

Technology Used

- **Flutter** with Dart Language for Mobile App
- **Laravel** for Admin Panel & API
- **MySQL** for Database

Please follow the below steps to set up the project on your server.

(We have provided the steps to set up using Visual Studio Code Editor. You can use other editors also. Steps may vary based on your editor.)

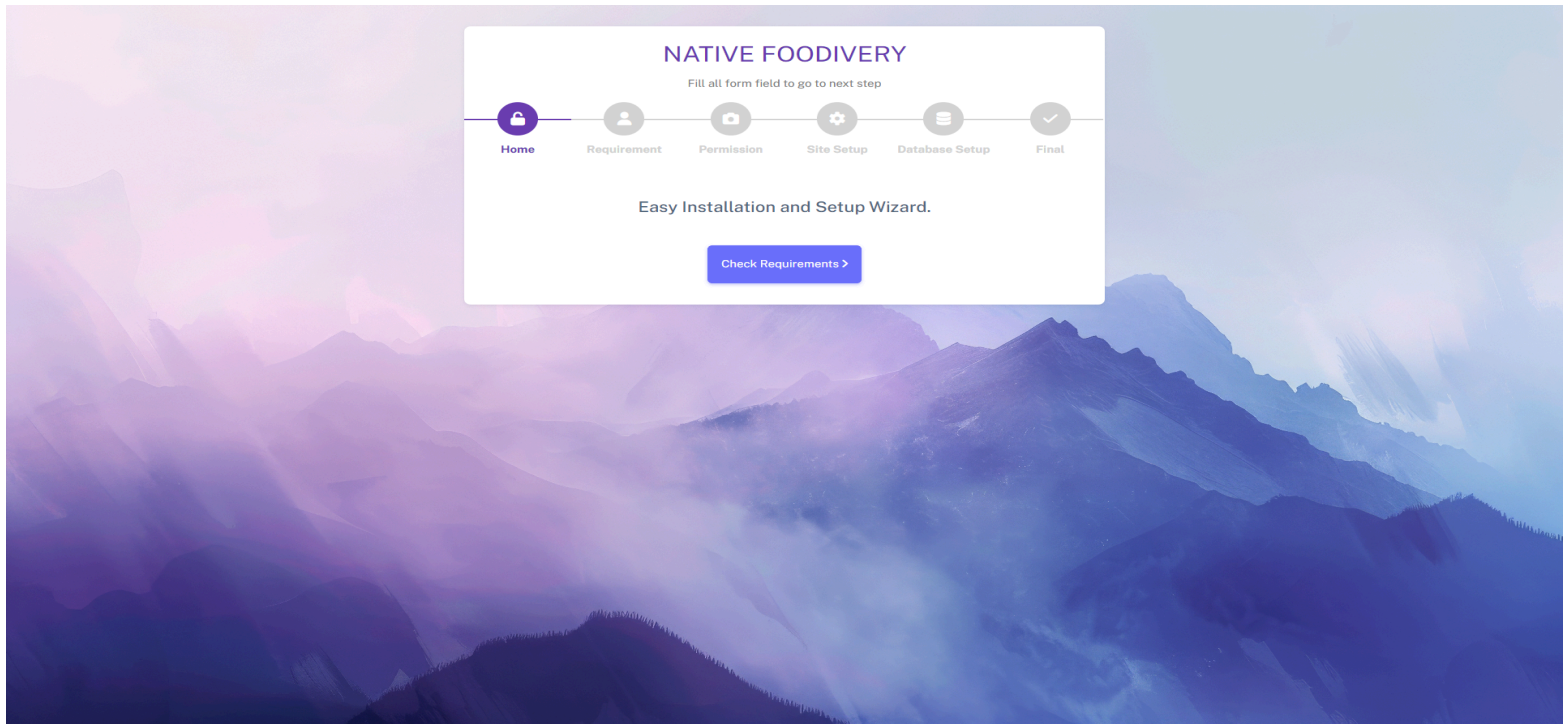
1. Set up Prerequisite (If not available)

- Install Visual Studio Code (VSCode) from [this link](#)
- Install Composer from [this link](#) with the latest version. (Minimum version 2.7.7)
- Install PHP(xampp) from [this link](#) with the latest version. (Minimum version 8.3.2)
- Install Laravel from [this link](#) (Minimum version 10 -11)
- Install and set up Flutter from [this link](#) (Minimum version 3.4.3)
- Install MySQL from [this link](#)
(You can choose the MySQL edition based on your needs)
- Install MySQL Workbench from [this link](#) (This is optional)

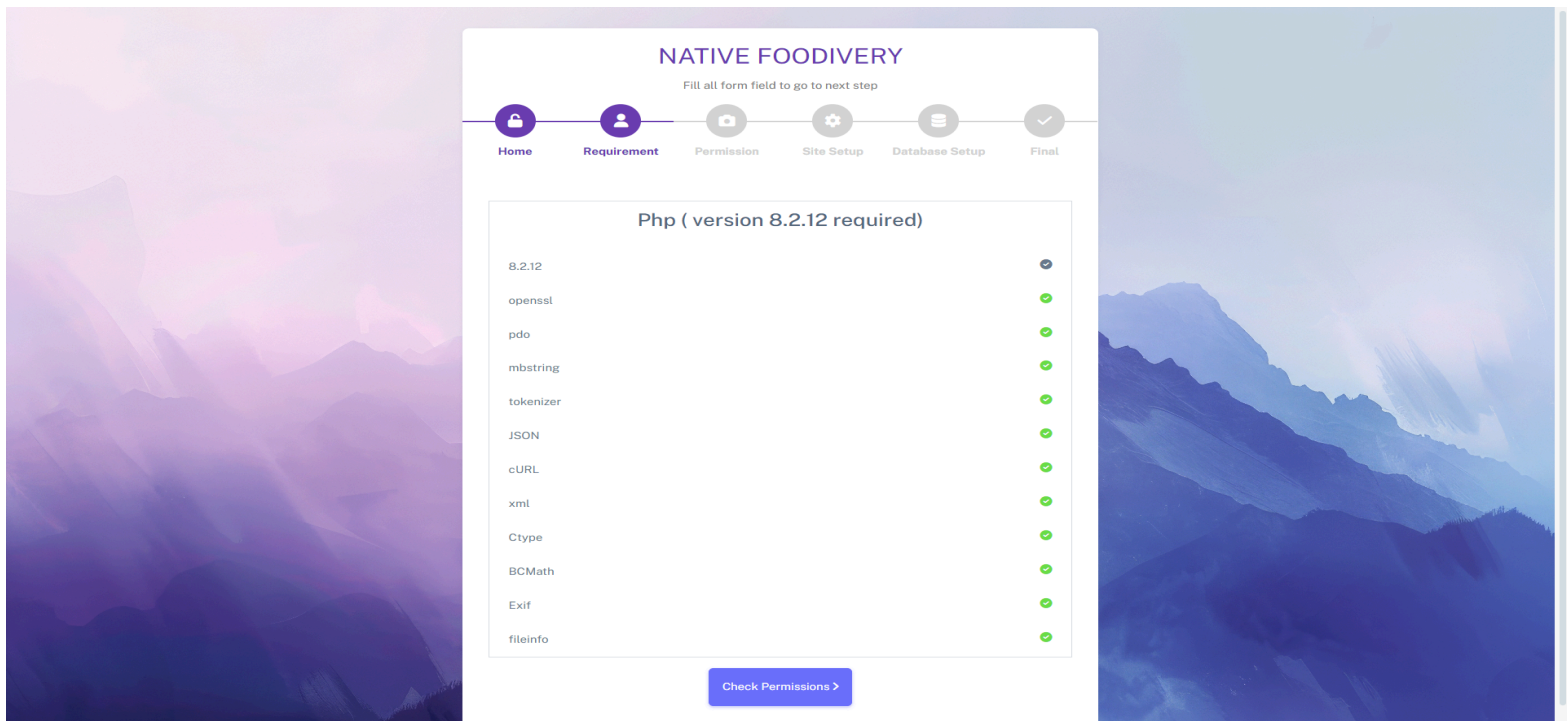
2. Set up the Database (Technology MySQL)

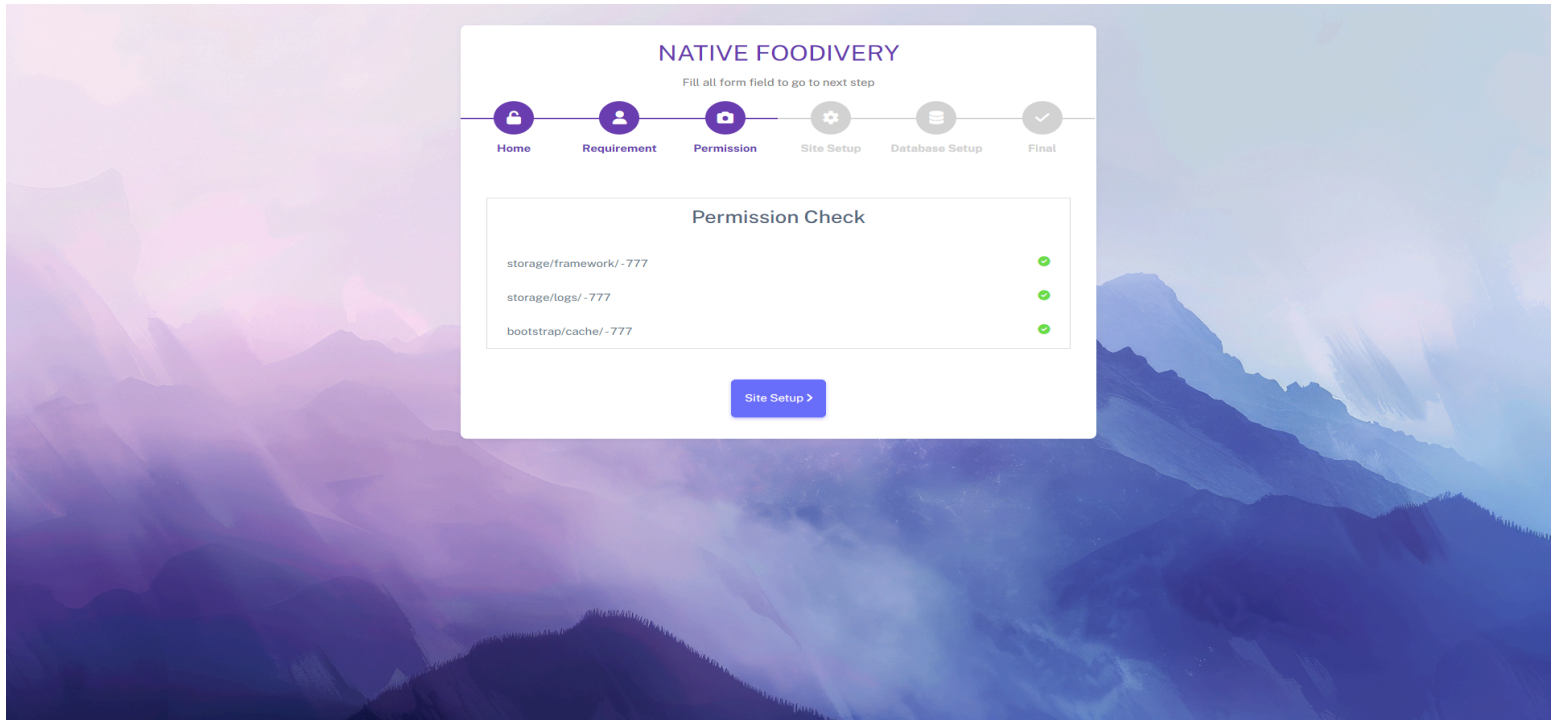
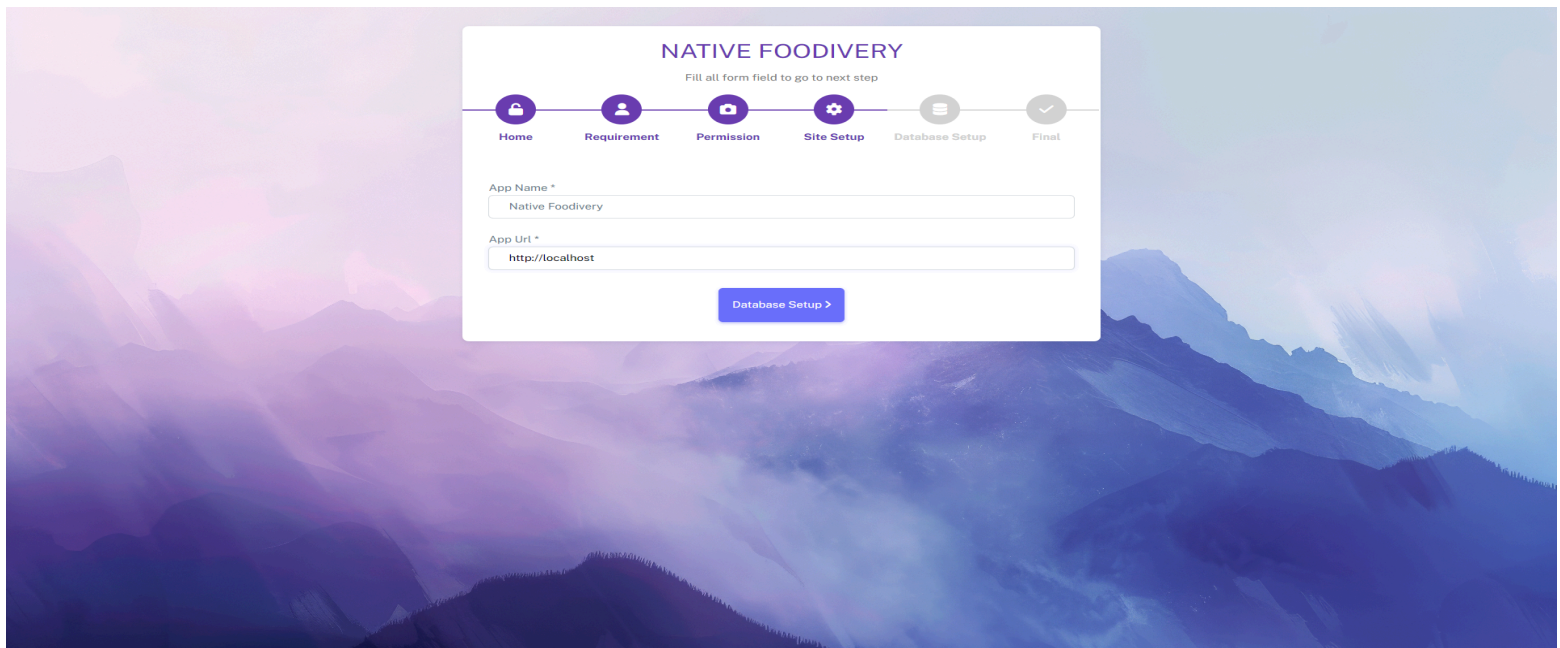
- Use your domain to setup database. (https://<YOUR_DOMAIN>)

a. Start Installation



b. Check Prerequisites Requirements.



c. Check Folder Permission for storing images,files.**d. Site Setup(Your APP_NAME,APP_URL)**

e. Database Setup

NATIVE FOODIVERY
Fill all form field to go to next step

Home Requirement Permission Site Setup **Database Setup** Final

Database Host *
localhost

Database Port*
3306

Database Name *
admin_foodiveryapp

Database Username *

Database Password *

Final Setup >

f. Final Setup

NATIVE FOODIVERY
Fill all form field to go to next step

Home Requirement Permission Site Setup Database Setup **Final**

Login Information

Email: admin@gmail.com

Password: info@1234

Finish >

3. Set up the API & Admin Panel (Technology Laravel)

- a. Open VSCode
- b. Open the **AdminPanel** folder into the VSCode from the ZIP file.
- c. Please execute the following command in the VSCode terminal to update the packages as per the package.json file.

```
composer update
```

Please make sure the installation gets completed successfully without any errors.

- d. To run an API please execute the following command

```
php artisan serve
```

NOTE:

- To run an API with your IP Address and Port number use the below command
php artisan serve --host=<YOUR_IP_ADDRESS>:8000
- For Accessing API use url

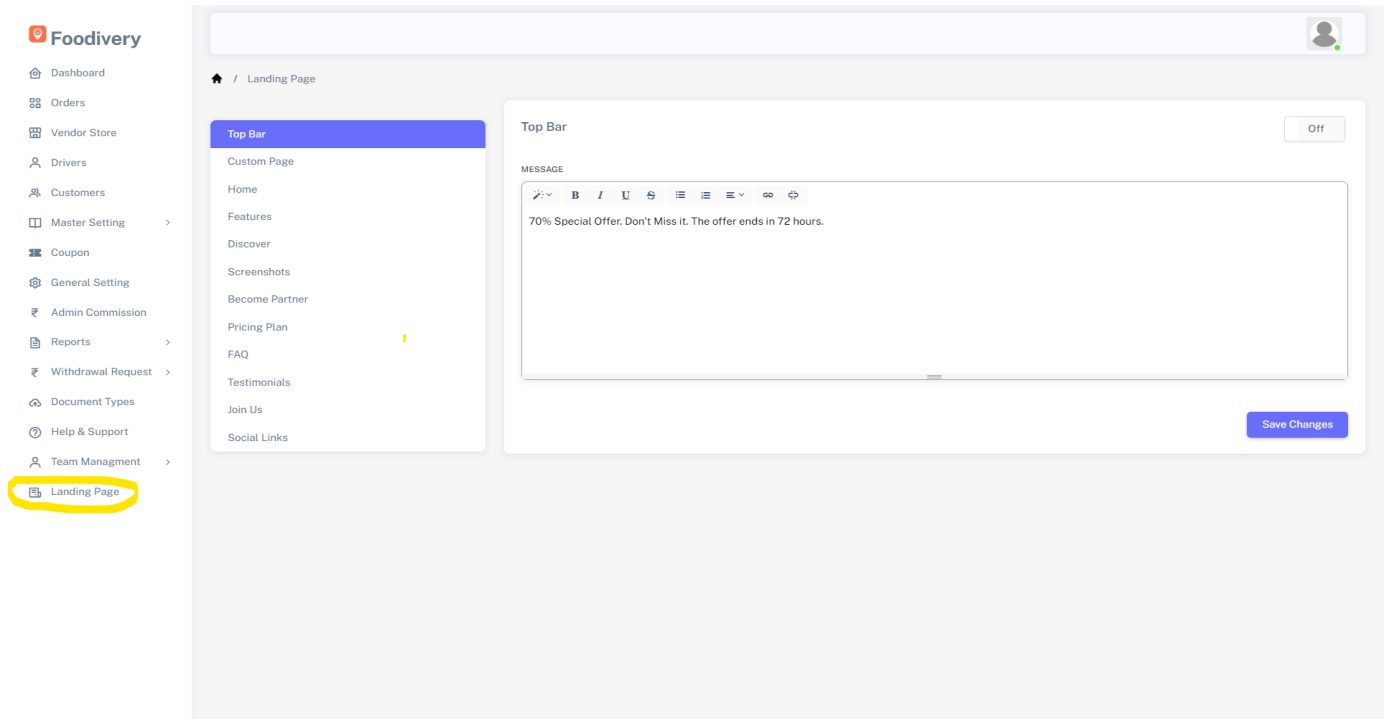
```
https://<YOUR_DOMAIN_NAME>/api
```

- For Accessing Admin use url

```
https://<YOUR_DOMAIN_NAME>/admin
```

4. Setup Landing Page of your site

- a. Login to admin panel using admin credential
- b. Update Landing page content using admin panel

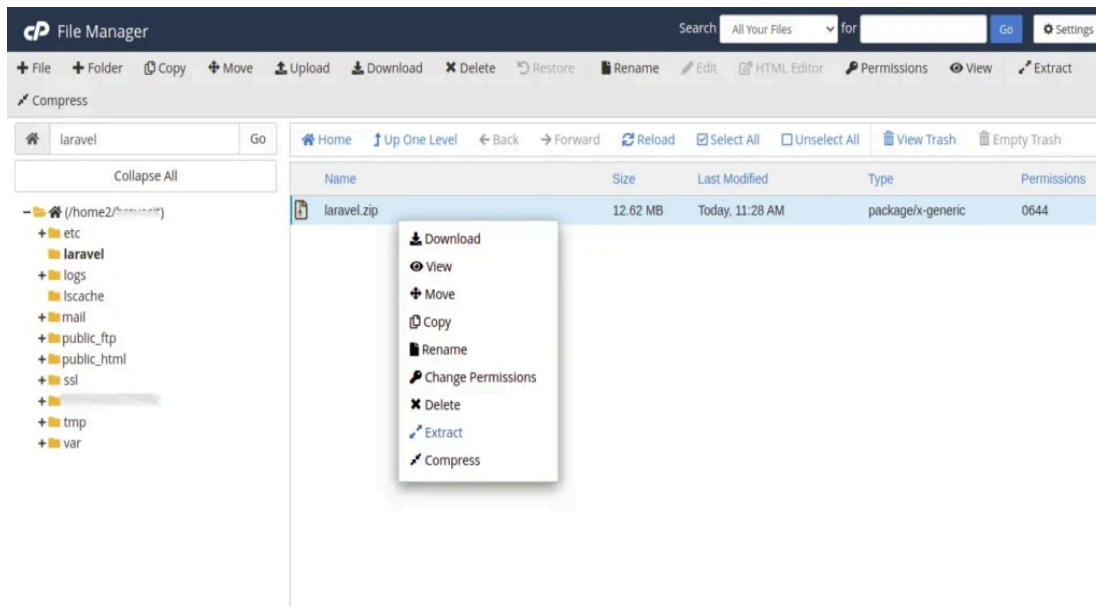


a. To set the API & Admin Panel on the Live Server

You can host on whichever hosting server is available to you, here we have provided steps for two hosting servers.

i. **Set up Cpanel**

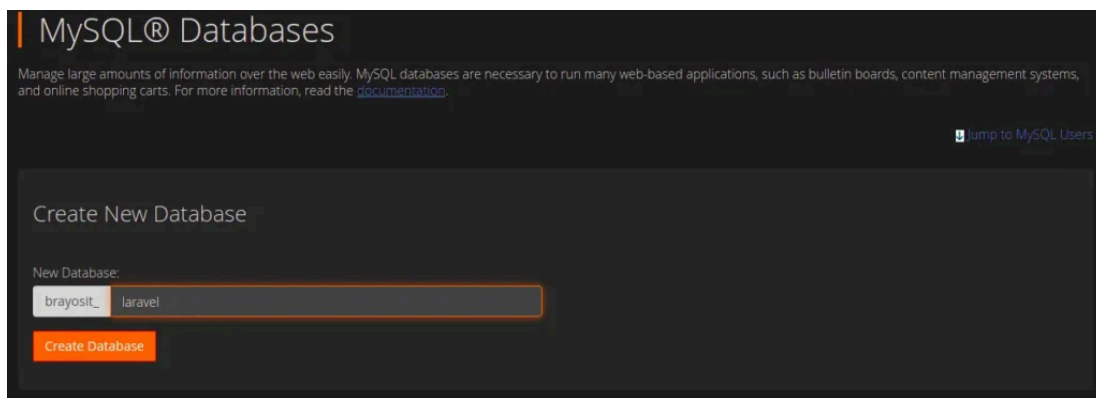
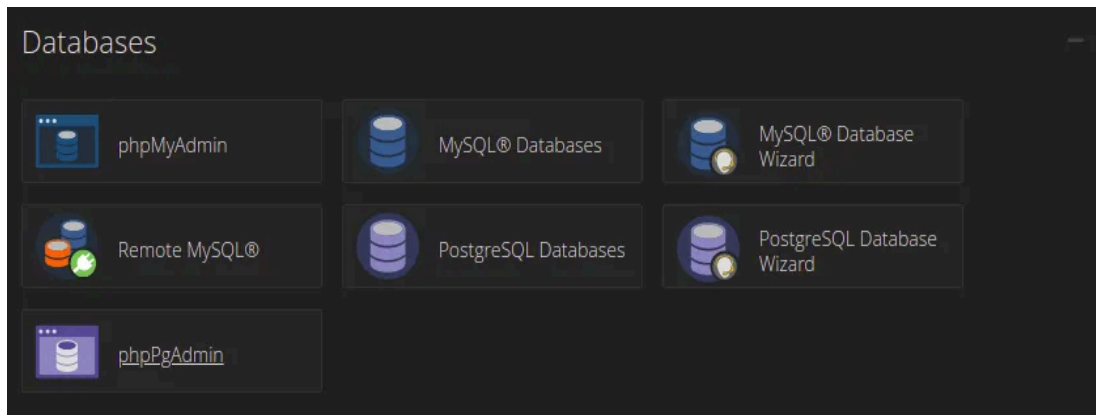
1. Log in to your shared hosting cPanel account.
2. Open the "File Manager" in cPanel.
3. Create a folder "YOUR_PROJECT_NAME".
4. Extract your project zip in this folder(Make zip except for public folder)



5. Upload the public folder inside the 'Public-Html' folder
6. Open index.php

```
1  * /
2
3  $app = require_once __DIR__.'./<YOUR-PROJECT-NAME>/bootstrap/app.php';
4
5  $kernel = $app->make(Kernel::class);
6
7  $response = $kernel->handle(
8      $request = Request::capture()
9  )->send();
10
11  $kernel->terminate($request, $response);
```

7. Write your project name to every **require_once __DIR__**.
8. Next, go to MySQL and create a new database



9. Access your domain in a web browser to check if your app is running.

ii. Set up Plesk Panel

1. Log in to your shared hosting Plesk account.
2. Open Plesk and navigate to the 'Add Domain' button. From the drop-down menu that appears, select 'Laravel site' to begin establishing a new domain.

