Thank you for your purchase! To ensure seamless after-sales support, please activate your product using the link below:

Activate Now

Your activation is essential for us to provide you with the best assistance. We appreciate your cooperation.

Technology Used

- **Flutter** with Dart Language for Mobile App
- **NodeJS** with Express Framework for API
- **MySQL** for Database
- **MariaDB** 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. Setup Prerequisite (If not available)

a. Install Visual Studio Code (VSCode) from this link

b. Install NodeJS from this link (Minimum version 16.14.0)

c. Install and set up Flutter from this link

d. Install MySQL from this link

(You can choose the MySQL edition based on your needs)

e. Install MySQL Workbench from this link (This is optional)

2. Setup Web API (Technology Node js)

a. Setup database credentials

i. Open **variable.json** file from **Source Code\Web API** And replace your credentials

3. Insert Master Data into the Database

a. After a successful web API setup, you just need to create an empty database with the same credentials that you have provided above in the **db.ts** file. If you have already done this then please skip this step. Or you can refer **Accounts-Pro-Quick-Start-Guide** document for how to create a database

b. After the above step. Please go to the web API and run the command **npm start**. While running the web API it will check and create all the needed tables in an empty database

c. After all tables are created then you need to import CSV files for the master data. Go to the database and follow below steps

i. Open **valuetypes** table and import **Source Code\Master Data csv\valueTypes.csv**

ii. Open **authproviders** table and import **Source Code\Master Data csv\authProviders.csv**

iii. Open **country** table and import **Source Code\Master Data csv\country.csv**

iv. Open **currency** table and import **Source Code\Master Data csv\currency.csv**

v. Open **expensecategories** table and import **Source Code\Master Data csv\ExpenseCategories.csv**

vi. Open **flaggroup** table and import **Source Code\Master Data csv\flagGroups.csv**

vii. Open **gstslabs** table and import **Source Code\Master Data csv\GstSlabs.csv**

viii. Open **modulegroups** table and import **Source Code\Master Data csv\moduleGroups.csv**

ix. Open **modulepermissions** table and import **Source Code\Master Data csv\modulePermissions.csv**

x. Open **modules** table and import **Source Code\Master Data csv\modules.csv**

xi. Open **systemflags** table and import **Source Code\Master Data csv\systemFlags.csv**

xii. Open **unitcombinations** table and import **Source Code\Master Data csv\UnitCombinations.csv**

xiii. Open **Units** table and import **Source Code\Master Data csv\Units.csv**

xiv. Open **userflaggroup** table and import **Source Code\Master Data csv\userFlagGroups.csv**

xv. Open **userflags** table and import **Source Code\Master Data csv\userFlags.csv**

xvi. Open **userroles** table and import **Source Code\Master Data csv\UserRoles.csv**

4. Setup Mobile App (Technology Flutter)

a. Initial steps to set up and run mobile app

i. Open the **App** 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
```

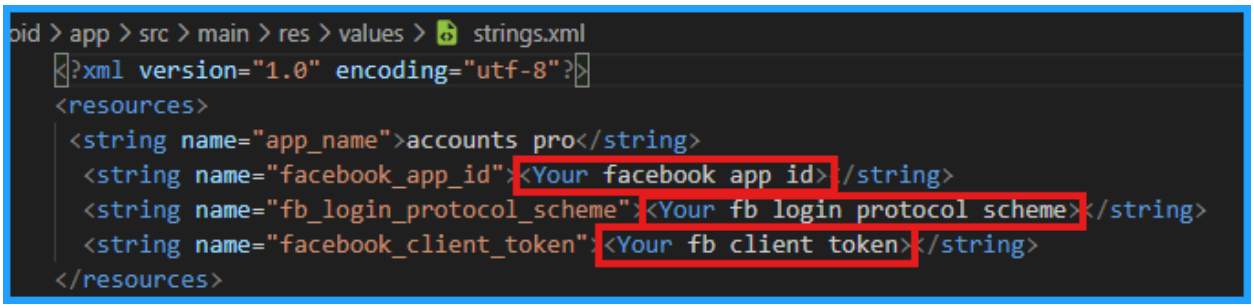
vi. To upgrade the Flutter version run the following command in the VSCode Terminal

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

```
flutter upgrade
```

b. Setup Facebook app Credentials

Open string.xml file from **App\android\app\src\main\res\values** and set credentials. If you do not have setup face book app then please visit https://developers.facebook.com/docs/development/create-an-app/ for more information



c. Change Credentails

After the setup of your API and Admin panel, Open **variable.json** file from **Source Code\App\assets\jsondata\variable.json** And replace your credentials where value starting from **<Your**

