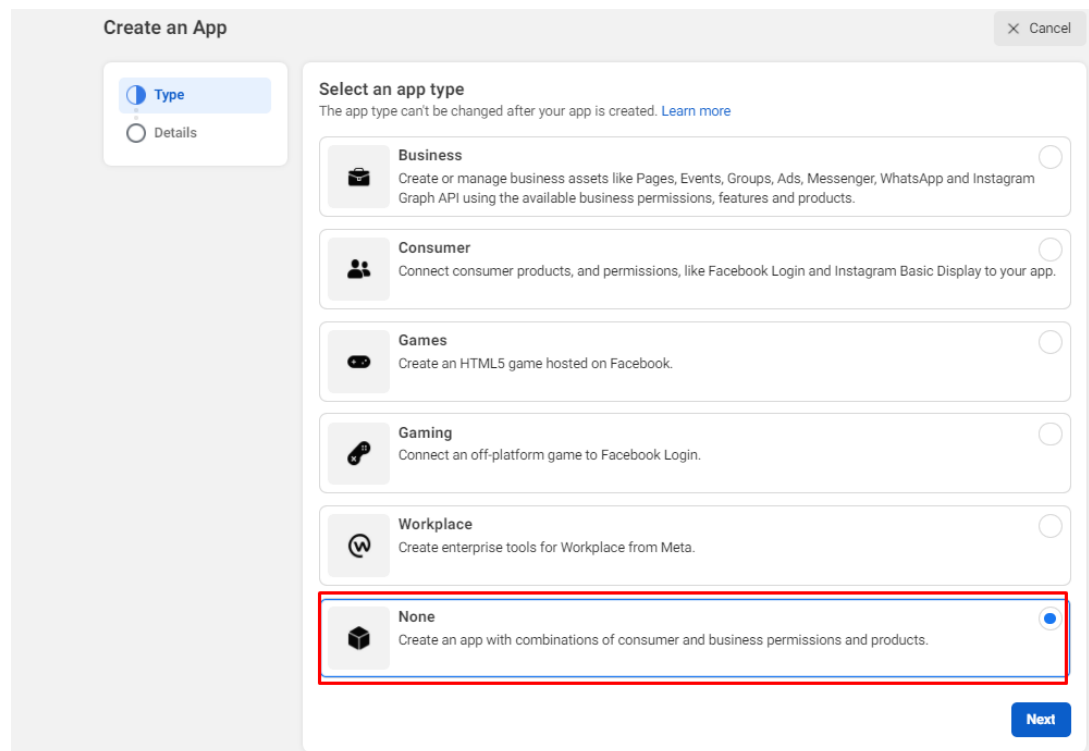
- A) First you have to login facebook in your browser.
- B) Goto <https://developers.facebook.com/>
- C) Goto My Apps.



D) Create an app.



Select App Type None.



E) Add your app details

- App Name.
- App Contact Email.

- Business Account. (Don't Change it).

Create an App [X] Cancel

Type (checked) **Details**

Add details

Display name
This is the app name associated with your app ID. You can change this later.
[Input field] *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.
[Input field] *Email*

Business Account · Optional
To access certain permissions or features, apps need to be connected to a Business Account.
[Dropdown: No Business Account selected] *Don't change this option*

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

F) Add product “Facebook Login”.

- ❖ Click the “Set Up” button.

Add a Product

<p>Facebook Login</p> <p>The world's number one social login product.</p> <p>Read Docs Set Up</p>	<p>Audience Network</p> <p>Monetize your app and grow revenue with ads from Facebook advertisers.</p> <p>Read Docs Set Up</p>	<p>App Events</p> <p>Understand how people engage with your business across apps, devices, platforms and websites.</p> <p>Read Docs Set Up</p>
<p>Messenger</p> <p>Customize the way you interact with people on Messenger.</p> <p>Read Docs Set Up</p>	<p>Webhooks</p> <p>Subscribe to changes and receive updates in real time without calling the API.</p> <p>Read Docs Set Up</p>	<p>Games</p> <p>Create a cross-platform HTML5 game hosted on Facebook.</p> <p>Read Docs Set Up</p>

G) Click the “Basic” Option.

Dashboard **Settings** **Basic** *Advanced* Roles Alerts App Review Products Facebook Login Activity Log

App ID [Input field]

App Secret [Input field] Show

Display Name [Input field: testapp]

App Domains [Input field]

Privacy Policy URL [Input field: Privacy policy for Login dialog and App Details]

App Icon (1024 x 1024) [Image placeholder]

Namespace [Input field]

Contact Email [Input field: mdtusaraddmob@gmail.com]

Terms of Service URL [Input field: Terms of Service for Login dialog and App Details]

Category [Dropdown: Choose a Category] Find out more information about app categories here

H) 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.

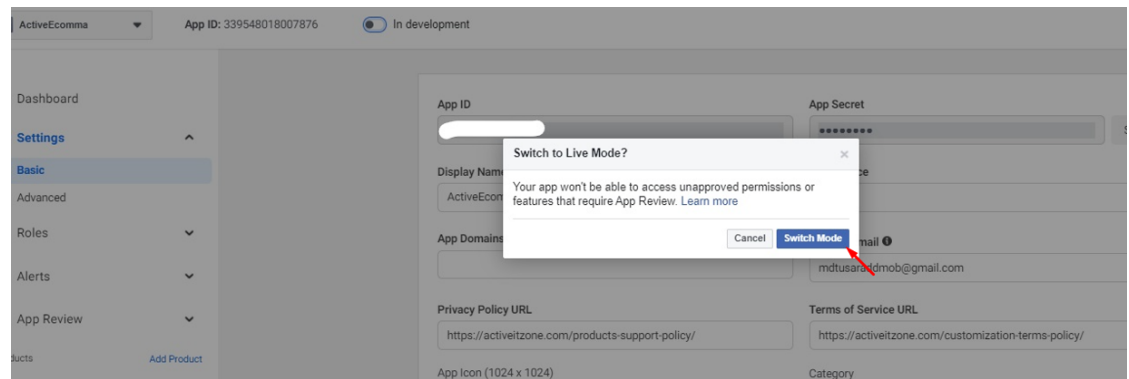
The screenshot shows a form for configuring an app. Red arrows point to the following fields:

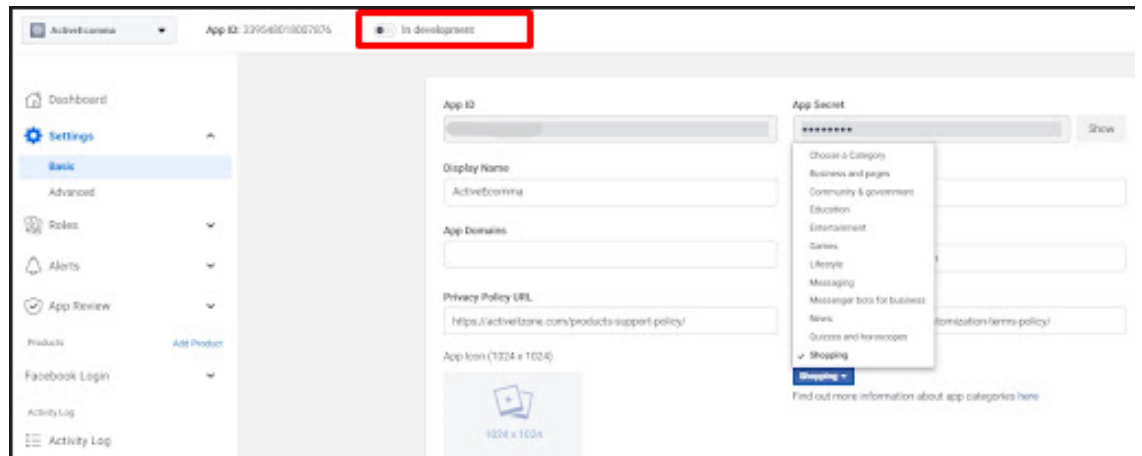
- Privacy Policy URL:** Labeled "Privacy Policy Url", it contains the text "Privacy policy for Login dialog and App Details".
- Terms of Service URL:** Labeled "Terms of service url", it contains the text "Terms of Service for Login dialog and App Details".
- App Icon:** Labeled "App icon", it shows a placeholder icon with a plus sign and the text "1024 x 1024".
- Category:** Labeled "Choose your app category", it shows a dropdown menu with the text "Choose a Category".

