



Motors – Car Dealership & Classified Listings Mobile App for Android & IOS Documentation

Also available [Online manual](#) and [Video tutorials](#).

Table of Contents

Getting Started	4
Introduction.....	4
Motors Application Plugin	5
Installation of the plugin	5
Motors Application Plugin Settings	6
Global Settings	6
Main Page.....	7
Add a Car	8
AdMob.....	8
Inventory	9
Translations	12
Mobile Application Build	13
Prerequisites and Environment.....	13
Step 1 – Open the app folder.....	13
Step 2 – Install Node.js and NPM.....	14
Step 3 – Run npm install commands.....	15
Step 4 – Run react-native commands.....	16
Step 5 – Put your site link	18
Step 6 – Upload your logo.....	18
Google AdMob integration.....	19
Push Notification Setup.....	21
Android Application Build.....	22
Step 1 – Set up environment	22
Step 2 – Application Name.....	24
Step 3 – Upload app icon	25
Step 4 – Change package name	26
Step 5 – Change “import” name	29
Step 6 – Edit build.gradle file	30
Step 7 – Edit AndroidManifest file	30
Android App Release	32
Step 1 – Test your app	32
Step 2 – Generate Android App Bundle	35
Step 2.1 – Generate APK	40
IOS App Build	41
Prerequisites	41

Step 1 – App Registration on App Store Connect	41
Step 2 – Register a Bundle ID.....	41
Step 3 – Create an app	44
Step 4 – Prepare for app distribution	44
Step 5 – Setting up the project	44
Step 6 – Add an App Icon	47
iOS App Release	49
Step 1 – Create a build archive	49
Step 2 – Upload an app to App Store Connect	51

Getting Started

Introduction

Motors mobile app is a new extension to Motors WordPress theme. It is the next step on vehicles classified advertisement. This application is built on React Native and that makes it so resilient and cross-platform.

Motors app contains a vast amount of marvelous features to buy and sell cars.

- Add listings - no need to add a car from inside the web. Submit all the needed information and create vehicles ad page instantly.
- Simple car search - keep a car search simple to make the process fast and smooth. In Motors App it's easy to search for cars by make or model.
- Simple Authentication - WP users can easily sign in to Motors App with WP credentials. And new users can register right in the app and later authorize on WP website.
- Integration plugin - Motors app comes with a special WordPress plugin that you will have to install and activate in order to make changes to your application from one place inside WordPress
- Your Personal Pocket Business - You can create an unlimited number of apps for a different type of clients

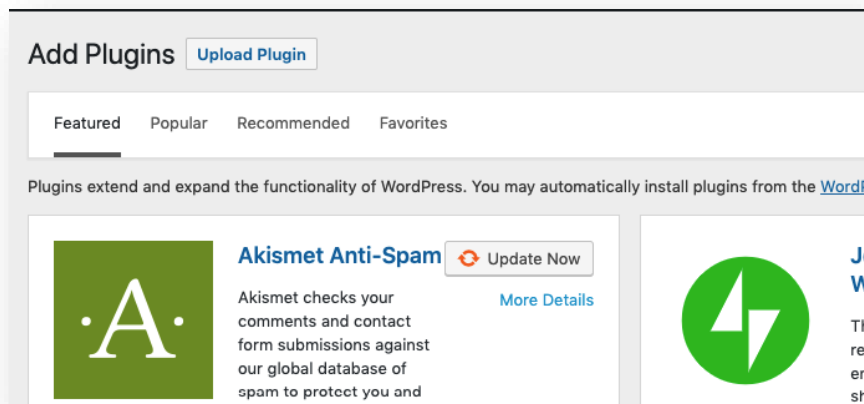
Motors Application Plugin

Installation of the plugin

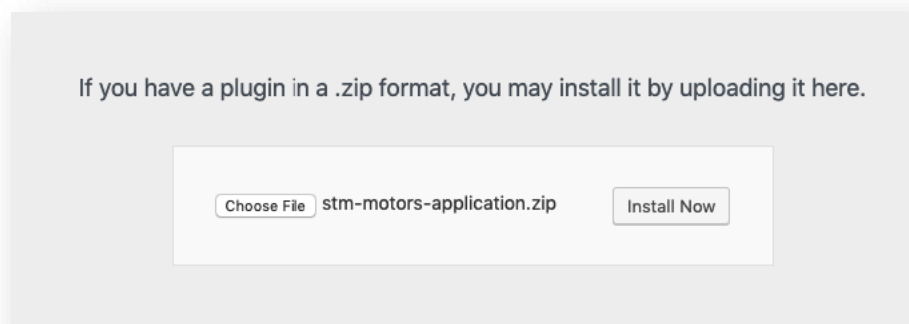
The Mobile App extension comes with the special **Motors Application** plugin. The plugin needs to customize the mobile app features such as color scheme, Google Services API keys, placeholder image, Inventory settings, and Add Car options.

To install the Motors Application plugin:

1. Click on the **Dashboard > Plugins > Add New** section of WordPress menu.

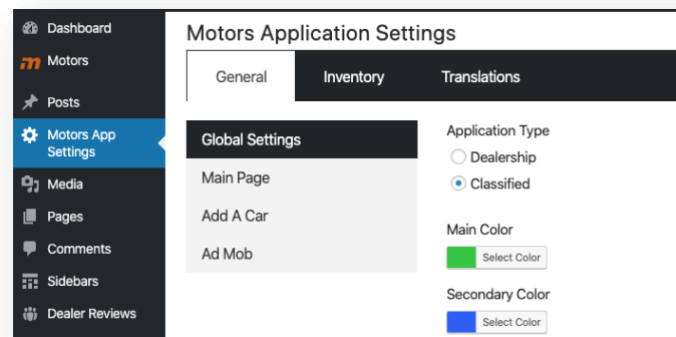


2. Click on the **Upload Plugin** button and upload the zipped plugin file you have received from ThemeForest.



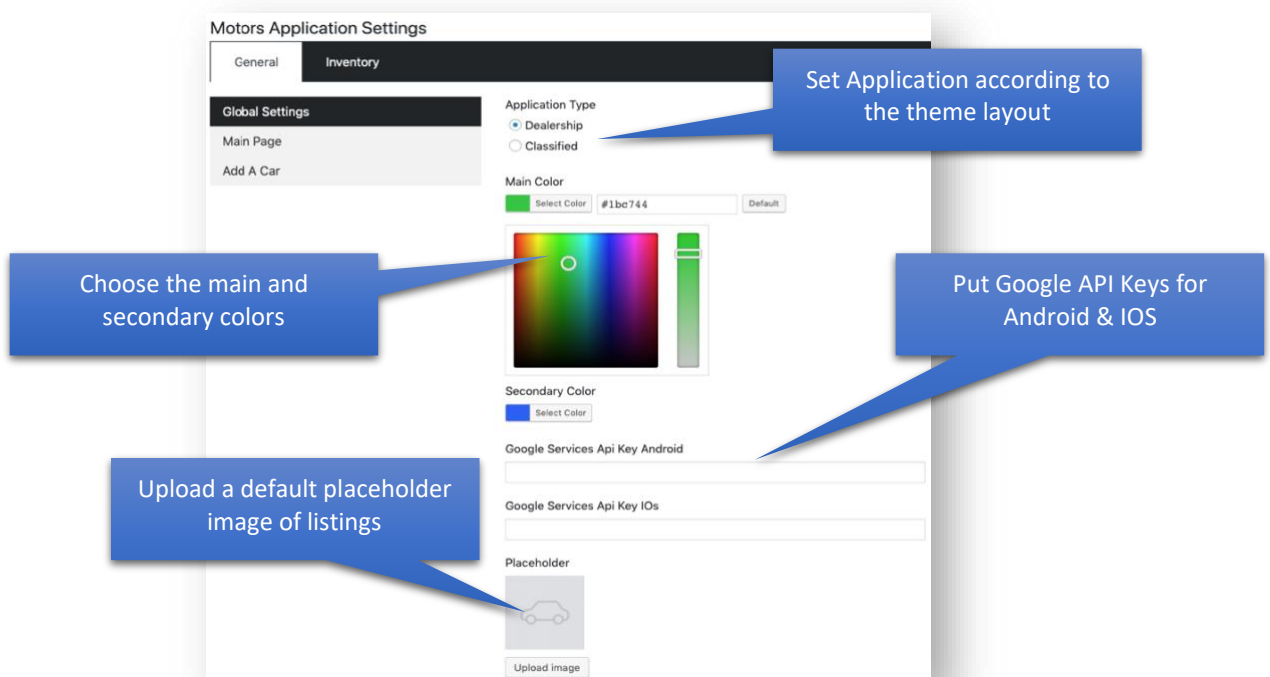
Motors Application Plugin Settings

After installation of the **Motors Application** plugin, the new **Motors App Settings** section appears in WordPress dashboard menu. The plugin allows to change color scheme of the mobile app, set up Google Services, inventory layout and much more.



Global Settings

Global settings allow you set up the general styling of the mobile application visually. Here you can customize the appearance of the mobile app with the following options:



- **Application Type:** You can select the application type according to the active layout of the theme;
- **Main & Secondary Color:** You can choose the main and secondary color of the application;
- **Google Services:** You can put API keys of Google Services for Android & IOS;
- **Placeholder:** You can upload a default placeholder image for listings without images.

Tip: For detailed overview of getting Google API key, see the [Get an API Key for Android](#) and [iOS](#) guides

Main Page

This section includes features for setting up a number of listings per page in Recommended Posts and Recently Added Posts modules. Also, you can select the view type of the Recently Added Posts module. Those modules appear on main page of your mobile app.

The screenshot shows the 'Motors Application Settings' window with the 'Inventory' tab selected. The left sidebar contains 'Global Settings', 'Main Page', and 'Add A Car'. The main content area has three settings: 'Recommended Listings Per Page' (set to 10), 'Recently Added Listings Per Page' (set to 10), and 'View Type for Recently Added se' (set to 'List View'). Three blue callout boxes provide instructions: one points to the 'Recommended Listings Per Page' input field with the text 'Set a number of recommended listings per page'; another points to the 'Recently Added Listings Per Page' input field with the text 'Set a number of recently added listings per page'; and a third points to the 'View Type for Recently Added se' dropdown menu with the text 'Choose a view type of the recently added listings (List/Grid)'.

Add a Car

The Add a Car section allows to set up steps of the add listing process in mobile application. You can select the options which will be displayed in the certain steps of the Add a Car page.

Motors Application Settings

General Inventory

Global Settings

Main Page

Add A Car

Step One

Location
Condition
Body
Fuel type
Engine
Price
Fuel consumption
Transmission

Exterior Color
Add Media
Make
Model
Mileage
Year

Step Two

Add Media
Condition
Make
Model
Mileage
Year
Fuel consumption
Fuel economy

Drive
Location
Body
Fuel type
Engine
Price
Transmission

Step Three

Add Media
Location
Condition
Body
Make
Model
Mileage
Fuel type

Seller notes

Select options which will be displayed in the step One, Two, and Three respectively

AdMob

The AdMob section incorporates settings for using the Google AdMob platform. Google AdMob is an easy way to monetize mobile apps with targeted, in-app advertising. As the AdMob is a part of the Firebase service, you need to sign up on [Firebase](#).

Motors Application Settings

General Inventory Translations

Global Settings

Main Page

Add A Car

Ad Mob

Show Ads ☐

Ads Type ☒ Banner ☐ Interstitial

Banner position ☒ Top ☐ Bottom

ios banner id ca-app-pub-0123456789/012345678

Android banner id ca-app-pub-0123456789/012345678

SAVE CHANGES

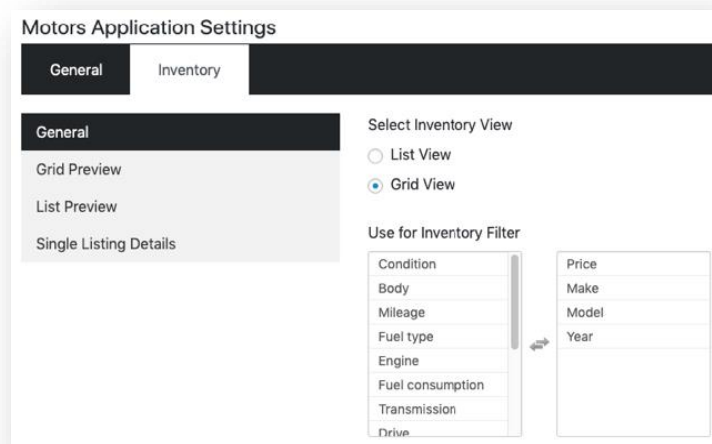
There are following options in the AdMob section:

- Show Ads – allows enabling/disabling ads in app;
- Ads Type – you can select ads type Banner or Interstitial;
- Banner position – you can set a position of ads;
- iOS & Android Banner ID – fields for adding special IDs from Google AdMob platform.

For detailed overview on how to find banner IDs, please have a look at the AdMob [user guide](#). For interstitial IDs, you can follow instructions in [this manual](#).

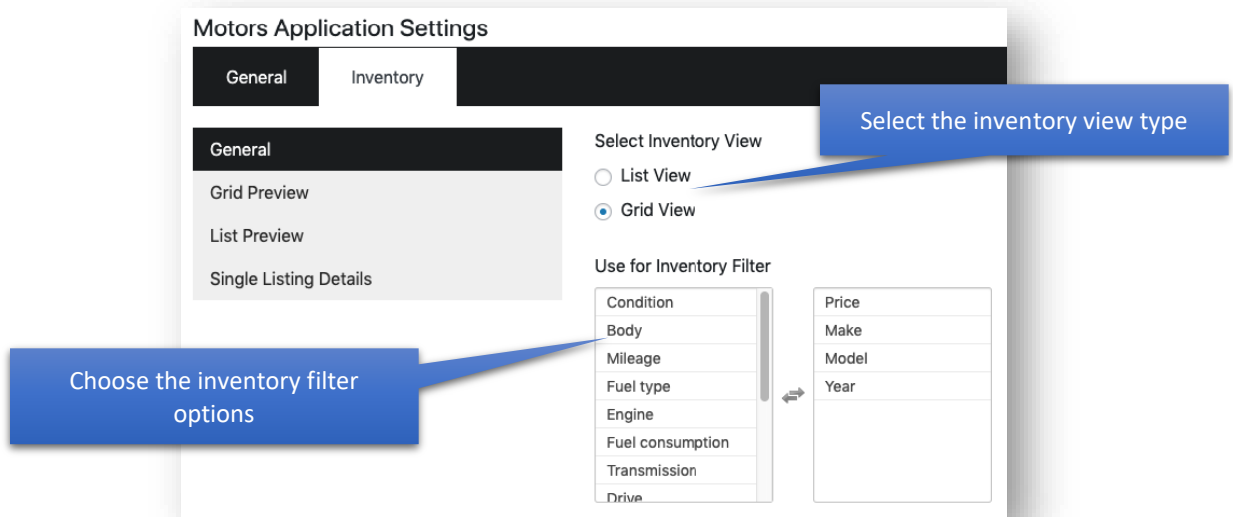
Inventory

The **Inventory** section contains global settings to set up the appearance of Inventory and single list. You can set up the layout of your inventory and single listing pages in the **Motors App Settings -> Inventory** section.