```
{
    "baseURL": "<Your Base URL Here>",
    "apiBaseURL": "/api/",
    "imageBaseurl":"/",
    "facebookIdForWeb": "<Your Facebook Id Here>",
    "firebaseConfig": {
        "apiKey": "<Your Firebase API Key Here>",
        "authDomain": "<Your Firebase Auth Domain Here>",
        "databaseURL": "<Your Firebase Database URL Here>",
        "projectId": "<Your Firebase Project Id Here>",
        "storageBucket": "<Your Firebase Storage Bucket URL Here>",
        "messagingSenderId": "<Your Firebase Messaging Sender Id Here>",
        "appId": "<Your Firebase App Id Here>",
        "measurementId": "<Your Firebase Measurement Id Here>"
    }
}
```

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/AndoidManifest.xml**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.demo.accountspro">
```

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

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.demo.accountspro">
```

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

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.demo.accountspro">
```

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

```
defaultConfig {
    // TODO: Specify your own unique Applicatio
    applicationId "com.demo.accountspro"
    minSdkVersion 25
```

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

   android\app\src\main\java\com\demo\accountspro

   Ex. If your package name is com.demo.accountspro

   android\app\src\main\java\com\app\accountspro

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

```
> app > src > main > java > com > demo > accountspro > MainActivity.java
package com.demo.accountspro;
import io.flutter.embedding.android.FlutterActivity;
import io.flutter.embedding.android.FlutterFragmentActivity;
```

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>
```

c. Change the credentials of face book app