Other visible fields include: Display Name (testapp), Namespace, App Domains, Contact Email (mdtusraddmob@gmail.com), and a link "Find out more information about app categories here".

J) Now activate your app.

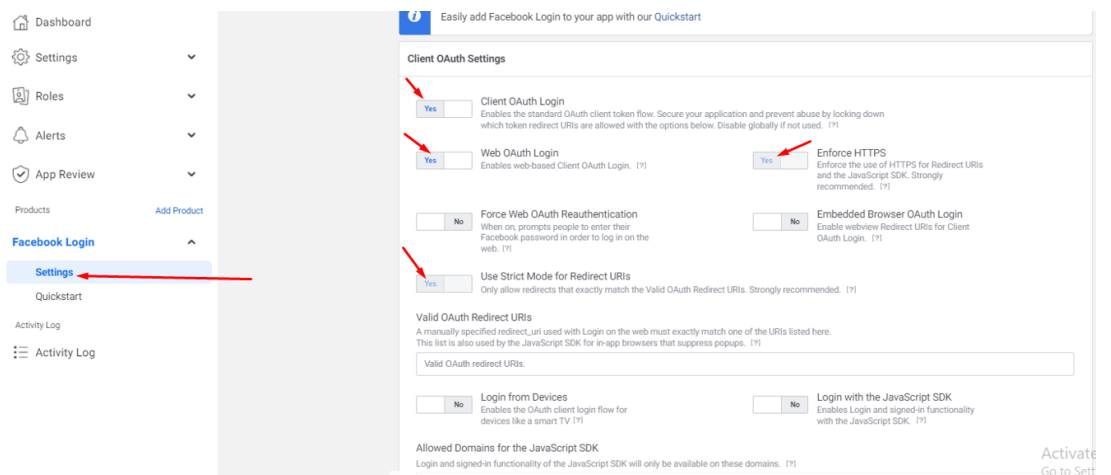
K) Click the "Switch Mode" Button.





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

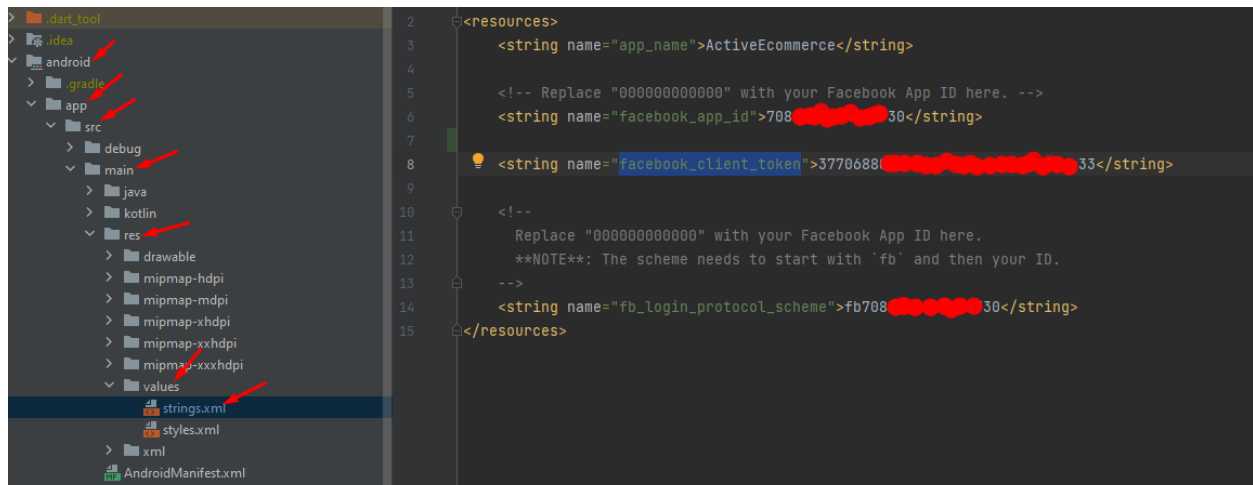


Integration

For Android:

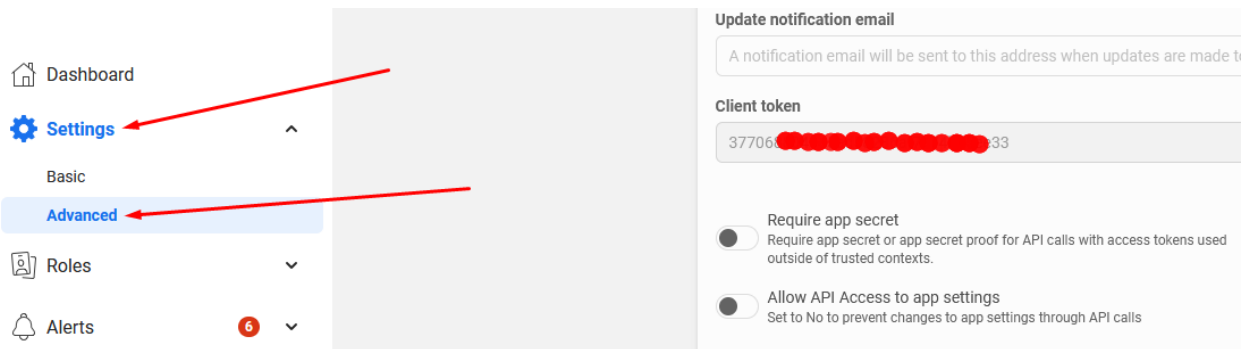
M) Add your facebook app id, app name, **facebook_client_token** and fb login protocol scheme (**NOTE**: The scheme needs to start with `fb` and then your ID.example: fb123456789)

into your **project->android->app->src->main->res->values->string.xml** file.



Client Token: Where do you get the client token?

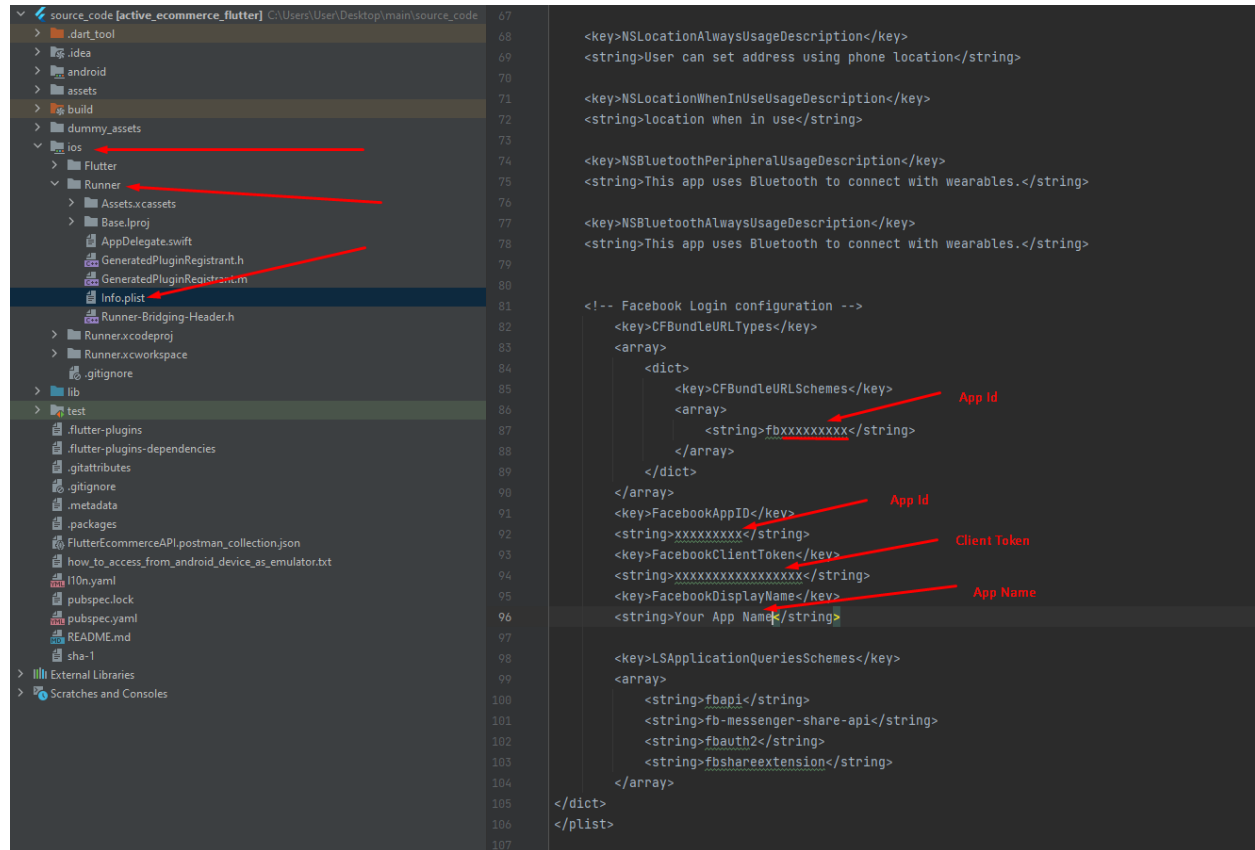
Ans: Goto your facebook app setting->advance finde the Client token