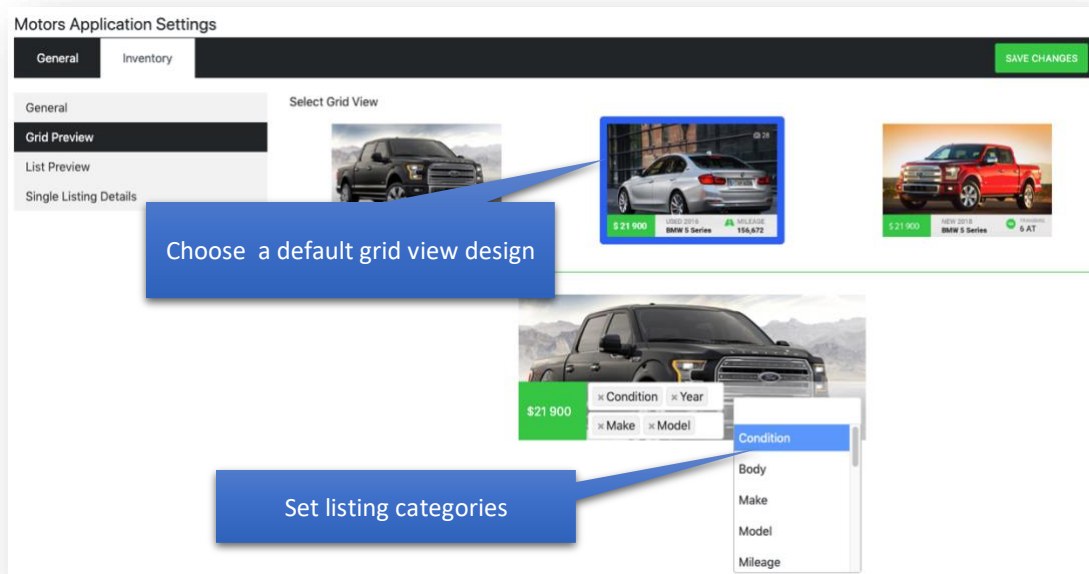
General:

- **Select Inventory View:** You can select a view type of the Inventory page;
- **Use for Inventory Filter:** You can choose your taxonomies for displaying them as the filter parameters on Inventory.



Grid View:

- **Select Grid View:** You can select a default design of the grid view type of your Inventory;
- Set listing categories which will be displayed on the frontend.



List View:

- Set up listing categories which will be displayed on the frontend;
- Define positions of listing categories – **top**, **middle-left**, **middle-right**, **bottom-left**, and **bottom-right**.

The screenshot shows the 'List View' configuration page. On the left, a sidebar contains links: General, Grid Preview, List Preview (selected), and Single Listing Details. The main area features a car image with a price tag of \$21,900. To the right of the image are input fields for 'Condition', 'Make', 'Year', 'Mileage', and 'Fuel type'. A dropdown menu is open, showing a list of categories: Condition, Body, Make, Model, and Mileage, with 'Condition' selected.

Single Listing Details:

- Define listing options which will display as Title, Subtitle, and in the Info section;

The screenshot shows the 'Single Listing Details' configuration page. The sidebar on the left has links: General, Grid Preview, List Preview, and Single Listing Details (selected). The main area is divided into sections: 'Title' with fields for 'Make' and 'Model'; 'Subtitle' with fields for 'Condition' and 'Year'; and 'Listing Info' with fields for 'Body', 'Mileage', and 'Fuel type'. Below these fields is a list of categories: Condition, Body, Make (selected), Model, and Mileage.

Translations

The Translations section contains a list of used static strings in the app where you can translate or replace with another text.

General	Inventory	Translations	SAVE CHANGES
Translations			
Sign In		Logg inn	
Sign Up		Registrer deg	
LOGIN		LOGG INN	
PASSWORD		PASSORD	
Vehicle for sale		Biler til salgs	
RECOMENDED		ANBEFALT	
RECENTLY ADDED		NYLIG LAGT TIL	
Choose Image		Velg Bilde	
Choose Photo		Velg Foto	
Open Camera		Åpent kamera	
Save		Lagre	
Build Your Ad		Bygg din annonse	

Mobile Application Build

Once you have installed and set up the Motors Application plugin on your website, you can proceed to build your mobile application for Android and IOS. Before starting, you need to download and install some prerequisite development environments and packages which make app building process much easier.

Prerequisites and Environment

Step 1 – Open the app folder

You need to unzip the **motors-app.zip** file which comes in the Motors Application plugin package to your desktop or another location/folder. After, you need to run **Terminal** in macOS or **CMD (Command Prompt)** in Windows and go to **motors-app/** folder.

Tip: All commands are identical in both Terminal and Command Prompt

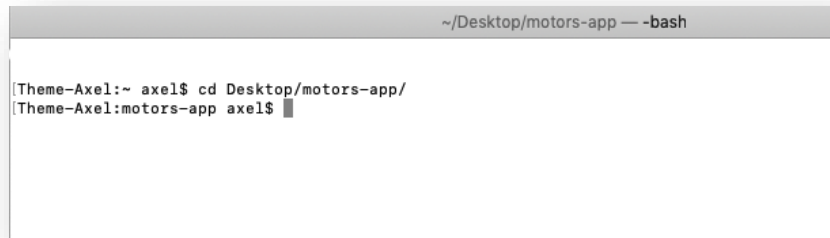
To open the folder, go in to your Terminal/CMD and run this command:

```
cd Desktop/motors-app/
```

In case you unzipped the file to another place, run command like:

```
cd folder-name/motors-app/
```

Terminal/CMD outcome:

A screenshot of a terminal window with a title bar that reads '~ / Desktop / motors-app — -bash'. The terminal content shows two lines: the first line is '[Theme-Axel:~ axel\$ cd Desktop/motors-app/' and the second line is '[Theme-Axel:motors-app axel\$' followed by a cursor. The terminal has a white background and a grey title bar.

```
~ / Desktop / motors-app — -bash
[Theme-Axel:~ axel$ cd Desktop/motors-app/
[Theme-Axel:motors-app axel$
```

Step 2 – Install Node.js and NPM

You need to download the Node.js pre-built installer and install it to your computer. You can get the latest version of the Node.js here - <https://nodejs.org/en/download/>. NPM is distributed with Node.js - which means that when you download Node.js, you automatically get NPM installed on your computer.

To check if you have Node.js installed, run this command in your Terminal/CMD:

```
node -v
```

To confirm that you have npm installed you can run this command in your Terminal/CMD:

```
npm -v
```

Terminal/CMD outcome:

```
~/Desktop/motors-app — -bash
[Theme-Axel:~ axel$ cd Desktop/motors-app/
[Theme-Axel:motors-app axel$ node -v
v10.16.0
[Theme-Axel:motors-app axel$ npm -v
6.9.0
[Theme-Axel:motors-app axel$ ]
```

Step 3 – Run npm install commands

After installing the Node.js package, you need to run the following command in Terminal/CMD:

```
npm install
```

Terminal/CMD outcome:

```
~/Desktop/motors-app — npm TERM_PROGRAM=Apple_Terminal SHELL=/bin/bash
Last login: Mon Jul  1 08:54:21 on console
You have new mail.
[Theme-Axel:~ axel$ cd Desktop/motors-app/
[Theme-Axel:motors-app axel$ node -v
v10.16.0
[Theme-Axel:motors-app axel$ npm -v
6.9.0
[Theme-Axel:motors-app axel$ npm install
( [REDACTED] ) i loadDep:decamelize: sill install loadAllDepsIntoIdealTree
```

When the installation process is done, you need to run this command in Terminal/CMD:

```
npm install -g react-native-cli
```

Terminal/CMD outcome:

```
+ react-native-cli@2.0.1
updated 1 package and audited 893771 packages in 13.081s
found 11 low severity vulnerabilities
  run `npm audit fix` to fix them, or `npm audit` for details
[Theme-Axel:motors-app axel$ npm install react-native-cli
(( )) : rollbackFailedOptional: npm-session 5ebe9aef891623da
```

Note. If you get error such as “**Error: EACCES: permission denied ...**”, you need to run this command: ***sudo npm install -g react-native-cli***

Step 4 – Run react-native commands

Now you need link native dependencies with the following command:

```
react-native link
```


Terminal/CMD outcome:

```
Theme-Axel:motors-app axel$ react-native link
warn Running 'react-native link' without package name is deprecated and will be removed in next release. If you use this command to link your project assets, please let us know about your use case here: https://goo.gl/RKTeoc
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info iOS module "@react-native-community/async-storage" is already linked
info Android module "@react-native-community/async-storage" is already linked
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
info Linking assets to ios project
info Linking assets to android project
info Assets have been successfully linked to your project
```

Finally, you need to run this command:

```
npm run start
```

Terminal/CMD outcome:

```
~ — Metro Bundler — node • launchPackager.command

Running Metro Bundler on port 8081.

Keep Metro running while developing on any JS projects. Feel free to
close this tab and run your own Metro instance if you prefer.

https://github.com/facebook/react-native

Looking for JS files in
/Users/axel/Desktop/motors-app
Loading dependency graph, done.
```

Step 5 – Put your site link

You need to edit this “/motors-app/src/helpers/settings.json” file and put your site link. You can use any text editor which you familiar with and the file.



```
JS settings.js x
src ▸ helpers ▸ JS settings.js ▸ liveUri
1  const liveUri = 'http://motors.stylemixthemes.com';
2
3  export default {
4    BASE_URL: liveUri,
5    REST_URL: liveUri + 'wp-json/stm-mra/v1'
6  }
```

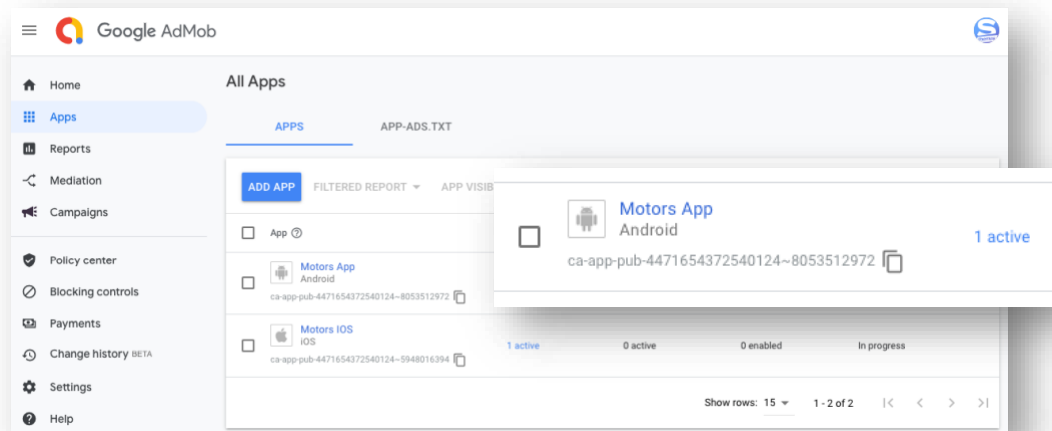
Step 6 – Upload your logo

The last but not the least step is uploading your logo files. You need rename your logo files to **logo-white.png** and **logo-dark.png**. After, upload them to this “/motors-app/src/assets/img/” folder and confirm the replacement.

Finally, all the needed dependencies have been installed and set up. Now you can proceed directly to Android and IOS application building process.

Google AdMob integration

You need to copy your App ID from the Google AdMob account to integrate AdMob features in your app. You can find your App ID by following instructions in [this article](#).



You need to copy your App IDs (Android/iOS) and paste them to the Motors App templates.

For Android:

You need paste your Android App Id in **`"/motors-app/android/app/src/main/AndroidManifest.xml"`** line 46

```
43      <!--AdMob-->
44      <meta-data
45          android:name="com.google.android.gms.ads.APPLICATION_ID"
46          android:value="ca-app-pub-1689195480083998~5151673612"/>
47      <!--Notifications-->
```

and **`"/motors-app/android/app/src/main/java/com/stylemixthemes/motors/MainApplication.java"`** files line 83.

```
79  @Override
80  public void onCreate() {
81      super.onCreate();
82      SoLoader.init(this, /* native exopackage */ false);
83      MobileAds.initialize(this, "ca-app-pub-1689195480083998~5151673612");
84  }
```

For iOS:

You need to paste your iOS App ID to **`"/motors-app/ios/motorsReactApp/Info.plist"`** file line 28.

```
26  <string>1.0.4</string>
27  <key>GADApplicationIdentifier</key>
28  <string>ca-app-pub-1689195480083998~1340199344</string>
29  <key>LSApplicationCategoryType</key>
```

Push Notification Setup

Push notification feature is integrated with Google Firebase service. You need to sign up on the Firebase website and create a new project. Then, add your app to your Firebase project to register app. For the detailed overview, please have a look at [this manual](#).

After the app registration is done, you need to upload Firebase configuration file to the Motors App.

For Android: it needs to replace the “**google-services.json**” file in this folder “**.../motors-app/android/app**”.

For iOS: it needs to replace the “**GoogleService-Info.plist**” file in this folder “**.../motors-app/ios/motorsReactApp**”.

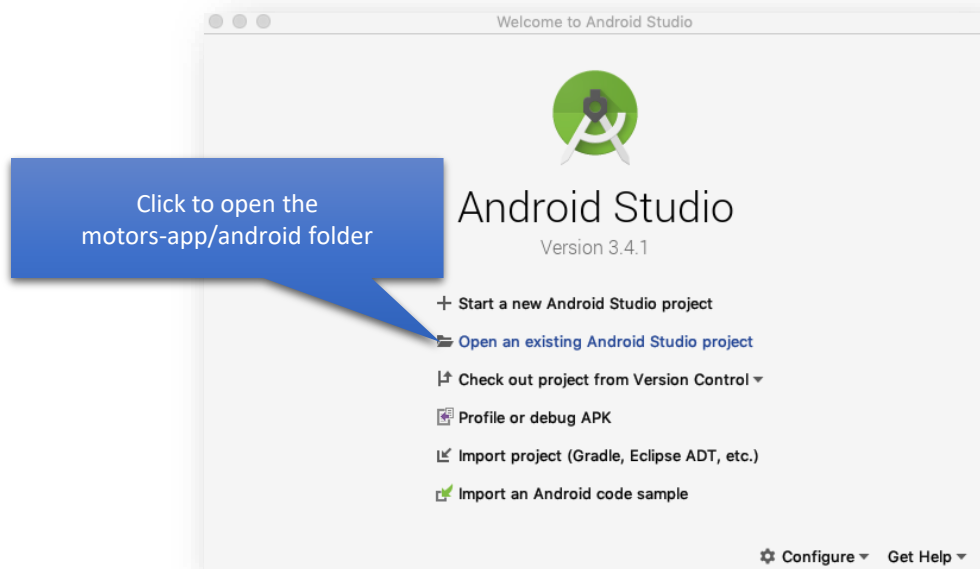
You can download Firebase configuration files for your apps by following instructions in this [user guide](#).

Android Application Build

Step 1 – Set up environment

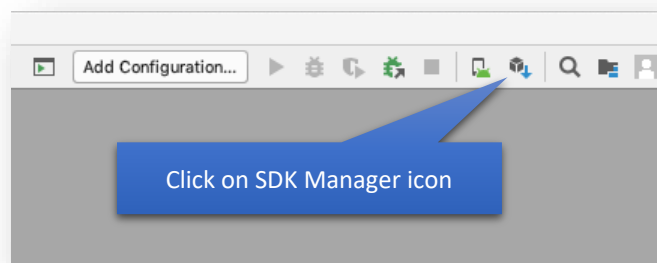
You need to download and install Android Studio in order to build your android application. You can download it here - <https://developer.android.com/studio>.