```
25    <key>CFBundleURLTypes</key>
26    <array>
27        <dict>
28            <key>CFBundleTypeRole</key>
29            <string>Editor</string>
30            <key>CFBundleURLName</key>
31            <string></string>
32            <key>CFBundleURLSchemes</key>
33            <array>
34                <string>com.googleusercontent.apps.<Your google app id></string>
35                <string><Your fb app id></string>
36            </array>
37        </dict>
38    </array>
39    <key>FacebookAppID</key>
40    <string><Your fb app id></string>
41    <key>FacebookClientToken</key>
42    <string><Your fb client token></string>
```

2. In XCode

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

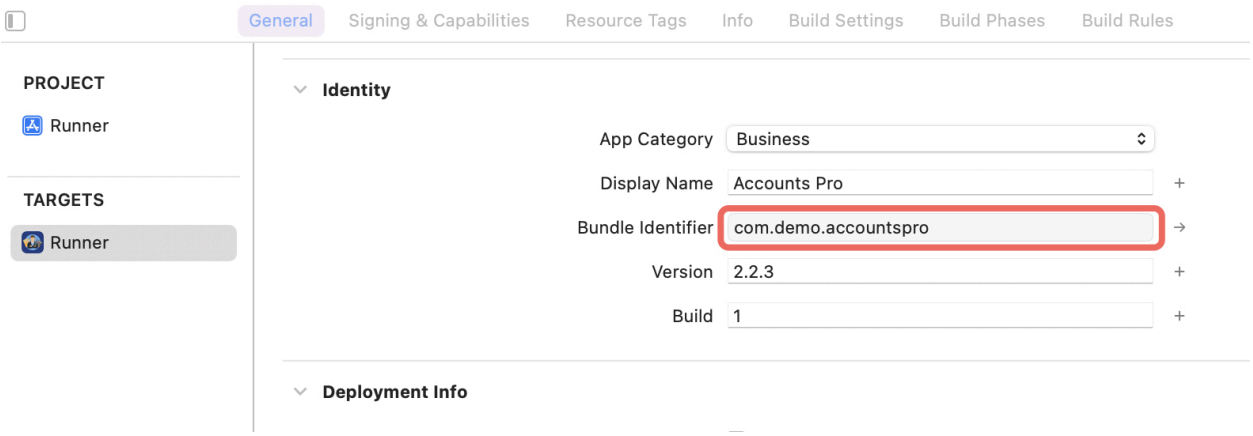
b. Click on the folder icon left side of the XCode window