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.



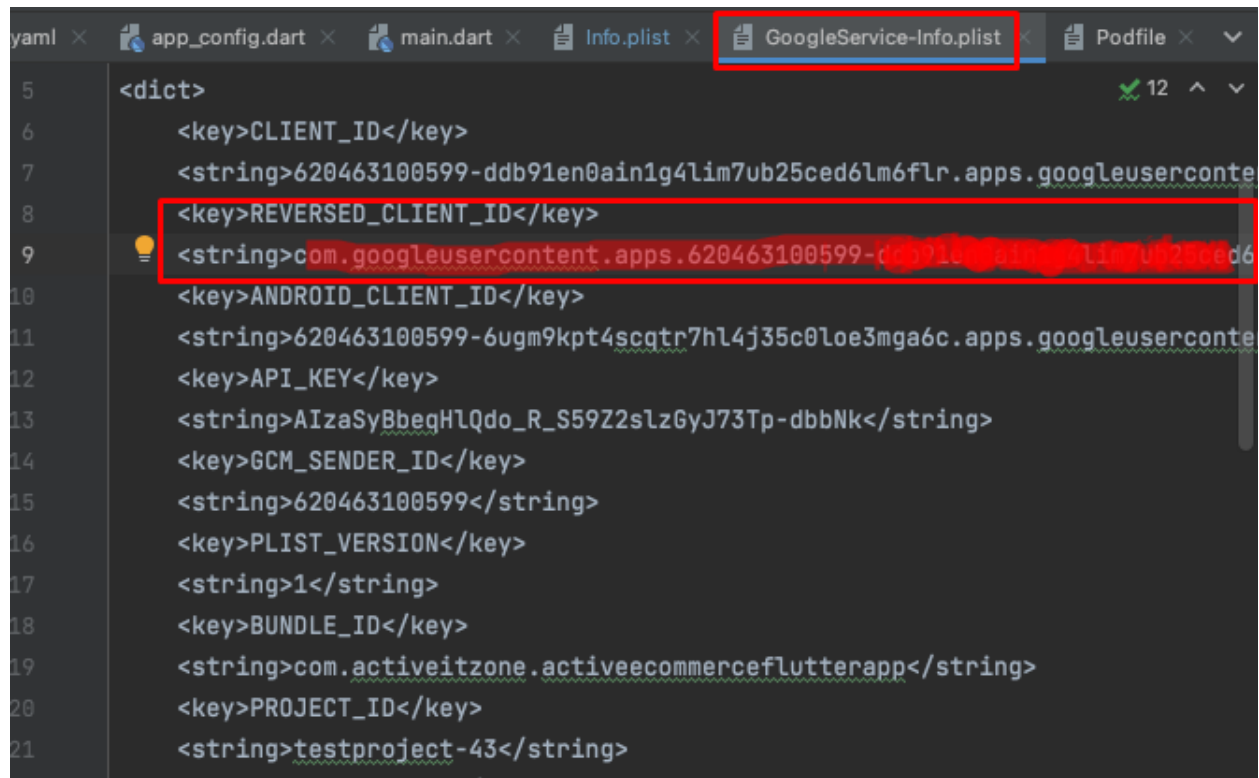
Google: Package Used

https://pub.dev/packages/google_sign_in

Integration

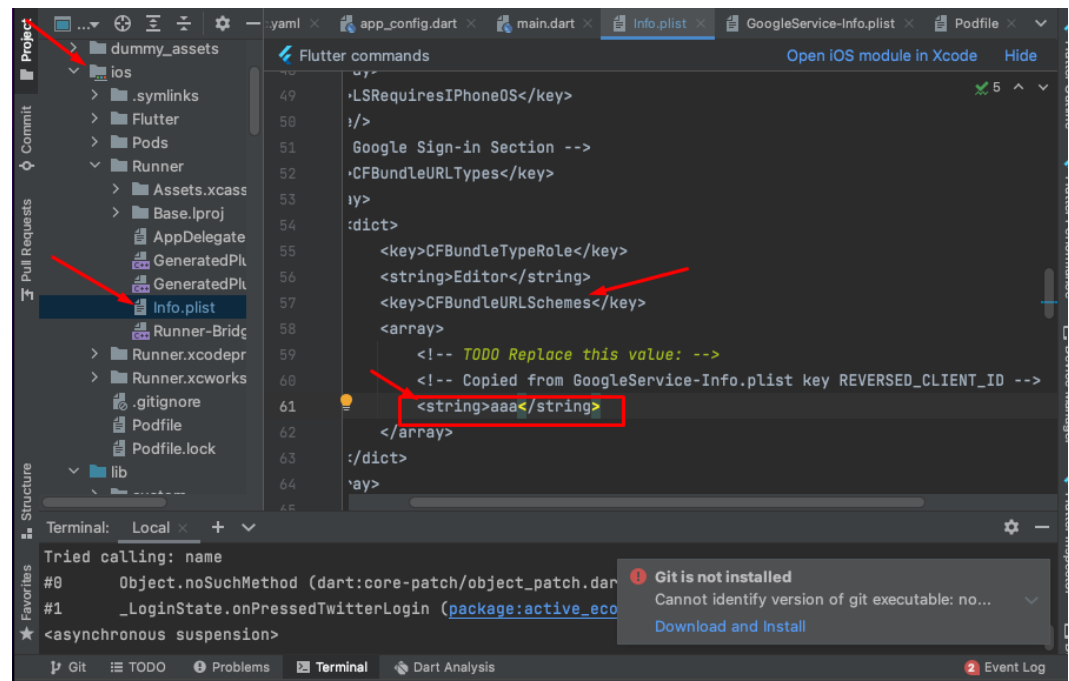
For IOS:

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



```
<dict>
  <key>CLIENT_ID</key>
  <string>620463100599-ddb91en0ain1g4lim7ub25ced6lm6flr.apps.googleusercontent.com</string>
  <key>REVERSED_CLIENT_ID</key>
  <string>com.googleusercontent.apps.620463100599-ddb91en0ain1g4lim7ub25ced6lm6flr</string>
  <key>ANDROID_CLIENT_ID</key>
  <string>620463100599-6ugm9kpt4scqtr7hl4j35c0loe3mga6c.apps.googleusercontent.com</string>
  <key>API_KEY</key>
  <string>AIzaSyBbegHlQdo_R_S59Z2slzGyJ73Tp-dbbNk</string>
  <key>GCM_SENDER_ID</key>
  <string>620463100599</string>
  <key>PLIST_VERSION</key>
  <string>1</string>
  <key>BUNDLE_ID</key>
  <string>com.activeitzone.activecommerceflutterapp</string>
  <key>PROJECT_ID</key>
  <string>testproject-43</string>
</dict>
```