After, run the Android Studio and select the “**Open an existing Android Studio project**” option. In the popup window, you need to select the “**motors-app/android/**” folder.

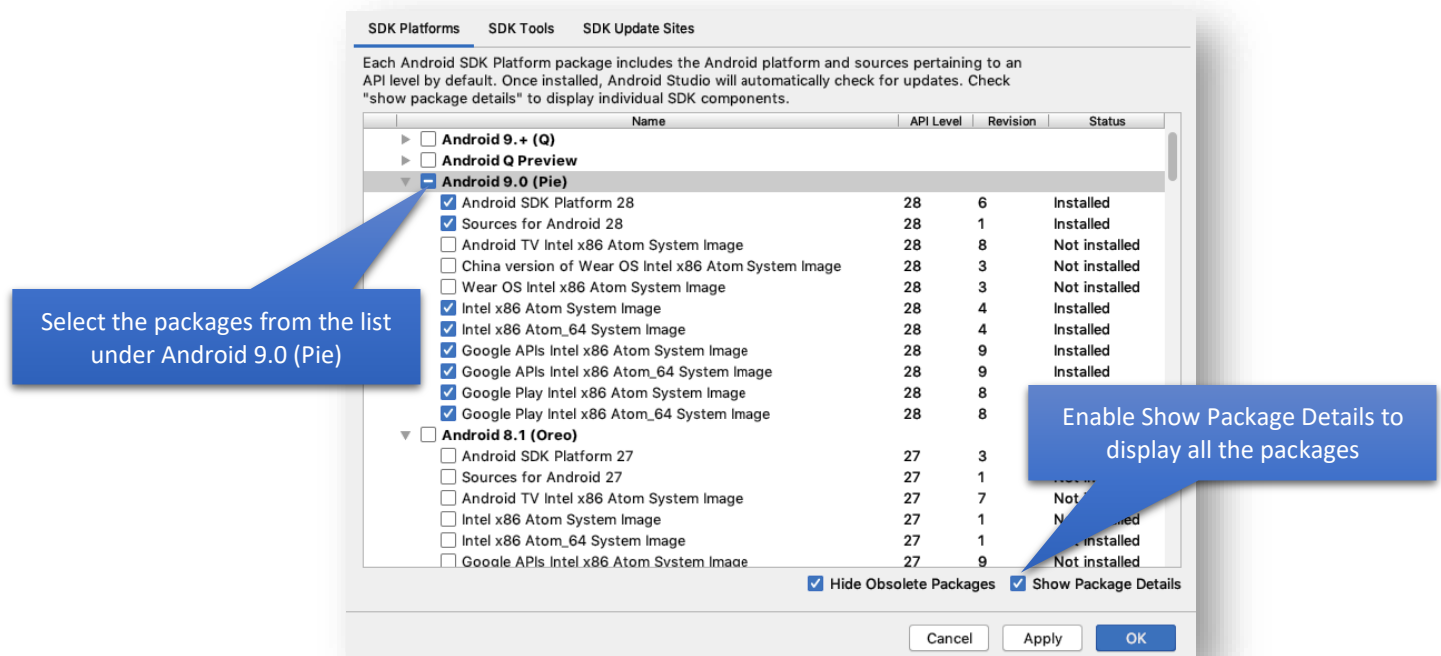


Note. You should select the “**android**” folder in order to open the Android Studio project

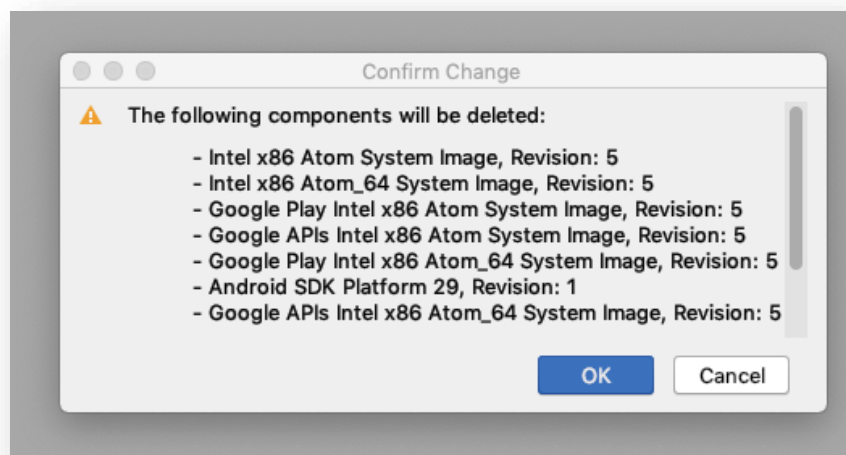
After, you need to install **Android SDK** platform package to your Android Studio. To install Android SDK platform, click on the **SDK Manager** icon in the top-right toolbar.



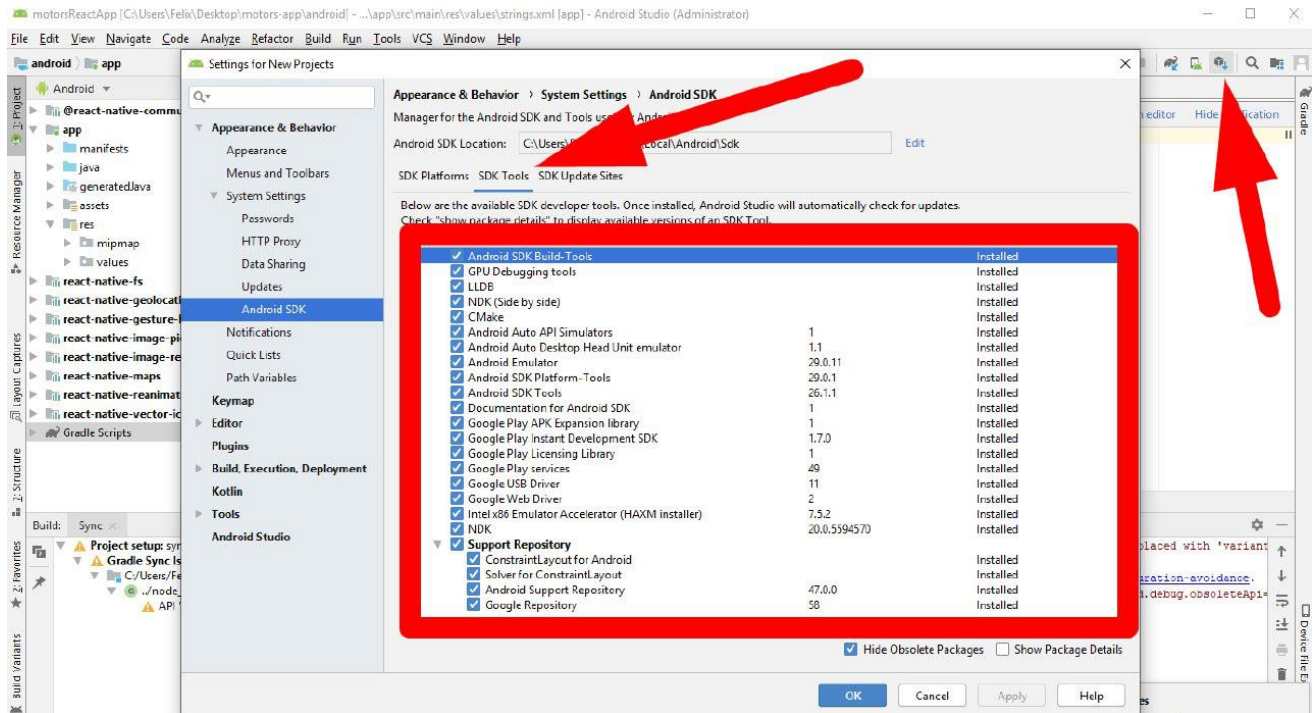
In the next window, enable **Show Package Details** option and select the packages under **Android 9.0 (Pie)**, then click on **Ok** button.



After, click on the **Ok** button of the **Confirm Change** window in order to start the download process.



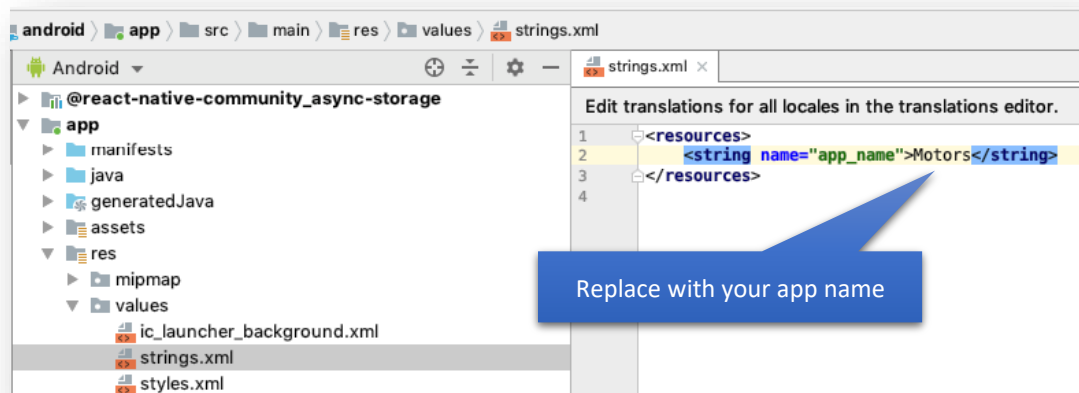
After the installation is completed, you can proceed to the next step.



Note: If you did not install the SDK **Packages** on the SDK Tools tab, **please** download them, and then install them **in order** to work without error in the next steps.

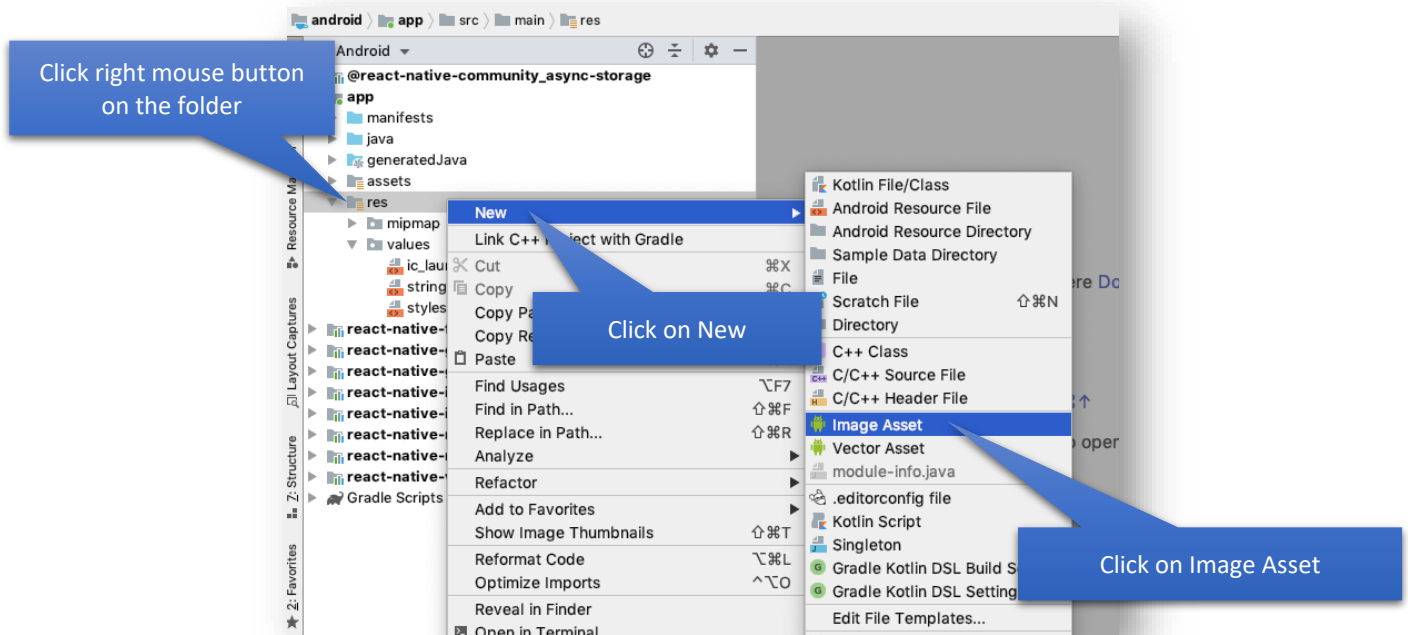
Step 2 – Application Name

You can set a name of your application by editing this “/app/res/values/strings.xml” file in Android Studio editor. Press **cmd+S/ctrl+S** buttons on a keyboard in order to save your changes.



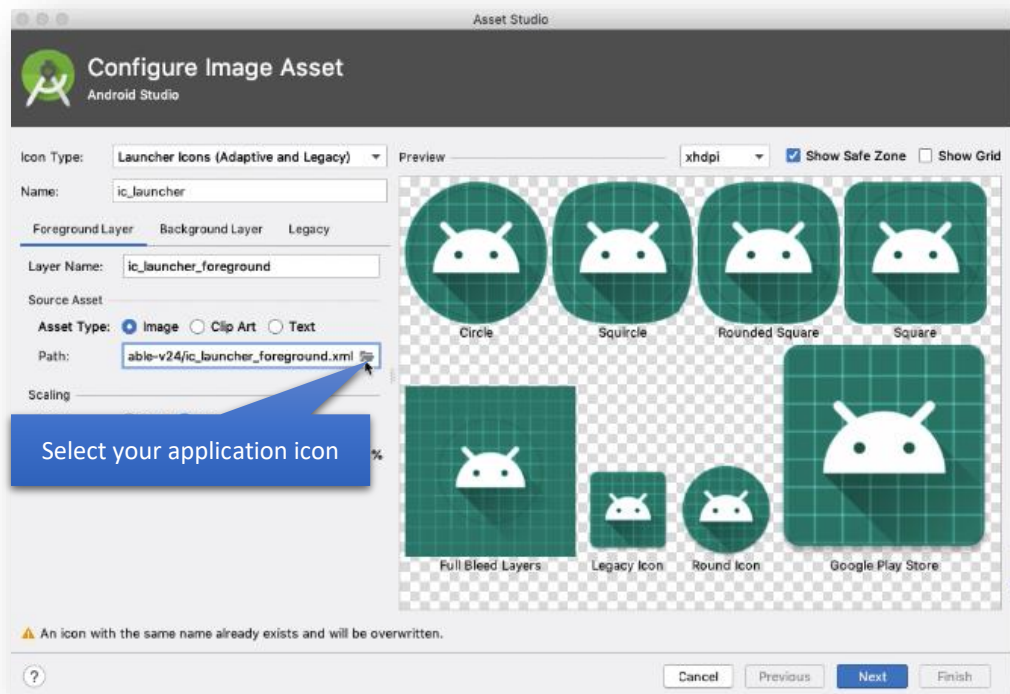
Step 3 – Upload app icon

To upload your application icon, click right mouse button on this folder “/app/res” > **New** > **Image Asset**.