c. Select Runner.

d. Select Target runner

e. Go to identity

f. Change Bundle Identifier



e. Create and set Keystore file for Android

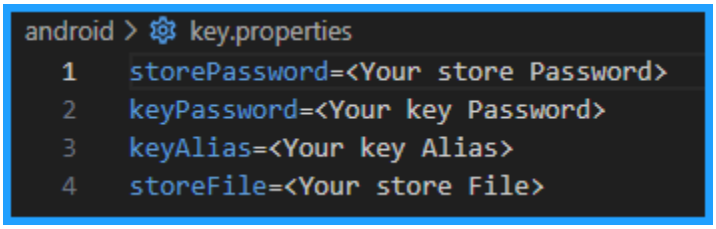
i. Create a keystore.jks file, if not exist, use the below command in the terminal

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 key.properties file in the **android** folder and add the details in the file as per the below screenshot.



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 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. Firestore Database

iv. Firebase Analytics

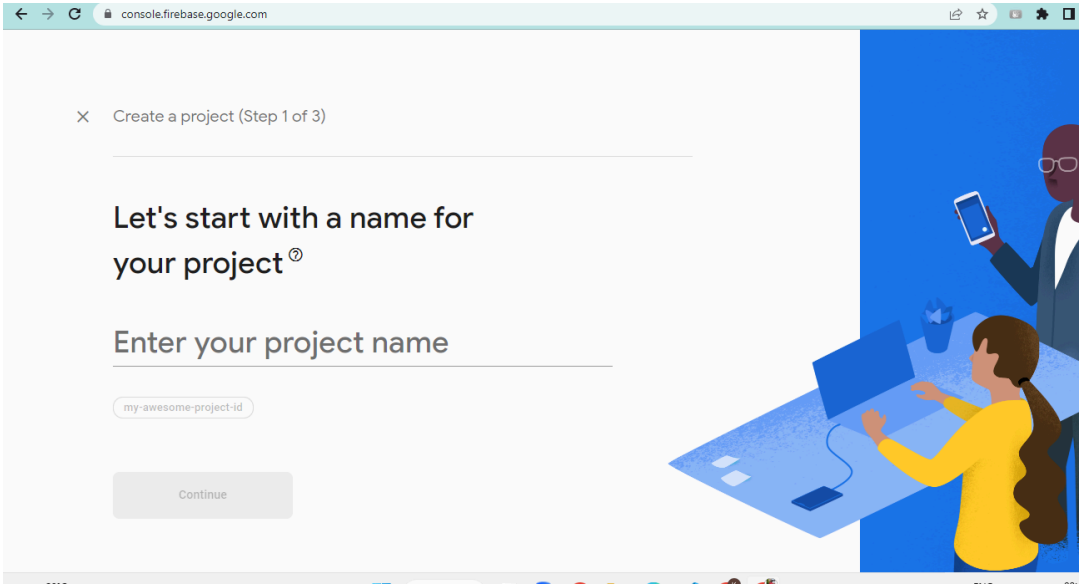
For this, you need a Firebase account and a project set up in the Firebase. Please follow the below steps for this,