Adding New Domain



Choose a way of creating a website



Blank website

A starter page for your HTML or PHP site (the previous default)



Upload files

From a local machine



Deploy using Git

Pull the files from a Git repository



WordPress site

A website running on the latest WordPress version



Import website

From another hosting server



Laravel site

Create a new website by installing a Laravel application



Node.js application

A website running on Node.js

Cancel

3. If you don't have a domain name for your website yet, you can use a temporary domain name.

Select your domain name



Registered domain name

I already have a registered domain name





Temporary domain name

I don't have a registered domain name yet

4. Click on the temporary domain name and you will get a domain name as shown

Select your domain name

 **Registered domain name**
I already have a registered domain name

 **Temporary domain name**
I don't have a registered domain name yet

Your temporary domain name
brave-lewin.103-150-136-240.plesk.page

Temporary domain names are used for accessing your site before you register and purchase a
[Show more](#)

☐ Assign this domain to a customer

Webspace *

Create a new subscription

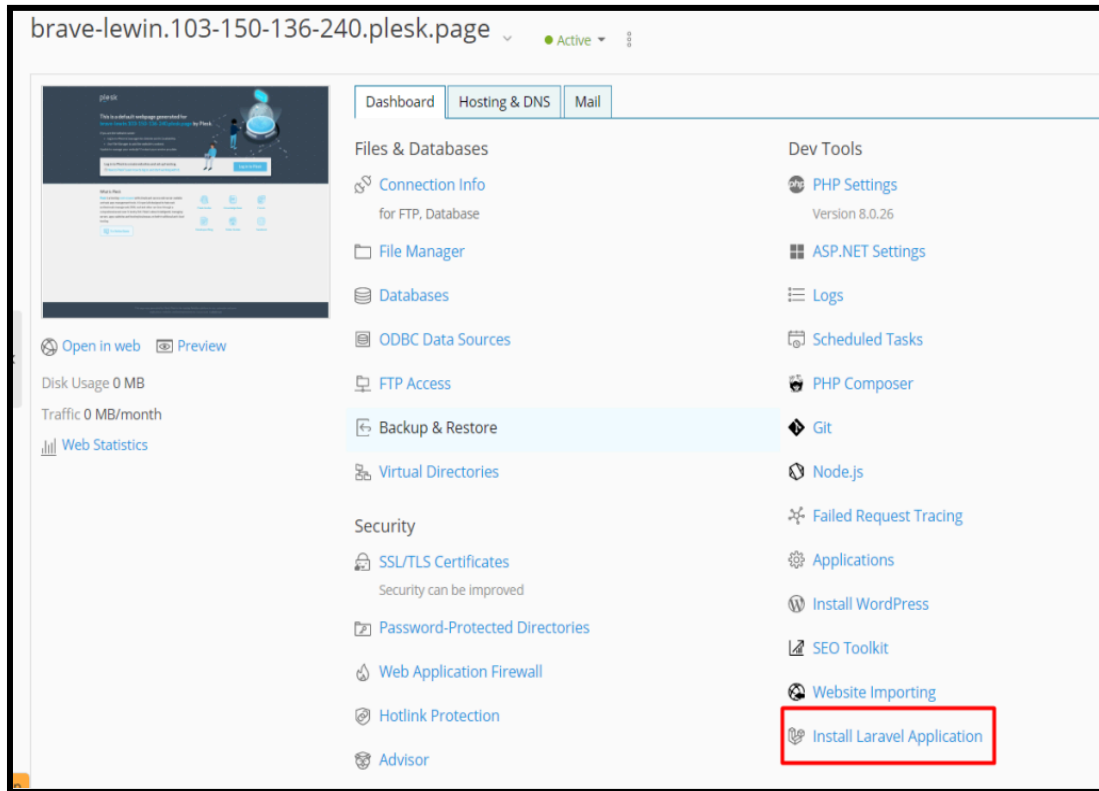
^ Webspace settings

5. Click Add Domain and you will see the below screenshot

Creating domain brave-lewin.103-150-136-240.plesk.page

- ✓ Creating a subscription
- ✓ Adding a domain
- Creating physical hosting
- Configuring PHP

6. After the website has been created, Plesk will prompt you to select between installing a default skeleton Laravel application or pulling the application from a remote Git repository. Your selection will be saved automatically. Installing the skeleton is the best way to get acquainted with the Laravel Toolkit's capabilities for the time being.



7. Wait for a few seconds.

Deploying project...

✔

Creating a Git repository

Done

- Creating Laravel skeleton
- Committing to the Git repository

Queued
- Installing Composer dependencies

Queued
- Configuring Laravel application

Queued
- Configuring hosting settings

Queued
- Configuring PHP environment

Queued
- Preparing Node.js environment

Queued

8. And it is ready

Laravel application for [brave-lewin.103-150-136-240.plesk.page](#)

Dashboard

Artisan

Composer

Node.js

Deployment

Application Info

URL
[brave-lewin.103-150-136-240.plesk.page](#)

Repository
[https://brave-l_0tzxe0hhj97c@brave-lewin.103-150-136-240.plesk.page/plesk-git/laravel_e4b34a](#)

Last commit

Date: Fri Dec 30 09:33:33 2022 +0530

Laravel skeleton

Settings

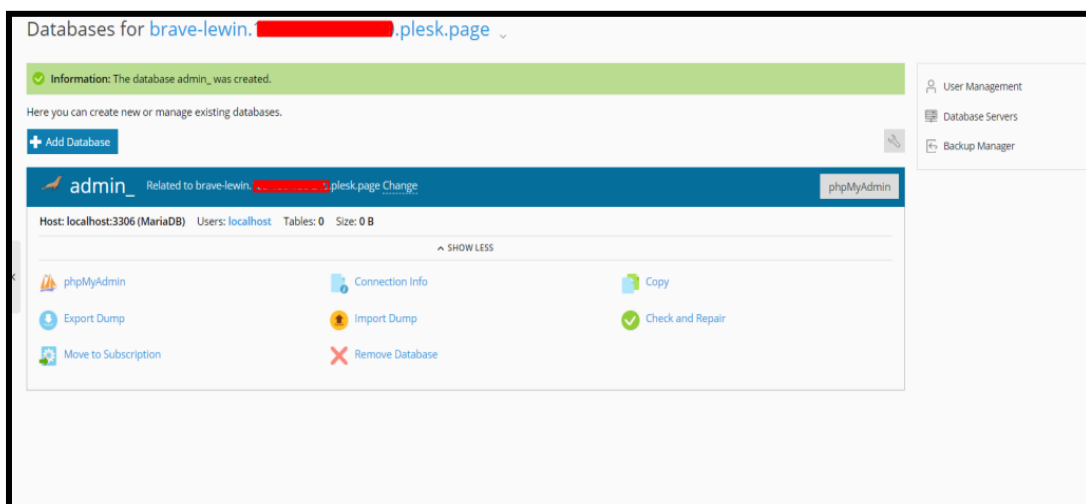
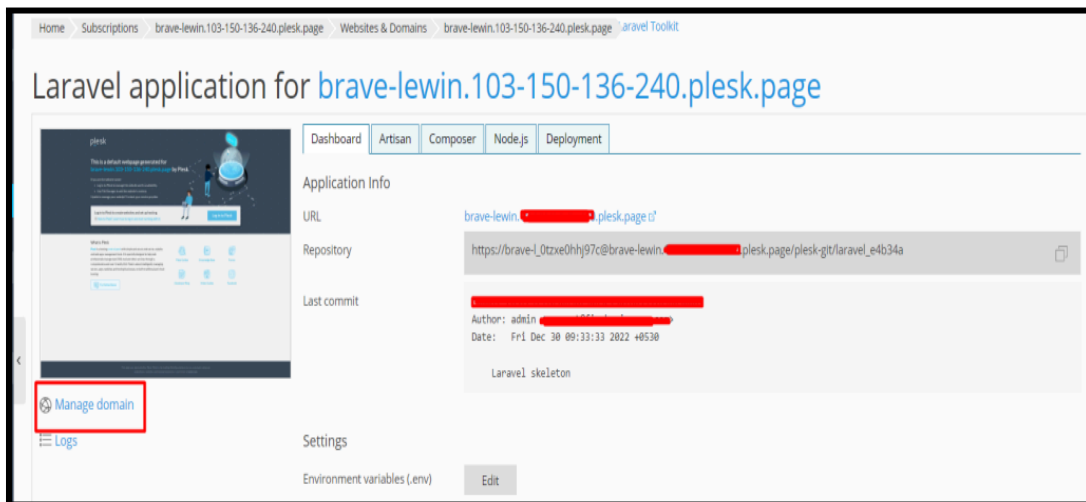
Environment variables (.env)

Edit

Maintenance mode

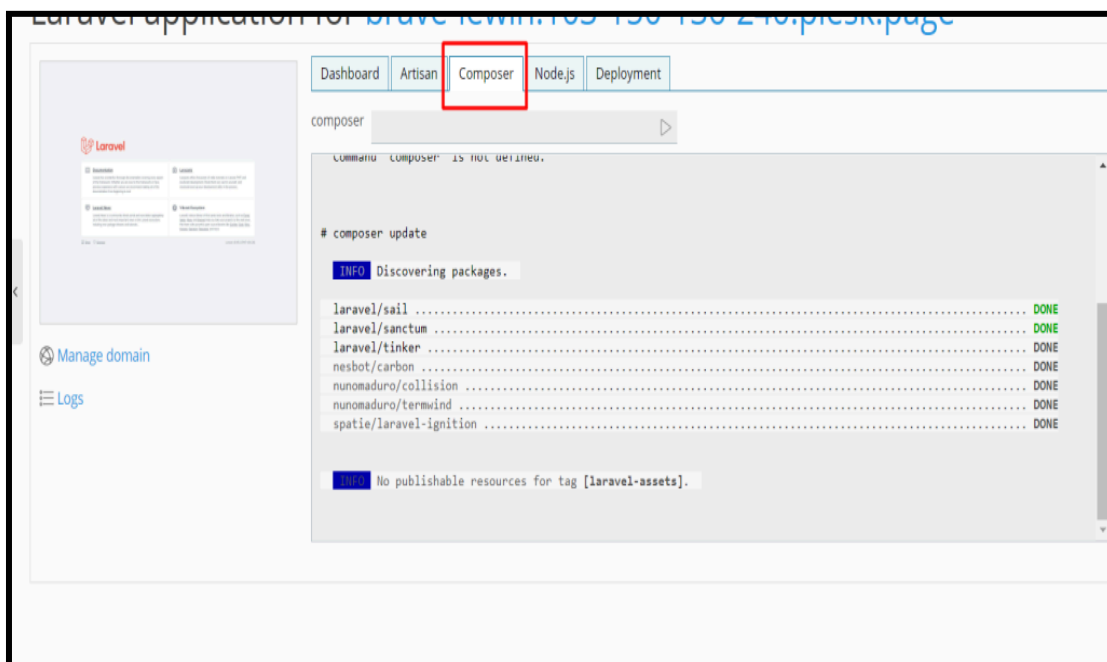
Disabled

9. Your very first Laravel application has been published on the World Wide Web!
10. Navigate to the Laravel dashboard labeled "Information," then click the "Manage domain" button. Following this link will bring you to the "Hosting" dashboard.
11. After arriving at that location, select "Databases," and then after that, "Add Database".



12. When you are finished creating the database, you will need to modify the configuration file for the Laravel application so that it contains information about how to connect to the newly created database.

13. Click the button labeled "Manage Laravel Application" located on the "Hosting" dashboard. Clicking on this link will take you to the Laravel card.
14. However, there is more to come! In particular, let's look at some additional features that save you time when hosting Laravel applications.
15. It is important to point out that you can rapidly invoke the 'composer' and 'npm' commands by simply selecting them from the list of preloaded * commands, which I discovered to be an indispensable resource:



-
16. If your Laravel application makes use of Laravel Task Scheduling, then in addition to the typical 'artisan schedule: list' command, you can quickly review all scheduled jobs by switching to a different tab.

b. Set up Payments

- i. Set up Razorpay payment gateway in the project (Anyone payment gateway is required) please follow **6. a**
- ii. Set up Stripe payment gateway in the project (Anyone payment gateway is required) please follow **6. b**.
- iii. Set up PhonePe payment gateway in the project (Anyone payment gateway is required) please follow **6. c**.

5. Set up Customer App (Technology Flutter)

a. Initial steps to set up and run mobile app

- i. Open the **CustomerApp** folder in the VSCode
- ii. Run the following commands in the VSCode Terminal

```
flutter clean  
flutter pub get
```

- iii. Additional steps to set up for iOS (You can skip these steps if you don't want to set up for iOS)

1. In the VSCode terminal, go to the ios directory

(using the command **cd ios**)

2. Run the following command to install pods

```
pod install
```

- iv. Connect your Android or iOS device with your machine

(To run on an Apple device, you must have an Apple computer)

- v. Run the following command to run on an Android or iOS device

```
flutter run
```

- vi. To upgrade the Flutter version run the following command in the VSCode Terminal

(Only if your Flutter version is lower than mentioned in this document)

flutter upgrade

b. Change API base URL

After the set up of your API and Admin panel, you have to change your API base URL, for that go to the file located at **lib\utils\global.dart**

```
String baseUrl = "https://foodivery.native.software/api";  
String imageUrl = "https://foodivery.native.software";
```

c. Change App Name

i. Change the app name in the Android App

1. Change the app name in the file located at **android/app/src/main/AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    android:versionCode="1" android:versionName="1.0">  
    <uses-permission android:name="android.permission.ACCESS_NETWORK_INTERNET">  
    </uses-permission>  
    <application  
        android:label="Foodivery"  
        android:icon="@mipmap/ic_launcher">  
        <activity  
            android:name=".MainActivity"  
            android:launchMode="singleTop"  
            android:theme="@style/LaunchTheme"  
            android:configChanges="orientation|screenSize|uiMode"  
            android:windowSoftInputMode="adjustResize">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN">  
                </action>  
                <category android:name="android.intent.category.LAUNCHER">  
                </category>  
            </intent-filter>  
        </activity>  
    </application>  
</manifest>
```

ii. Change the app name in the iOS App

1. In VSCode

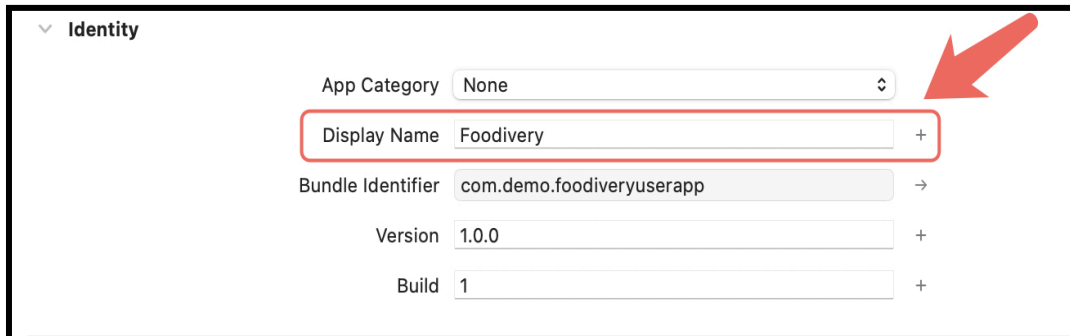
- a. Go to **ios/Runner/info.plist**
- b. Change string of key **CFBundleDisplayName**

```
<?xml version="1.0" encoding="UTF-8" ?>  
<plist version="1.0">  
    <dict>  
        <key>CFBundleIdentifier</key>  
        <string>com.example.foodivery</string>  
        <key>CFBundleDisplayName</key>  
        <string>Foodivery</string>  
        <key>CFBundleExecutable</key>  
        <string>Runner</string>  
    </dict>  
</plist>
```

2. In XCode

- a. Right-click on the **iOS** folder and Choose Open in Xcode Option

- b. Click on the folder icon left side of the XCode window
- c. Select Runner.
- d. Select Target runner
- e. From the General Tab Go to Identity
- f. Change Display Name

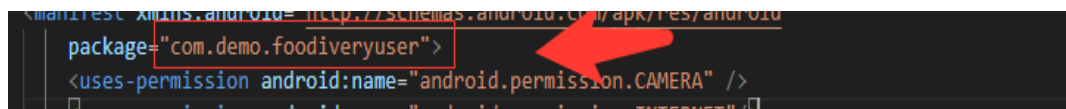


d. Change Package Name/Bundle ID

An app's package name is a unique identifier that is automatically created when you create an app. The term used for iOS apps is "**bundle ID**" and for Android apps, it is "**package name**"

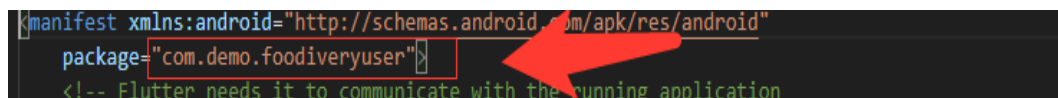
i. Set Package Name for Android App

1. Change the package name in the file located at **android/app/src/main/AndroidManifest.xml**



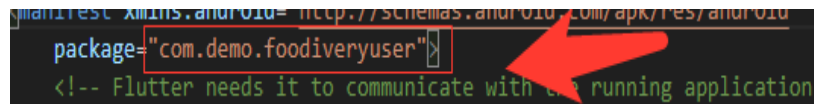
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.demo.foodiveryuser">
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.INTERNET" />
```

2. Change the package name in the file located at **android/app/src/debug/AndroidManifest.xml**



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.demo.foodiveryuser">
    <!-- Flutter needs it to communicate with the running application
```

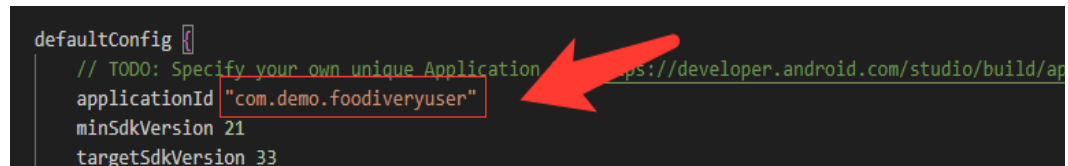
3. Change Package Name in file which is located at **android/app/src/Profile/AndroidManifest.xml**



```
manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.demo.foodiveryuser"
<!-- Flutter needs it to communicate with the running application
```

A red box highlights the package name "com.demo.foodiveryuser" in the build.gradle file. A red arrow points to the package name.

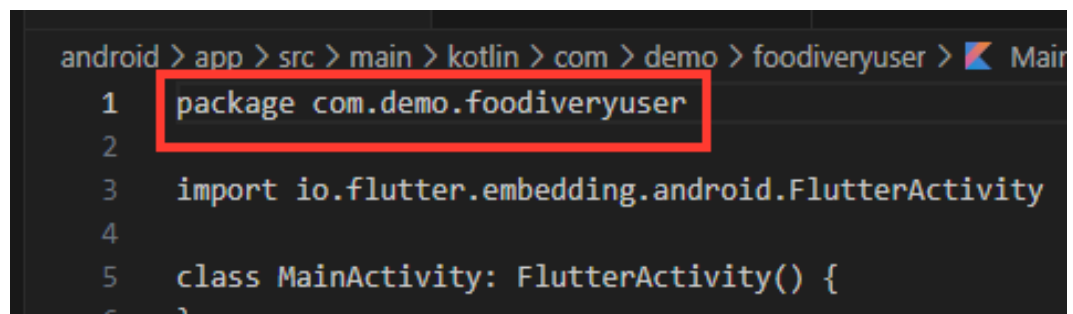
4. Change the Package Name in a file that is located at **android/app/build.gradle**