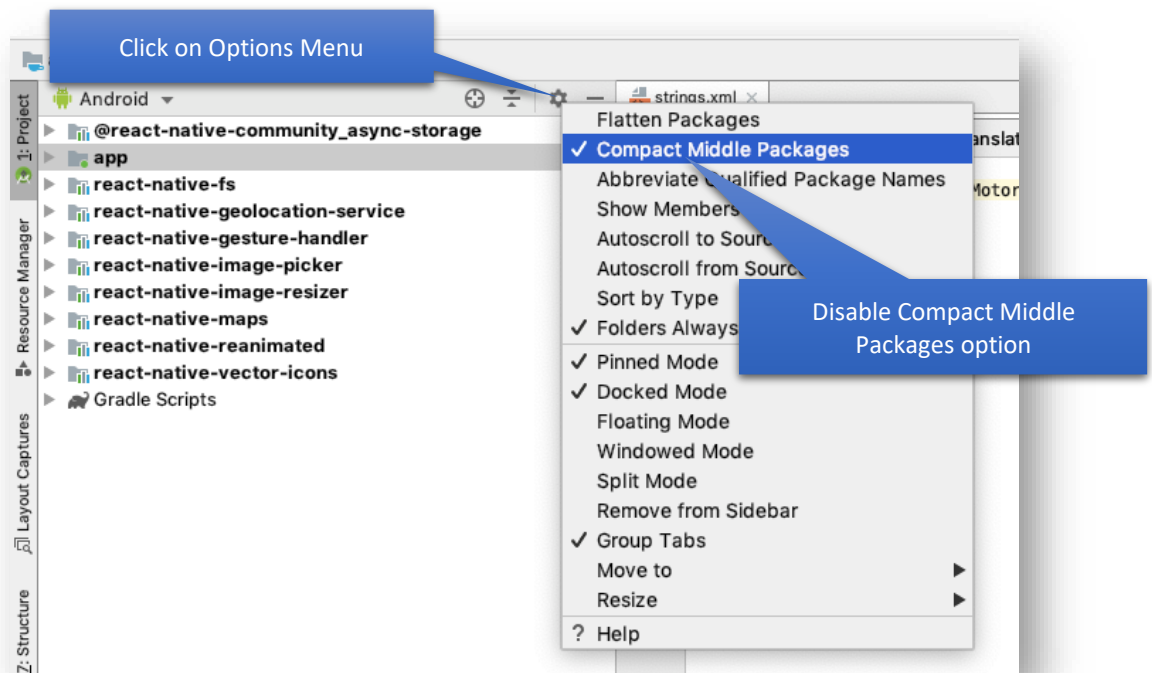
When you click on the **Image Asset**, **Asset Studio** window appears where you can configure your application icon.

Tip: For detailed overview of the Asset Studio, have a look at the [Asset Studio](#) user guide.

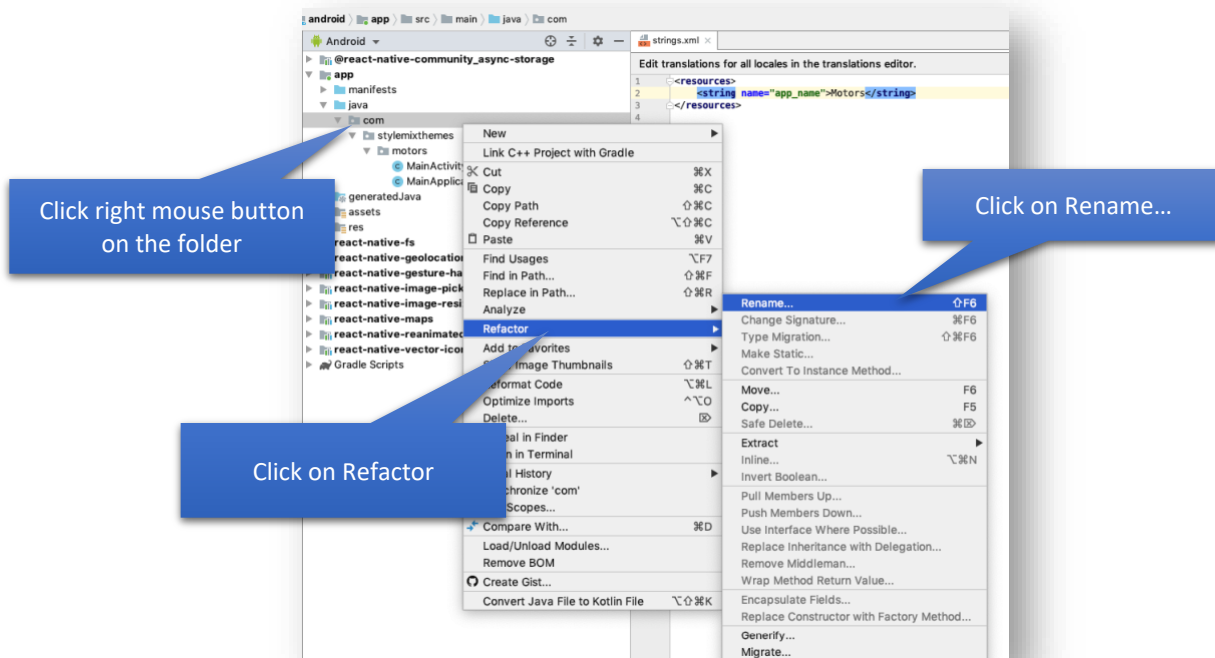


Step 4 – Change package name

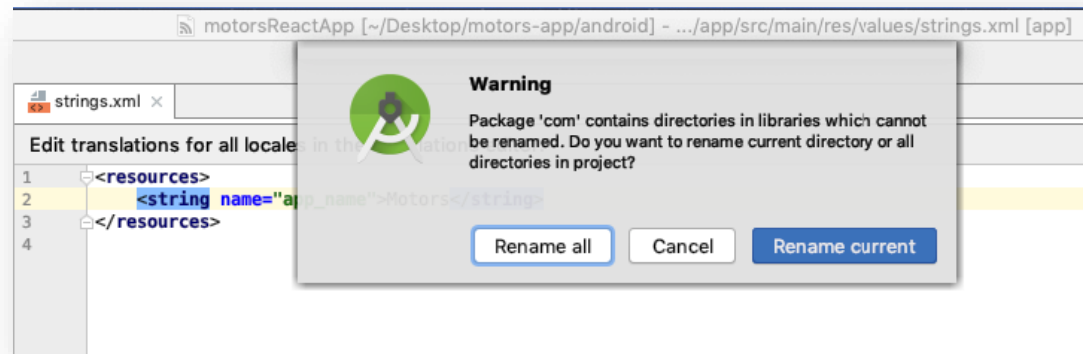
You need to disable the “**Compact Middle Packages**” option in the **Options Menu** in order to display sub-folders of the project.



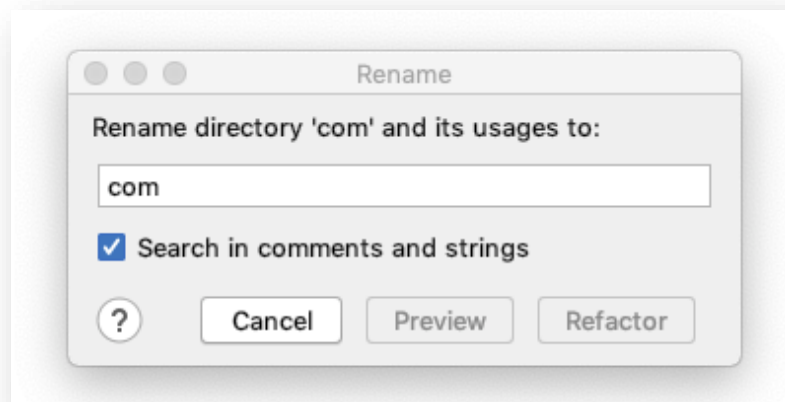
After, go to this “**app/java/com/**” folder and click right mouse button on folder “**/com**” > **Refactor** > **Rename** option.



Click on **Rename current** button in the **Warning** popup window.



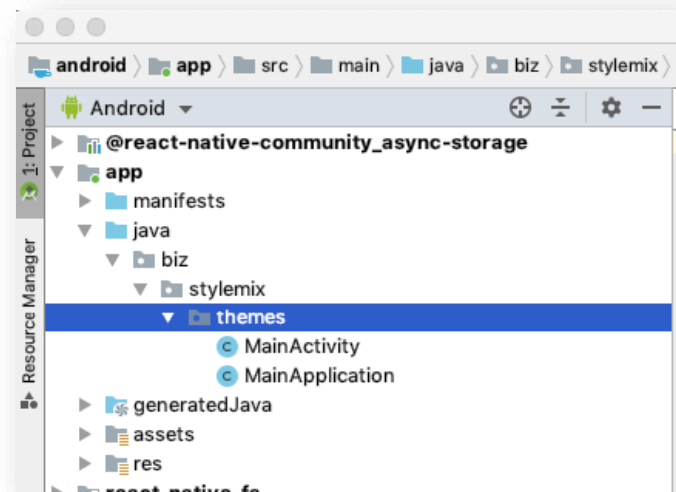
Replace **“com”** with your website domain name and click on **Refactor** button.



You need to do the same actions for **“/stylemixthemes/”** and **“/motors/”** folders and replace them with your domain name.

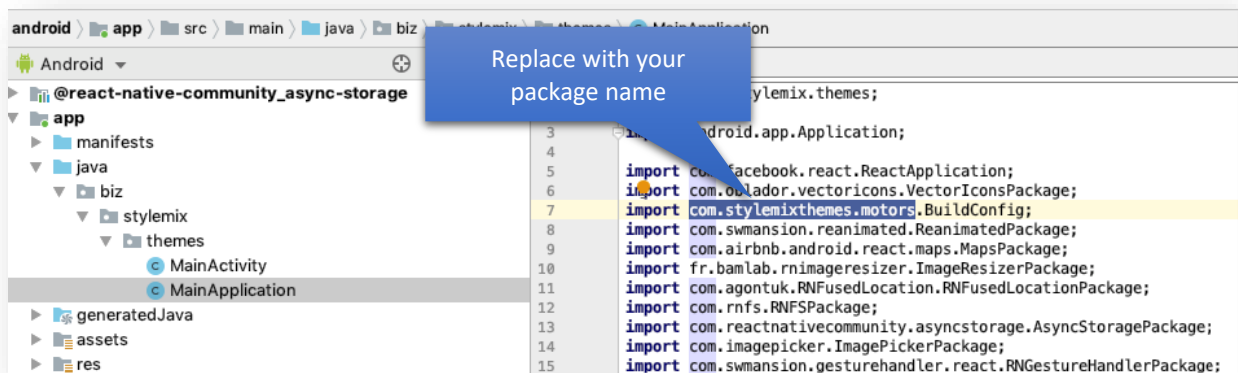
Tip: You need to enter a reverse-domain name. I.E., folders name **“com > stylemixthemes > motors”** can be renamed to **“biz > stylemix > themes”** or **“com > yourcompanyname”**.

Outcome:



Step 5 – Change “import” name

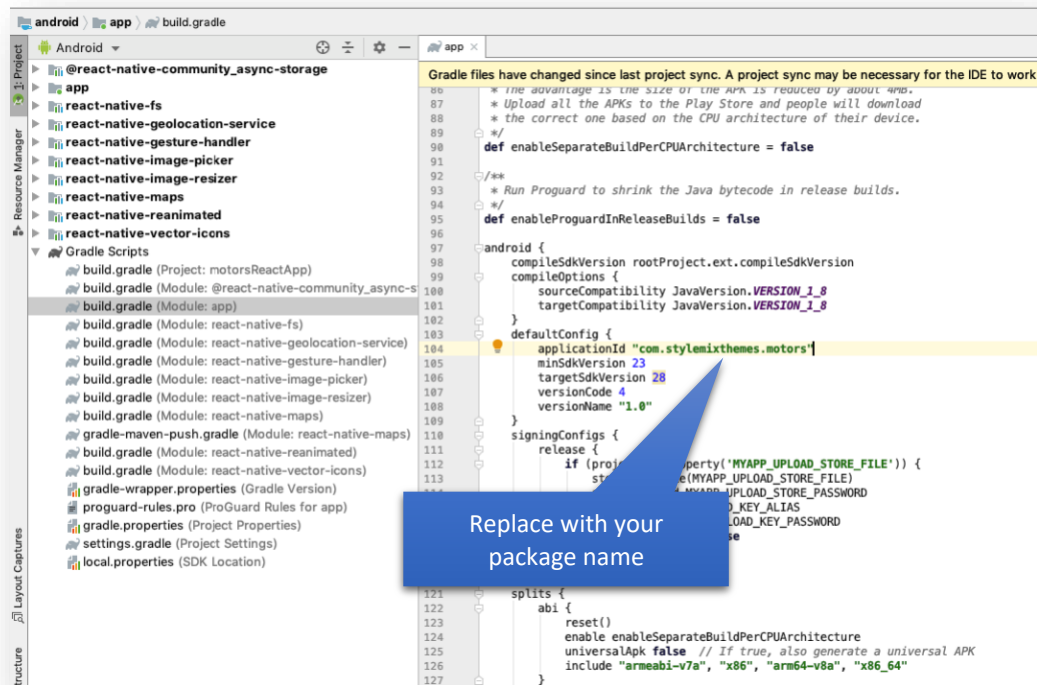
Now you need to rename package name in the “**app > java > com > yourdomain > MainApplication.java**” file.



Press **cmd+S/ctrl+S** buttons on a keyboard in order to save your changes.