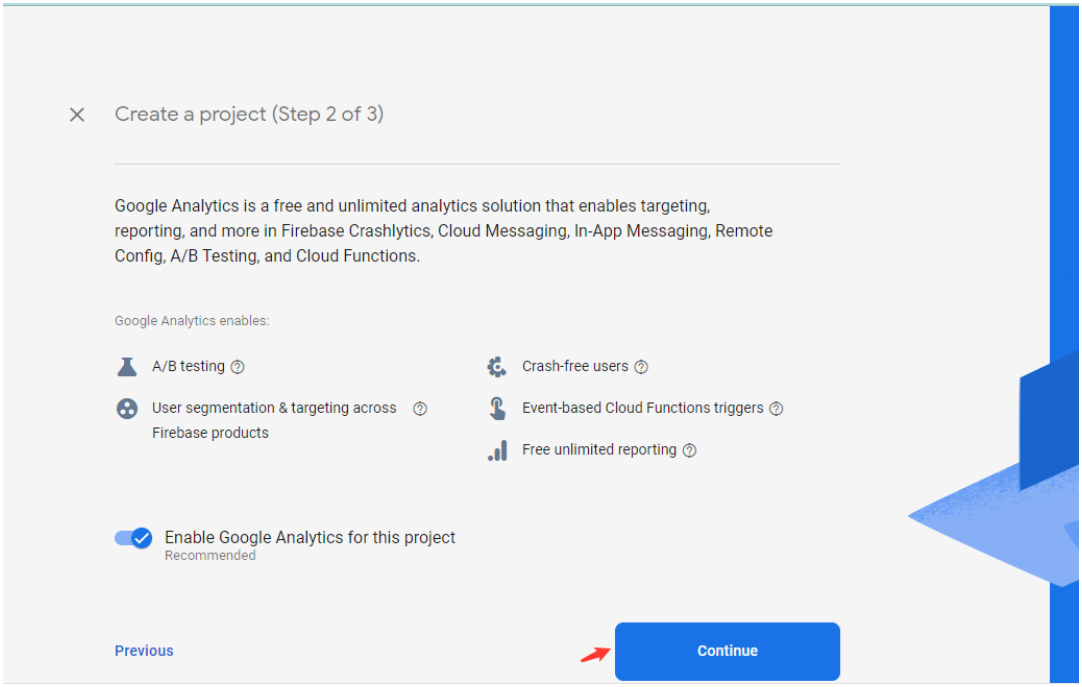
i. Go to the [Firebase console]

ii. Sign up if you don't have a Google Account or you want to create a new account for your project. Otherwise, sign in with your Google Account.

iii. Click on **Add Project**

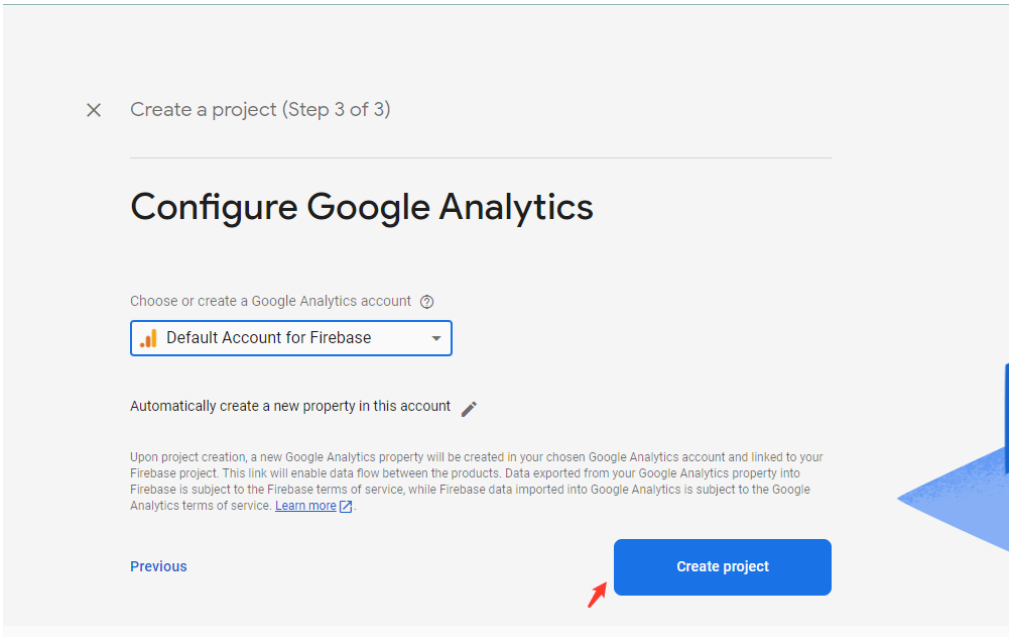
iv. Enter your project name





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.d.vi

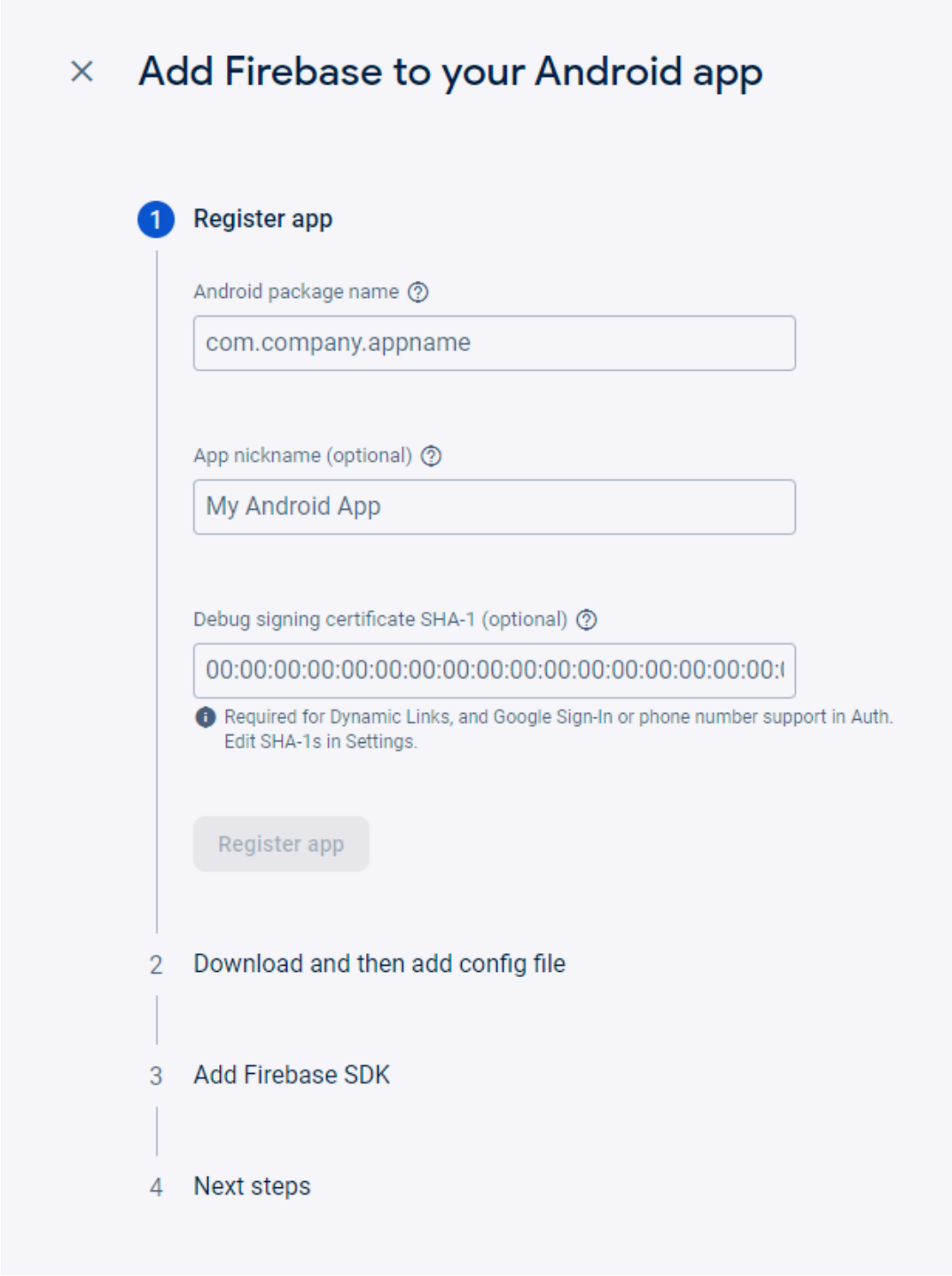
iii. Go to **Project Setting**

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.

1. To Generate debug SHA use the below command

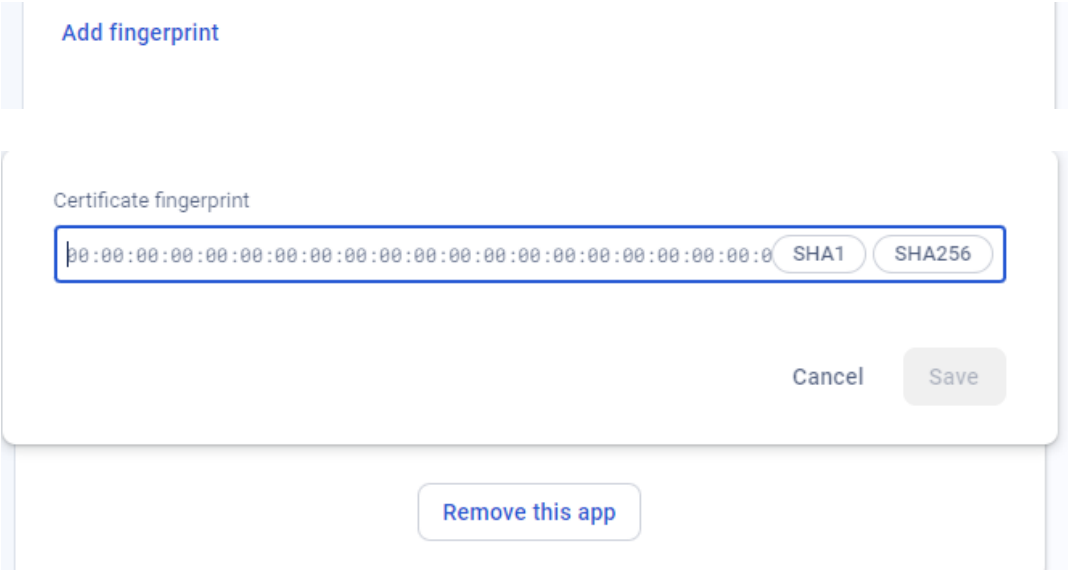
keytool -list -v -keystore "Your directory path\debug.jks" -alias androiddebugkey -storepass android -keypass android

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.