```
defaultConfig {
    // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/apk-application-id)
    applicationId "com.demo.foodiveryuser"
    minSdkVersion 21
    targetSdkVersion 33
}
```

A red box highlights the applicationId "com.demo.foodiveryuser" in the build.gradle file. A red arrow points to the applicationId.

5. Change the folder structure for the below path as per your package name. **android\app\src\main\kotlin\com\demo\foodiveryuser**
6. Change Package Name in the file which is located at **android\app\src\main\kotlin\com\demo\foodiveryuser\MainActivity.kt**



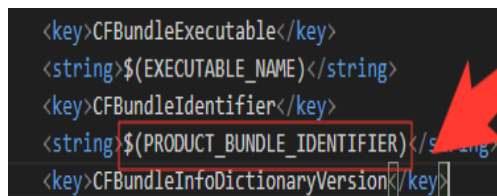
```
android > app > src > main > kotlin > com > demo > foodiveryuser > MainActivity.kt
1 package com.demo.foodiveryuser
2
3 import io.flutter.embedding.android.FlutterActivity
4
5 class MainActivity: FlutterActivity() {
6 }
```

A red box highlights the package name "com.demo.foodiveryuser" in the MainActivity.kt file. A red arrow points to the package name.

ii. Set Bundle ID for iOS App

1. In VSCode

- a. Go to **ios/Runner/info.plist**
- b. Change the string of key **CFBundleIdentifier**



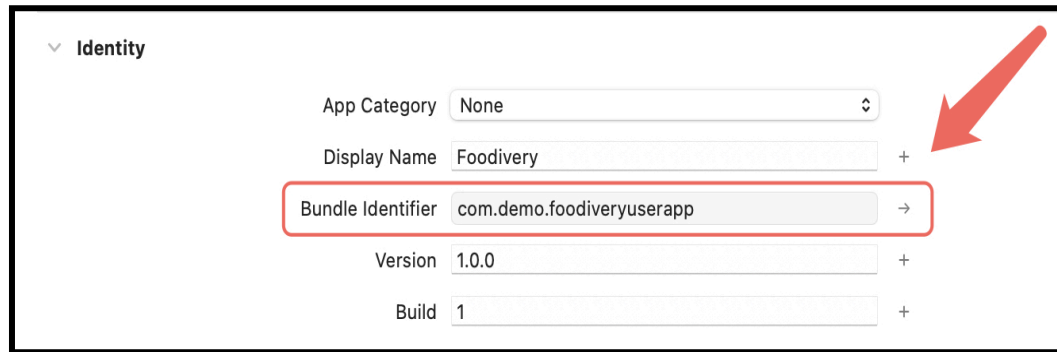
```
<key>CFBundleExecutable</key>
<string>$(EXECUTABLE_NAME)</string>
<key>CFBundleIdentifier</key>
<string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
<key>CFBundleInfoDictionaryVersion</key>
```

A red box highlights the CFBundleIdentifier key in the info.plist file. A red arrow points to the CFBundleIdentifier key.

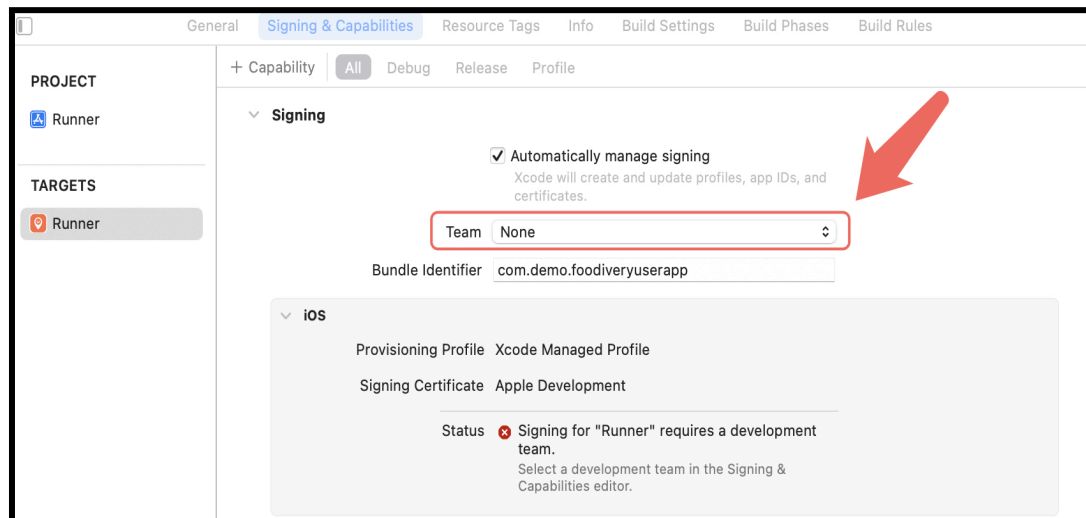
2. In XCode

- a. Right-click on the **iOS** folder and Choose Open in Xcode Option
- b. Click on the folder icon left side of the XCode window

- c. Select Runner.
- d. Select Target runner
- e. In general, Tab Go to identify
- f. Change Bundle Identifier



- g. In Signing & Capabilities Go to Signing
- h. Change Bundle Identifier



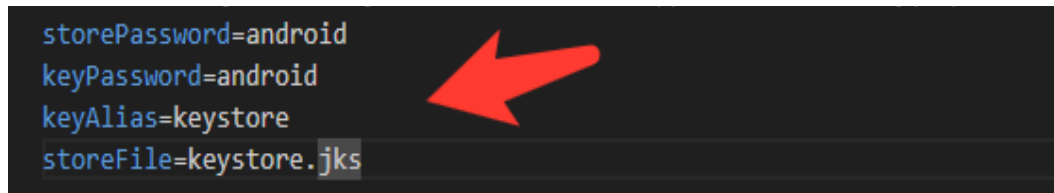
e. Create and set Keystore file for Android

- i. Create a Keystore.jks file if not exist using the below command in the terminal

```
keytool -genkey -v -keystore "path\keystore.jks" -storetype JKS  
-keyalg RSA -keysize 2048 -validity 10000 -alias keystore
```

- ii. Fill in all the details asked while executing the above command

- iii. Recommended. After creating your keystore.jks file, please put it in the **android/app** folder
- iv. Create a key.properties file in the **Android** folder and add the details in the file as per the below screenshot.



```
storePassword=android
keyPassword=android
keyAlias=keystore
storeFile=keystore.jks
```

A screenshot of a code editor showing the contents of a key.properties file. The text is: storePassword=android, keyPassword=android, keyAlias=keystore, storeFile=keystore.jks. A large red arrow points from the right towards the file name 'keystore.jks'.**NOTE:**

- If you have changed any default value for any of these keys (storePassword, keyPassword, keyAlias, storeFile) while creating the keystore.jks file, then please also change them to the key.properties file.
- If you place your keystore.jks file somewhere else in the project than mentioned in step 5.c.iii then please change storeFile key value accordingly.
- For more details please refer to [this link](#)

f. Create Firebase Account & Project

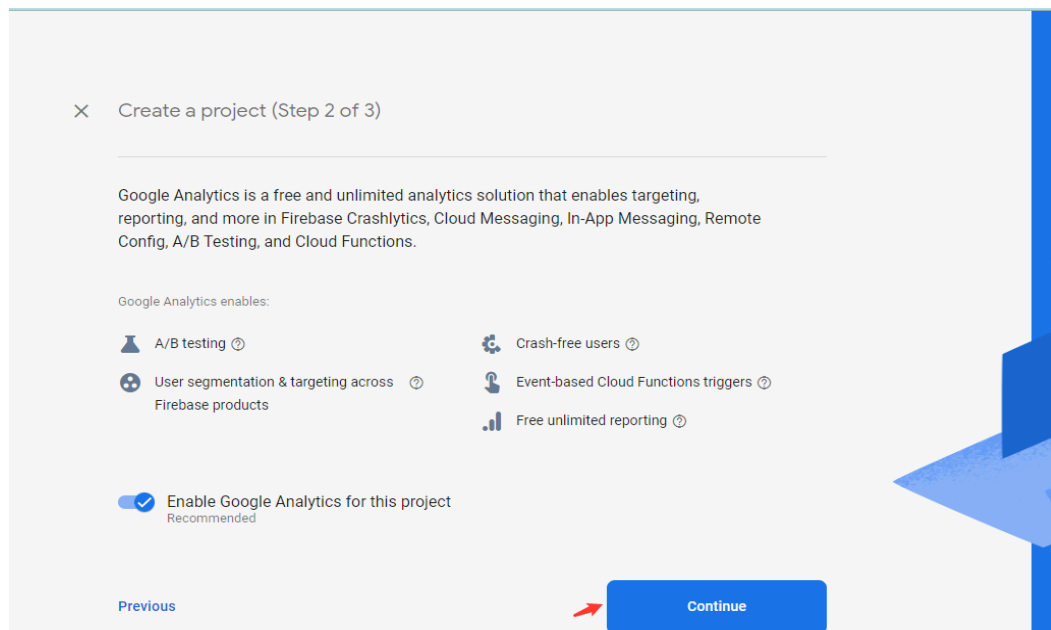
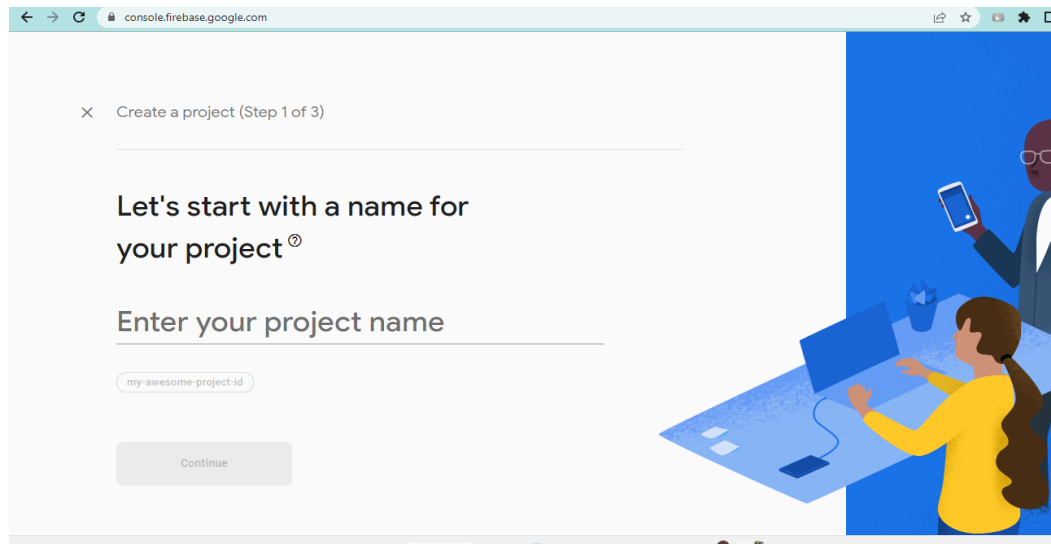
In this project, we are using the following Firebase services.

- i. Push Notification
- ii. Phone Authentication
- iii. Firebase Analytics
- iv. Firebase Firestore
- v. Firebase Dynamic Link

For this, you need a Firebase account and a project set up in the Firebase. Please follow the below steps for this,

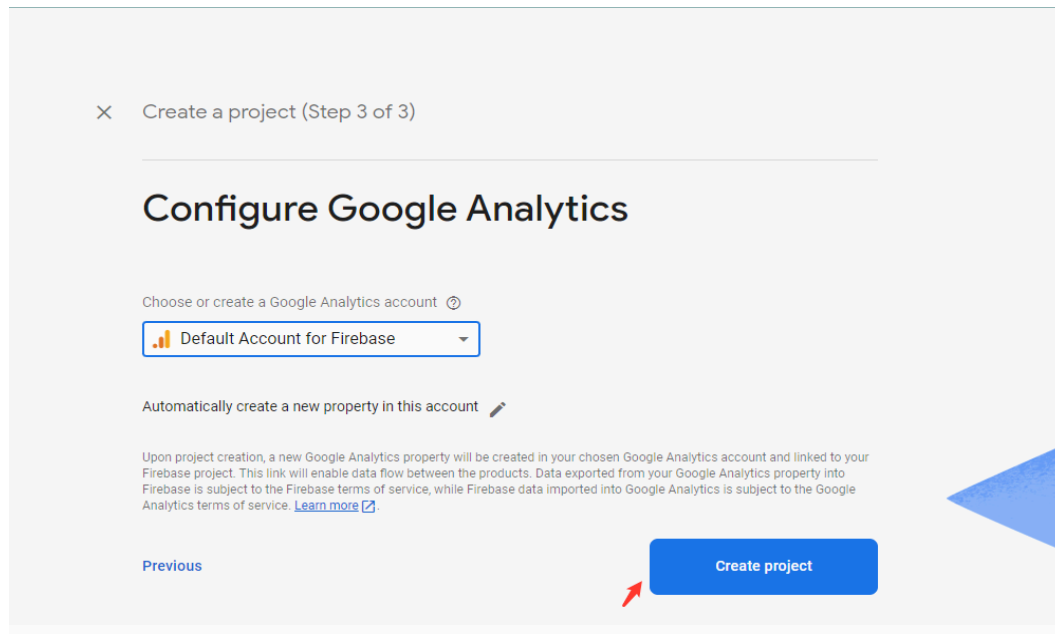
- i. Go to the [Firebase console](#)
- ii. Sign up if you don't have a Google Account or want to create a new account for your project. Otherwise, sign in with your Google Account.
- iii. Click on **Add Project**

iv. Enter your project name



v. Select Default Account for Firebase

(or you can create a new account)



vi. Create project

g. Set up Android App in Firebase Project

- i. Go to the Firebase console
- ii. Select the project you created in step 5.f.vi.
- iii. Go to **Project Setting**
- iv. In the **General** Tab click on the **Add App** button
- v. Select **Android**
- vi. Fill out the form and click on the **Register App Button**

(Please check the below screenshot for reference)

- vii. You need SHA keys (SHA-1 and SHA-256) to add once you create the Android App in the above steps.

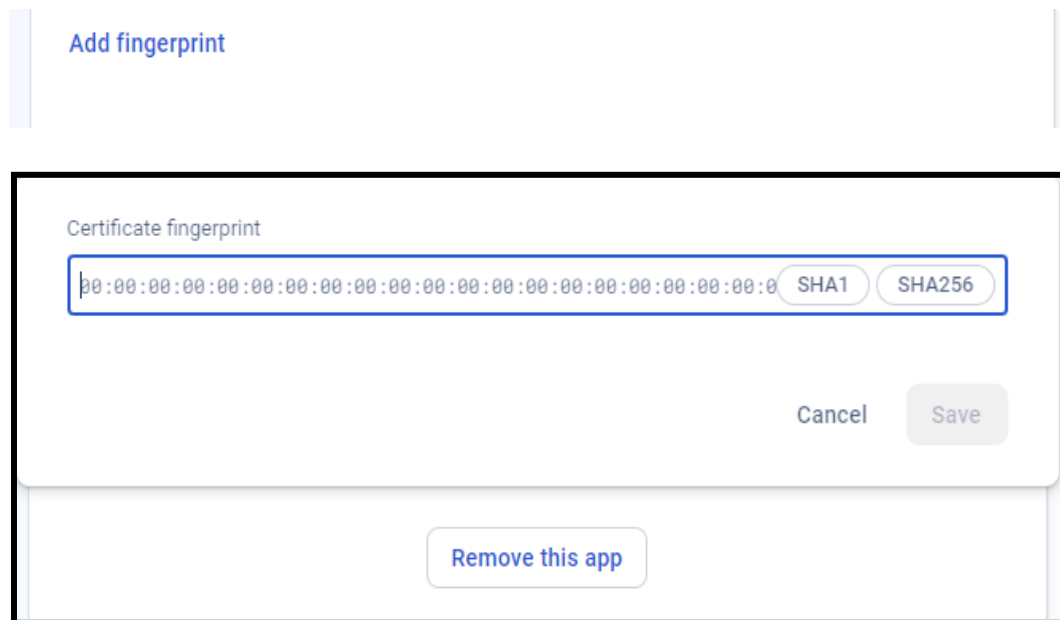
- ```
keytool -list -v -keystore "Your directory path\debug.jks" -alias
androiddebugkey -storepass android -keypass android
```

2. To Generate release SHA use the below command

```
keytool -list -v -keystore "your directory path\keystore.jks" -alias
androidreleasekey -storepass your store password -keypass your
key password
```

After generating the debug and release SHA, you have to add them in the Firebase Console where you have created the Android app.

Please check the screenshot below for the reference.



- viii. Download the google-services.json file from Firebase project settings and paste it at the **android/app** location.
- ix. Add Firebase SDK Add the plugin as a build script dependency to your project-level **build.gradle** file:



```
buildscript {
 repositories {
 // Make sure that you have the following two repositories
 google() // Google's Maven repository
 mavenCentral() // Maven Central repository
 }
 dependencies {
 ...
 // Add the dependency for the Google services Gradle plugin
 classpath 'com.google.gms:google-services:4.3.15'
 }
}

allprojects {
 ...
 repositories {
 // Make sure that you have the following two repositories
 google() // Google's Maven repository
 mavenCentral() // Maven Central repository
 }
}
```

- x. Then, in your module (app-level) `build.gradle` file, add both the google-services plugin and any Firebase SDKs that you want to use in your app:

```
plugins {
 id 'com.android.application'
 // Add the Google services Gradle plugin
 id 'com.google.gms.google-services'
 ...
}

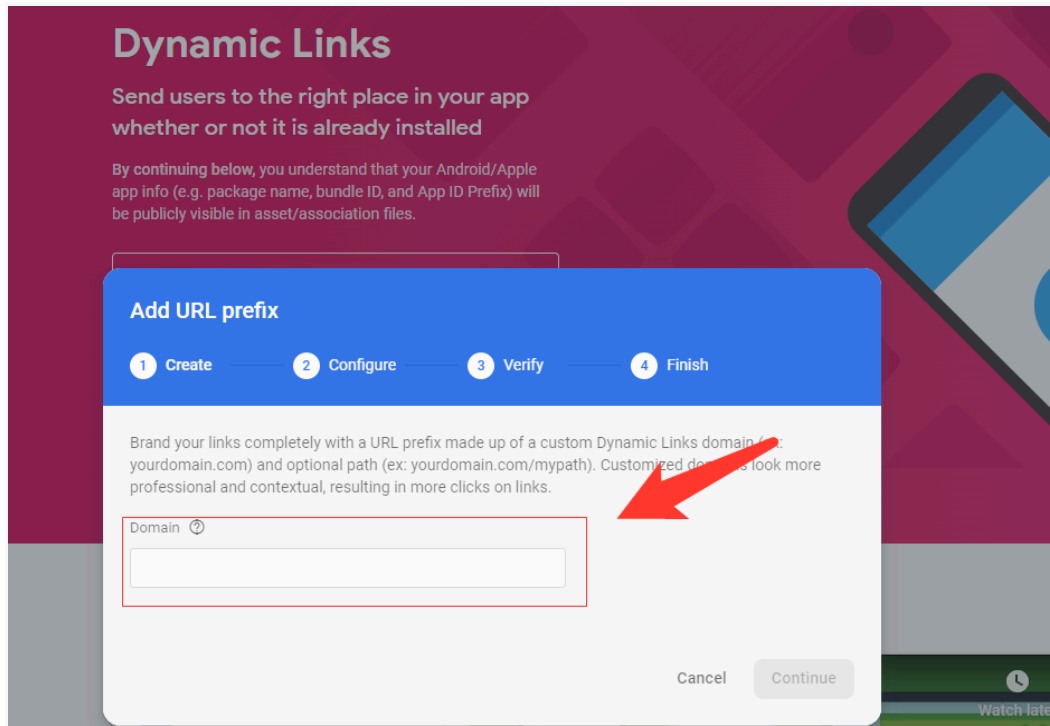
dependencies {
 // Import the Firebase BoM
 implementation platform('com.google.firebase:firebase-bom:31.5.0')

 // TODO: Add the dependencies for Firebase products you want to use
 // When using the BoM, don't specify versions in Firebase dependencies
 implementation 'com.google.firebase:firebase-analytics-ktx'

 // Add the dependencies for any other desired Firebase products
 // https://firebase.google.com/docs/android/setup#available-libraries
}
```

## h. To Set up Dynamic Links

- i. In the Firebase console click on "Get Started" or "Set Up" to start the process of setting up Dynamic Links for your project.

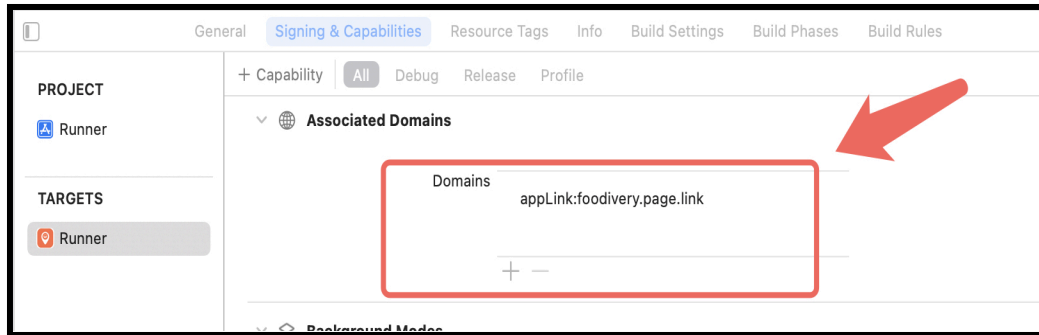


- ii. After you create Dynamic Links, you need to set up your app codes  
**lib\utils\global.dart**

```
String dynamicLink = 'https://foodivery.page.link';
```

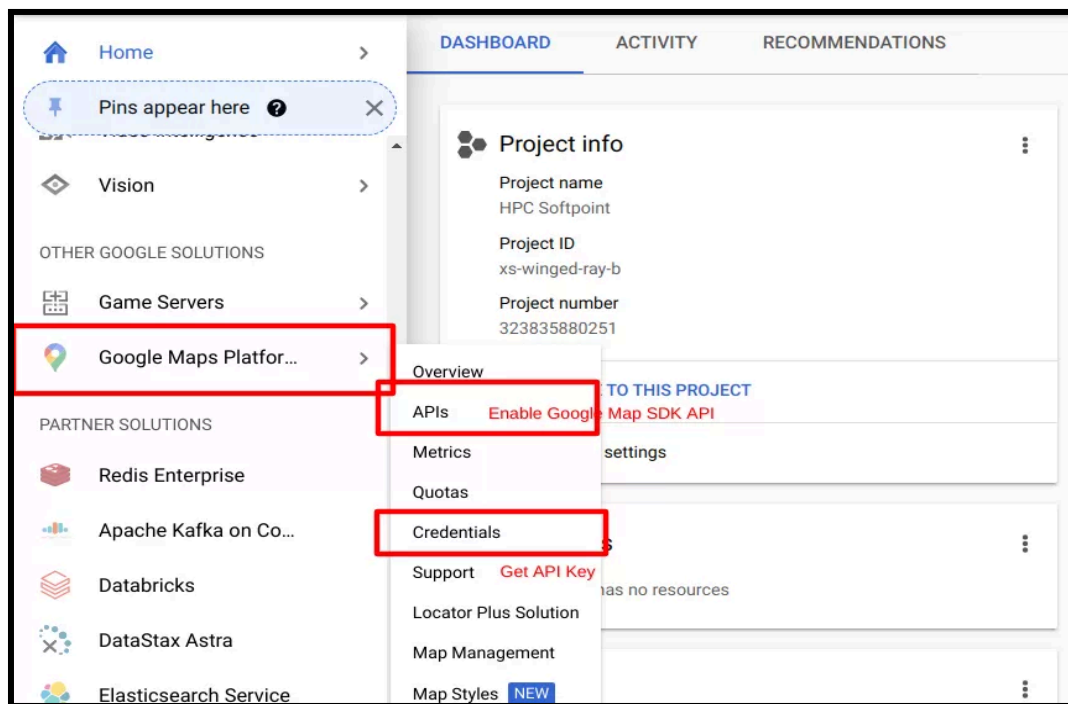
### In Xcode

- Add associated domain capabilities and set the domain

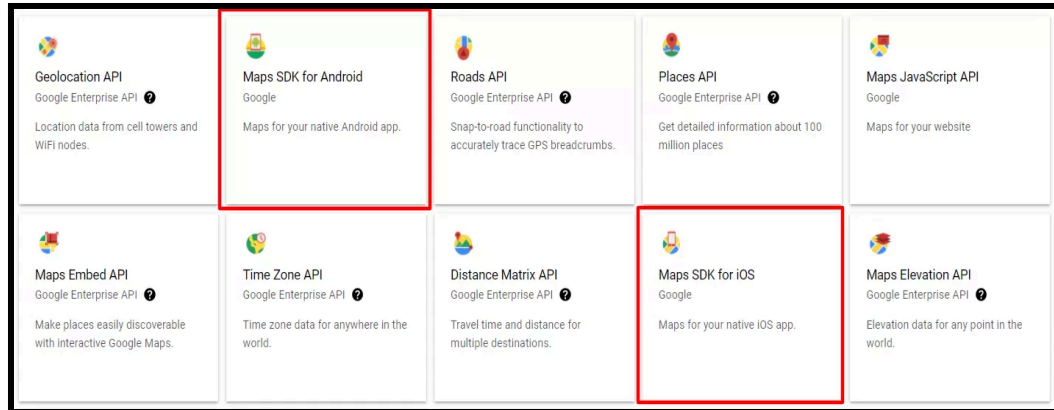


i. To add Google Maps in the app

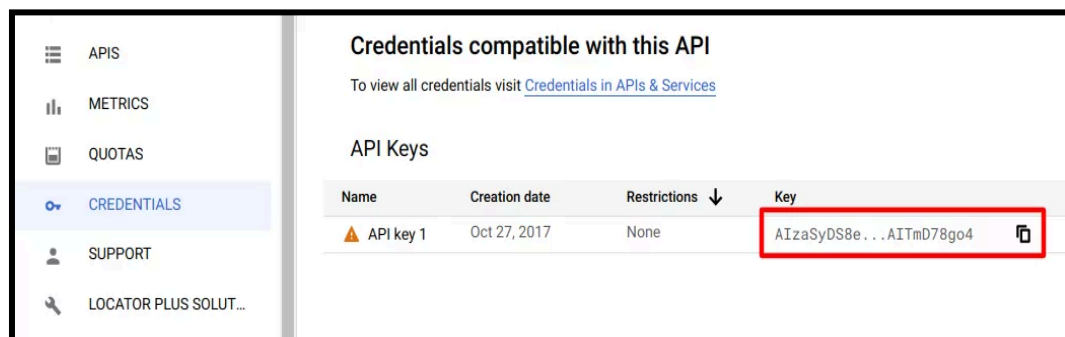
i. Enable Google Map SDK in the [Google Cloud Console](#) platform.



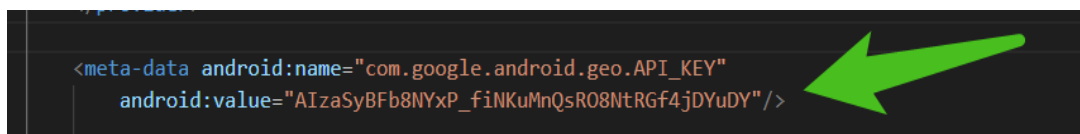
ii. In the API Section, Enable Maps SDK




iii. Go to Credentials to get the API Key.



iv. To add Google Maps API key in Android:  
**android/app/src/main/AndroidManifest.xml**



v. To add Google Maps API key in iOS:  
**ios\Runner\AppDelegate.swift**