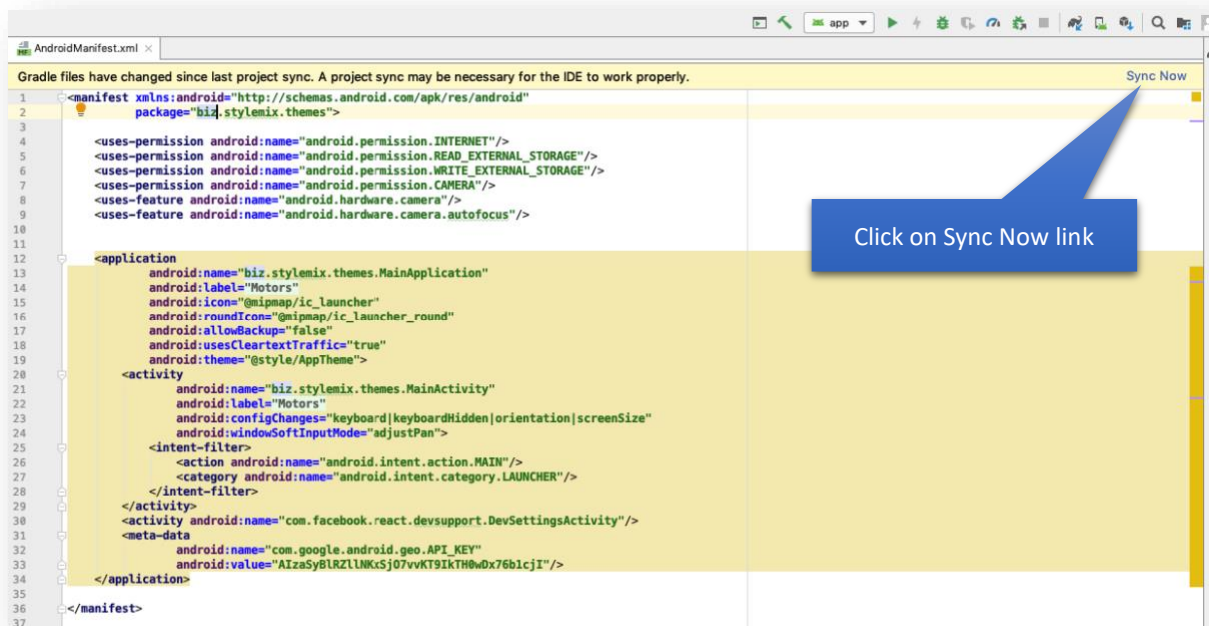
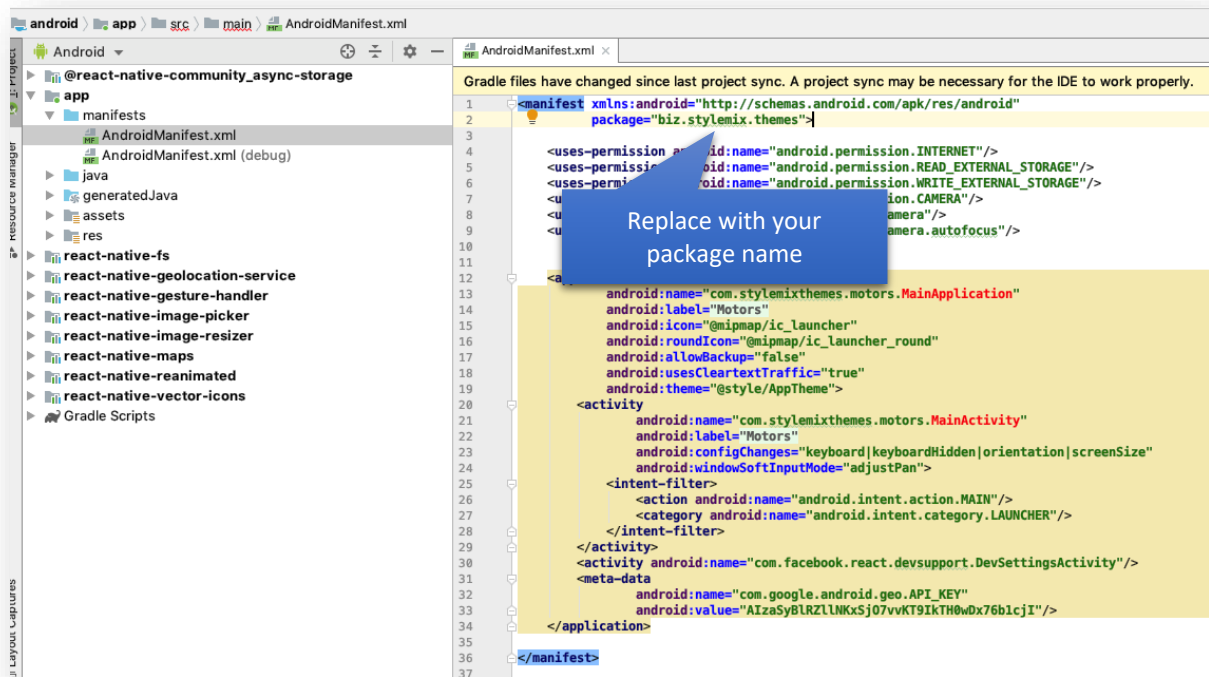
Step 6 – Edit build.gradle file

You need to add your package name in the “**Android > Gradle Scripts > build.gradle (Module: app)**” file, then press **cmd+S/ctrl+S** buttons on a keyboard in order to save your changes.



Step 7 – Edit AndroidManifest file

You need to edit this “**App > manifests > AndroidManifest.xml**” file and change the package name with yours. Then press **cmd+S/ctrl+S** buttons on a keyboard in order to save your changes. After, click on the Sync Now link at the top-right corner.



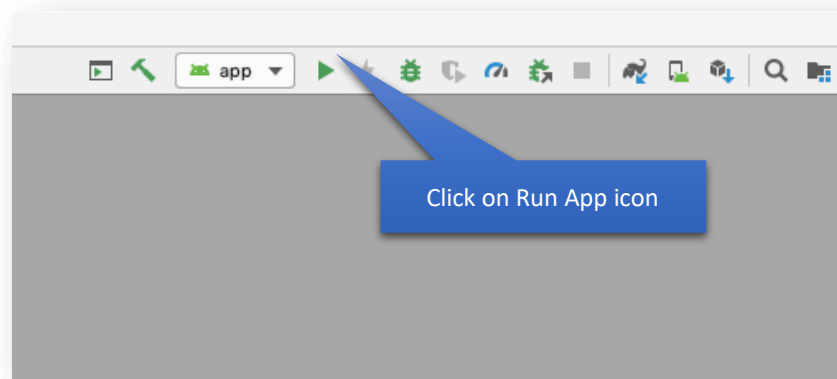
Finally, the android application building process is completed. Now you can start to test your app in Android Emulator and release your app.

Android App Release

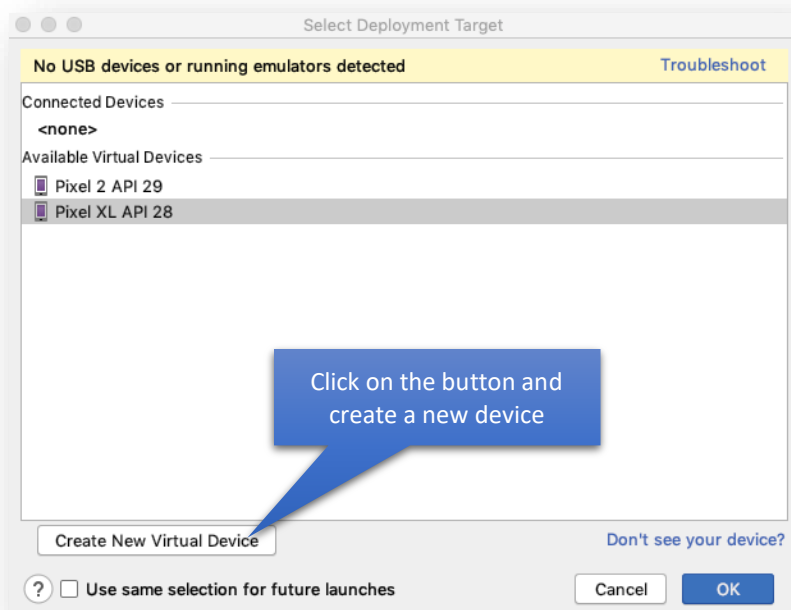
Step 1 – Test your app

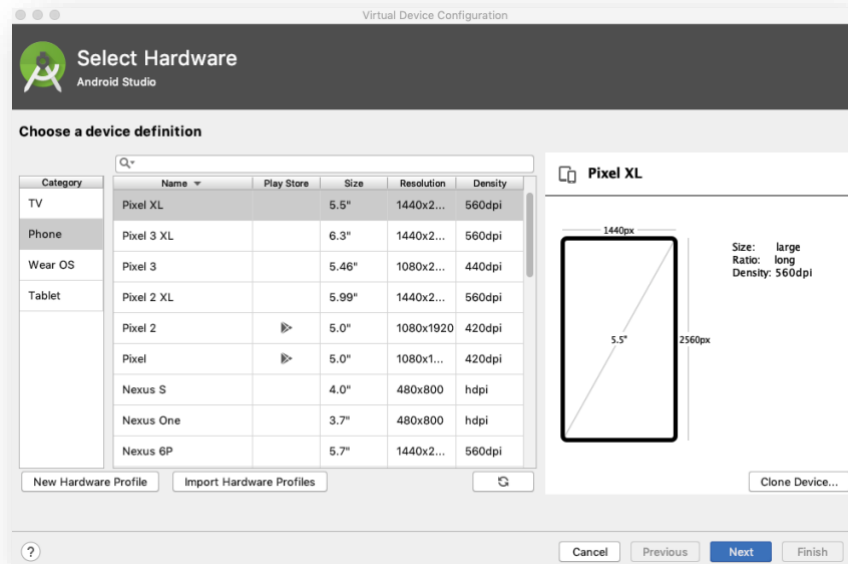
You can test your app via running emulator in Android Studio. To run the emulator click on Run App icon in the tool bar.

Note. In order to use virtual device emulator you need to install the [Java SDK 8u211](#) package.

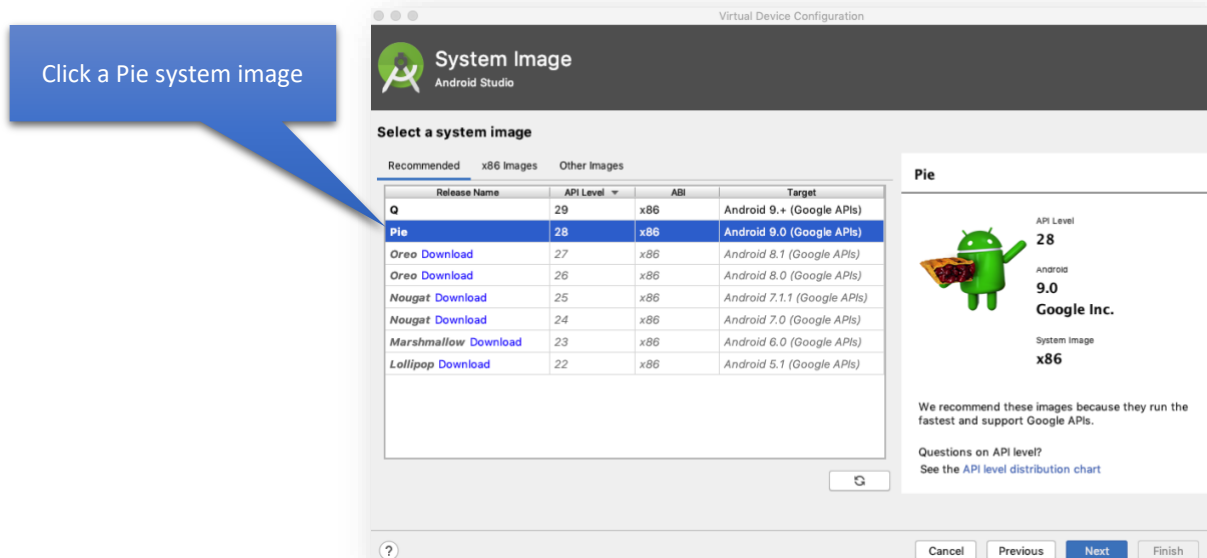


Click on “**Create New Virtual Device**” button in the appeared popup window.





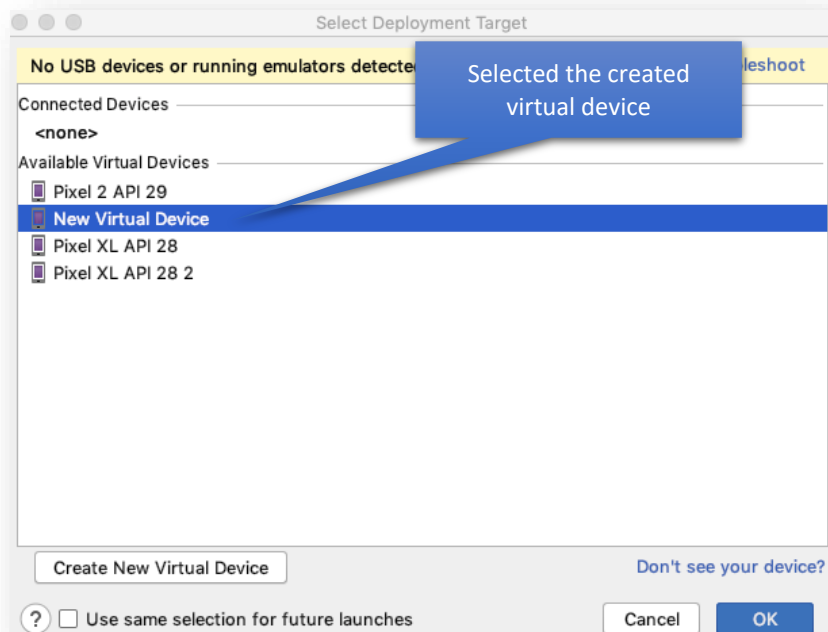
After selecting the needed device, click on **Next** button and select a **Pie** system image.



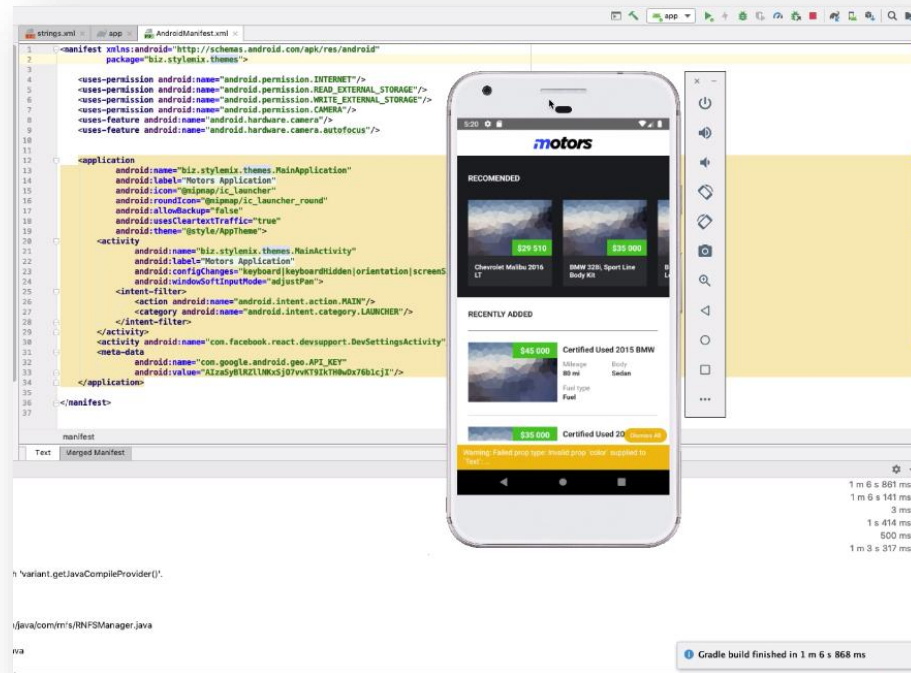
Click on **Next** button and verify your device configurations. After, click on **Finish** button to add your virtual device.



Select your virtual device and click on **OK** button to run the emulator.