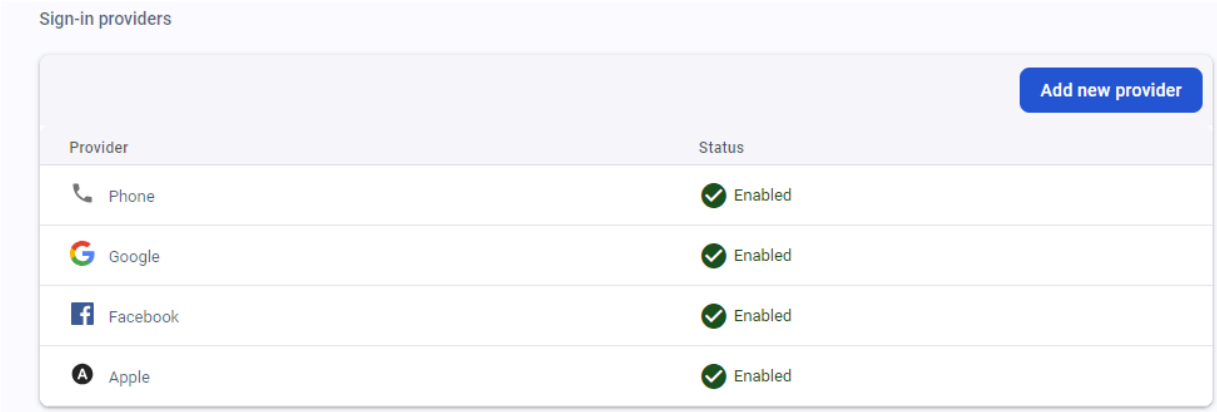
Please check the screenshot below for the reference.



viii. Download the google-services.json file from Firebase project settings and paste it at **android/app** location.

ix. Setup Authentication

- Enable Sign in methods:
  a. In the Firebase console's **Authentication** section, open the Sign in method page.
  b. From the **Sign in method** page, enable the methods which are shown in the image.



h. Setup Firebase iOS App

   i. Go to the Firebase console

   ii. Select the project you created in step 5.d.vi

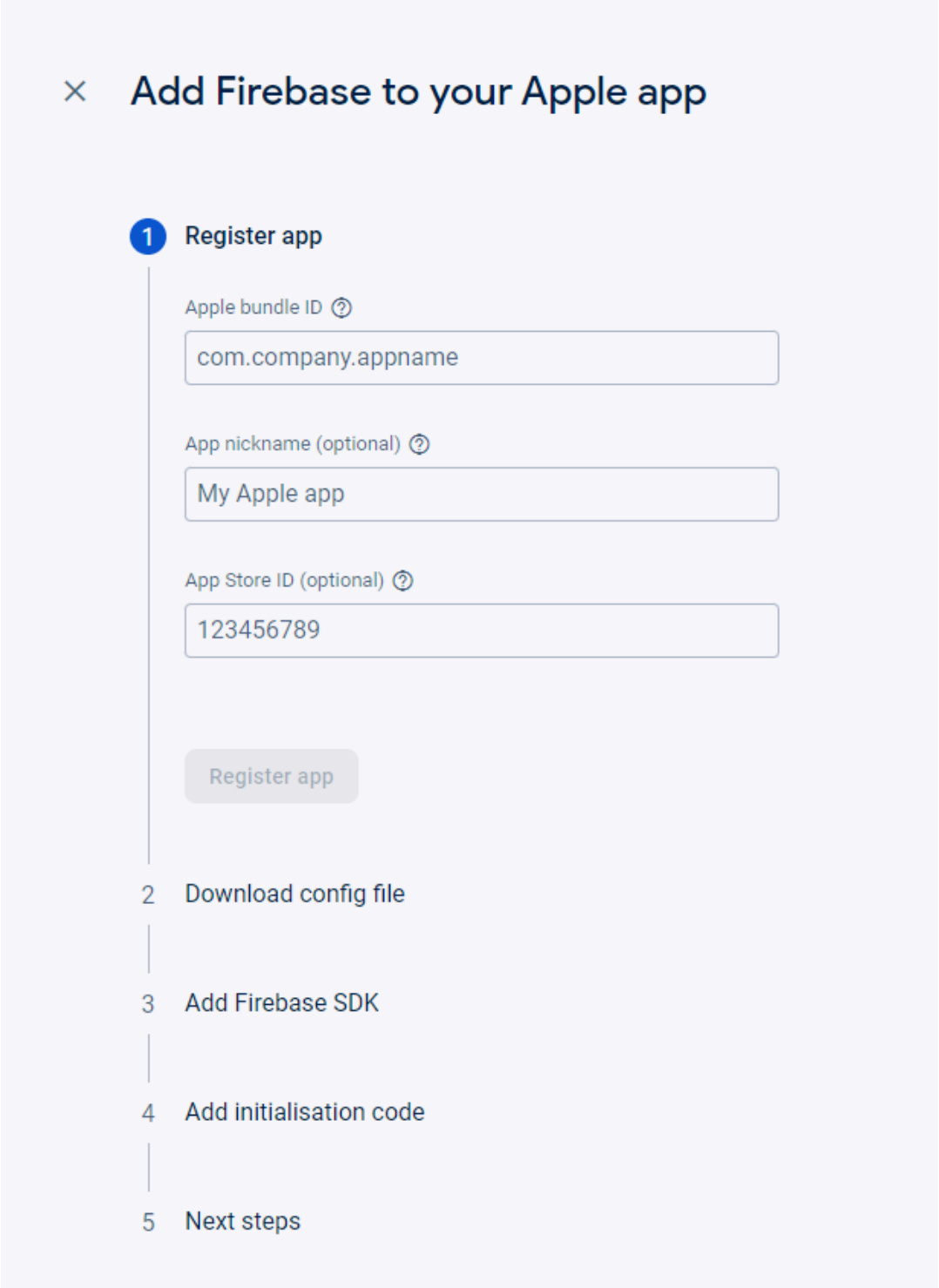
   iii. Go to **Project Setting**

   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)



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

viii. Change the credentials of google app