```
import UserNotifications
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
 override func application(
 _ application: UIApplication,
 didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey]?)
 -> Bool {
 GMSServices.provideAPIKey("AIzaSyBe99x-g_IKpL3arkSIhRuG94Fhd76l0pQ")
 FirebaseApp.configure()
 GeneratedPluginRegistrant.register(with: self)
 return true
 }
}
```

#### j. Set up Firebase iOS App

- i. Go to the Firebase console
- ii. Select the project you created in step 5.d.vi
- iii. Go to **Project Setting**
- iv. In the **General** Tab click on the Add App button
- v. Select **iOS**
- vi. Fill out the form and click on the **Register App** Button

(Please check the below screenshot for reference)



The screenshot shows the 'Add Firebase to your Apple app' wizard. At the top, there is a close button (X) and the title 'Add Firebase to your Apple app'. Below the title is a progress indicator with five steps: 1. Register app (highlighted with a blue circle), 2. Download config file, 3. Add Firebase SDK, 4. Add initialisation code, and 5. Next steps. The 'Register app' step contains three input fields: 'Apple bundle ID' with a help icon and the value 'com.company.appname', 'App nickname (optional)' with a help icon and the value 'My Apple app', and 'App Store ID (optional)' with a help icon and the value '123456789'. Below these fields is a 'Register app' button.

×

## Add Firebase to your Apple app

1 Register app

Apple bundle ID ⓘ

com.company.appname

App nickname (optional) ⓘ

My Apple app

App Store ID (optional) ⓘ

123456789

Register app

2 Download config file

3 Add Firebase SDK

4 Add initialisation code

5 Next steps

- vii. Download the GoogleService-info.plist file from Firebase project settings and paste it at the **ios/Runner** location in the app
- viii. XCode Project Setting

This step covers reviewing the most important settings in the XCode workspace. For detailed procedures and descriptions, see [Prepare for App Distribution](#)

1. Navigate to your target's settings in XCode:

- a. Open the default Xcode workspace in your project by running the below command in a terminal window from your Flutter project directory.

```
open ios/Runner.xcworkspace
```

- b. To view your app's settings, select the Runner target in the Xcode navigator.

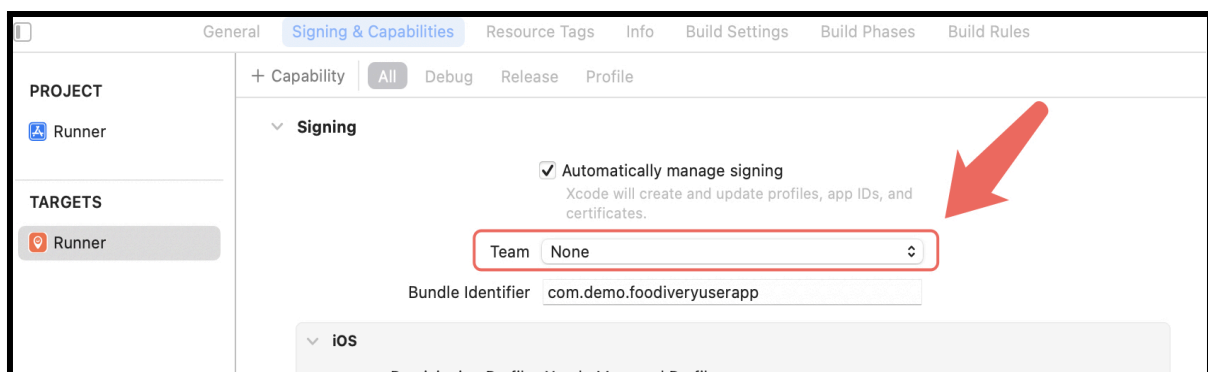
2. Verify the most important settings

- a. In the Identity section of the General tab

- i. **Display Name** (The display name of your app.)
- ii. **Bundle Identifier** (The App ID you registered on App Store Connect.)

- b. In the Signing & Capabilities tab

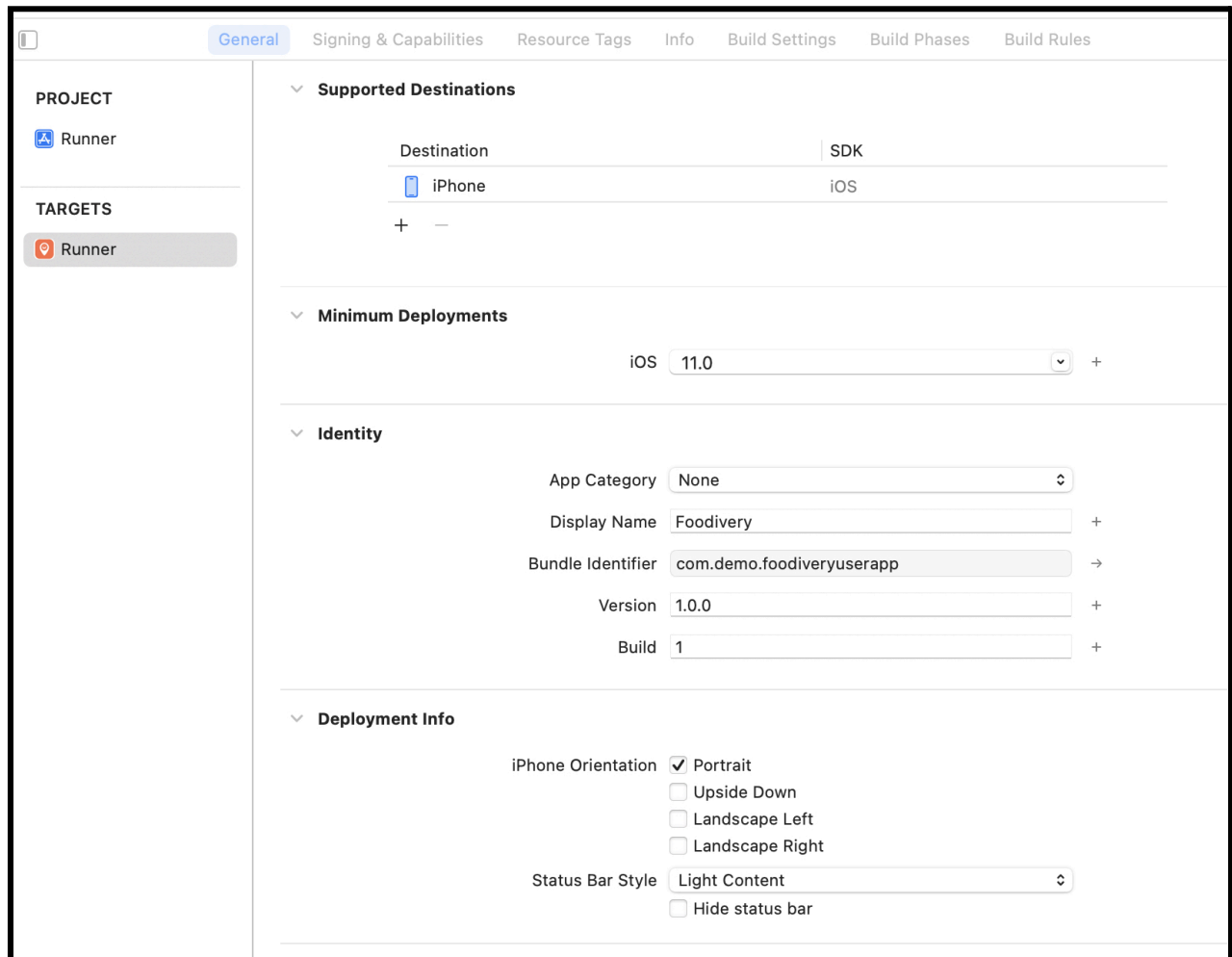
- i. **Automatically manage signing** (Xcode should automatically manage app signing and provisioning. This is set true by default, which should be sufficient for most apps. For more complex scenarios, see the [Code Signing Guide](#))
- ii. **Team** (Select the team associated with your registered Apple Developer account. If required, select Add Account..., then update this setting.)



c. In the deployment section of the build settings tab:

i. iOS Deployment Target

1. The minimum iOS version that the app supports is 11.0.
2. The General tab of your project settings should resemble the following:

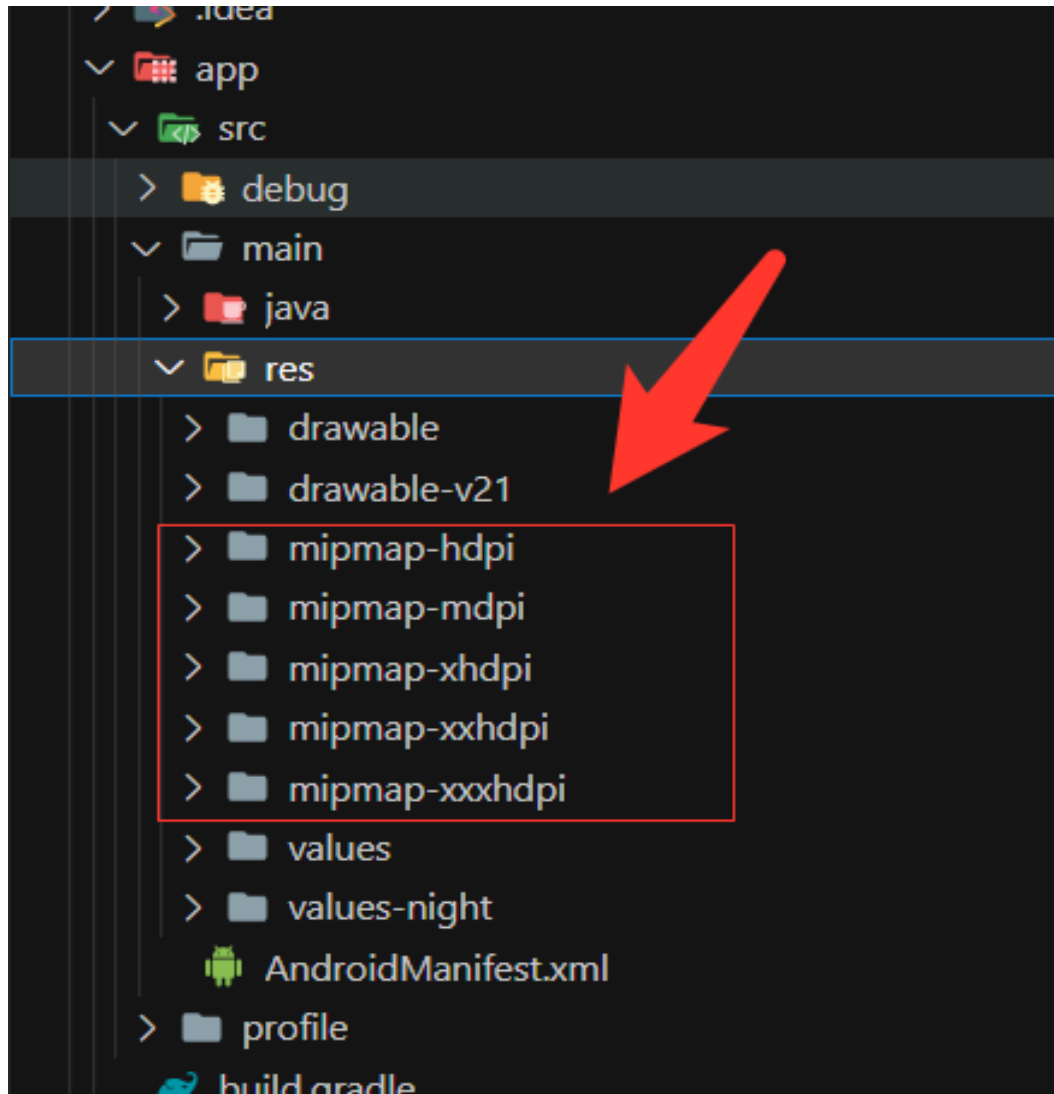


3. For a detailed overview of app signing, see [Create, export, and Delete signing certificates](#).

## k. Change App Icon

### i. For Android

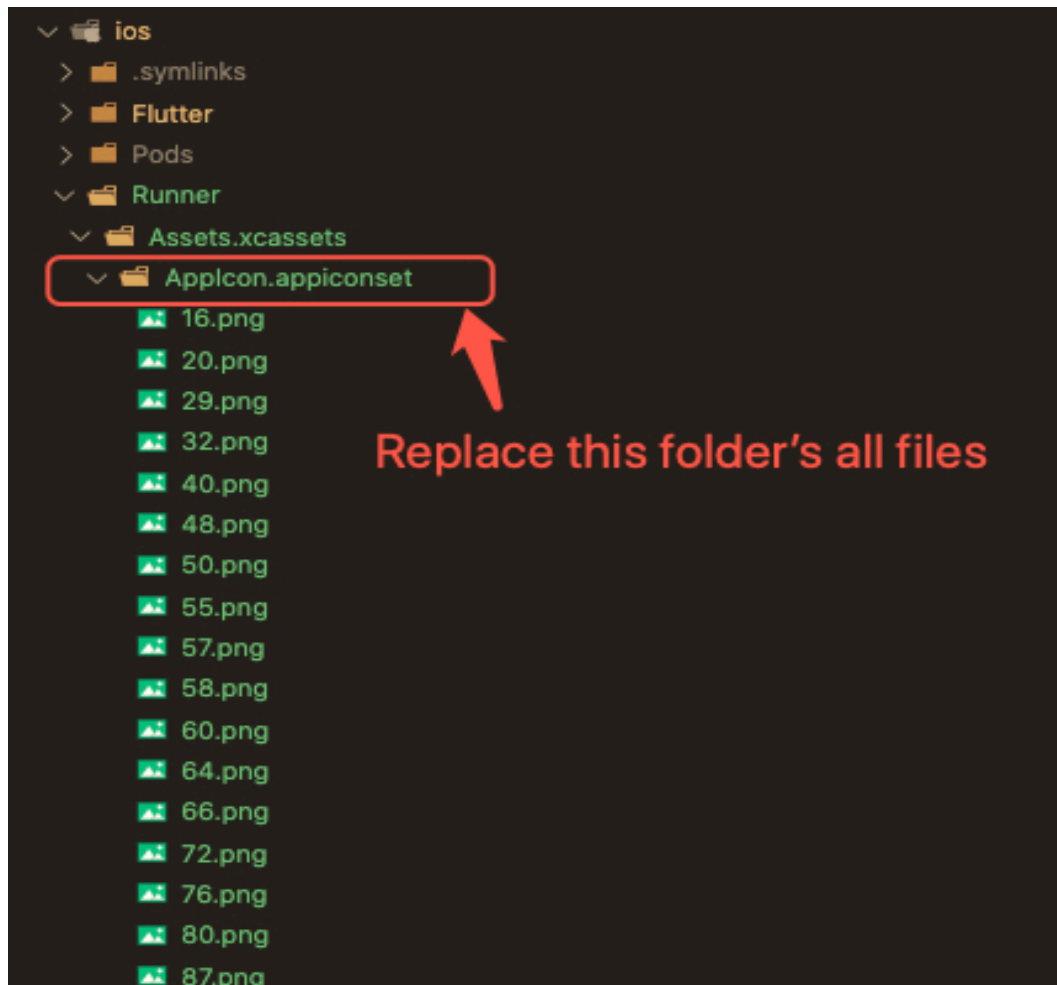
Replace the icons in the **android\app\src\main\res** folder as shown in the below image.



### ii. For iOS

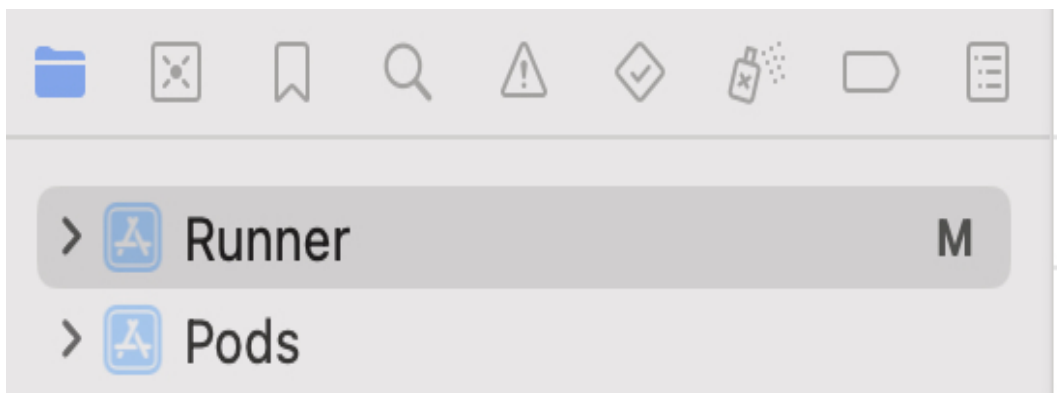
1. Replace the icons in the below folder as shown in the below image

**ios\Runner\Assets.xcassets\AppIcon.appiconset**

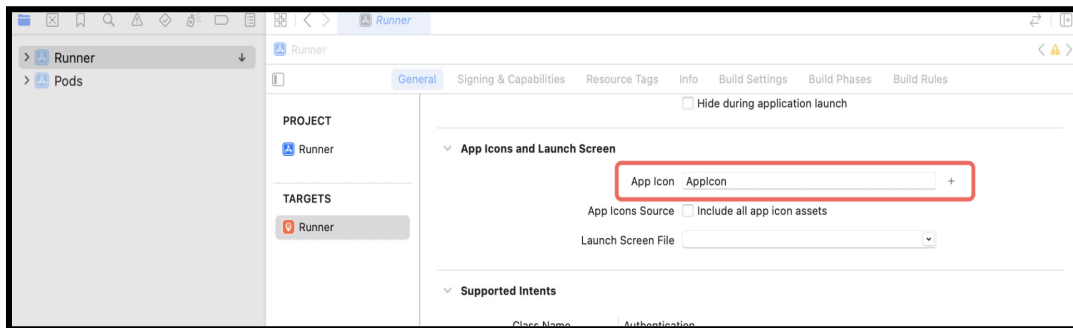


## 2. Change icons using XCode

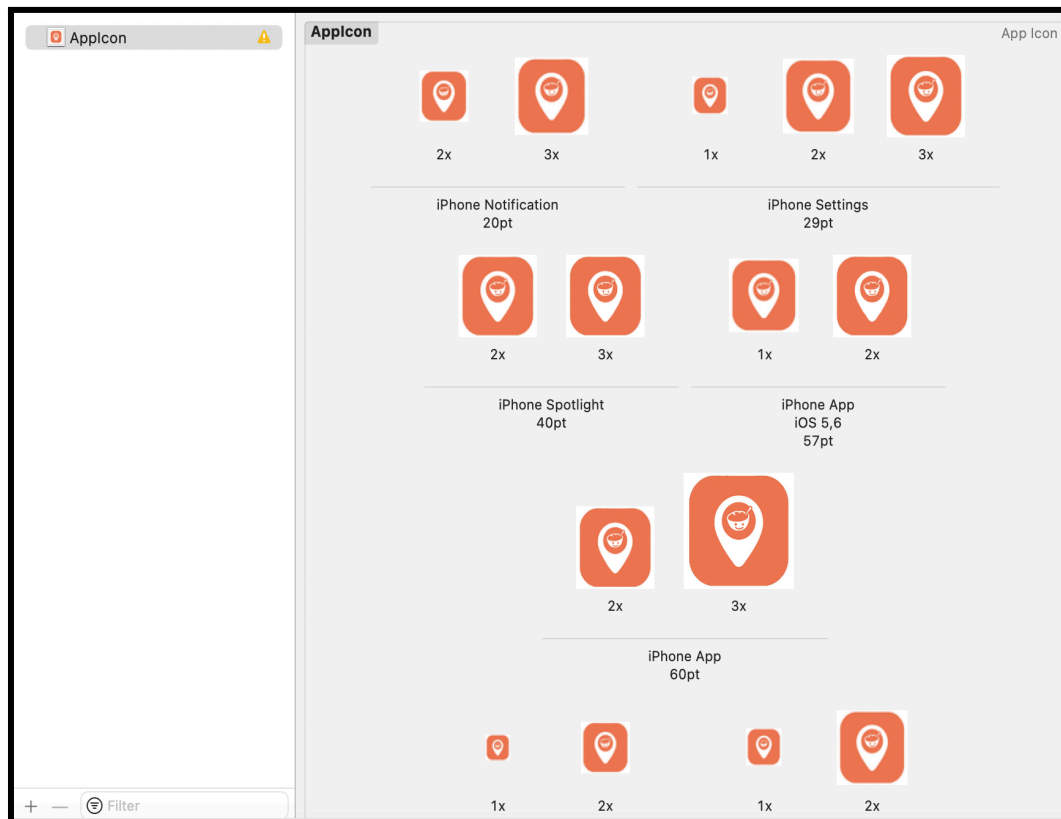
- a. Right-click on the iOS folder Choose Open in Xcode Option
- b. Click on the folder icon on the left side of the XCode window



- c. Select Runner.
- d. Select Target runner
- e. Go to App Icons And Launch Images
- f. Click the right arrow button of the app icon source



- g. Replace all the icons according to their size





**NOTE:**

- If you want to generate the App icon bundle from any image you have, you can generate it from publicly available websites like

<https://www.appicon.co/>

**I. Build Release for Android**

- Open Project in VS Code
- In Terminal Execute the below commands