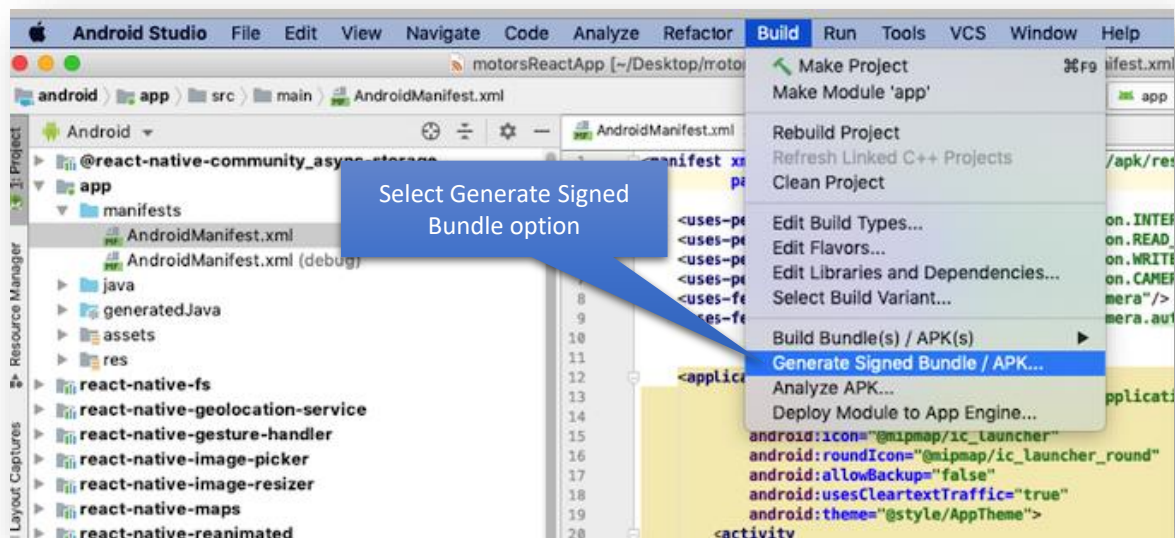


Android Virtual Device outcome:

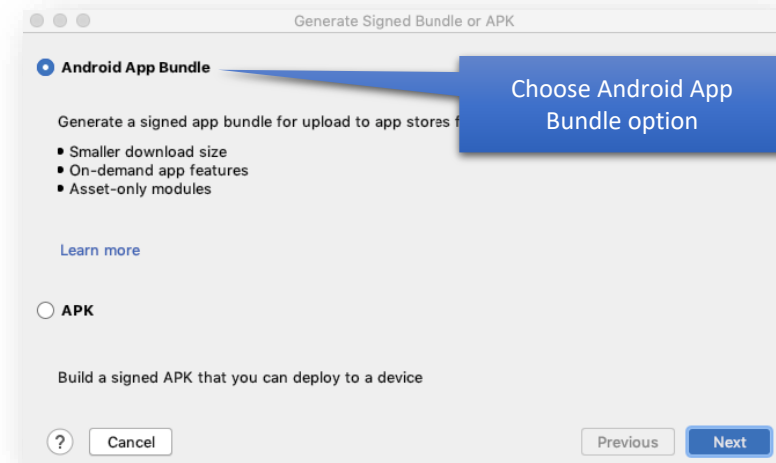


Step 2 – Generate Android App Bundle

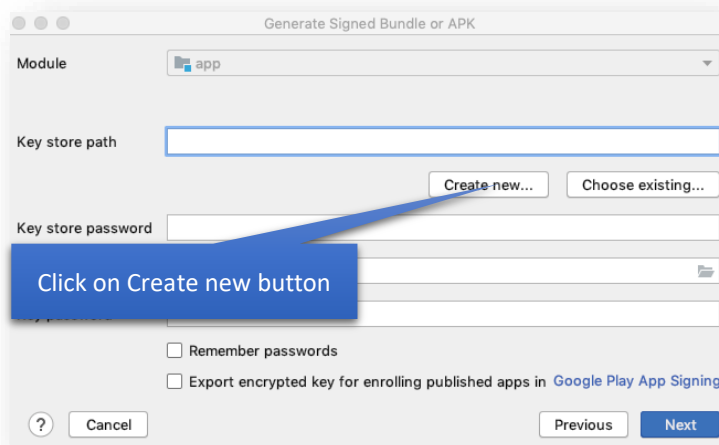
After testing your app in emulator, you can generate signed APK file for uploading to Google Play. To generate APK, on the top menu click on **Build > Generate Signed Bundle / APK** option.



In the appeared window, select **Android App Bundle** option and click on Next button.



In the next window, you can either create a new Certificate in (*.jks) format or use your existing certificate.



Click on **Create new...** button and put appropriate data to fields in the **New Key Store** window. You need to write name of your certificate with (.jks) format.

The image shows two screenshots of the 'New Key Store' dialog box. The top screenshot shows the 'Save As' field with the text 'motors-app.jks' and a callout that says 'Type a name of your certificate with .jks format'. The bottom screenshot shows the 'Key store path' field with the text '/Users/axel/Desktop/motors-app.jks', the 'Password' and 'Confirm' fields, the 'Alias' field with the text 'motors-app', and the 'Certificate' section with fields for 'First and Last Name', 'Organizational Unit', 'Organization', 'City or Locality', 'State or Province', and 'Country Code (XX)'. Callouts point to the 'Password' field ('Create a password'), the 'Alias' field ('Create certificate alias'), and the 'Certificate' section ('Add information about your organization').

New Key Store

Save As:

Tags:

Desktop

Type a name of your certificate with .jks format

New Key Store

Key store path:

Password: Confirm:

Create a password

Key

Alias:

Create certificate alias

password: Confirm:

Validity (years):

Add information about your organization

Certificate

First and Last Name:

Organizational Unit:

Organization:

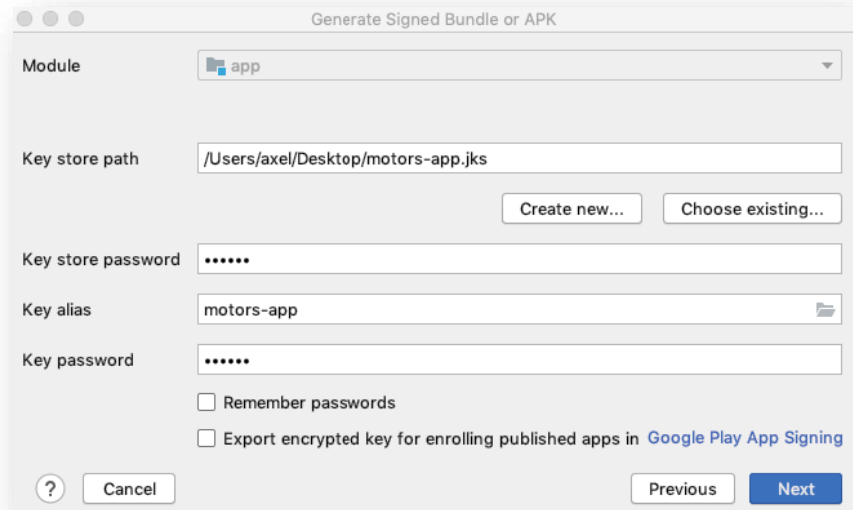
City or Locality:

State or Province:

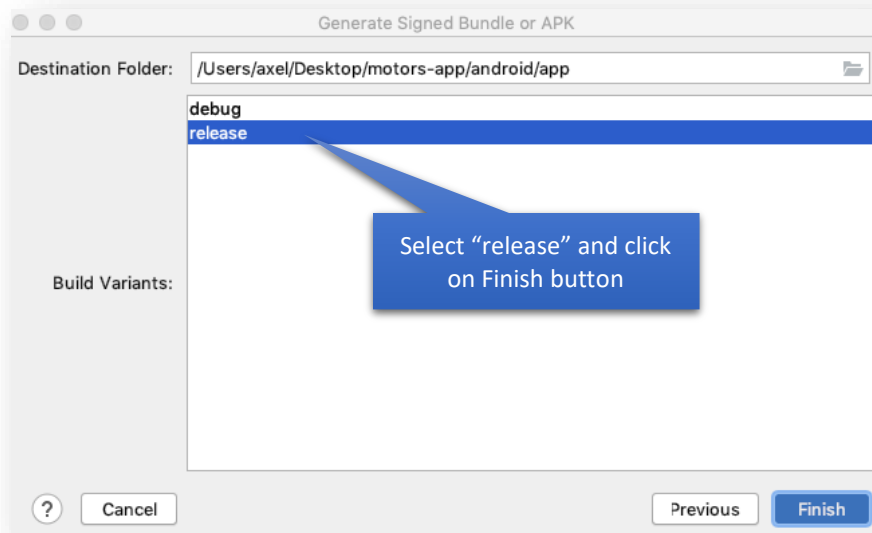
Country Code (XX):

Cancel OK

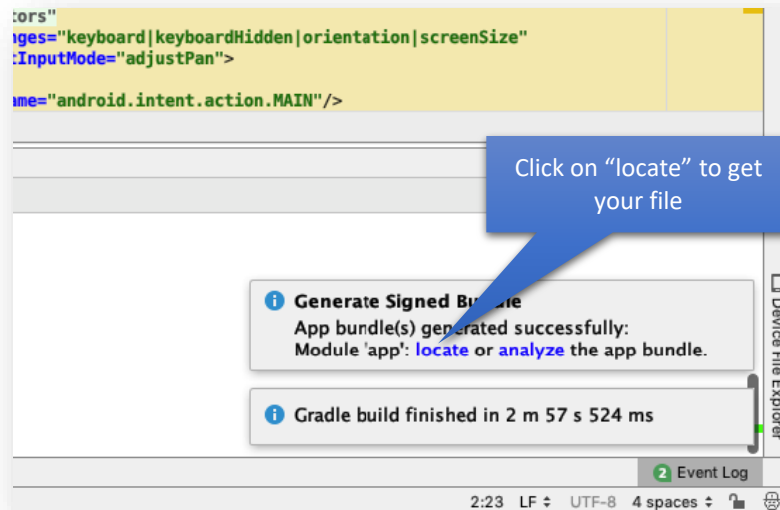
After, click on **OK** button and go to next step by clicking on **Next** button.



Select “**Release**” option in the next window and click on **Finish** button.



After Android Studio finishes the generating Android Signed Bundle, it displays the success message in **Event Log** section.

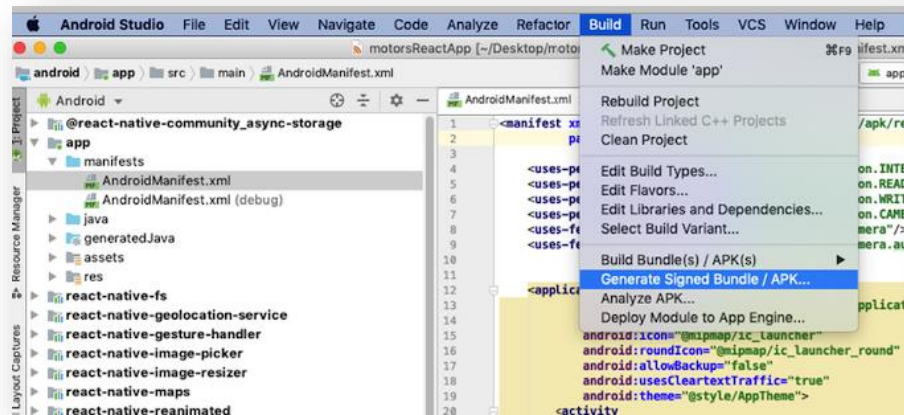


You can click on “**locate**” link in order to get your APK file. The file is located in “**motors-app/android/app/release**” folder.

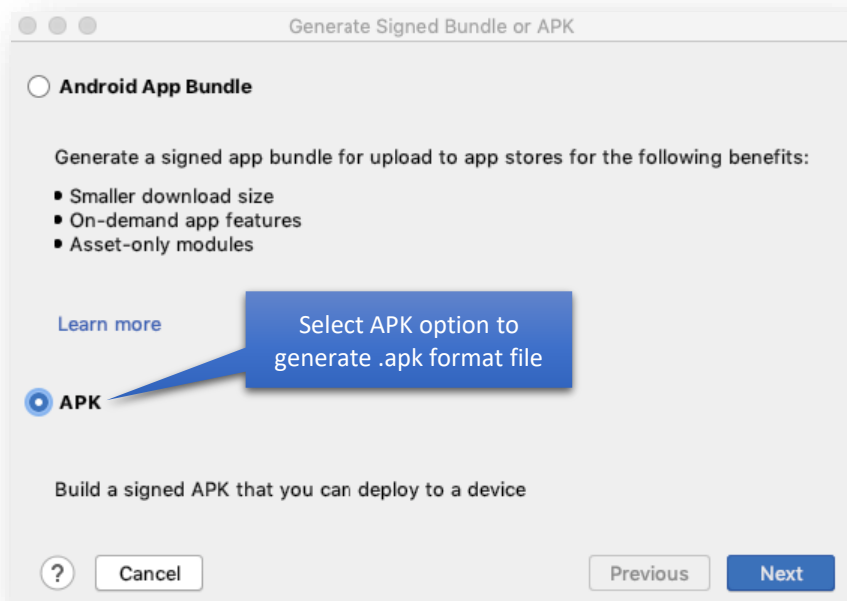
Tip: The generated file format is (*.aab). An Android App Bundle is a new upload format that includes all your app’s compiled code and resources, but defers APK generation and signing to Google Play. For a detailed overview, see the [Android App Bundles](#) user guide.

Step 2.1 – Generate APK

In case you need to generate a signed APK file in (*.apk) format to test your app in your android devices without uploading to Google Play, you need to click on **Build > Generate Signed Bundle / APK** option on the top menu.



After, select APK option in the window and perform next step as described in the **Step 2 – Generate Android App Bundle**.



iOS App Build

This manual provides a step-by-step demonstration of releasing a Motors app to the [App Store](#) and [TestFlight](#).

Prerequisites

You need to enroll in the **Apple Developer Program** in order to publish your app to the App Store. You can read more about developer program and its options [here](#).

Step 1 – App Registration on App Store Connect

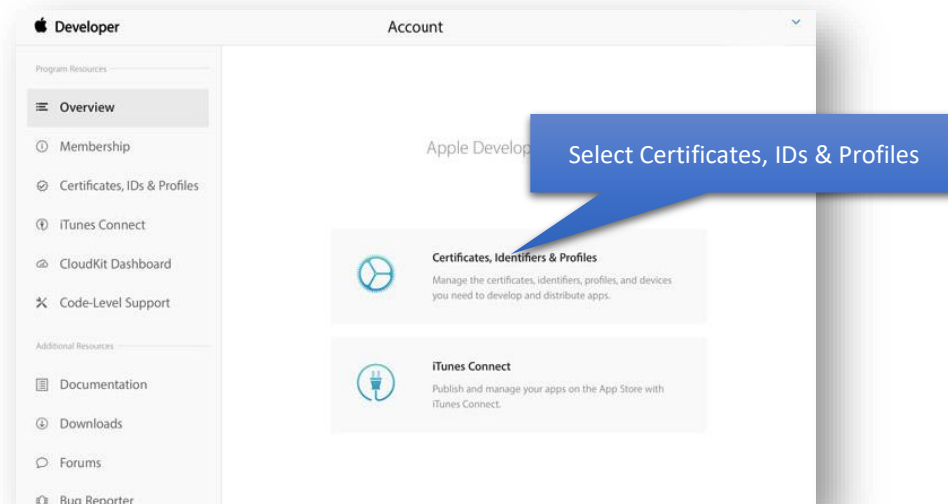
App Store Connect is a web-based tools kit for controlling apps sold on the App Store. As a member of the Apple Developer Program, you will set up your app name, description, screenshots, submit and manage apps, invite users to test with TestFlight, monitor sales reports, and more.

Tip: You can review the [App Store Connect](#) tutorial for more detailed overview.