```
<array>
    <dict>
        <key>CFBundleTypeRole</key>
        <string>Editor</string>
        <key>CFBundleURLName</key>
        <string></string>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>com.googleusercontent.apps.<Your google app id></string>
            <string><Your fb app id></string>
        </array>
    </dict>
</array>
```

ix. Go to the **Source Code\App\web\index.html** file and replace **<!-- <meta name="google-signin-client_id" content="YOUR_GOOGLE_SIGN_IN_OAUTH_CLIENT_ID.apps.googleusercontent.com"> -->** with your google app id.

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/cropperjs/1.6.2/cropper.css" />
<script src="https://cdnjs.cloudflare.com/ajax/libs/cropperjs/1.6.2/cropper.min.js"></script>
<script type="application/javascript" src="/assets/packages/flutter_inappwebview_web/assets/web/web_support.js" defer></script>
<script src="https://apis.google.com/js/platform.js" async defer></script>

<!-- <meta name="google-signin-client_id" content="YOUR_GOOGLE_SIGN_IN_OAUTH_CLIENT_ID.apps.googleusercontent.com"> -->

<title>Accounts Pro</title>
<link rel="manifest" href="manifest.json">
```

x. Go to the **ios\Runner\AppDelegate.m** file and replace “YOUR-API-KEY” with your Api key.

```
- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  [GMSServices provideAPIKey:@"YOUR-API-KEY"];
  [GeneratedPluginRegistrant registerWithRegistry:self];
  // Override point for customization after application launch.
```