```
flutter clean
flutter pub get
flutter build apk --release
```

- After making the release, to generate the release bundle Execute the below command

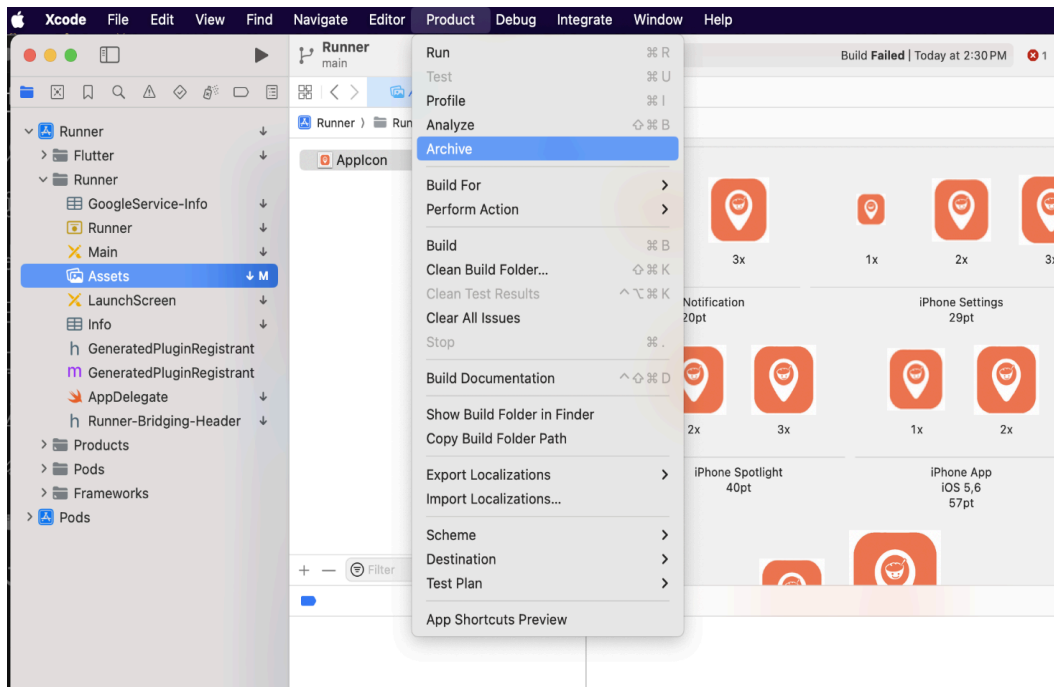
```
flutter build appbundle --release
```

- Get the APK from the below path

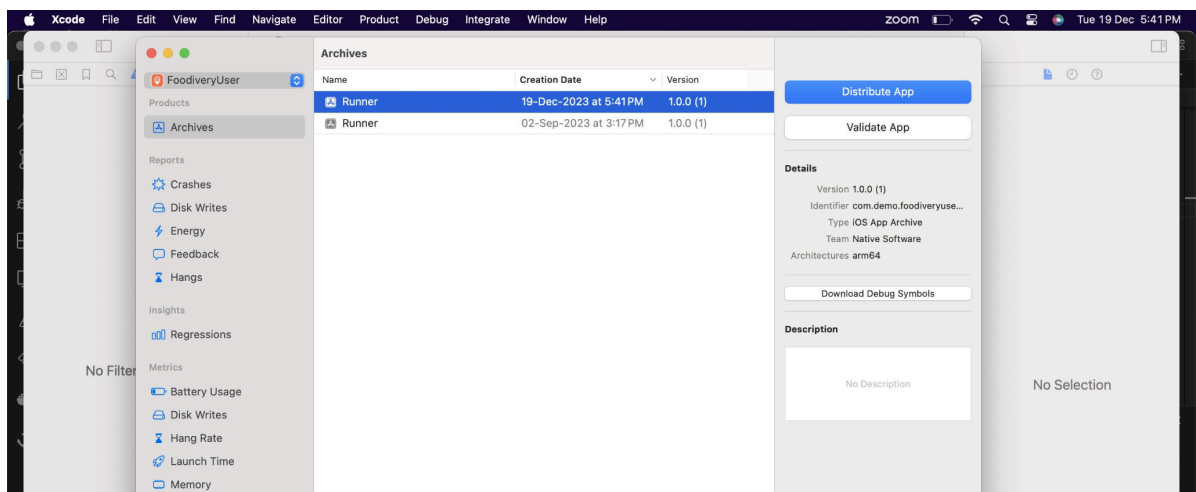
**build\app\outputs\flutter-apk\app-release.apk**

**m. Build Release for iOS**

- Open Project in XCode
- Select **Archive** from the **Product Menu**



- iii. After successfully archiving select the **Organizer** option from the **Windows** menu
- iv. After clicking on it opens one popup for Archive, Click on the **Distribute App** Button



- v. After successfully done, you can upload this app to your Apple developer account in the TestFlight
- vi. To publish your app from TestFlight please follow [this link](#)

## n. Other Options for the Advanced User

## i. Paths to the images used in the app

| Images                  | Path                                                                                                                            | Screen Path                                       |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| Splash screen           | assets\images\logo.png                                                                                                          | lib\views\splash\splashScreen.dart                |
| Card screen             | assets\images\appleImage.png<br>assets\images\googleLogo.png<br>assets\images\masterCardLogo.png<br>assets\images\visaImage.png | lib\views\home\myCardScreen.dart                  |
| Coupon screen           | assets\images\coupon_logo.png                                                                                                   | lib\views\drawer\myCouponScreen.dart              |
| Add New Card Screen     | assets\images\creditCard.png                                                                                                    | lib\views\home\addNewCardScreen.dart              |
| Checkout Screen         | assets\images\discountImage.png                                                                                                 | lib\views\home\checkOutScreen.dart                |
| Intro Screen            | assets\images\intro_Image_1.png<br>assets\images\intro_Image_2.png<br>assets\images\intro_Image_3.png                           | lib\views\onBoarding Screen\introScreen.dart      |
| Wallet Screen           | assets\images\noOrderImage.png                                                                                                  | lib\views\drawer\walletScreen.dart                |
| My Order Screen         | assets\images\noProduct.png                                                                                                     | lib\views\drawer\myOrderScreen.dart               |
| Add Order Review Screen | assets\images\order-review.png                                                                                                  | lib\views\addOrderReviewScreen.dart               |
| Order Detail Screen     | assets\images\placeholder.png                                                                                                   | lib\views\drawer\setting\orderDetailScreen.dart   |
| Order successful screen | assets\images\successPaymentImage.png                                                                                           | lib\views\successScreen\successPaymentScreen.dart |

ii. Fonts used in the app. If you want to change, you can make the changes in the **pubspec.yaml** file and the **Assets** folder.

|        |                                                                                                                                       |
|--------|---------------------------------------------------------------------------------------------------------------------------------------|
| Gilroy | assets/fonts/Gilroy-Bold.ttf<br>assets/fonts/Gilroy-Medium.ttf<br>assets/fonts/Gilroy-Regular.ttf<br>assets/fonts/Gilroy-SemiBold.ttf |
|--------|---------------------------------------------------------------------------------------------------------------------------------------|

- iii. Colors used in the app. If you want to change the colors you can make the changes in the file **lib/Theme/nativeTheme.dart**

| #                    | Color code |
|----------------------|------------|
| Primary color        | #FF6C44    |
| Background color     | #FFFFFF    |
| Primary light color  | #FFDDCC    |
| Secondary color      | #111A2C    |
| Text field color     | #F5F5F8    |
| Error color          | #FF1717    |
| Success color        | #27AE60    |
| Hint color           | #BBBDC1    |
| Counter color        | #898B9A    |
| Facebook color       | #0047B3    |
| Google color         | #F5F5F8    |
| Delete account color | #D74722    |
| Cancel button color  | #898481    |

- iv. Packages used in the app are listed below. You can find them in **pubspec.yaml** file.

| Package Name - Version        | Description                                                                                                       |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------|
| firebase_core: ^2.25.5        | To use the Firebase Core API, which enables connecting to multiple Firebase apps.                                 |
| firebase_auth: ^4.17.6        | To use the <a href="#">Firebase Authentication API</a> .                                                          |
| firebase_messaging: ^14.7.17  | To use the <a href="#">Firebase Cloud Messaging API</a> .                                                         |
| smooth_page_indicator: ^1.1.0 | Customizable animated page indicator with a set of built-in effects.                                              |
| shimmer_animation: ^2.1.0+1   | This shimmer animation widget can help you bring simple yet beautiful skeleton loaders to your project with ease. |
| image_picker: ^1.0.7          | For iOS and Android for picking images from the image library, and taking new pictures with the camera.           |

|                                          |                                                                                                                                                                           |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| webview_flutter: ^4.7.0                  | Flutter Pininput is a package that provides an easy-to-use and customizable Pin code input field                                                                          |
| carousel_slider: ^4.2.1                  | A carousel slider widget.                                                                                                                                                 |
| flutter_advanced_drawer: ^1.3.6          | An advanced drawer widget, that can be fully customized with size, text, color, and radius of corners.                                                                    |
| google_nav_bar: ^5.0.6                   | A modern Google style nav bar for flutter                                                                                                                                 |
| cached_network_image: ^3.3.1             | To show images from the internet and keep them in the cache directory.                                                                                                    |
| email_validator: ^2.1.17                 | A simple (but correct) Dart class for validating email addresses without using RegEx. Can also be used to validate emails within Flutter apps                             |
| get: ^4.6.6                              | To use for state management, intelligent dependency injection, and route management quickly and practically                                                               |
| get_storage: ^2.1.1                      | A fast, extra light and synchronous key-value in memory, which backs up data to disk at each operation                                                                    |
| connectivity_plus: ^5.0.2                | This plugin allows Flutter apps to discover network connectivity and configure themselves accordingly. It can distinguish between cellular vs WiFi connection.            |
| http: ^1.1.2                             | A composable, Future-based library for making HTTP requests                                                                                                               |
| fl_country_code_picker: ^0.1.9+1         | A Flutter package for showing a modal that contains country dial code                                                                                                     |
| shared_preferences: ^2.2.2               | Wraps platform-specific persistent storage for simple data                                                                                                                |
| phone_number_parser: ^8.2.1              | Phone Number is a Flutter plugin that allows you to parse, validate, format and other utilities for to international phone numbers                                        |
| intl: ^0.19.0                            | Provides internationalization and localization facilities, including message translation, plurals and genders, date/number formatting and parsing, and bidirectional text |
| permission_handler: ^11.1.0              | This plugin provides a cross-platform (iOS, Android) API to request permissions and check their status                                                                    |
| geolocator: ^11.0.0                      | A Flutter geolocation plugin which provides easy access to platform specific location services                                                                            |
| geocoding: ^2.2.0                        | A Flutter Geocoding plugin which provides easy geocoding and reverse-geocoding features                                                                                   |
| google_maps_flutter: ^2.5.0              | To provides a <a href="#">Google Maps</a> widget                                                                                                                          |
| flutter_polyline_points: ^2.0.0          | To decodes encoded google polyline string into list of geo-coordinates suitable for showing route/polyline on maps                                                        |
| razorpay_flutter: ^1.3.6                 | For Razorpay SDK.                                                                                                                                                         |
| material_design_icons_flutter: ^7.0.7296 | The <a href="#">Material Design Icons</a> Icon pack available as a set of Flutter Icons                                                                                   |

|                                      |                                                                                                                                                |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| font_awesome_flutter: ^10.7.0        | The <a href="#">Font Awesome</a> Icon pack available as a set of Flutter Icons                                                                 |
| flutter_rating_bar: ^4.0.1           | A simple yet fully customizable rating bar which also includes a rating bar indicator, supporting any fraction of rating                       |
| open_filex: ^4.4.0                   | Can call native APP to open files with string result in flutter, support iOS(DocumentInteraction) / android(intent) / PC(ffl) / web(dart:html) |
| flutter_local_notifications: ^16.3.3 | For displaying local notifications.                                                                                                            |
| share_plus: ^7.2.2                   | To share content from your Flutter app via the platform's share dialog                                                                         |
| firebase_dynamic_links: ^5.4.15      | To use the <a href="#">Firebase Dynamic Links API</a>                                                                                          |
| url_launcher: ^6.2.5                 | To launch a URL                                                                                                                                |
| image_cropper: ^5.0.1                | For Android, iOS and Web supports cropping images                                                                                              |

## 6. Set up Driver App (Technology Flutter)

### a. Initial steps to set up and run mobile app

- i. Open the **DriverApp** folder in the VSCode
- ii. To complete the other setup, please follow the instructions from above steps **4.a.ii** to **4.a.vi**.

### b. Change API base URL

After the setup of your API and Admin panel, you have to change your API base URL for that, go to the file located at **lib\utils\global.dart**

```
String appMode = "LIVE";

Map<String, dynamic> appParameters = {
 "LIVE": {
 "apiUrl": "https://foodivery.native.software/api/",
 },
 "DEV": {
 "apiUrl": "http://192.168.29.118:8080/api/",
 }
};
```

### c. Change App Name

#### i. Change the app name in the Android App

1. Change the app name in the file located at **android/app/src/main/AndroidManifest.xml**


```
<!-- <uses-permission android:name="android.permission.INTERNET" /> -->
<application
 android:label="Foodivery Driver"
 android:usesCleartextTraffic="true"
 android:icon="@mipmap/ic_launcher">
```

#### ii. Change the app name in the iOS App

##### 1. In VSCode

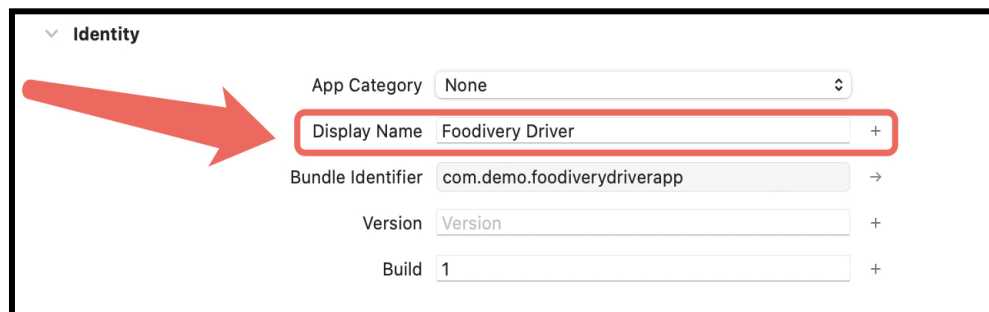
- a. Go to **ios/Runner/info.plist**
- b. Change string of key **CFBundleDisplayName**

```
<string>$(EXECUTABLE_NAME)</string>
<key>CFBundleDisplayName</key>
<string>Foodivery Driver</string>
<key>CFBundleExecutable</key>
```



## 2. In XCode

- a. Right-click on the **iOS** folder and Choose Open in Xcode Option
- b. Click on the folder icon left side of the XCode window
- c. Select Runner.
- d. Select Target runner
- e. From the General Tab Go to the identity
- f. Change Display Name




## d. Change Package Name/Bundle ID

An app's package name is a unique identifier that is automatically created when you create an app. The term used for iOS apps is "**bundle ID**"; for Android apps, it is "**package name**".

### i. Set Package Name for Android App

1. Change the package name in the file located at **android/app/src/main/AndroidManifest.xml**

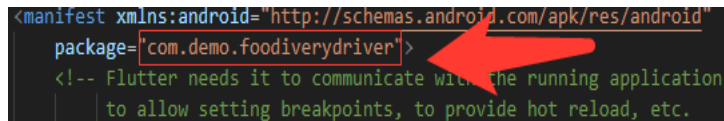
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.demo.foodiverydriver">
 <uses-permission android:name="android.permission.INTERNET"/>
```



2. Change the package name in the file located at **android/app/src/debug/AndroidManifest.xml**

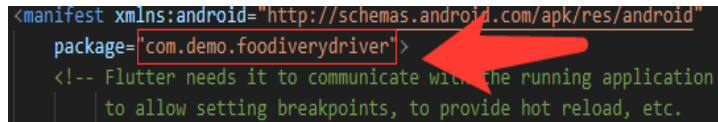


```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.demo.foodiverydriver">
 <!-- Flutter needs it to communicate with the running application
 to allow setting breakpoints, to provide hot reload, etc.
-->
```



3. Change Package Name in file which is located at **android/app/src/Profile/AndroidManifest.xml**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.demo.foodiverydriver">
 <!-- Flutter needs it to communicate with the running application
 to allow setting breakpoints, to provide hot reload, etc.
-->
```



4. Change the Package Name in the file which is located at **android/app/build.gradle**

```
defaultConfig {
 // TODO: Specify your own unique Application ID (see https://developer.android.com/studio/build/apk-signing)
 applicationId "com.demo.foodiverydriver"
 minSdkVersion 21
 targetSdkVersion 33
}
```



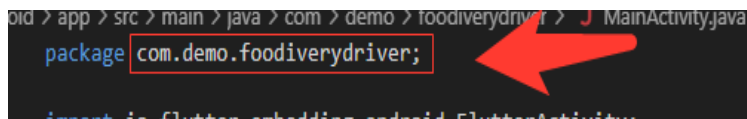
5. Change the folder structure for the below path as per your package name.

**android\app\src\main\java\com\demo\foodiverydriver\**

6. Change Package Name in file which is located at **android\app\src\main\java\com\demo\foodiverydriver\MainActivity.java**

```
package com.demo.foodiverydriver;

import io.flutter.embedding.android.FlutterActivity;
```

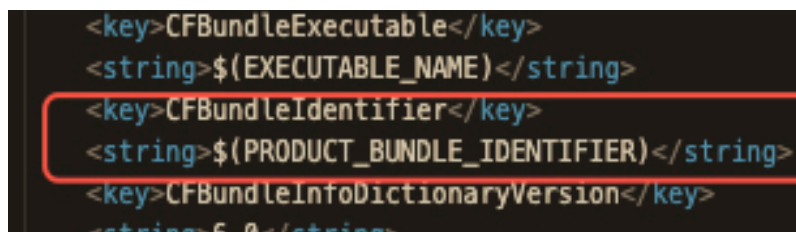


- ii. Set Bundle ID for iOS App

1. In VSCode

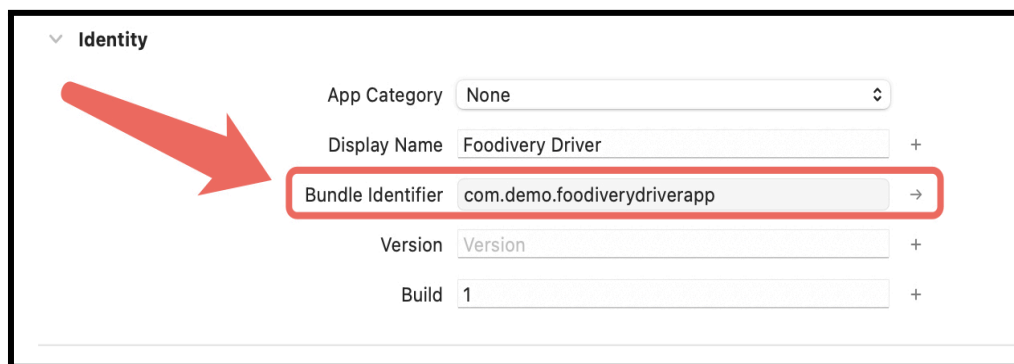
- a. Go to **ios/Runner/info.plist**
- b. Change the string of key **CFBundleIdentifier**

```
<key>CFBundleExecutable</key>
<string>$(EXECUTABLE_NAME)</string>
<key>CFBundleIdentifier</key>
<string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
<key>CFBundleInfoDictionaryVersion</key>
<string>6.0</string>
```

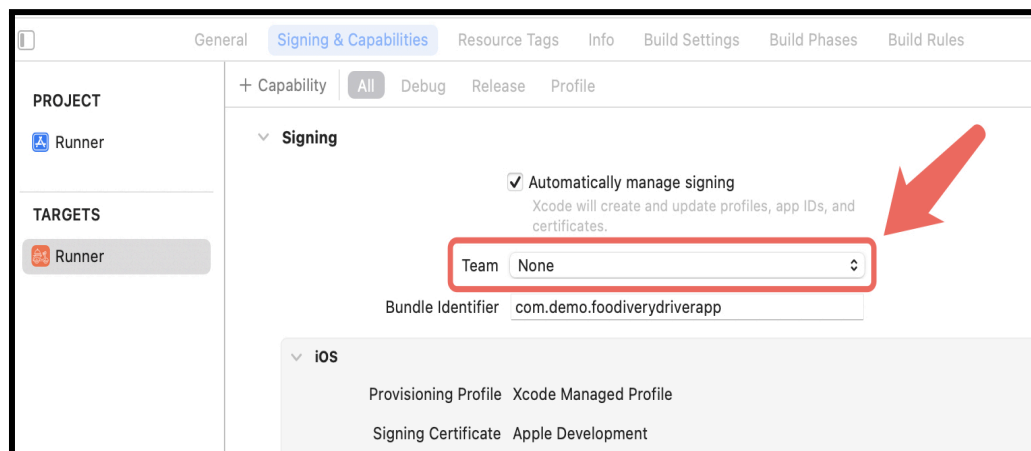


## 2. In XCode

- a. Right-click on the **iOS** folder and Choose Open in Xcode Option
- b. Click on the folder icon on the left side of the XCode window
- c. Select Runner.
- d. Select Target runner
- e. Go to identity
- f. Change Bundle Identifier



- g. In Signing & Capabilities Go to Signing
- h. Change Bundle Identifier



## e. Create and set the Keystore file for Android

To complete this setup, please follow the instructions from steps **4.e.i** to **4.a.iv** above.

## f. Create Firebase Account & Project

You can use the same account you created for the Customer App. ( If you want to create a different account for this App, then follow the above steps from 4.f. )

## g. Set up Android App in Firebase Project

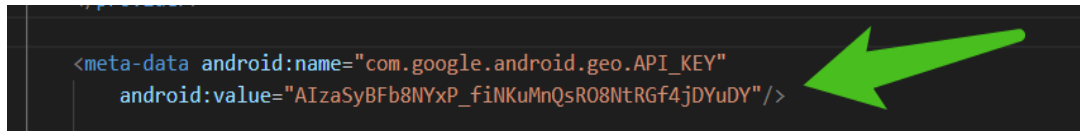
For this setup you can follow the complete steps mentioned in 4.g.

## h. To add Google Maps in the app

i. For this setup you can refer to the steps from **4.i.i** to **4.i.iii**.

ii. To add Google Maps API key in Android:

**android/app/src/main/AndroidManifest.xml**

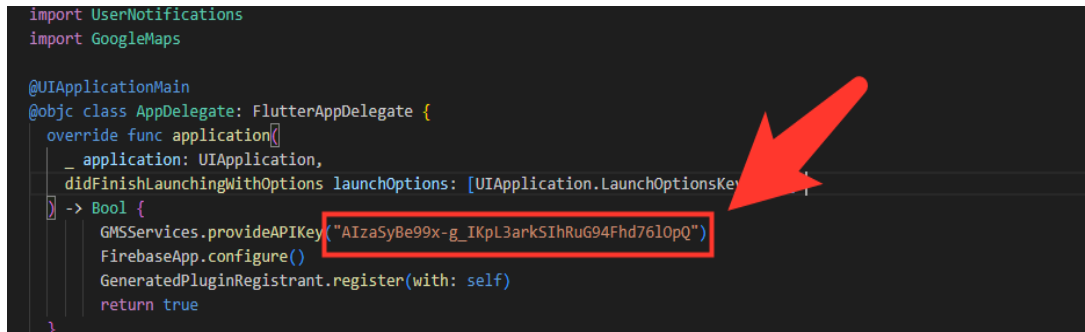