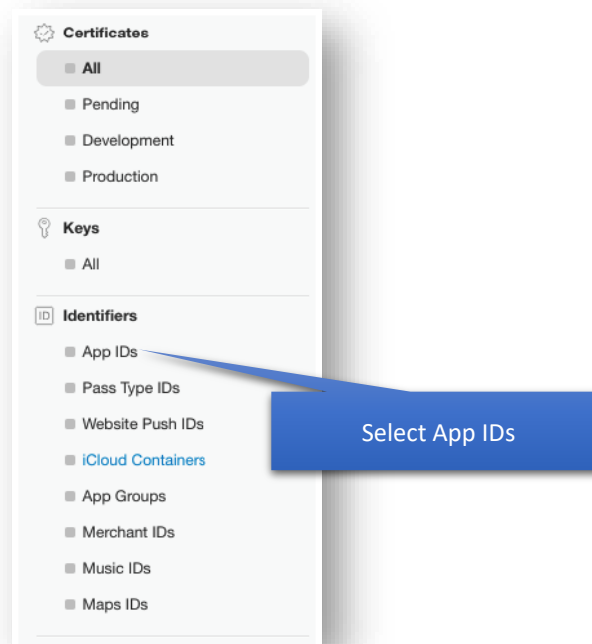
Step 2 – Register a Bundle ID

Bundle ID is a required identifier for IOS apps registered with Apple. It consists of strings which supplied by you during development, usually in form of **com.your-company-domain.app-name**.

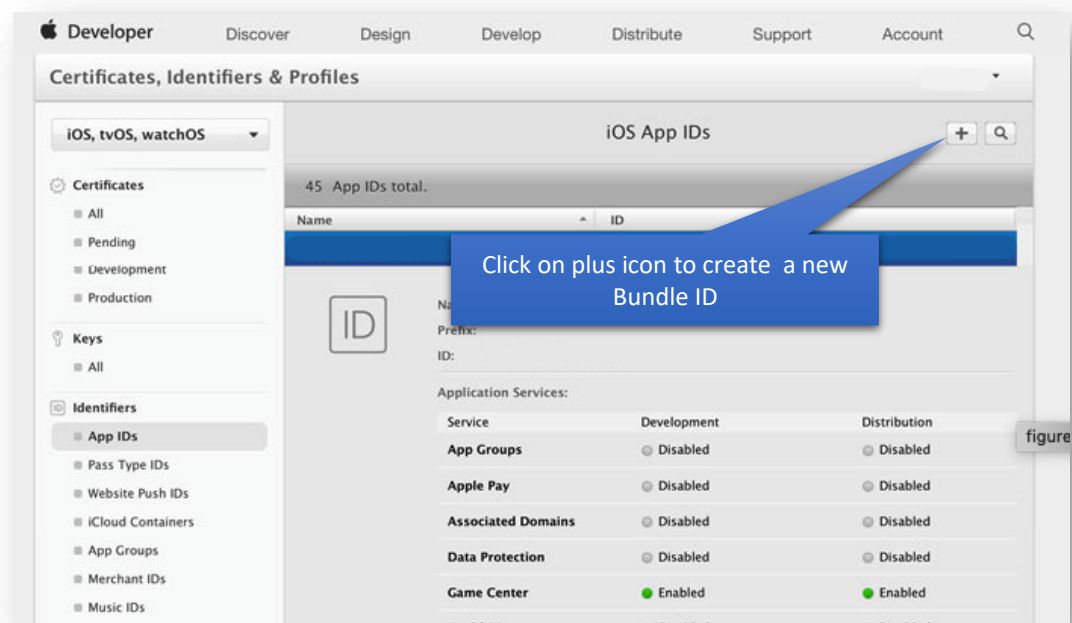
To register a Bundle ID open your developer account on Apple's [developer website](#). Click on **Certificates, IDs & Profiles** section.



From the left menu, select **App IDs** in **Identifiers** section.



Click on the “+” button in order to create a new Bundle ID.



Enter all the required information - write an app name, select **Explicit App ID** in **App ID Suffix**, enter Bundle ID and click on **Continue** button.

Tip: Enter a reverse-domain name in **Bundle ID** field (i.e., com.your-company-domain-name.appname), as Apple suggests.

The screenshot shows the 'Registering an App ID' form with the following sections and annotations:

- App ID Description:** Includes a 'Name' text field. A blue callout bubble points to it with the text 'Enter a name of your app'.
- App ID Prefix:** Includes a 'Value' dropdown menu. A blue callout bubble points to it with the text 'Select Explicit App ID'.
- App ID Suffix:** Features two radio button options:
 - Explicit App ID:** This option is selected. Below it, a 'Bundle ID' text field is present. A blue callout bubble points to it with the text 'Enter a reverse-domain name'. Below the field, a note states: 'We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*)'.
 - Wildcard App ID:** This option is unselected. Below it, another 'Bundle ID' text field is present, with an example 'Example: com.domainname.*' shown below.

After, confirm your App settings and click on **Submit** button to complete the registration of your App ID.

Step 3 – Create an app

Now you need to register your app on App Store Connect. Before you can upload a build of your app to App Store Connect, you must first create it in your App Store Connect account.

Tip: For detailed instructions of registering your app, have a look at the [Add an app to your account](#) tutorial.

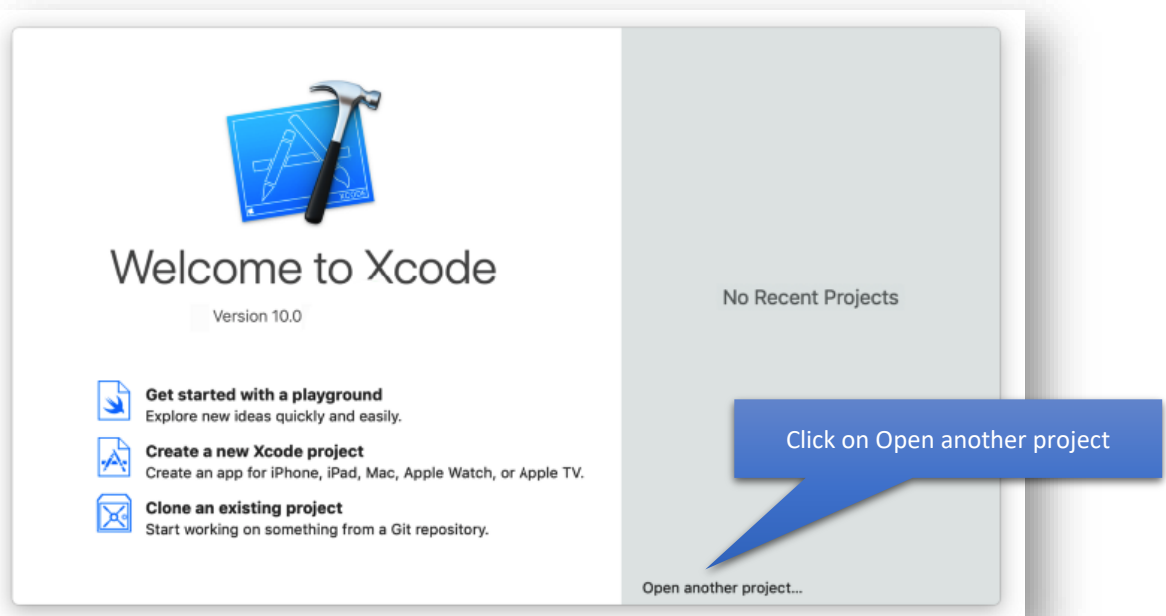
Step 4 – Prepare for app distribution

To prepare your iOS app for distribution on the App Store, you'll need to set app icons, update the build string, verify build settings etc. You can review the step by step guidelines [here](#).

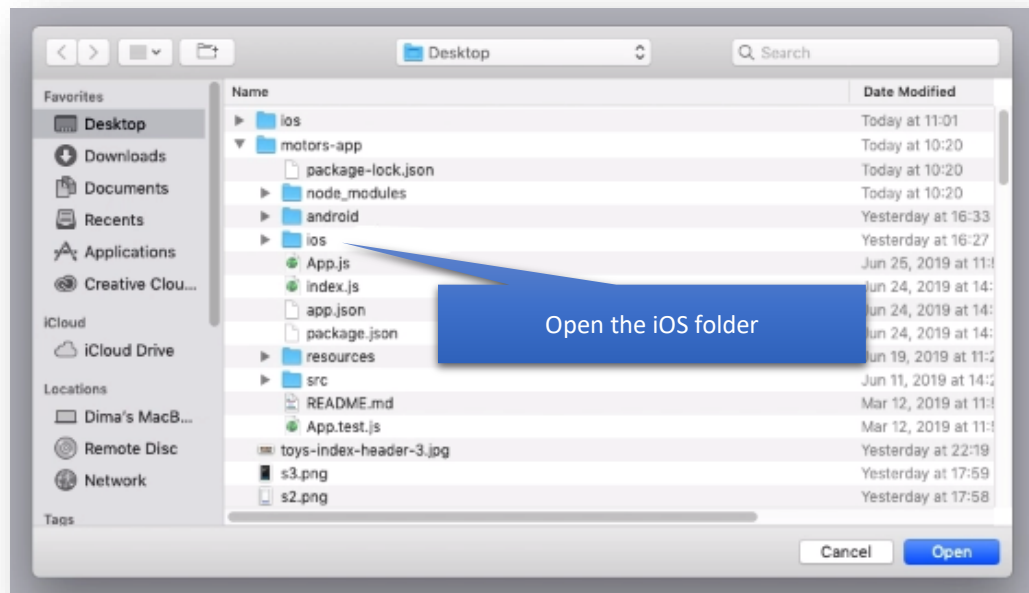
Step 5 – Setting up the project

In order to set display name, Bundle identifier, version, build, etc. of your app, you need to install the **Xcode** tool. You can download the **Xcode** [here](#).

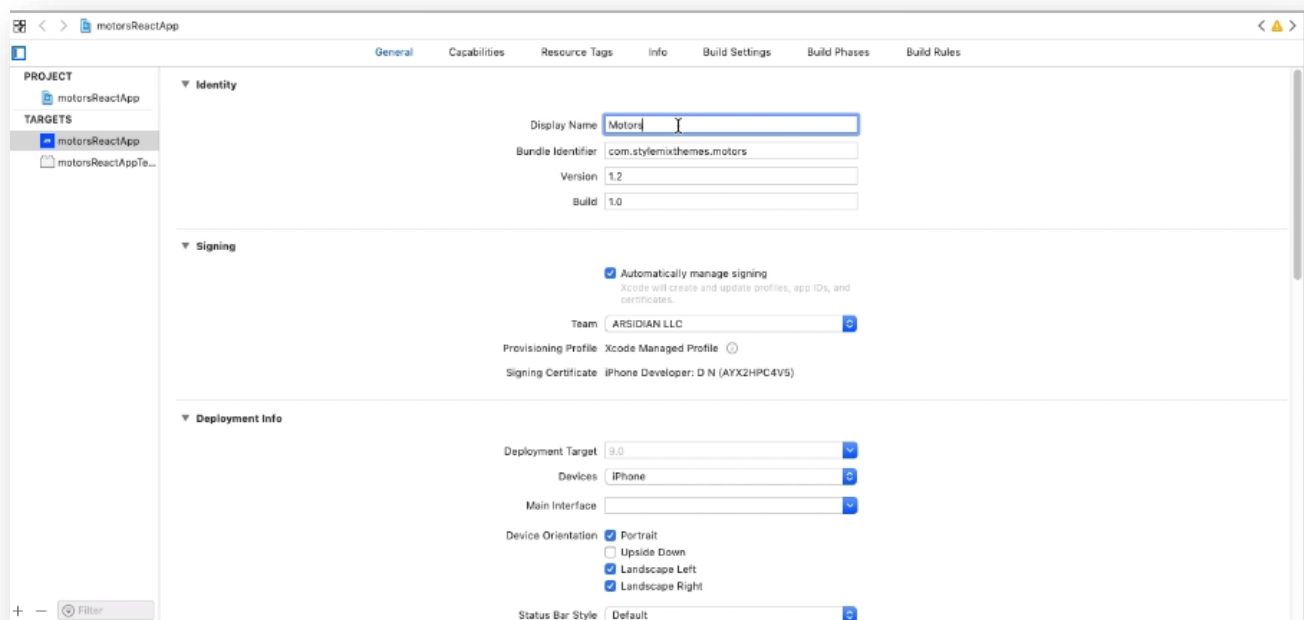
After, run the Xcode and click on “**Open another project...**” option in the welcome window.



In the popup window, select and open the “**motors-app/iOS**” folder.



Next, you need to enter display name, bundle identifier, version, and build in the **Identity** section.



- **Display Name** – the name of your app which will be displayed on the home screen/elsewhere;
- **Bundle Identifier** – the Bundle ID registered on App Store Connect (*com.yourcompanydomain.appname*);
- **Version** – the current version of your app;
- **Build** – the build version of your app.

The screenshot shows the 'Identity' section of the Xcode interface. It contains four text input fields: 'Display Name' (containing 'Motors App'), 'Bundle Identifier' (containing 'biz.stylemix.motorsapp'), 'Version' (containing '1.0'), and 'Build' (containing '1.0'). Four blue callout boxes with white text point to these fields: 'Enter your app display name' points to 'Display Name', 'Enter Bundle ID of your app' points to 'Bundle Identifier', 'Enter current version and build version of your app' points to 'Version', and another 'Enter current version and build version of your app' points to 'Build'.

In the **Signing** section, you need to select your team which connected to your Apple Developer account.

Note. **Automatically manage signing** option is active by default. This option allows Xcode automatically managing app signing and supplying.

The screenshot shows the 'Signing' section of the Xcode interface. It features a checked checkbox labeled 'Automatically manage signing' with a subtext 'Xcode will create and update profiles, app IDs, and certificates.' Below this is a 'Team' dropdown menu. A blue callout box with white text points to the 'Team' dropdown, stating 'Select your team connected to Apple Developer account'. Below the 'Team' field, it shows 'Provisioning Profile' set to 'Xcode Managed Profile' and 'Signing Certificate' set to 'iPhone Developer:'.

Tip: For detailed overview of app signing, have a look at the [Create, export, and delete signing certificates](#).

The next section is the **Deployment Info** section where you need to set the minimum iOS version your app will support. Flutter supports iOS 8.0 and later.

▼ **Deployment Info**

Deployment Target: 8.0

Devices: Universal

Main Interface: Main

Device Orientation:
☒ Portrait
☐ Upside Down
☒ Landscape Left
☒ Landscape Right

Status Bar Style: Default

☐ Hide status bar
☐ Requires full screen

Step 6 – Add an App Icon

You can add icons to your app in the App Icons and Launch Images section of the Xcode General panel. To add icons click on the **arrow** icon and open the asset catalog.

▼ **App Icons and Launch Images**

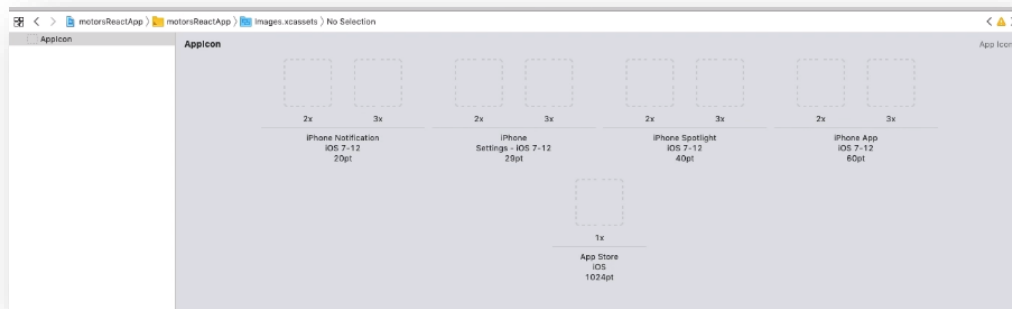
App Icons Source: AppIcon

Launch Images Source: Use Asset Catalog...