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



```
Flutter commands
49  .LSRequiresIPhoneOS</key>
50  />
51  Google Sign-in Section -->
52  .CFBundleURLTypes</key>
53  <array>
54  :dict>
55    <key>CFBundleTypeRole</key>
56    <string>Editor</string>
57    <key>CFBundleURLSchemes</key>
58    <array>
59      <!-- TODO Replace this value: -->
60      <!-- Copied from GoogleService-Info.plist key REVERSED_CLIENT_ID -->
61      <string>aaa</string>
62    </array>
63  </dict>
64  </array>
65  </dict>
```

Tried calling: name
#0 Object.noSuchMethod (dart:core-patch/object_patch.dart:32:11)
#1 _LoginState.onPressedTwitterLogin (package:active_eco...
<asynchronous suspension>

Git is not installed
Cannot identify version of git executable: no...
Download and Install

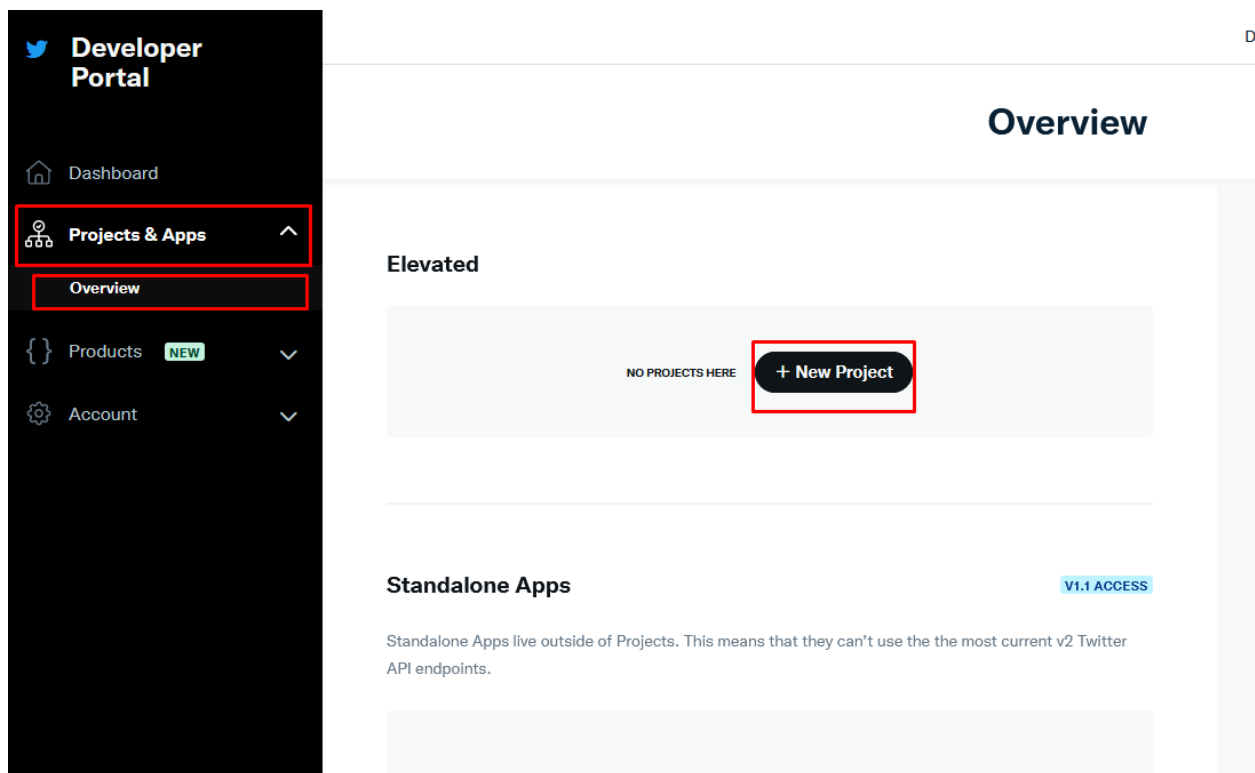
Follow the guideline from here https://pub.dev/packages/google_sign_in

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.



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

1 Project name **2 Use case** **3 Project description** **4 App s**

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

Your project name

21

1

Project description

Required

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)

2

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.)

1 App Environment **2 App name** **3 Keys & Tokens**

Development
For new work and in progress features

Staging
Used to test out changes

Production
Used for live products

1

Apps are where you get your access **keys & tokens**, plus set permissions. You can find them within your Projects.

DEVELOPMENT APP

App name

Required

2

For security, this will be the last time we'll fully display these. If something happens, you can always regenerate them. [Learn more](#)

API Key

pmF3fBBBlnFbGr

Copy

API Key Secret

ELRFY64JzIOPVjgQDFoUJnCFiwo

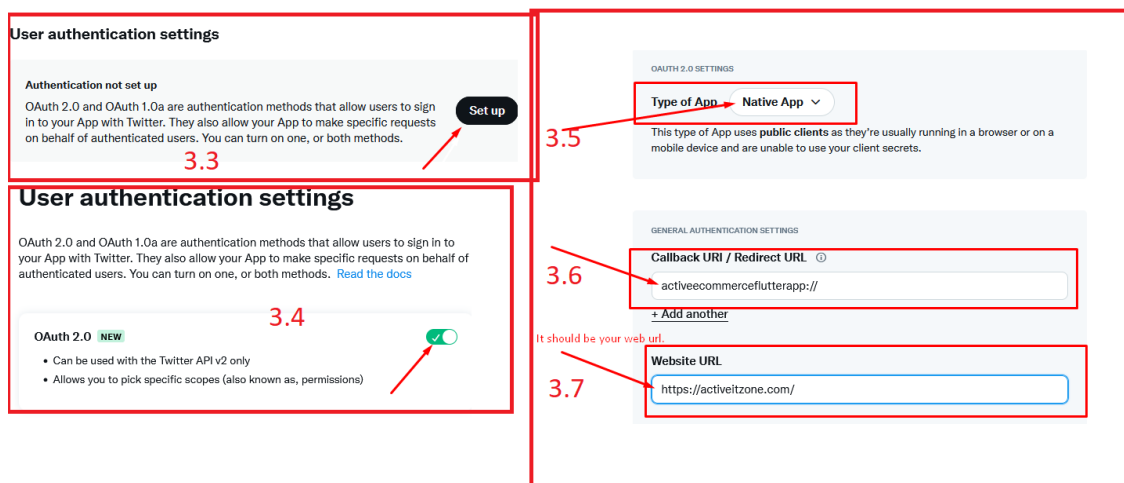
Copy

3

Go to dashboard

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 URL.(**activecommerceflutterapp://**).
- 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`.



Apple SignIn: package used

https://pub.dev/packages/sign_in_with_apple

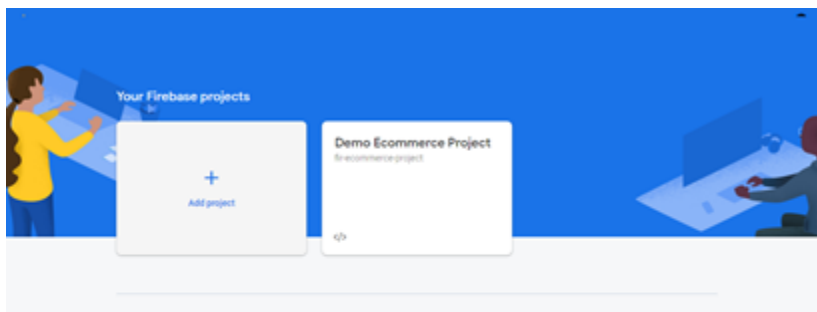
Configure properly as we mention in our ecommerce cms documentation. You don't need extra configuration for the app.