```
<meta-data android:name="com.google.android.geo.API_KEY"
 android:value="AIzaSyBFb8NYxP_fiNKuMnQsR08NtRGf4jDYuDY" />
```

A green arrow points to the API key value in the meta-data entry.

iii. To add Google Maps API key in IOS:

**ios\Runner\AppDelegate.swift**



```
import UserNotifications
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
 override func application(
 _ application: UIApplication,
 didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey]?,
 _ _ Bool {
 GMSServices.provideAPIKey("AIzaSyBe99x-g_IKpl3arkSIhRuG94Fhd76l0pQ")
 FirebaseApp.configure()
 GeneratedPluginRegistrant.register(with: self)
 return true
 }
 }
```

A red arrow points to the API key value in the GMServices.provideAPIKey method call.

## i. Set up Firebase iOS App

i. Follow the steps from **4.j.i** to **4.j.vii**.

ii. XCode Project Setting

This step covers reviewing the most important settings in the XCode workspace. For detailed procedures and descriptions, see [Prepare for App Distribution](#)

1. Navigate to your target's settings in XCode:

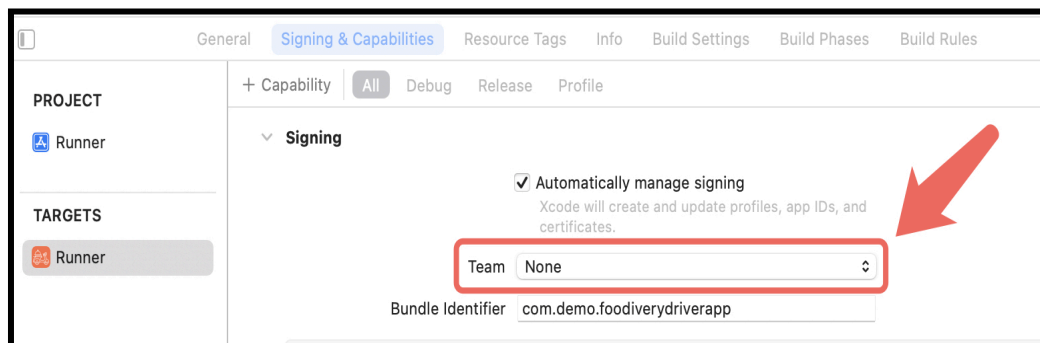
- a. Open the default Xcode workspace in your project by running the below command in a terminal window from your Flutter project directory.

**open ios/Runner.xcworkspace**

- b. To view your app's settings, select the Runner target in the Xcode navigator.

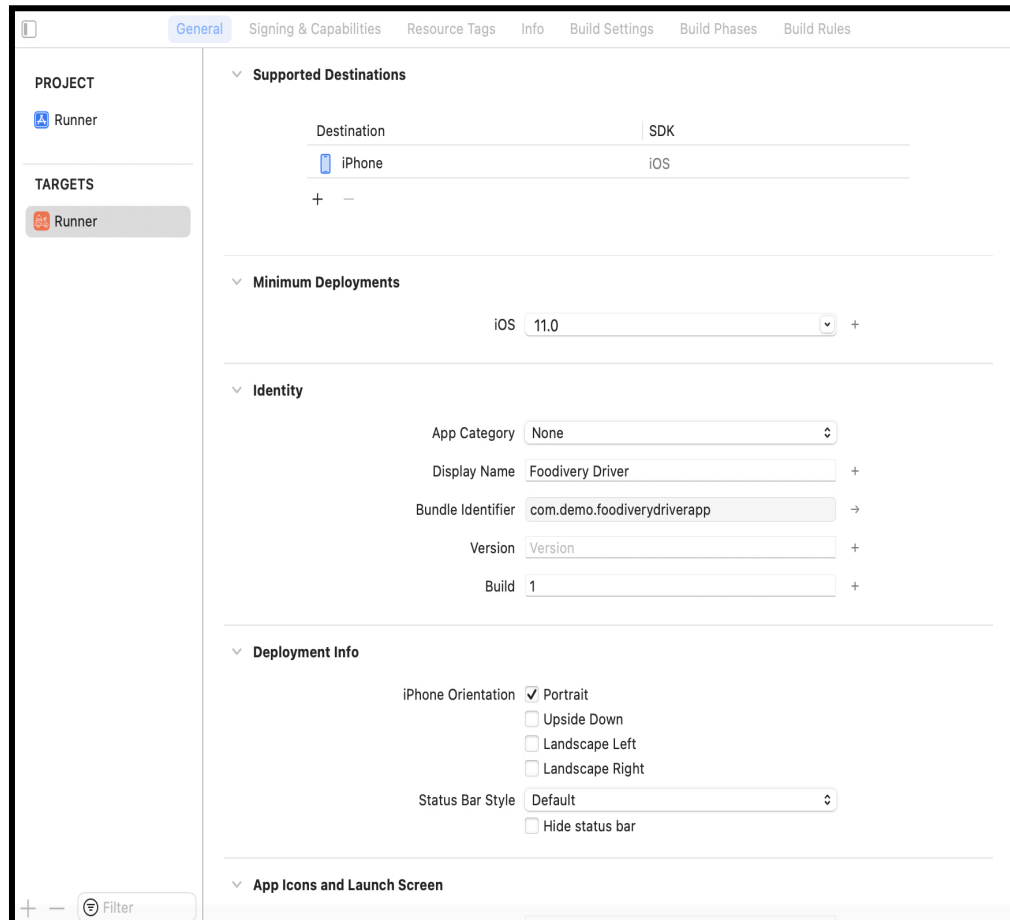
## 2. Verify the most important settings

- a. In the Identity section of the General tab
  - i. **Display Name** (The display name of your app.)
  - ii. **Bundle Identifier** (The App ID you registered on App Store Connect.)
- b. In the Signing & Capabilities tab
  - iii. **Automatically manage signing** (Xcode should automatically manage app signing and provisioning. This is set true by default, which should be sufficient for most apps. For more complex scenarios, see the [Code Signing Guide](#))
  - iv. **Team** (Select the team associated with your registered Apple Developer account. If required, select Add Account..., then update this setting.)



- c. In the deployment section of the build settings tab:
  - i. iOS Deployment Target

1. The minimum iOS version that the app supports is 11.0.
2. The General tab of your project settings should resemble the following:

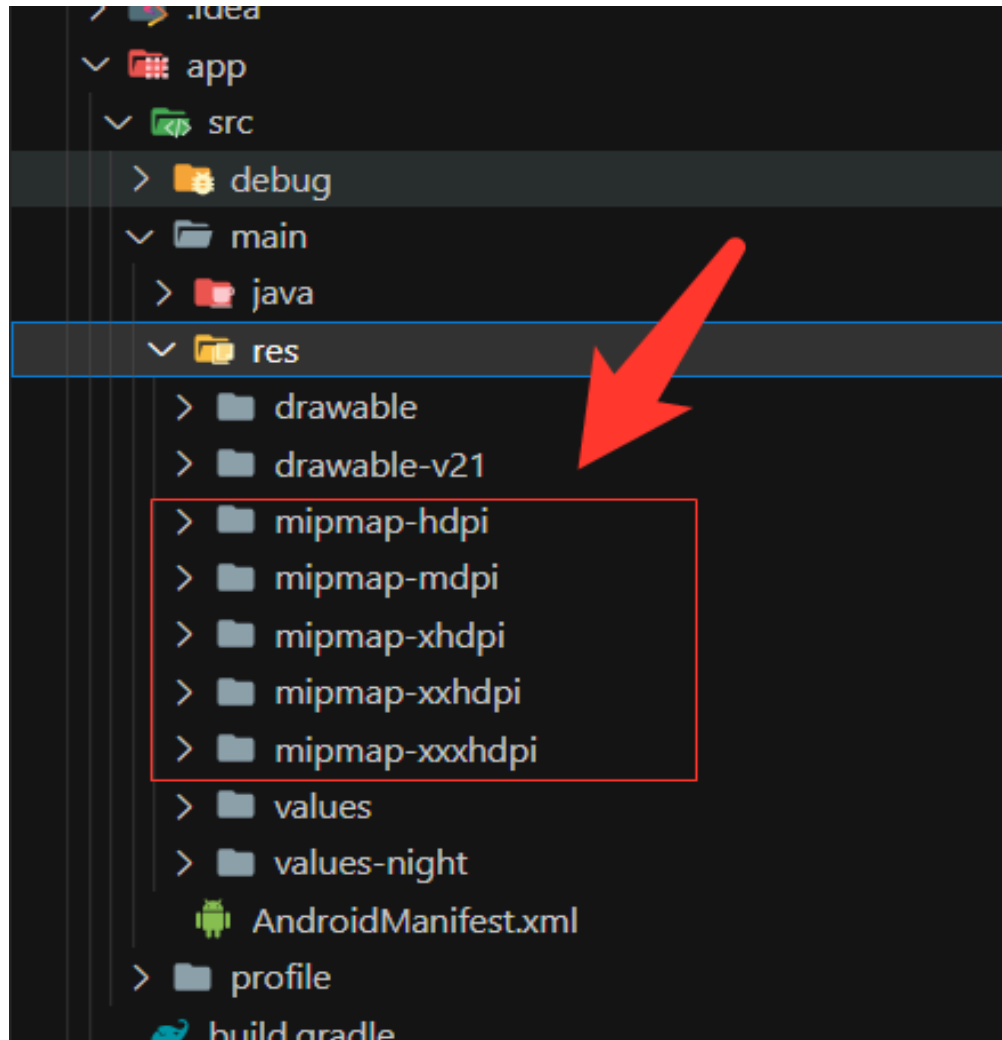


3. For a detailed overview of app signing, see [Create, export, and Delete signing certificates](#).

## j. Change App Icon

### i. For Android

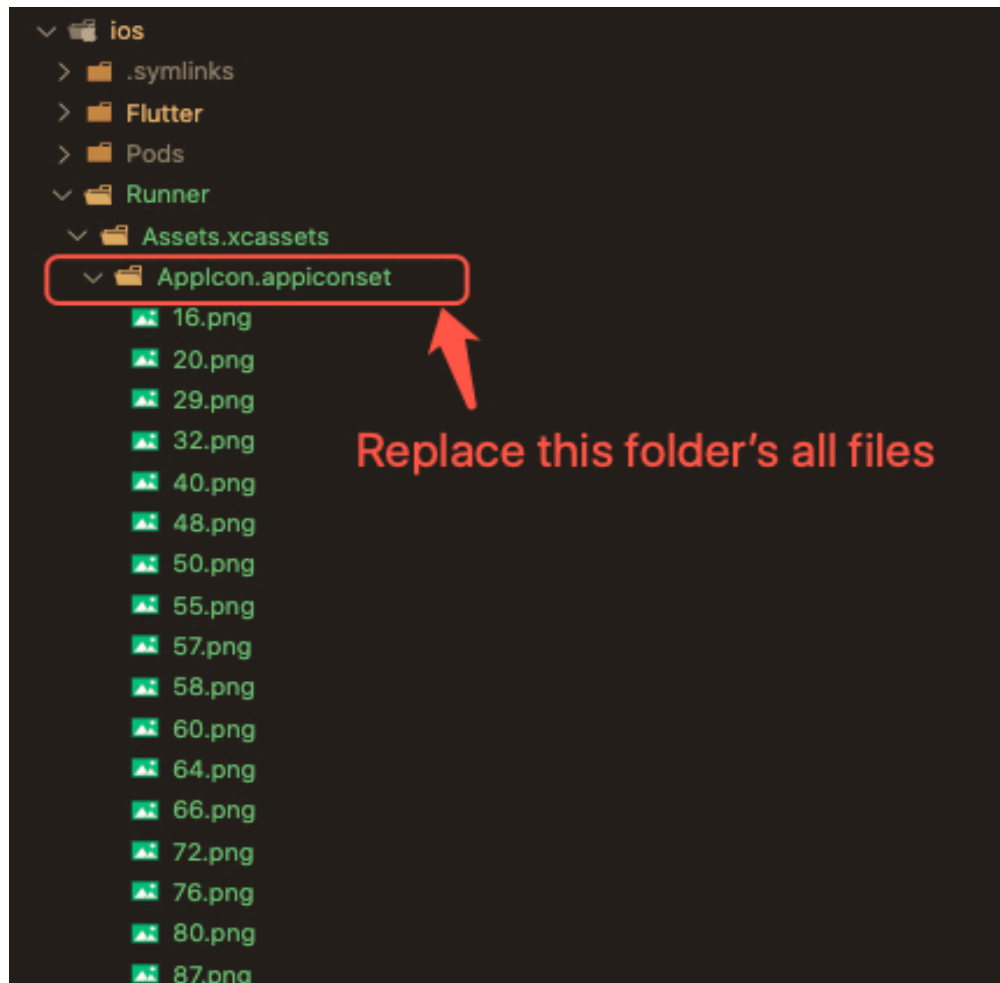
Replace the icons in the **android\app\src\main\res** folder as shown in the below image.



### ii. For iOS

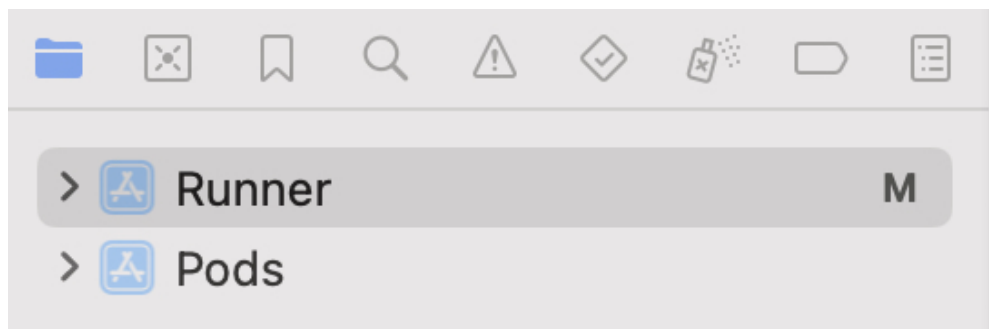
1. Replace the icons in the below folder as shown in the below image

**ios\Runner\Assets.xcassets\AppIcon.appiconset**



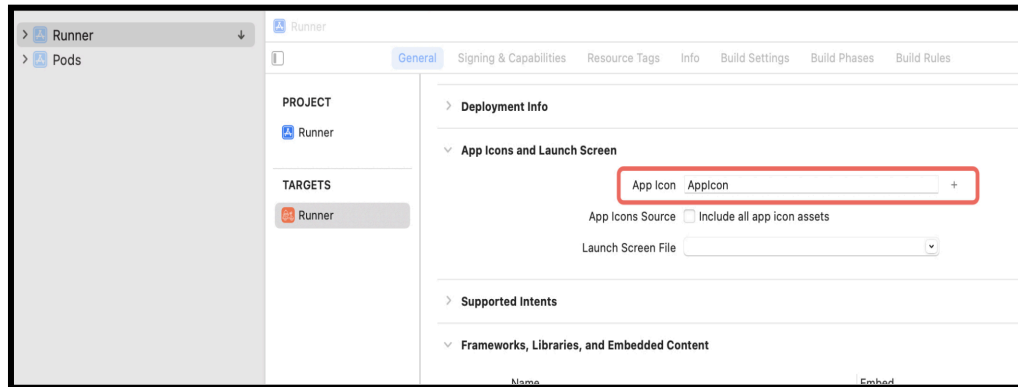
## 2. Change icons using XCode

- a. Right-click on the iOS folder Choose Open in Xcode Option
- b. Click on the folder icon on the left side of the XCode window

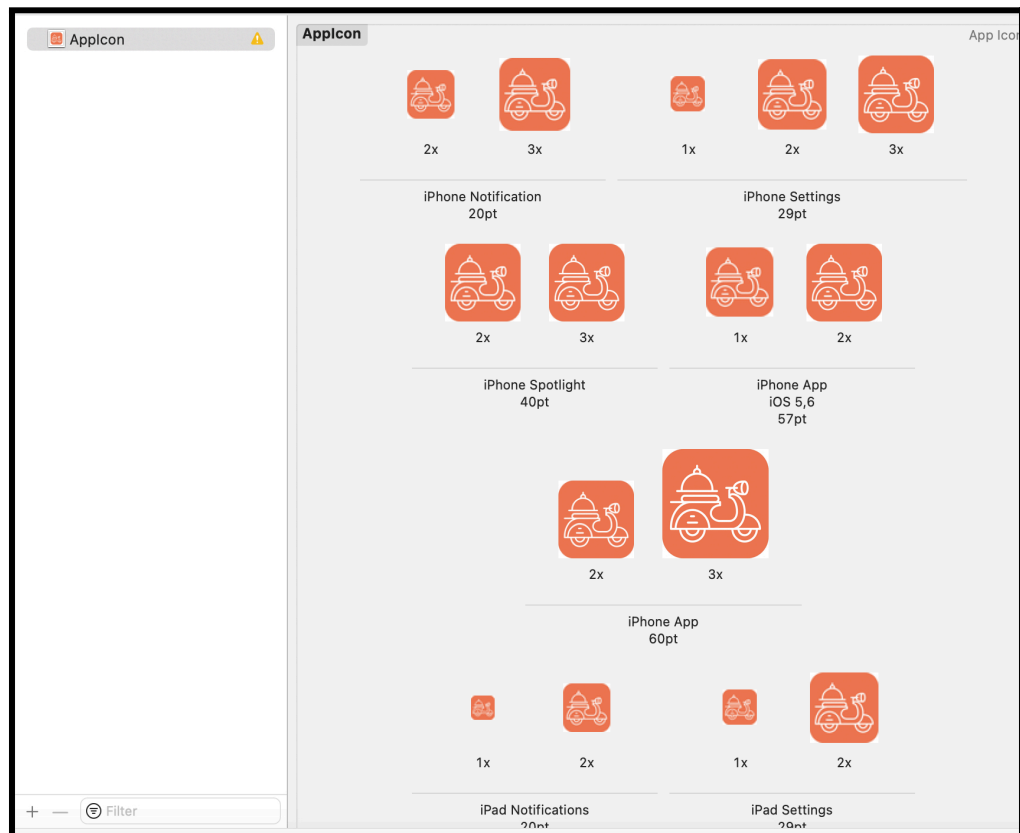


- c. Select Runner.

- d. Select Target runner
- e. Go to App Icons And Launch Images
- f. Click the right arrow button of the app icon source



- g. Replace all the icons according to their size





**NOTE:**

- If you want to generate the App icon bundle from any image you have, you can generate it from publicly available websites like

<https://www.appicon.co/>

**k. Build Release for Android**

For build release use the above complete steps from **4.l.**

**l. Build Release for iOS**

For build release use the above complete steps from **4.m.**

**m. Other Options for the Advanced User****i. Paths to the images used in the app**

| Images                  | Path                                                                                       | Screen Path                            |
|-------------------------|--------------------------------------------------------------------------------------------|----------------------------------------|
| Splash screen           | assets\images\delivery.gif                                                                 | lib\views\splash\splashScreen.dart     |
| Login Screen            | assets\images\splashLogo.jpg                                                               | lib\views\user\loginScreen.dart        |
| Order Detail Screen     | assets\images\noProduct.png                                                                | lib\views\Order\orderDetailScreen.dart |
| Home Screen             | assets\images\todayOrder.png<br>assets\images\cashInYourHand.png<br>assets\images\week.png | lib\views\home\homeScreen.dart         |
| Logout pop up           | assets\images\questionMark.png                                                             | lib\views\user\profileScreen.dart      |
| OTP verification screen | assets\images\otp.png                                                                      | lib\views\user\otpScreen.dart          |

- ii. Fonts used in the app. If you want to change, you can make the changes in the **pubspec.yaml** file and the **Assets** folder.

|        |                                                                                                                                    |
|--------|------------------------------------------------------------------------------------------------------------------------------------|
| Gilroy | assets/fonts/Gilroy-Bold.ttf<br>assets/font/Gilroy-Medium.ttf<br>assets/font/Gilroy-Regular.ttf<br>assets/font/Gilroy-SemiBold.ttf |
|--------|------------------------------------------------------------------------------------------------------------------------------------|

- iii. Colors used in the app. If you want to change the colors you can make the changes in the file **lib/Theme/nativeTheme.dart**

| #                    | Color code |
|----------------------|------------|
| Primary color        | #FF6C44    |
| Background color     | #FFFFFF    |
| Primary light color  | #FFDDCC    |
| Secondary color      | #111A2C    |
| Text field color     | #F5F5F8    |
| Error color          | #FF1717    |
| Success color        | #27AE60    |
| Hint color           | #BBBDC1    |
| Counter color        | #898B9A    |
| Facebook color       | #0047B3    |
| Google color         | #F5F5F8    |
| Delete account color | #D74722    |
| Cancel button color  | #898481    |

iv. Packages used in the app are listed below. You can find them in **pubspec.yaml** file.

| Package Name - Version       | Description                                                                                                                                   |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| firebase_core: ^2.24.2       | To use the Firebase Core API, which enables connecting to multiple Firebase apps.                                                             |
| firebase_auth: ^4.15.3       | To use the <a href="#">Firebase Authentication API</a> .                                                                                      |
| firebase_messaging: ^14.7.9  | To use the <a href="#">Firebase Cloud Messaging API</a> .                                                                                     |
| image_picker: ^1.0.5         | For iOS and Android for picking images from the image library, and taking new pictures with the camera.                                       |
| webview_flutter: ^4.4.2      | Flutter Pinput is a package that provides an easy-to-use and customizable Pin code input field                                                |
| cached_network_image: ^3.3.0 | To show images from the internet and keep them in the cache directory.                                                                        |
| email_validator: ^2.1.17     | A simple (but correct) Dart class for validating email addresses without using RegEx. Can also be used to validate emails within Flutter apps |
| get: ^4.6.6                  | To use for state management, intelligent dependency injection, and route management quickly and practically                                   |

|                                          |                                                                                                                                                                           |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| get_storage: ^2.1.1                      | A fast, extra light and synchronous key-value in memory, which backs up data to disk at each operation                                                                    |
| connectivity_plus: ^5.0.2                | This plugin allows Flutter apps to discover network connectivity and configure themselves accordingly. It can distinguish between cellular vs WiFi connection.            |
| fl_country_code_picker: ^0.1.9+1         | A Flutter package for showing a modal that contains country dial code                                                                                                     |
| shared_preferences: ^2.2.2               | Wraps platform-specific persistent storage for simple data                                                                                                                |
| intl: ^0.19.0                            | Provides internationalization and localization facilities, including message translation, plurals and genders, date/number formatting and parsing, and bidirectional text |
| permission_handler: ^11.1.0              | This plugin provides a cross-platform (iOS, Android) API to request permissions and check their status                                                                    |
| geolocator: ^9.0.2                       | A Flutter geolocation plugin which provides easy access to platform specific location services                                                                            |
| geocoding: ^2.1.0                        | A Flutter Geocoding plugin which provides easy geocoding and reverse-geocoding features                                                                                   |
| google_maps_flutter: ^2.4.0              | To provides a <a href="#">Google Maps</a> widget                                                                                                                          |
| flutter_polyline_points: ^1.0.0          | To decodes encoded google polyline string into list of geo-coordinates suitable for showing route/polyline on maps                                                        |
| razorpay_flutter: ^1.3.4                 | For Razorpay SDK.                                                                                                                                                         |
| material_design_icons_flutter: ^7.0.7296 | The <a href="#">Material Design Icons</a> Icon pack available as set of Flutter Icons                                                                                     |
| font_awesome_flutter: ^10.5.0            | The <a href="#">Font Awesome</a> Icon pack available as a set of Flutter Icons                                                                                            |
| flutter_rating_bar: ^4.0.1               | A simple yet fully customizable rating bar which also includes a rating bar indicator, supporting any fraction of rating                                                  |
| open_filex: ^4.3.2                       | Can call native APP to open files with string result in a flutter, support iOS(DocumentInteraction) / android(intent) / PC(ffi) / web(dart:html)                          |
| flutter_local_notifications: ^15.1.0+1   | For displaying local notifications.                                                                                                                                       |
| share_plus: ^7.1.0                       | To share content from your Flutter app via the platform's share dialog                                                                                                    |
| firebase_dynamic_links: ^5.3.4           | To use the <a href="#">Firebase Dynamic Links API</a>                                                                                                                     |
| url_launcher: ^6.1.12                    | To launch a URL                                                                                                                                                           |
| image_cropper: ^4.0.1                    | For Android, iOS and Web supports cropping images                                                                                                                         |

## 7. Set up Vendor App (Technology Flutter)

### a. Initial steps to set up and run mobile app

- i. Open the **VendorApp** folder in the VSCode
- ii. To complete the other setup, please follow the instructions from above steps **4.a.ii** to **4.a.vi**.

## b. Change API base URL

After the setup of your API and Admin panel, you have to change your API base URL for that, go to the file located at **lib\utils\global.dart**