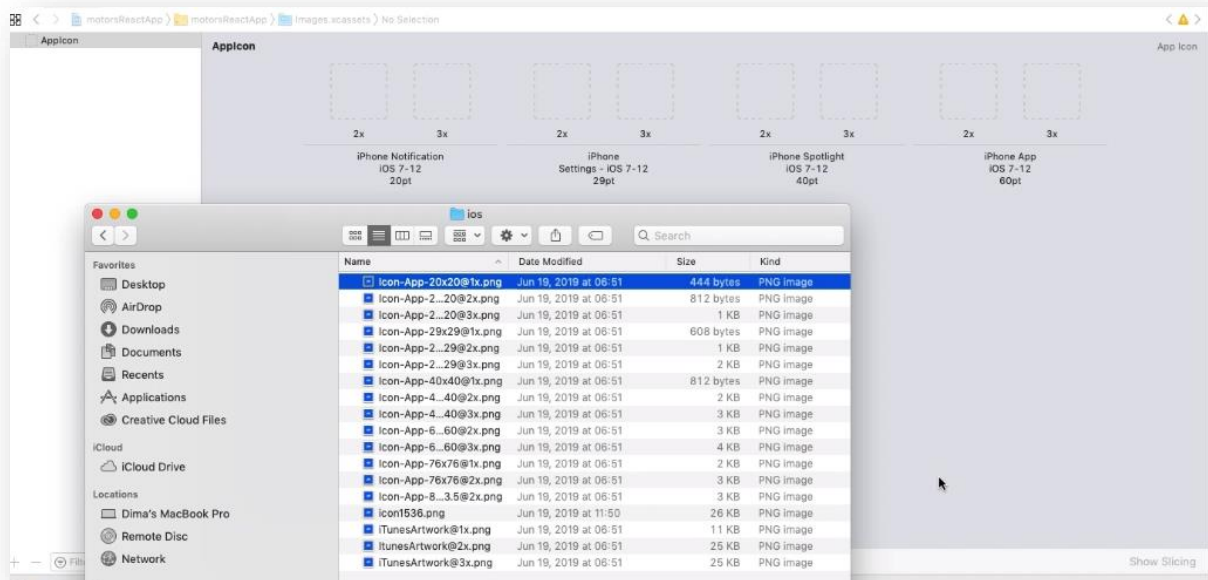
Launch Screen File: LaunchScreen

Click on the arrow icon to open the asset catalog

Tip: Follow the [Human Interface Guidelines](#) when creating the app icon. You can use the [App Icon Generator](#) tool.



In the **Finder**, drag variations of your app icon to the wells in the detail area match their resolutions.



Alternatively, select the asset catalog containing your app icon set in the [Project navigator](#) and drag variations to the wells.

Finally, review the icon set has been applied by running your app using Xcode device simulator.

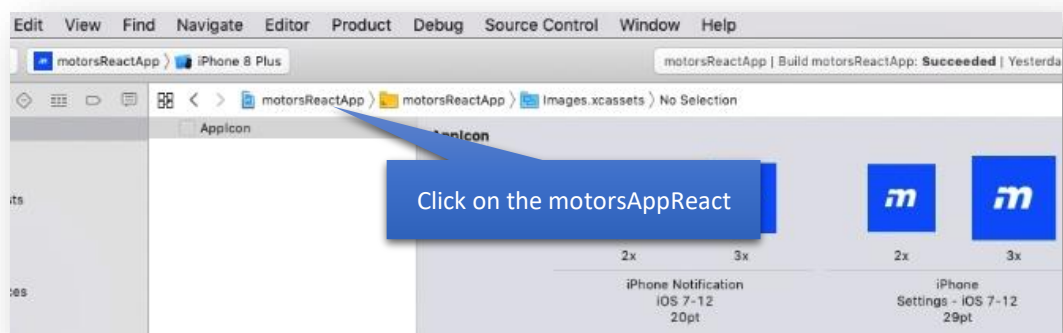
Tip: For detailed overview of using simulator, see the [Run your app in Simulator from Xcode](#).

iOS App Release

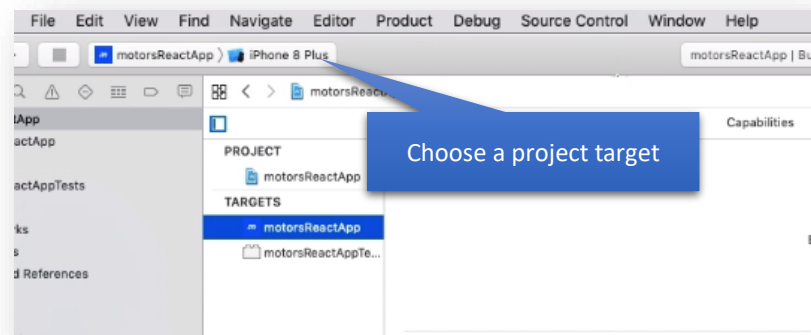
Step 1 – Create a build archive

You need to create a build archive in order to upload your build to App Store Connect. The current build is a debug build for building, debugging, and testing. When you are ready to publish your app on the App Store or TestFlight.

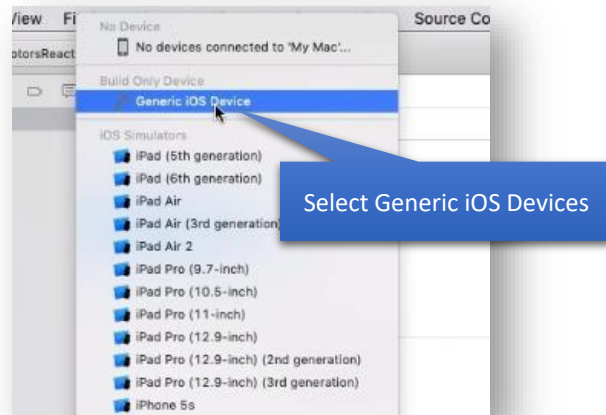
In the **Editor area**, select the **motorReactApp** project.



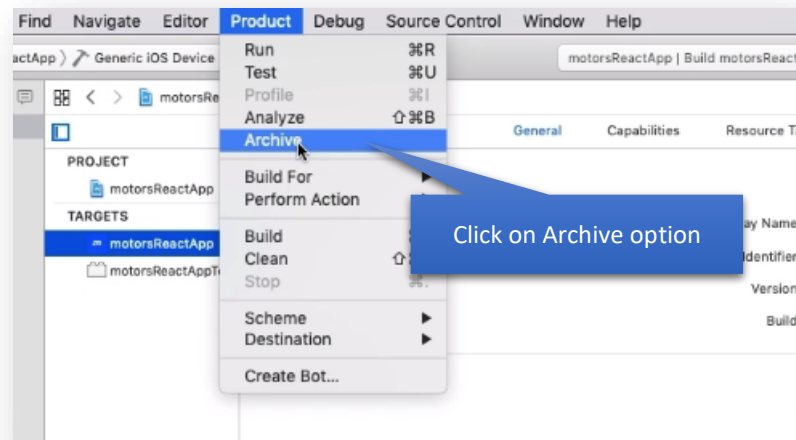
Next, choose a project target from the **Scheme toolbar** menu.



Select the **Generic iOS Device** option from the list.



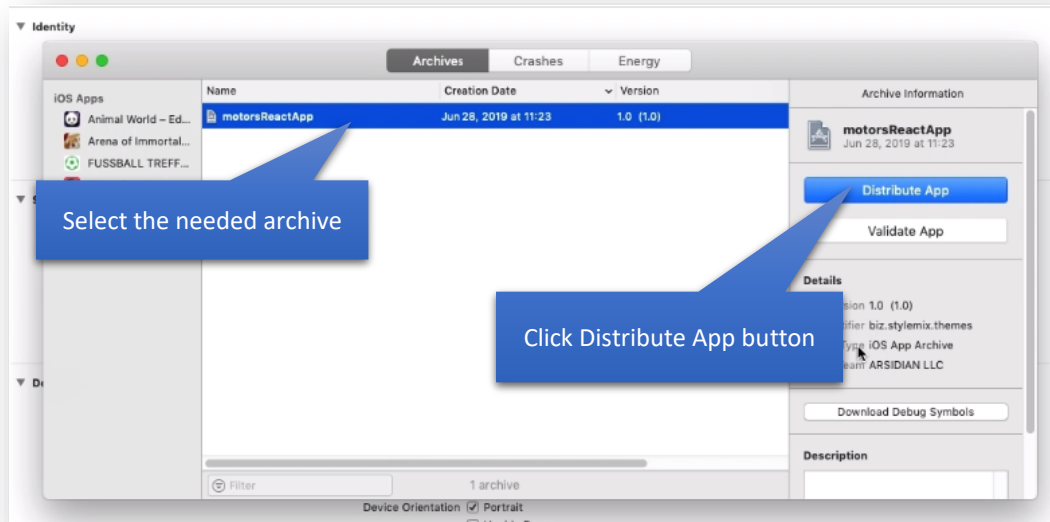
Finally, create a build archive by selecting **Product > Archive** from the Menu bar.



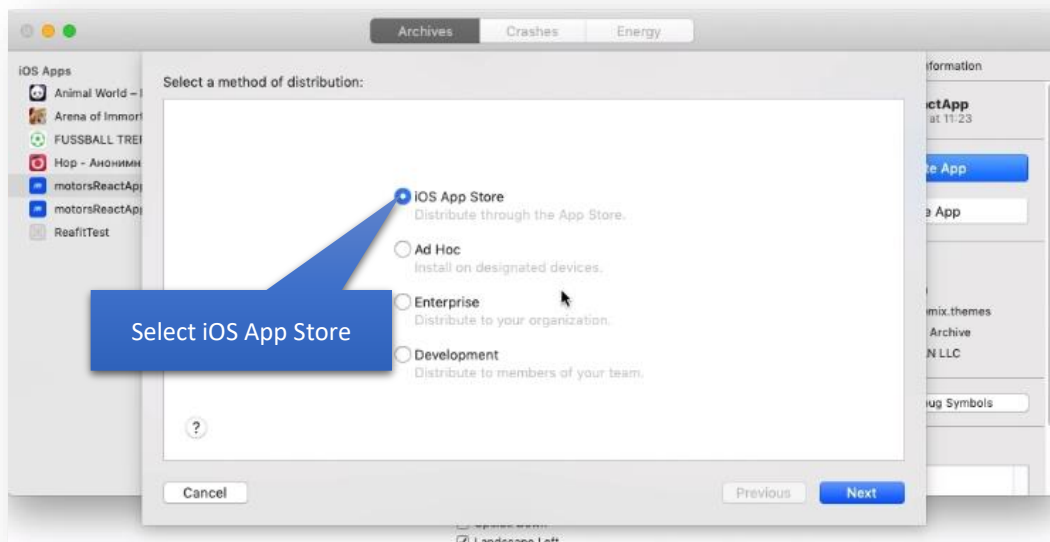
If the archive builds successfully, it appears in the [Archives organizer](#).

Step 2 – Upload an app to App Store Connect

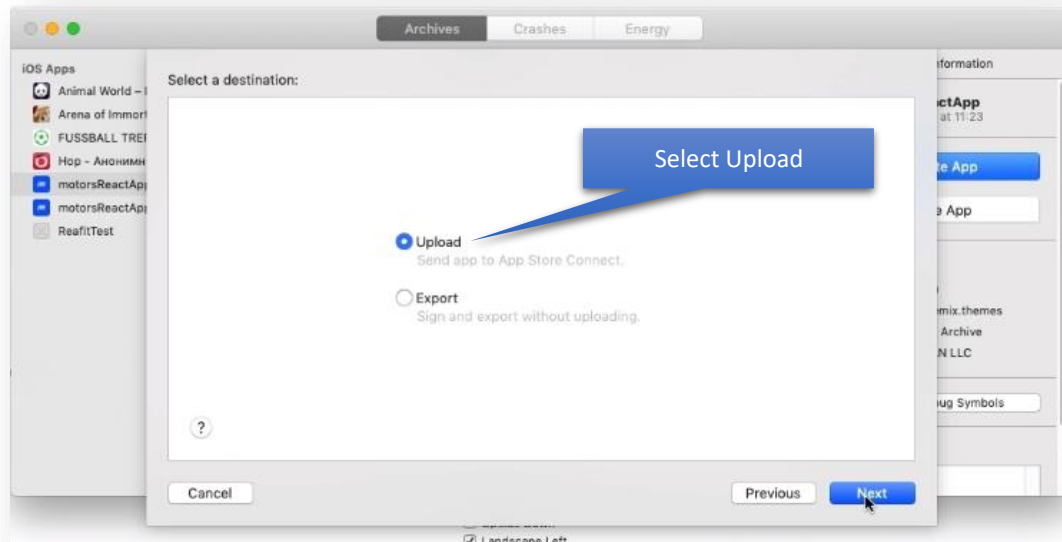
In the Archives organizer, select the created archive and click **Distribute App** button.



In the next window, select **App Store** as the distribution method, then click Next button.

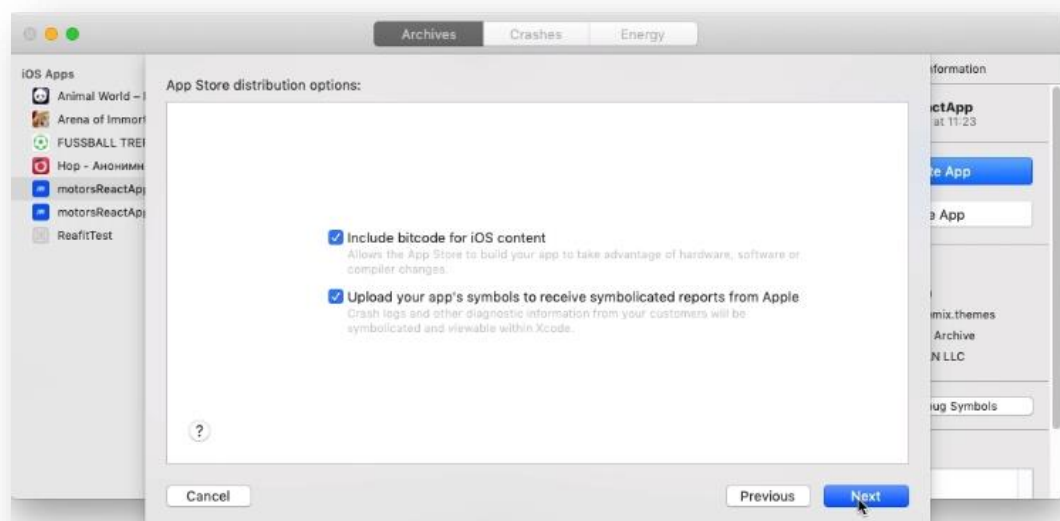


In the next sheet, select **Upload** and click Next button.



Alternatively, to export the app before uploading it, click Export, then click Next. Select a location for the files, then click Export. A folder containing the [archive export files](#) appears in Finder.

In the next sheet that appears, choose distribution options, then click Next.



Tip: For detailed overview of the distribution options, see the [Distribution options](#) guide.

In the next sheet, choose a signing option, then click Next.



Tip: For detailed overview of the signing options, have a look at the [Distribution signing options](#) guide.

Tip: If you select “**Manually manage signing**” option, follow alternate steps of the [Manually manage distribution signing](#) tutorial.

Review your app content – signing certificates, provisioning profiles, and entitlements in the next sheet.



Click Upload to run uploading process of your app to the App Store.