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

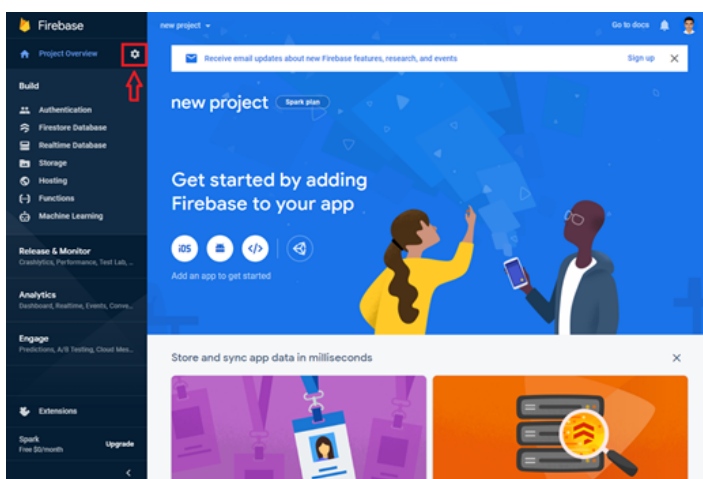
11. How to configure push notification?

To use firebase follow the procedure which are mentioned below

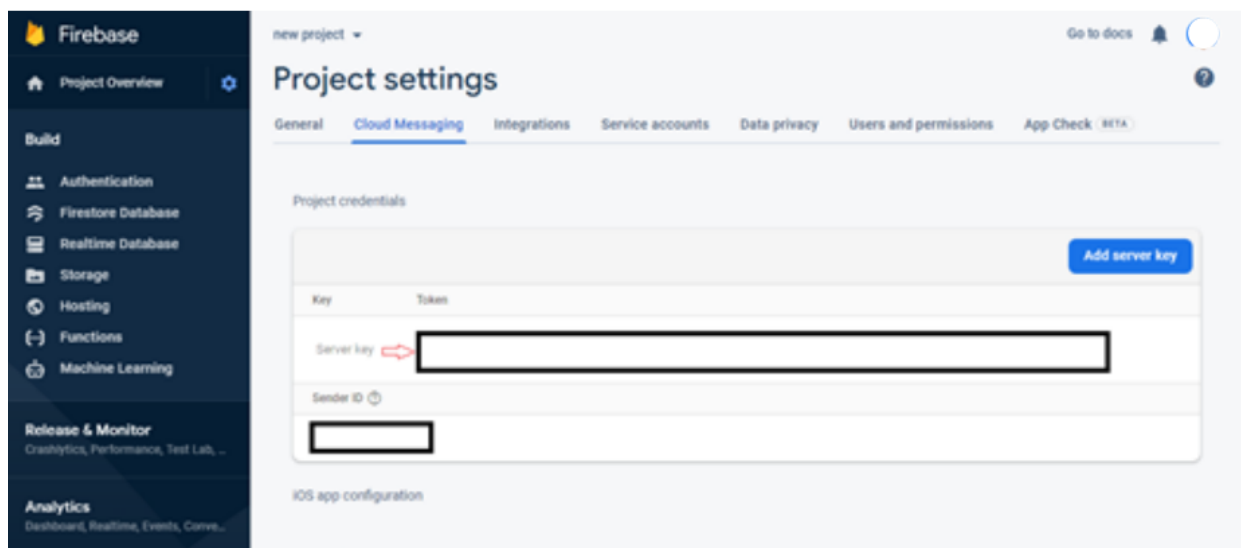
1. Go to this URL to create project <https://console.firebase.google.com/u/0/>
If you already have a project then continue with that.



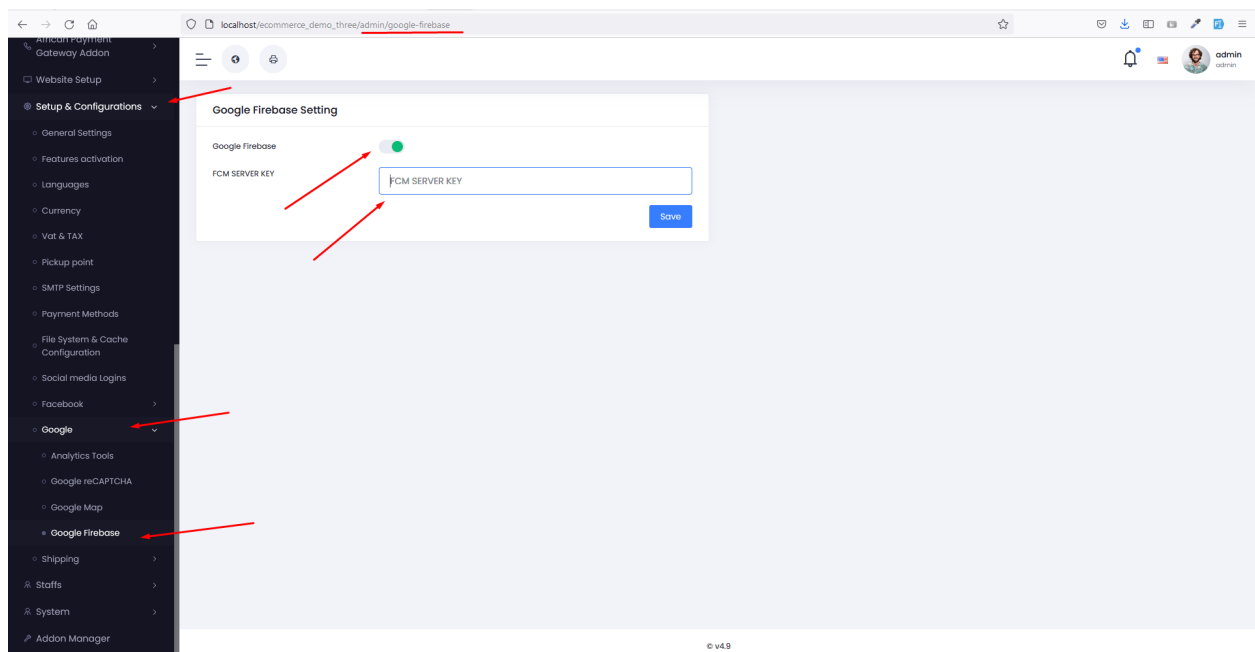
2. Now go to project settings to get server key