```
String appMode = "LIVE";

Map<String, dynamic> appParameters = {
 "LIVE": {
 "apiUrl": "https://foodivery.native.software/api/",
 },
 "DEV": {
 "apiUrl": "http://192.168.29.118:8080/api/",
 }
};
```

## c. Change App Name

- i. Change the app name in the Android App
  1. Change the app name in the file located at **android/app/src/main/AndroidManifest.xml**

```
<application
 android:label="Foodivery Vendor"
 android:name="${applicationName}"
 android:usesCleartextTraffic="true"
```

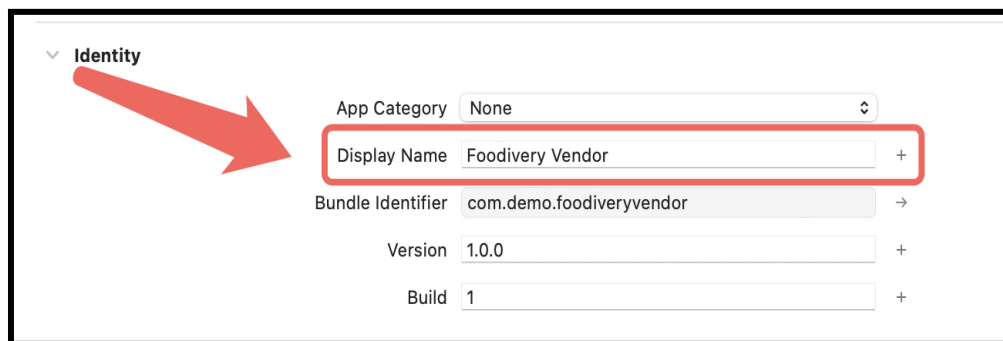
- ii. Change the app name in the iOS App

1. In VSCode
  - a. Go to **ios/Runner/info.plist**
  - b. Change string of key **CFBundleDisplayName**

```
<string>$(DEVELOPMENT_LANGUAGE)</string>
<key>CFBundleDisplayName</key>
<string>Foodivery Vendor</string>
<key>CFBundleExecutable</key>
<string>$(EXECUTABLE_NAME)</string>
```

## 2. In XCode

- a. Right-click on the **iOS** folder and Choose Open in Xcode Option
- b. Click on the folder icon left side of the XCode window
- c. Select Runner.
- d. Select Target runner
- e. From the General Tab Go to the identity
- f. Change Display Name



### d. Change Package Name/Bundle ID

An app's package name is a unique identifier that is automatically created when you create an app. The term used for iOS apps is "**bundle ID**"; for Android apps, it is "**package name**".

#### i. Set Package Name for Android App

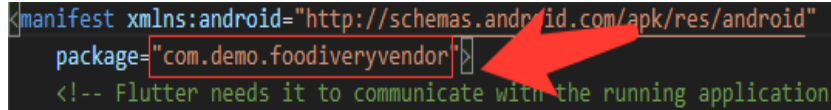
1. Change the package name in the file located at **android/app/src/main/AndroidManifest.xml**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.demo.foodiveryvendor"
 <uses-permission android:name="android.permission.CAMERA" />
```

2. Change the package name in the file located at **android/app/src/debug/AndroidManifest.xml**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.demo.foodiveryvendor"
 <!-- Flutter needs it to communicate with the running application
```

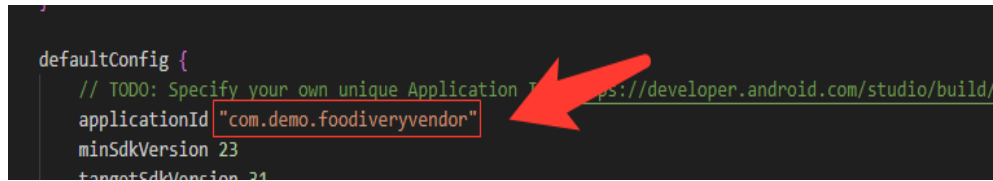
3. Change Package Name in file which is located at **android/app/src/Profile/AndroidManifest.xml**



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.demo.foodiveryvendor"
 <!-- Flutter needs it to communicate with the running application
```

A screenshot of the `android/build.gradle` file. The package name `com.demo.foodiveryvendor` is highlighted with a red box, and a red arrow points to it from the right.

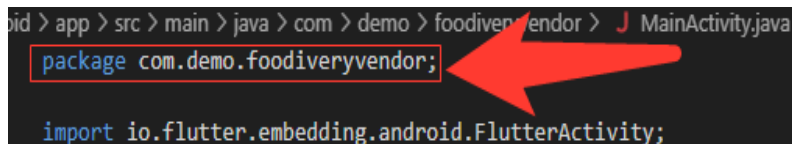
4. Change the Package Name in the file which is located at **android/app/build.gradle**



```
defaultConfig {
 // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/applicationId)
 applicationId "com.demo.foodiveryvendor"
 minSdkVersion 23
 targetSdkVersion 24
```

A screenshot of the `android/app/build.gradle` file. The `applicationId` value `"com.demo.foodiveryvendor"` is highlighted with a red box, and a red arrow points to it from the right.

5. Change the folder structure for the below path as per your package name.  
**android\app\src\main\java\com\demo\foodiveryvendor\**
6. Change Package Name in file which is located at **android\app\src\main\java\com\demo\foodiveryvendor\MainActivity.java**



```
package com.demo.foodiveryvendor;

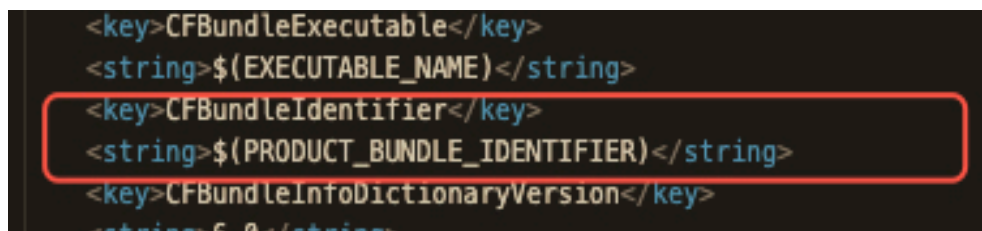
import io.flutter.embedding.android.FlutterActivity;
```

A screenshot of the `MainActivity.java` file. The package declaration `package com.demo.foodiveryvendor;` is highlighted with a red box, and a red arrow points to it from the right.

- ii. Set Bundle ID for iOS App

1. In VSCode

- a. Go to **ios/Runner/info.plist**
- b. Change the string of key **CFBundleIdentifier**



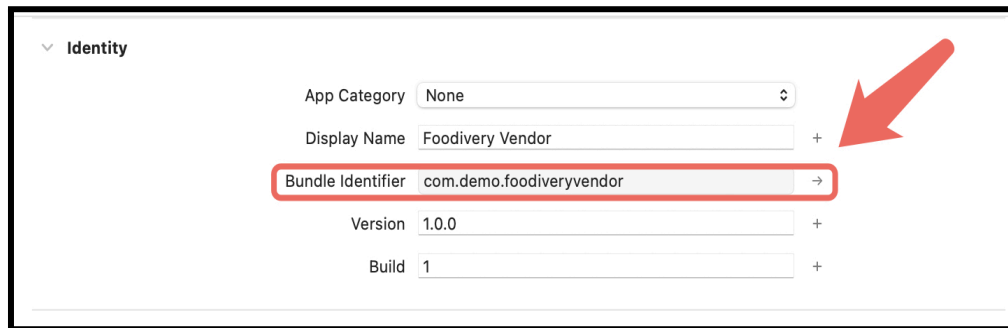
```
<key>CFBundleExecutable</key>
<string>$(EXECUTABLE_NAME)</string>
<key>CFBundleIdentifier</key>
<string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
<key>CFBundleInfoDictionaryVersion</key>
<string>5.0</string>
```

A screenshot of the `ios/Runner/info.plist` file. The `CFBundleIdentifier` key and its corresponding value `$(PRODUCT_BUNDLE_IDENTIFIER)` are highlighted with a red box.

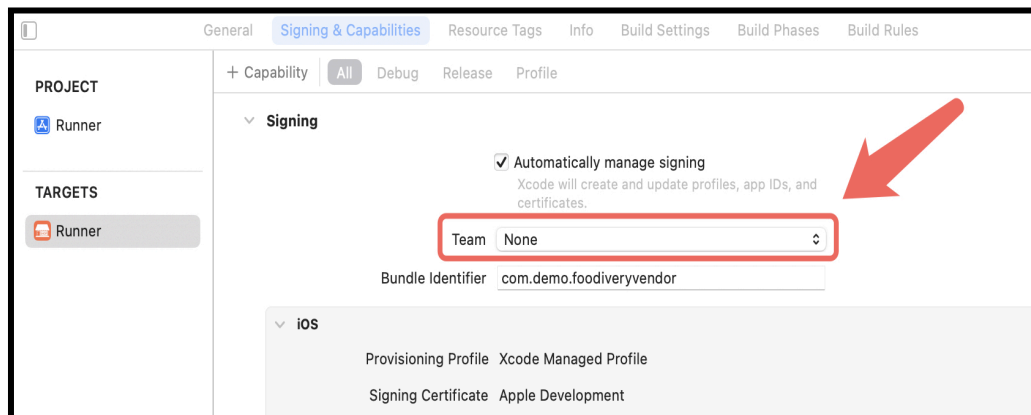
2. In XCode

- a. Right-click on the **iOS** folder and Choose Open in Xcode Option
- b. Click on the folder icon on the left side of the XCode window

- c. Select Runner.
- d. Select Target Runner
- e. Go to identity
- f. Change Bundle Identifier



- g. In Signing & Capabilities Go to Signing
- h. Change Bundle Identifier



- e. Create and set the Keystore file for Android

To complete this setup, please follow the instructions from steps **4.e.i** to **4.e.iv** above.

- f. Create Firebase Account & Project

You can use the same account you created for the Customer App. ( If you want to create a different account for this App, then follow the above steps from **4.f.** )

- g. Set up Android App in Firebase Project

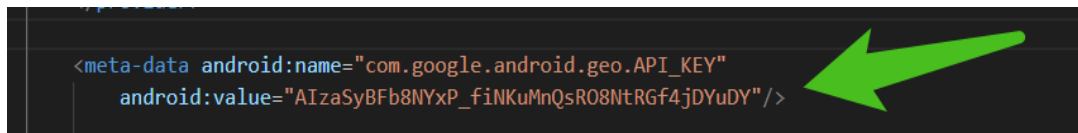
For this setup you can follow the complete steps mentioned in **4.g**.

h. To add Google Maps in the app

i. For this setup you can refer to the steps from **4.i.i** to **4.i.iii**.

ii. To add Google Maps API key in Android:

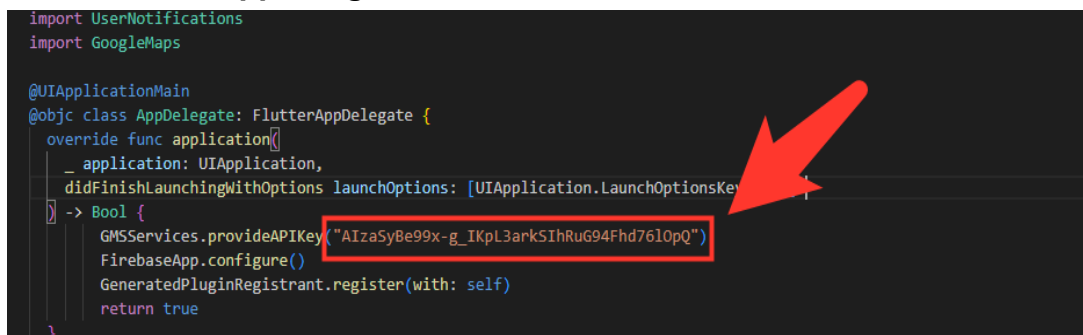
**android/app/src/main/AndroidManifest.xml**



```
<meta-data android:name="com.google.android.geo.API_KEY"
 android:value="AIzaSyBFb8NYxP_fiNKuMnQsR08NtRGf4jDYuDY" />
```

iii. To add Google Maps API key in iOS:

**ios\Runner\AppDelegate.swift**



```
import UserNotifications
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
 override func application(
 _ application: UIApplication,
 didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey]?)
 -> Bool {
 GMSServices.provideAPIKey("AIzaSyBe99x-g_IKpL3arkSIhRuG94Fhd76l0pQ")
 FirebaseApp.configure()
 GeneratedPluginRegistrant.register(with: self)
 return true
 }
}
```

i. Set up Firebase iOS App

i. Follow the steps from **4.j.i** to **4.j.vii**.

ii. XCode Project Setting

This step covers reviewing the most important settings in the XCode workspace. For detailed procedures and descriptions, see [Prepare for App Distribution](#)

3. Navigate to your target's settings in XCode:

a. Open the default Xcode workspace in your project by running the below command in a terminal window from your Flutter project directory.

**open ios/Runner.xcworkspace**

b. To view your app's settings, select the Runner target in the Xcode



navigator.

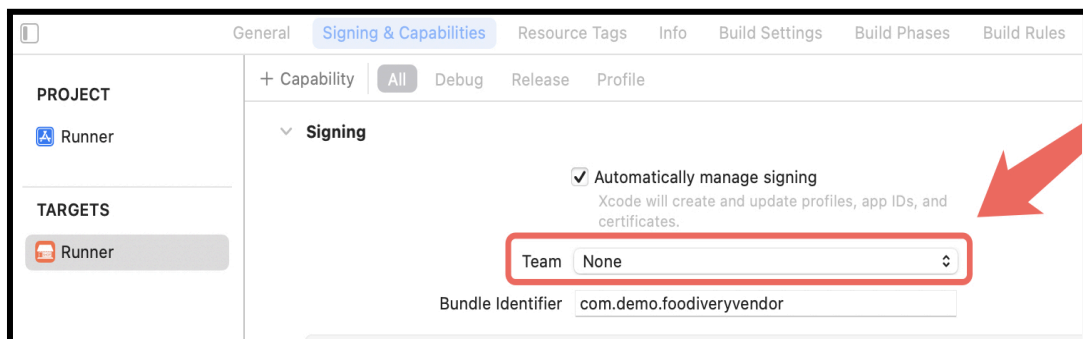
4. Verify the most important settings

a. In the Identity section of the General tab

- i. **Display Name** (The display name of your app.)
- ii. **Bundle Identifier** (The App ID you registered on App Store Connect.)

b. In the Signing & Capabilities tab

- i. **Automatically manage signing** (Xcode should automatically manage app signing and provisioning. This is set true by default, which should be sufficient for most apps. For more complex scenarios, see the [Code Signing Guide](#))
- ii. **Team** (Select the team associated with your registered Apple Developer account. If required, select Add Account..., then update this setting.)

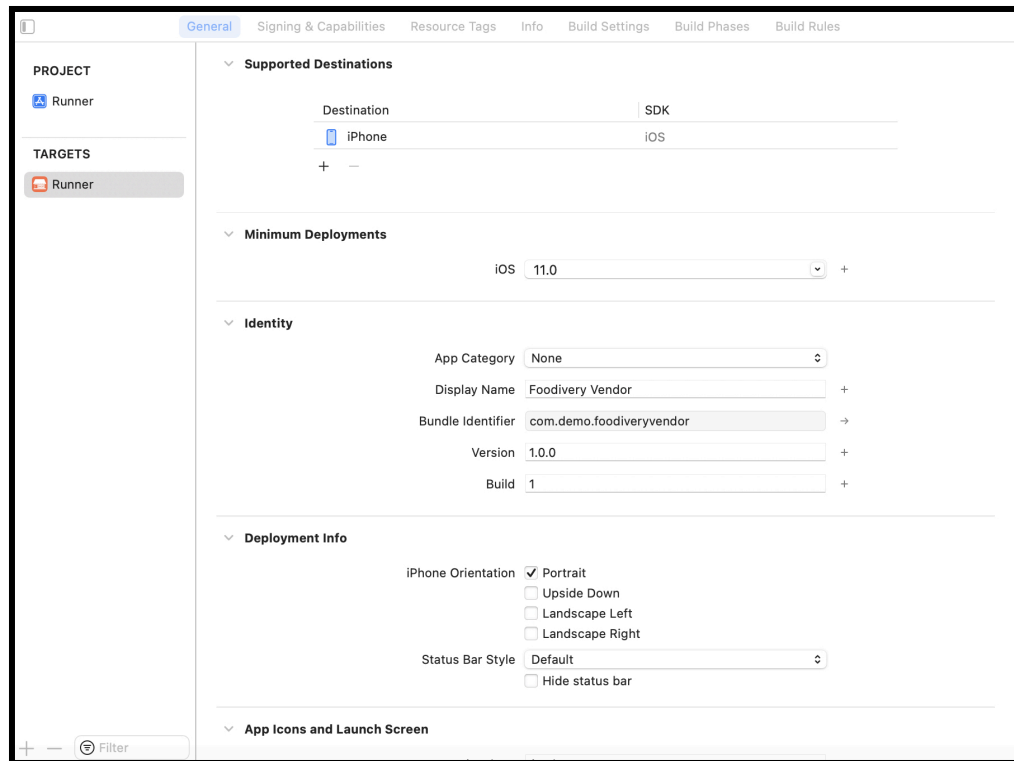


c. In the deployment section of the build settings tab:

i. iOS Deployment Target

4. The minimum iOS version that the app supports is 11.0.

5. The General tab of your project settings should resemble the following:

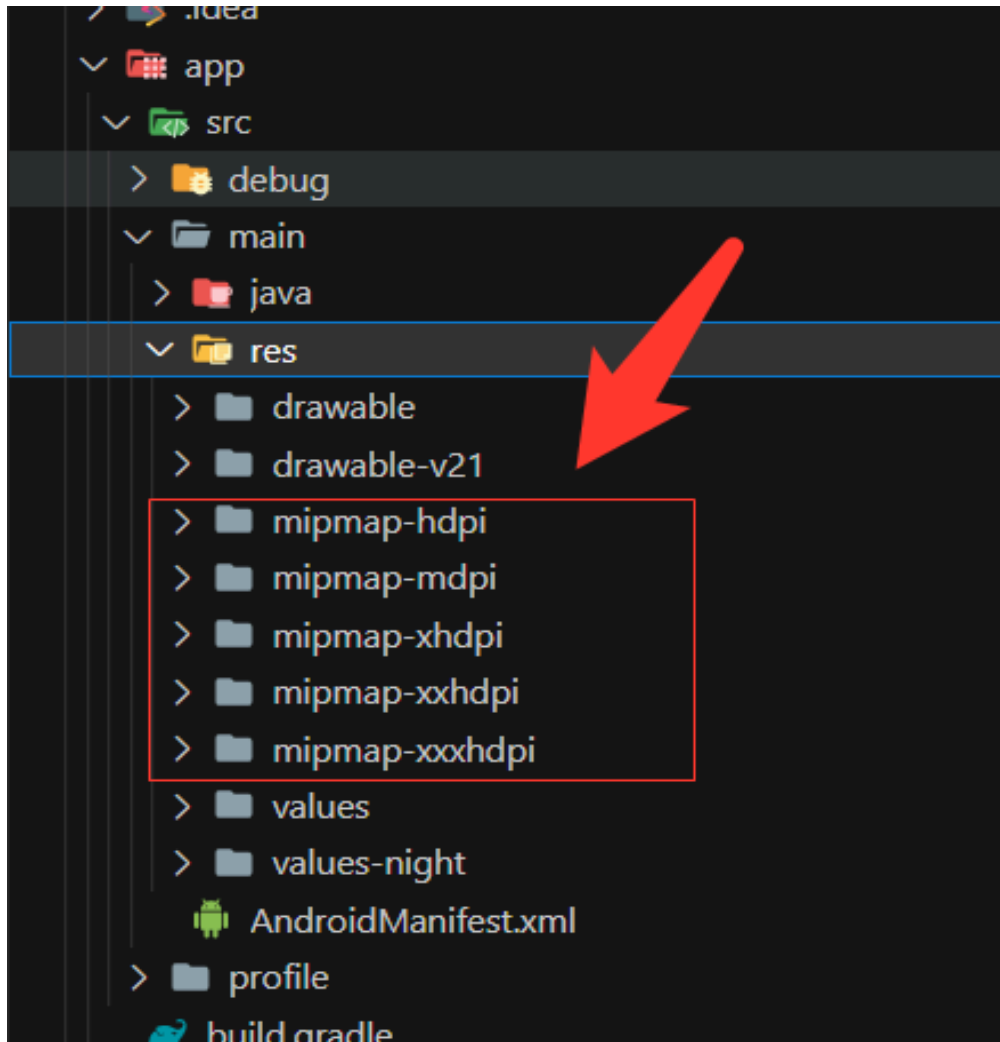


6. For a detailed overview of app signing, see [Create, export, and Delete signing certificates](#).

## j. Change App Icon

### i. For Android

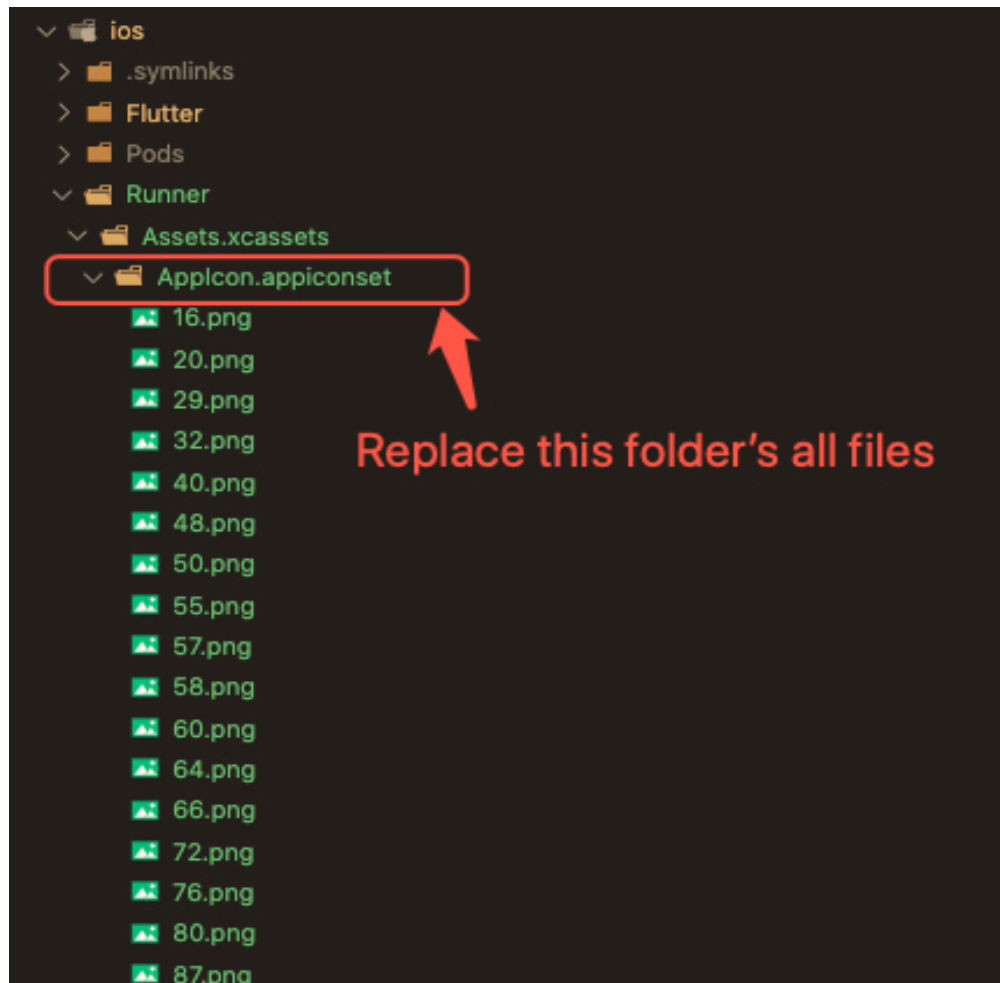
Replace the icons in the **android\app\src\main\res** folder as shown in the below image.