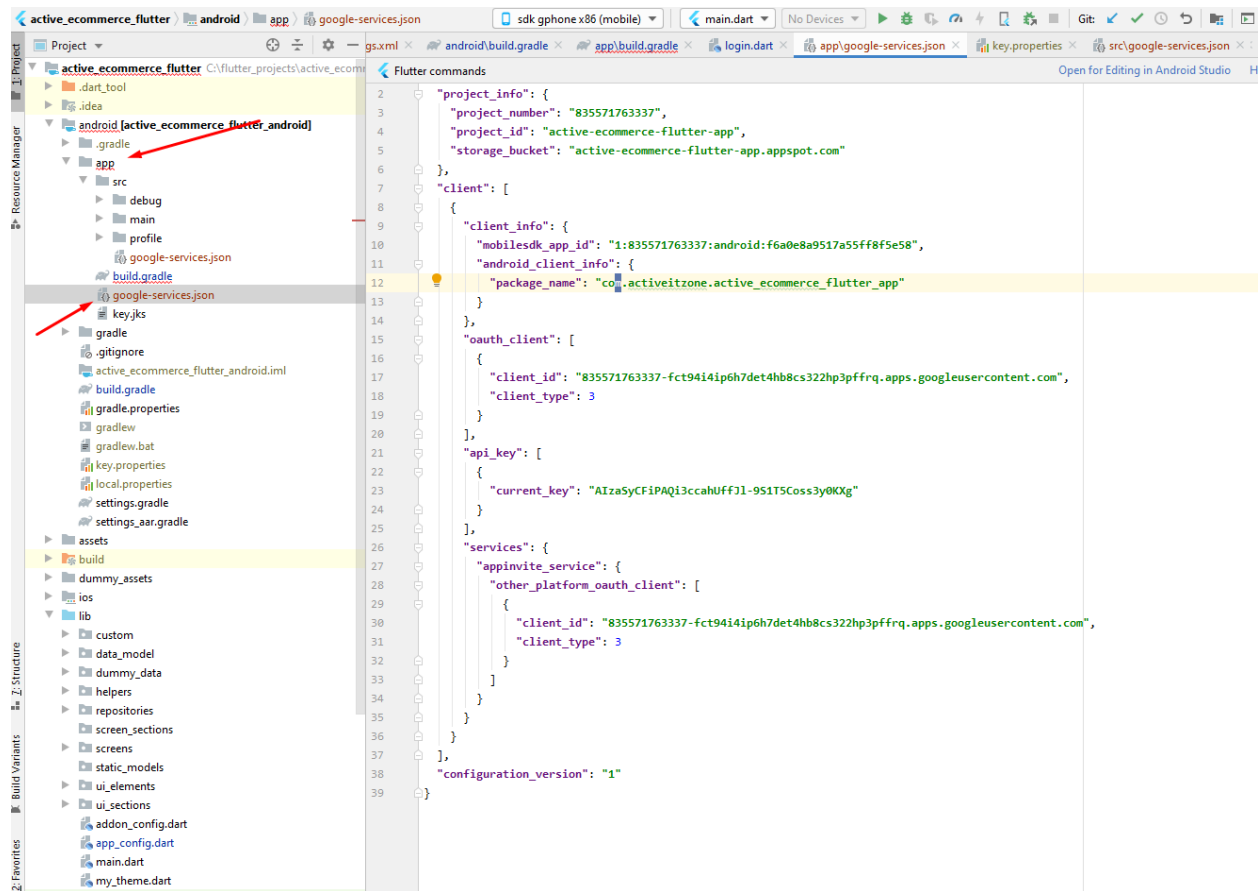
3. To get server key click on Cloud Messaging option



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

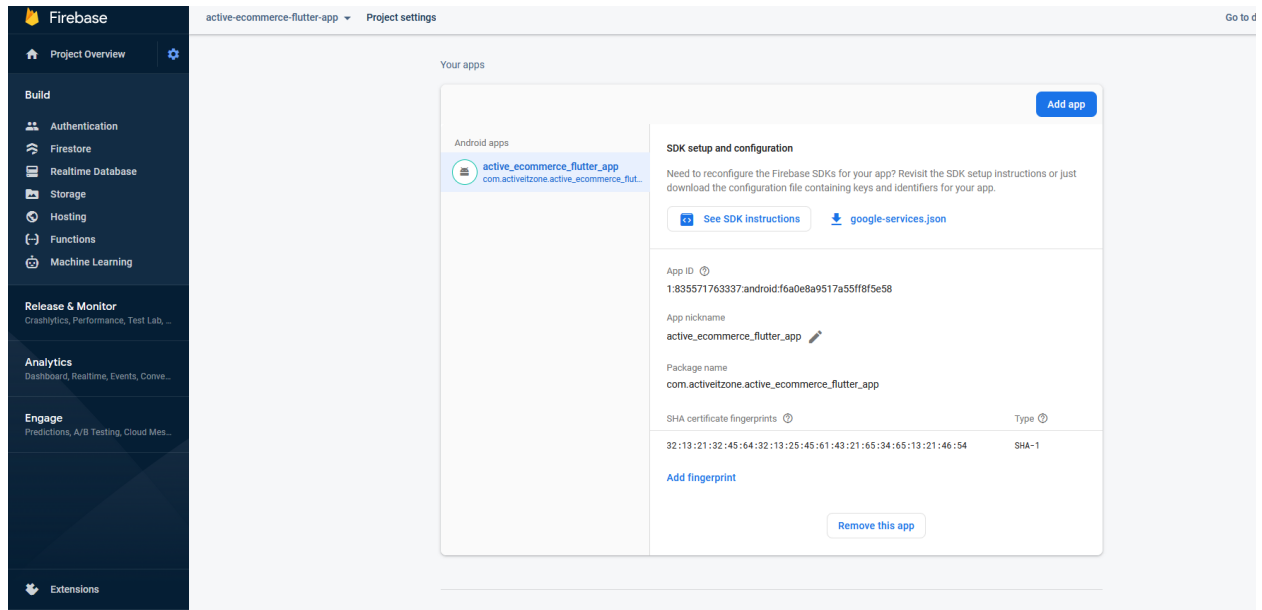


Firebase console:

<https://console.firebase.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)



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:CF:F8:9A:07:17:1F:9B:6E:14:8E:24:69:7C:EC:03:D5:2F
    SHA256: 92:09:F8:BF:56:F8:14:AB:AD:C8:C6:43:1D:79:FA:3F:66:2E:08:2D: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>

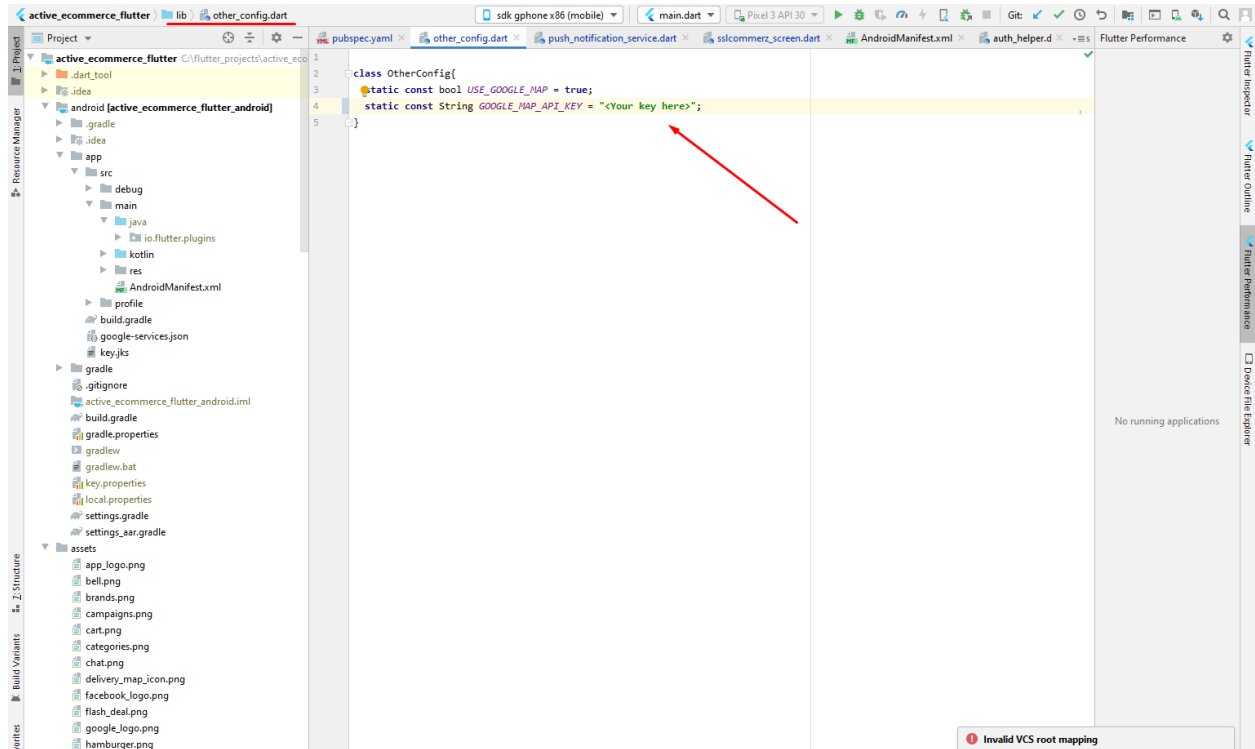
8. For ios follow this <https://firebase.google.com/docs/cloud-messaging/ios/client>

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.