### ii. For iOS

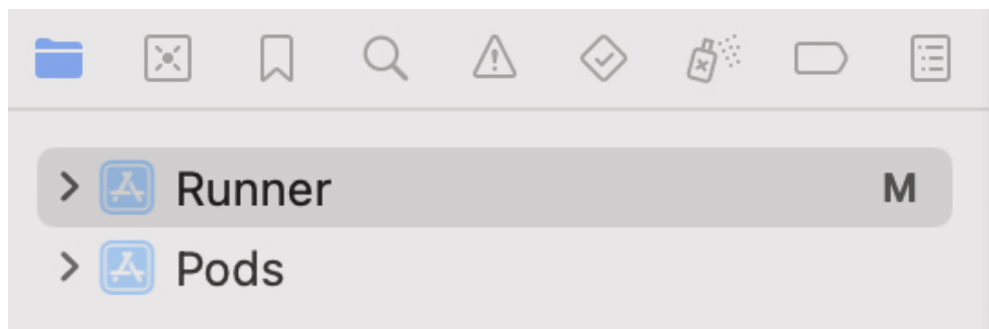
1. Replace the icons in the below folder as shown in the below image

**ios\Runner\Assets.xcassets\AppIcon.appiconset**



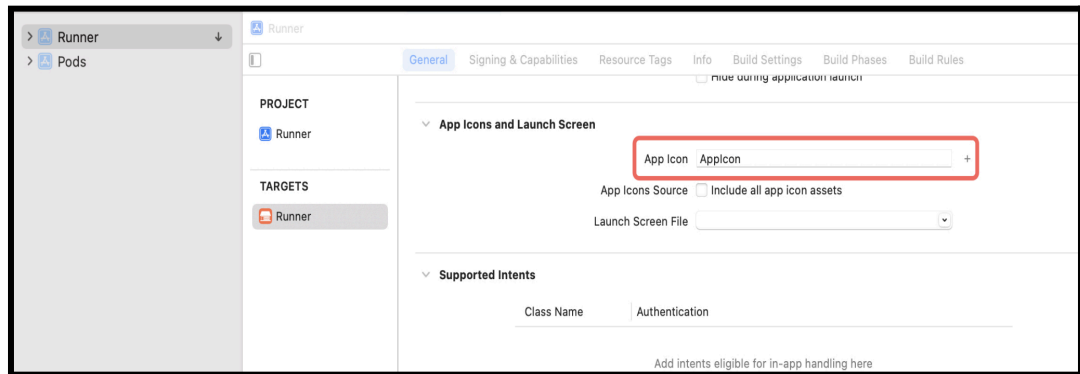
## 2. Change icons using XCode

- a. Right-click on the iOS folder Choose Open in Xcode Option
- b. Click on the folder icon on the left side of the XCode window

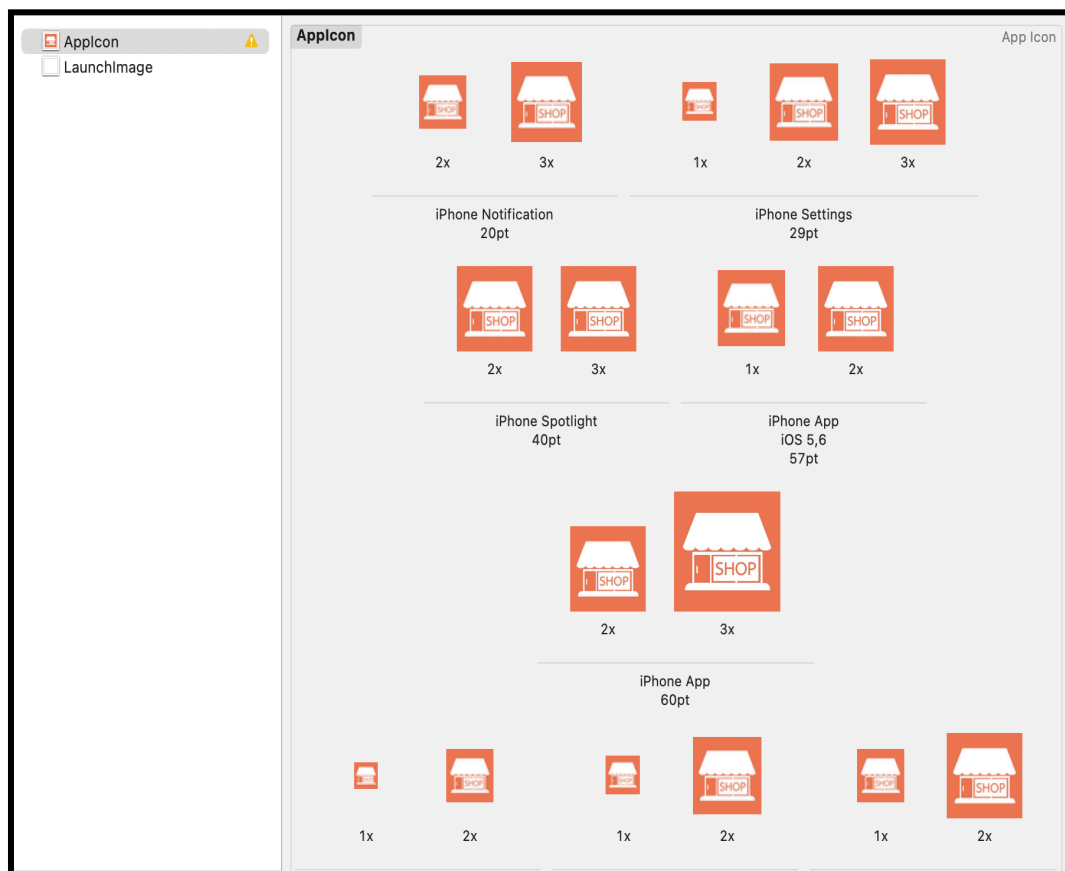


- c. Select Runner.

- d. Select Target runner
- e. Go to App Icons And Launch Images
- f. Click the right arrow button of the app icon source



- g. Replace all the icons according to their size



**NOTE:**

- If you want to generate the App icon bundle from any image you have, you can generate it from publicly available websites like

<https://www.appicon.co/>

**k. Build Release for Android**

For build release use the above complete steps from **4.l.**

**l. Build Release for iOS**

For build release use the above complete steps from **4.m.**

**m. Other Options for the Advanced User****i. Paths to the images used in the app**

| Images                  | Path                           | Screen Path                         |
|-------------------------|--------------------------------|-------------------------------------|
| Splash screen           | assets\images\splash.png       | lib\views\splash\splashScreen.dart  |
| Sign up screen          | assets\images\storeLogo.png    | lib\views\user\signUpScreen.dart    |
| Profile screen          | assets\images\profile.jpg      | lib\views\user\profileScreen.dart   |
| OTP verification screen | assets\images\otp.png          | lib\views\user\otpScreen.dart       |
| Order history screen    | assets\images\noOrderImage.png | lib\views\Order\myOrdersScreen.dart |

- ii. Fonts used in the app. If you want to change, you can make the changes in the **pubspec.yaml** file and the **Assets** folder.

|        |                                                                 |
|--------|-----------------------------------------------------------------|
| Gilroy | assets/font/Gilroy-Medium.ttf<br>assets/font/Gilroy-Regular.ttf |
|--------|-----------------------------------------------------------------|

- iii. Colors used in the app. If you want to change the colors you can make the changes in the file **lib/Theme/nativeTheme.dart**

| #                | Color code |
|------------------|------------|
| Primary color    | #FF6C44    |
| Background color | #FFFFFF    |

|                     |         |
|---------------------|---------|
| Primary light color | #FFDDCC |
| Secondary color     | #111A2C |
| Text field color    | #F5F5F8 |
| Error color         | #FF1717 |
| Success color       | #27AE60 |
| Hint color          | #BBBDC1 |
| Counter color       | #898B9A |
| Facebook color      | #0047B3 |
| Google color        | grey    |

- iv. Packages used in the app are listed below. You can find them in **pubspec.yaml** file.

| Package Name - Version       | Description                                                                                                                                                    |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| firebase_core: ^2.24.2       | To use the Firebase Core API, which enables connecting to multiple Firebase apps.                                                                              |
| firebase_auth: ^4.15.3       | To use the <a href="#">Firebase Authentication API</a> .                                                                                                       |
| firebase_messaging: ^14.7.9  | To use the <a href="#">Firebase Cloud Messaging API</a> .                                                                                                      |
| image_picker: ^1.0.5         | For iOS and Android for picking images from the image library, and taking new pictures with the camera.                                                        |
| webview_flutter: ^4.4.2      | Flutter Pinput is a package that provides an easy-to-use and customizable Pin code input field                                                                 |
| cached_network_image: ^3.3.0 | To show images from the internet and keep them in the cache directory.                                                                                         |
| email_validator: ^2.1.17     | A simple (but correct) Dart class for validating email addresses without using RegEx. Can also be used to validate emails within Flutter apps                  |
| get: ^4.6.6                  | To use for state management, intelligent dependency injection, and route management quickly and practically                                                    |
| connectivity_plus: ^5.0.2    | This plugin allows Flutter apps to discover network connectivity and configure themselves accordingly. It can distinguish between cellular vs WiFi connection. |
| http: ^1.1.2                 | A composable, Future-based library for making HTTP requests                                                                                                    |

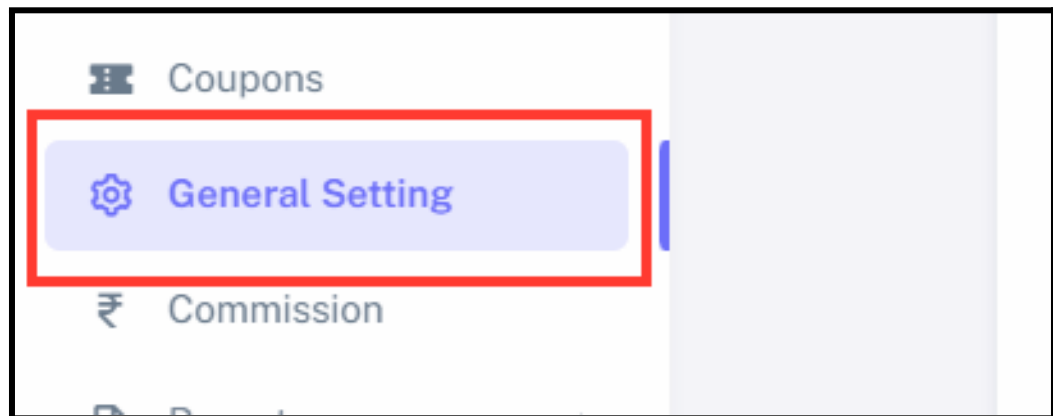
|                                      |                                                                                                                                                                           |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| country_code_picker: ^3.0.0          | A flutter package for showing a country code selector                                                                                                                     |
| shared_preferences: ^2.2.2           | Wraps platform-specific persistent storage for simple data                                                                                                                |
| intl: ^0.19.0                        | Provides internationalization and localization facilities, including message translation, plurals and genders, date/number formatting and parsing, and bidirectional text |
| permission_handler: ^11.1.0          | This plugin provides a cross-platform (iOS, Android) API to request permissions and check their status                                                                    |
| geolocator: ^10.1.0                  | A Flutter geolocation plugin which provides easy access to platform specific location services                                                                            |
| geocoding: ^2.1.1                    | A Flutter Geocoding plugin which provides easy geocoding and reverse-geocoding features                                                                                   |
| google_maps_flutter: ^2.5.0          | To provides a <a href="#">Google Maps</a> widget                                                                                                                          |
| font_awesome_flutter: ^10.6.0        | The <a href="#">Font Awesome</a> Icon pack available as set of Flutter Icons                                                                                              |
| flutter_rating_bar: ^4.0.1           | A simple yet fully customizable rating bar which also include a rating bar indicator, supporting any fraction of rating                                                   |
| flutter_local_notifications: ^16.2.0 | For displaying local notifications.                                                                                                                                       |
| image_cropper: ^5.0.1                | For Android, iOS and Web supports cropping images                                                                                                                         |
| firebase_analytics: ^10.7.4          | A Flutter plugin to use the <a href="#">Firebase Analytics API</a> .                                                                                                      |
| animated_snack_bar: ^0.4.0           | To show beautiful animated snackbars directly using overlay                                                                                                               |
| pointer_interceptor: ^0.10.0         | PointerInterceptor is a widget that prevents mouse events (in web) from being captured by an underlying <a href="#">HtmlElementView</a>                                   |
| date_format: ^2.0.7                  | A simple API to format dates                                                                                                                                              |
| flutter_animated_button: ^2.0.3      | A flutter package project which contains a collection of cool and beautiful button animations                                                                             |
| dotted_border: ^2.1.0                | A flutter package to easily added dotted borders around widgets                                                                                                           |
| location: ^5.0.3                     | This plugin for <a href="#">Flutter</a> handles getting a location on Android and iOS. It also provides callbacks when the location is changed                            |
| dotted_line: ^3.2.2                  | This package allows you to draw dotted lines with Flutter                                                                                                                 |
| jiffy: ^5.0.1                        | A date time package for parsing, manipulating, querying and formatting dates and time                                                                                     |
| device_info_plus: ^9.1.1             | Get current device information from within the Flutter application                                                                                                        |
| slider_button: ^2.0.0                | Customizable slider button widget for activating/deactivating some event                                                                                                  |
| open_app_settings: ^2.0.1            | Open App setting page by ObjC and java code                                                                                                                               |



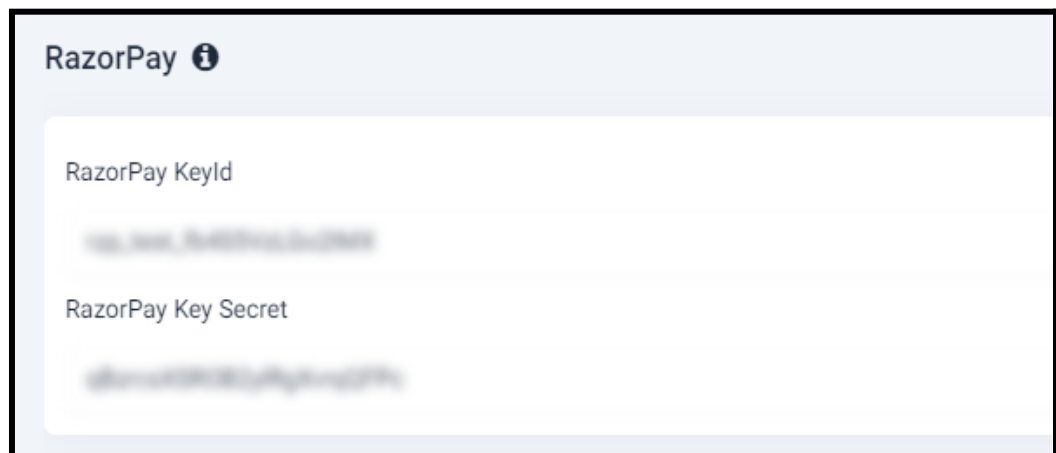
## 8. Payment Integration

### a. Set up RazorPay (Payment Gateway)

- i. Set up RazorPay from [this link](#).
- ii. After generating Razorpay KeyId and Razorpay Secret Key from the link, Set up them up in the **Admin Panel**.
- iii. In the **Admin Panel** Go to the General setting from the menu.



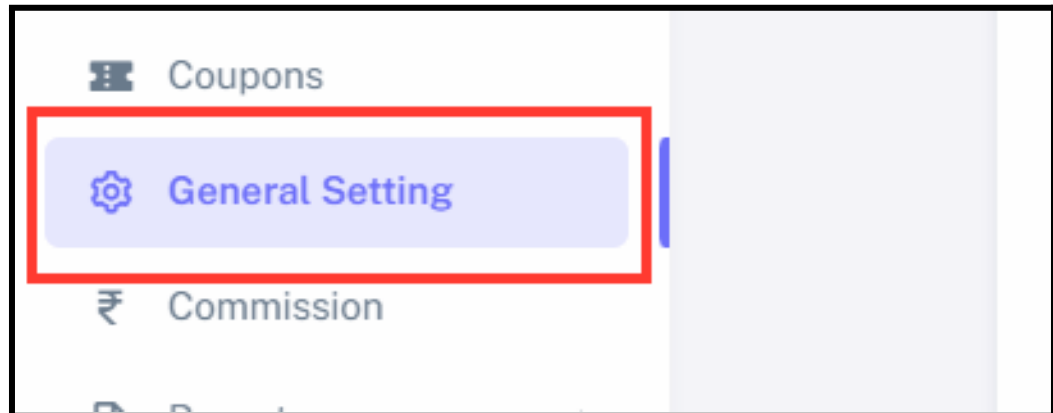
- iv. Then go to the payment Tab
- v. In the Razorpay Section add your RazorPay KeyId and RazorPay Secret Key.



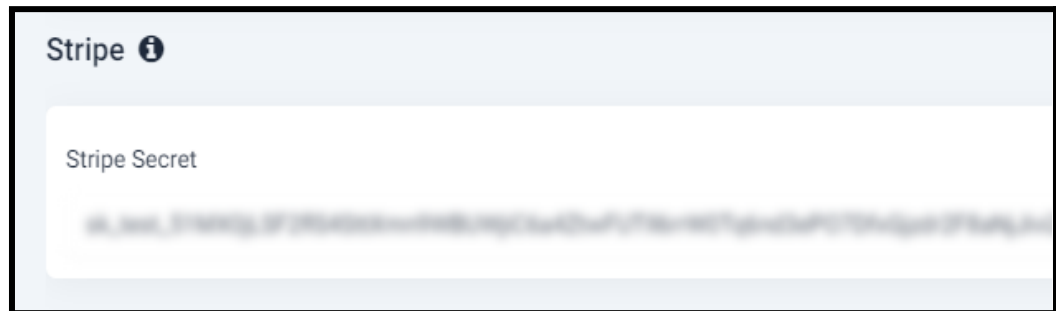
### b. Set up Stripe (Payment Gateway)

- i. Set up Stripe from [this link](#).
- ii. After generating the Stripe Secret key from the link, Set it up in the **Admin Panel**.

- iii. In the **Admin Panel** Go to the General setting from the menu.

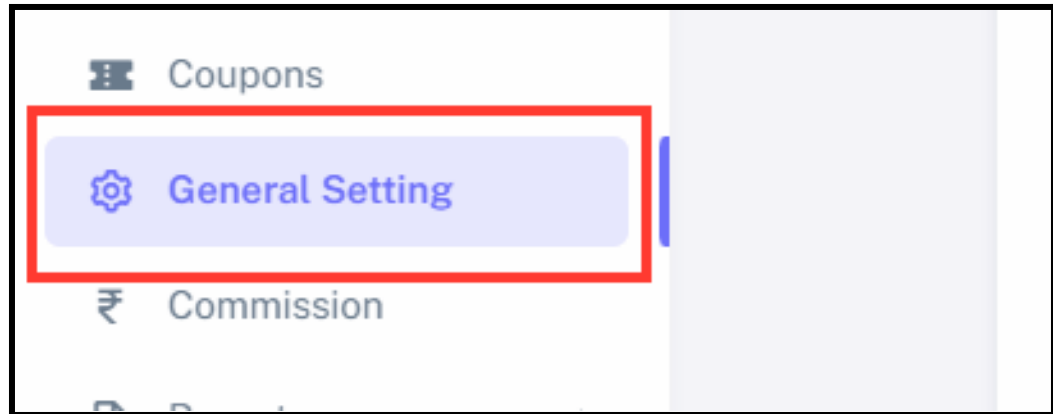


- iv. Then go to the payment tab.
- v. In the Stripe section add your Stripe Secret Key.



### c. Set up PhonePe (Payment Gateway)

- i. Set up PhonePe from [this link](#).
- ii. Add PhonePe MerchantId, Redirect Mode, Salt Key, Salt Index, and Environment Value from the link in the **Admin Panel**.
- iii. In the **Admin Panel** Go to the General setting from the menu.



- iv. Then go to the payment Tab.
- v. In the PhonePe section add your MerchantId, Redirect Mode, Salt Key, Salt Index, Environment Value, and API End Point.

A screenshot of the 'PhonePe' payment setup form. The form contains several input fields for the following fields: Merchant Id, Redirect Mode, Salt Key, Salt Index, Environment Value, and Api End Point. Each field has a corresponding label and a text input area.

## USEFUL LINKS

- To set up Laravel from scratch you can use [this link](#)
- To set up MySQL database you can use [this link](#)
- For more information on iOS refer to [this link](#)

This document was last updated on 22 Aug 2024.