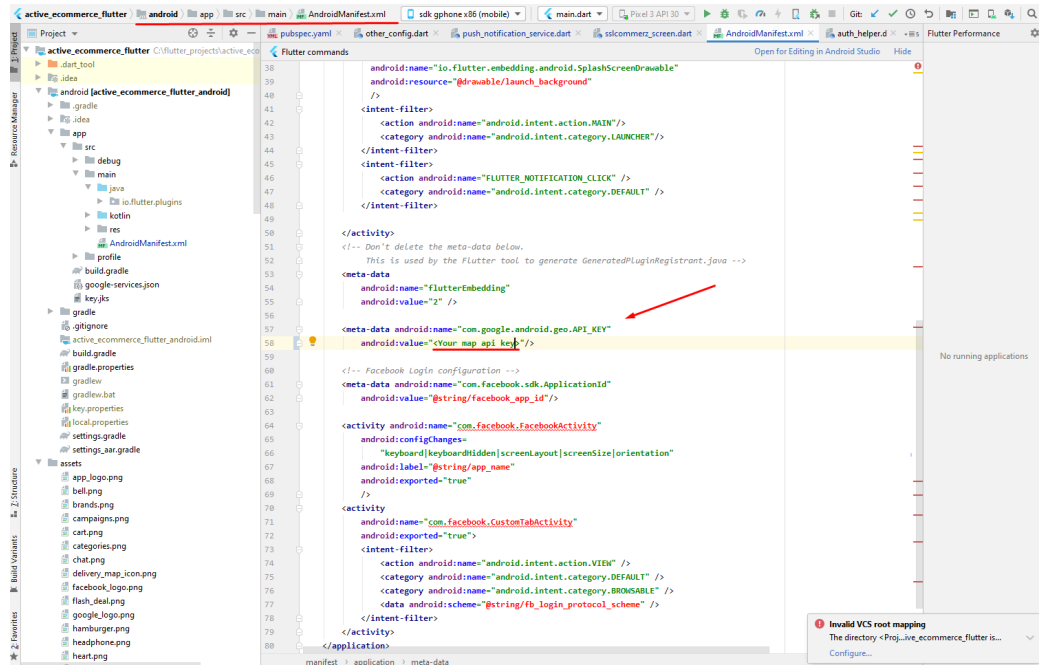
12. How to configure google map? (Read the whole thing before implementing)

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



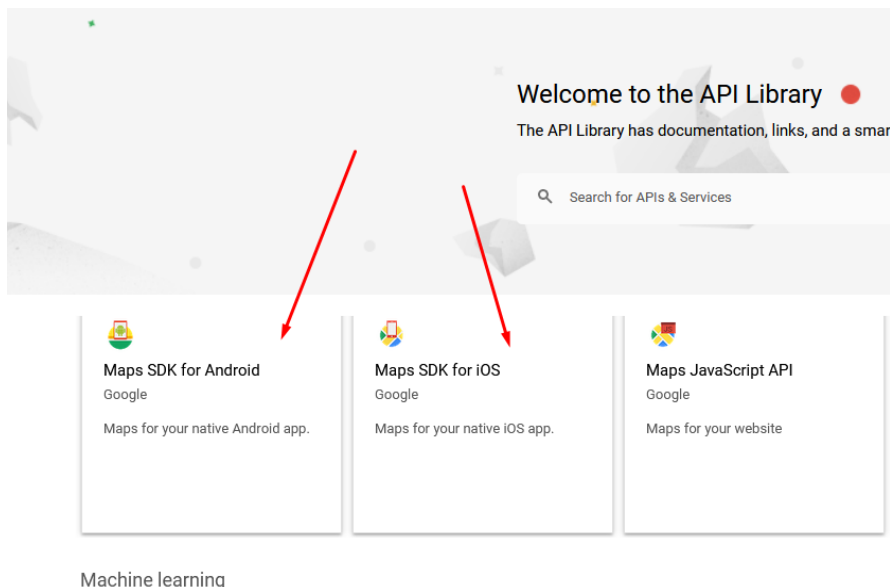
2. In main AndroidManifest.xml put the map api key

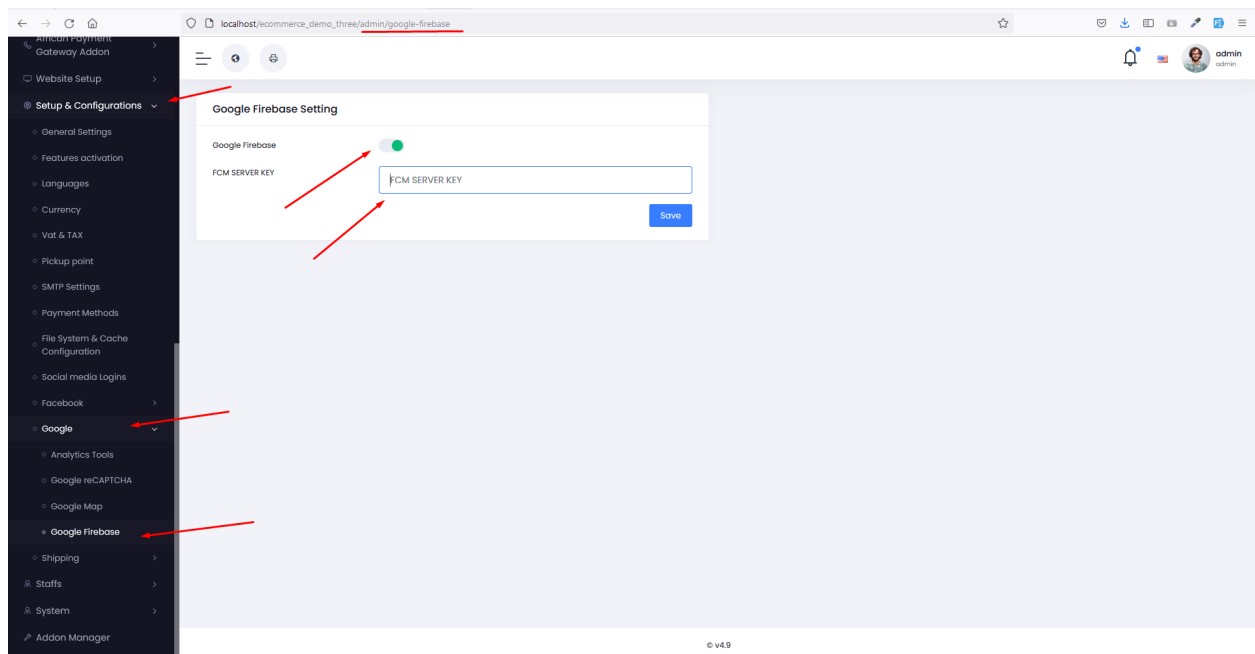


3. For ios follow this

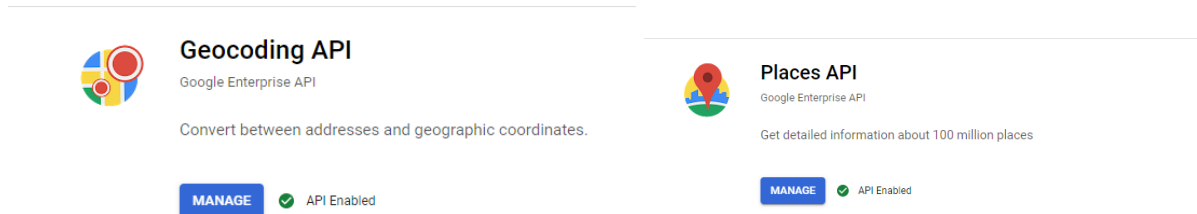
<https://blog.logrocket.com/adding-google-maps-to-a-flutter-app/#addinggooglemapstoflutterios>

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



13. 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;
}
```

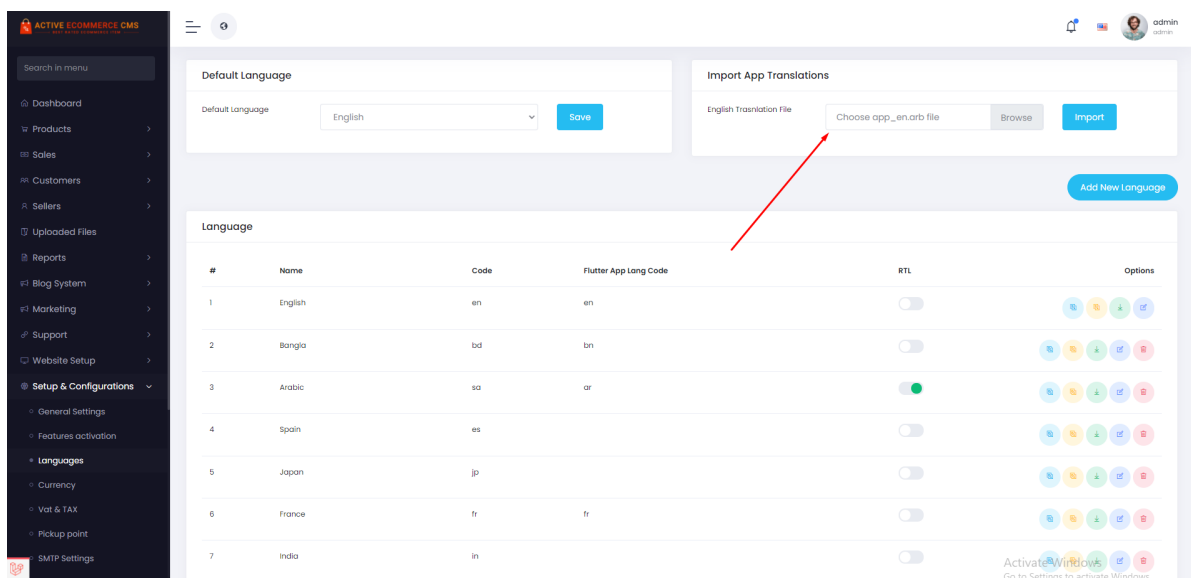
This value you can find in your admin panel. Go to your admin panel->setup & configurations->languages.

The screenshot displays the 'Active eCommerce CMS' admin interface. On the left is a dark sidebar with a search bar and a menu containing items like 'Dashboard', 'Products', 'Sales', 'Customers', 'Sellers', 'Uploaded Files', 'Reports', 'Blog System', 'Marketing', 'Support', 'Offline Payment System', 'Club Point System', 'Website Setup', and 'Setup & Configurations'. The 'Setup & Configurations' section is expanded, showing 'General Settings' and 'Features activation'. The main content area has a top bar with a 'Clear Cache' button and a user profile 'Admin admin'. Below this, there are two panels: 'Default Language' with a dropdown set to 'English' and a 'Save' button; and 'Import App Translations' with a file upload section. The primary section is a 'Language' table. It has columns: '#', 'Name', 'Code', 'Flutter App Lang Code', 'RTL', and 'Options'. Three languages are listed: English, Bangla, and Arabic. Red annotations highlight the 'Code', 'Flutter App Lang Code', and 'RTL' columns for each language row. For English, Code is 'en' and Flutter App Lang Code is 'en'. For Bangla, Code is 'bd' and Flutter App Lang Code is 'bn'. For Arabic, Code is 'sa' and Flutter App Lang Code is 'ar'. The RTL column shows toggle switches: off for English and Bangla, and on (green) for Arabic.

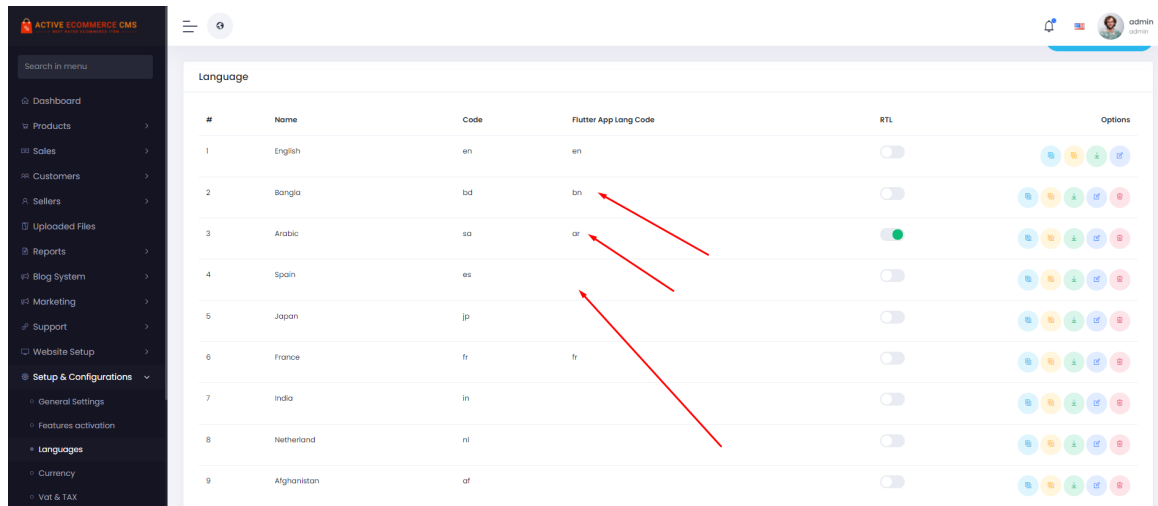
#	Name	Code	Flutter App Lang Code	RTL	Options
1	English	en	en	<input type="checkbox"/>	[Icons]
2	Bangla	bd	bn	<input type="checkbox"/>	[Icons]
3	Arabic	sa	ar	<input checked="" type="checkbox"/>	[Icons]

14. 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.



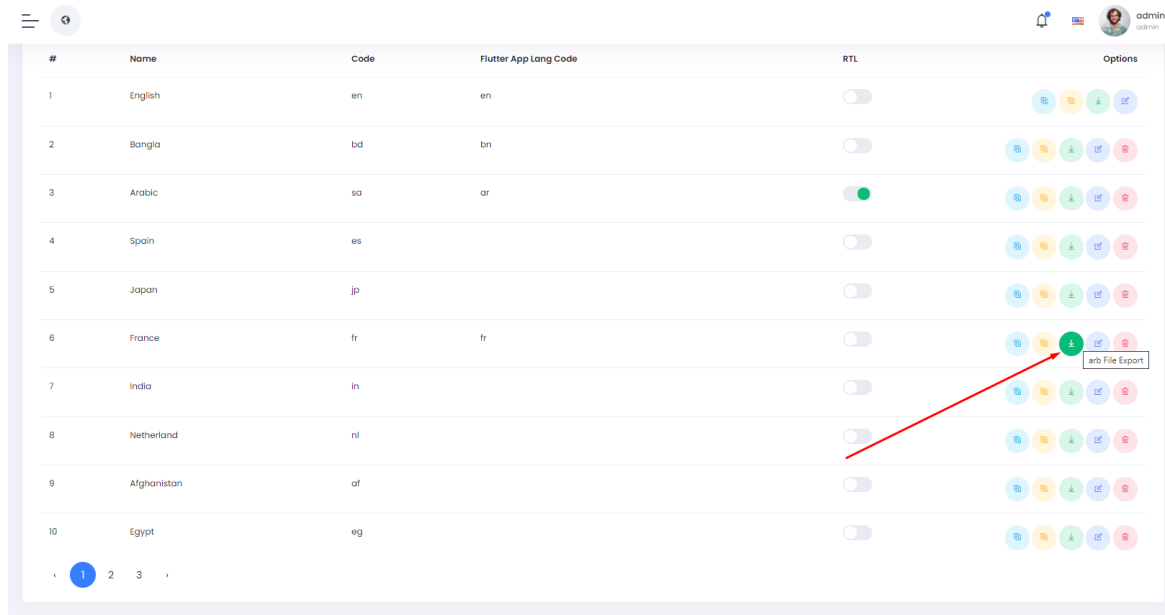
5. Make sure while adding/editing a language, your flutter app language code exists. The code must be in iso 639-1 format. Without a valid code, you will not see a translated output in the app.



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"/>	Arb File Export
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"/>	
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.

15. 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.