xi. Replace “YOUR-REVERSED-CLIENT-ID” with your reversed client ID. You can find this Id from the GoogleService-info.plist file you added from step **5.g.vii**.

```
    <string>Editor</string>
    <key>CFBundleURLName</key>
    <string>$(REVERSED_CLIENT_ID)</string>
    <key>CFBundleURLSchemes</key>
    <array>
        <string>YOUR-REVERSED-CLIENT-ID</string>
    </array>
</dict>
<dict>
```

xii. 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

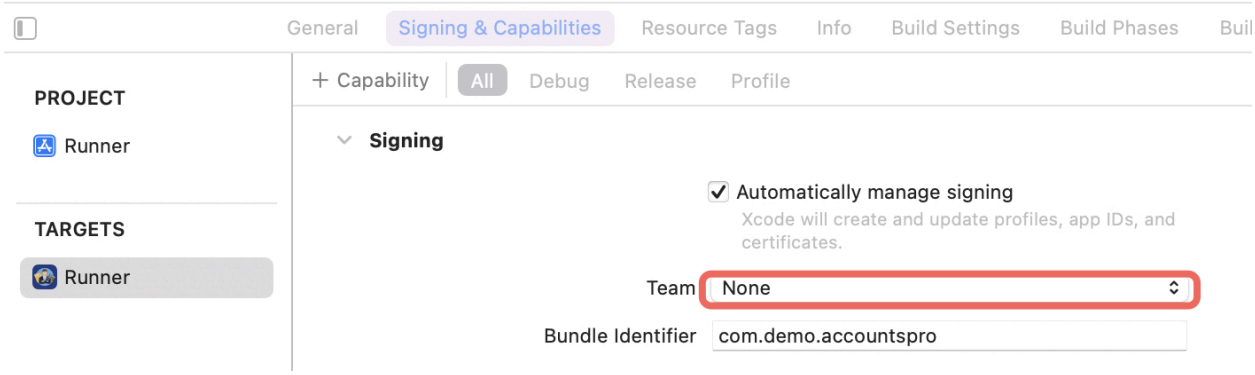
1. Navigate to your target's settings in XCode:
    a. Open the default Xcode workspace in your project by running the below command in a terminal window from your Flutter project directory.

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

    b. Select the Runner target in the Xcode navigator to view your app's settings.

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

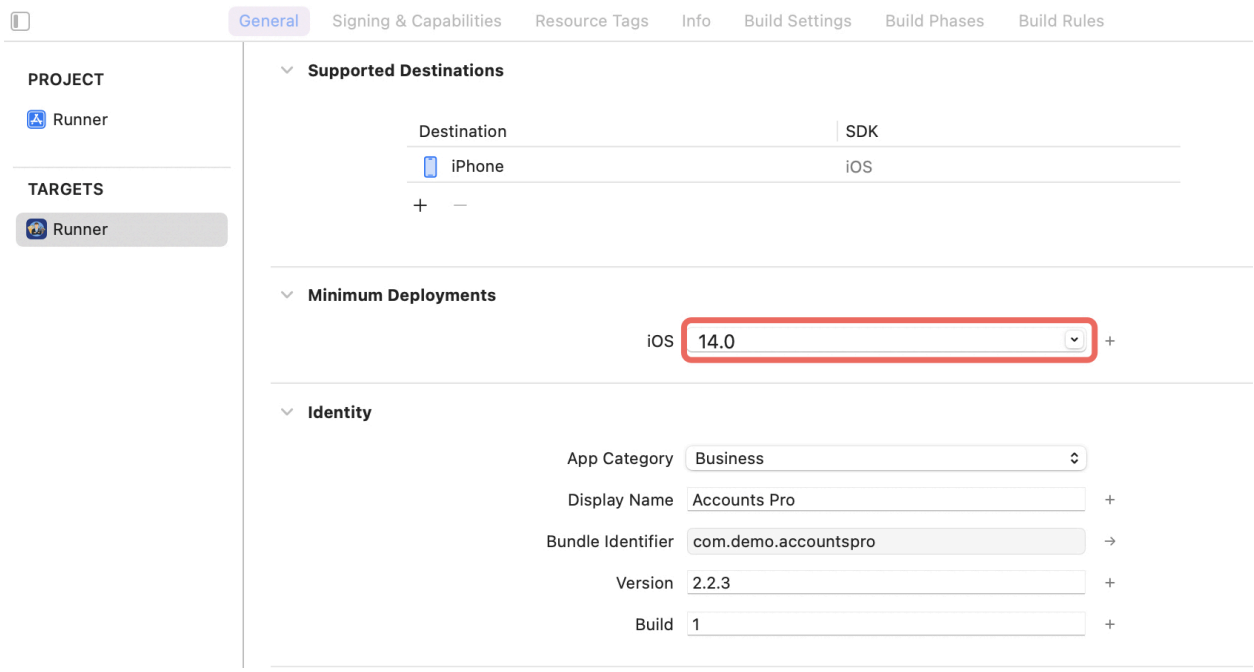
update this setting.)



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

   i. iOS Deployment Target

      1. The minimum iOS version that the app supports is 14.0.
      2. The General tab of your project settings should resemble the following:

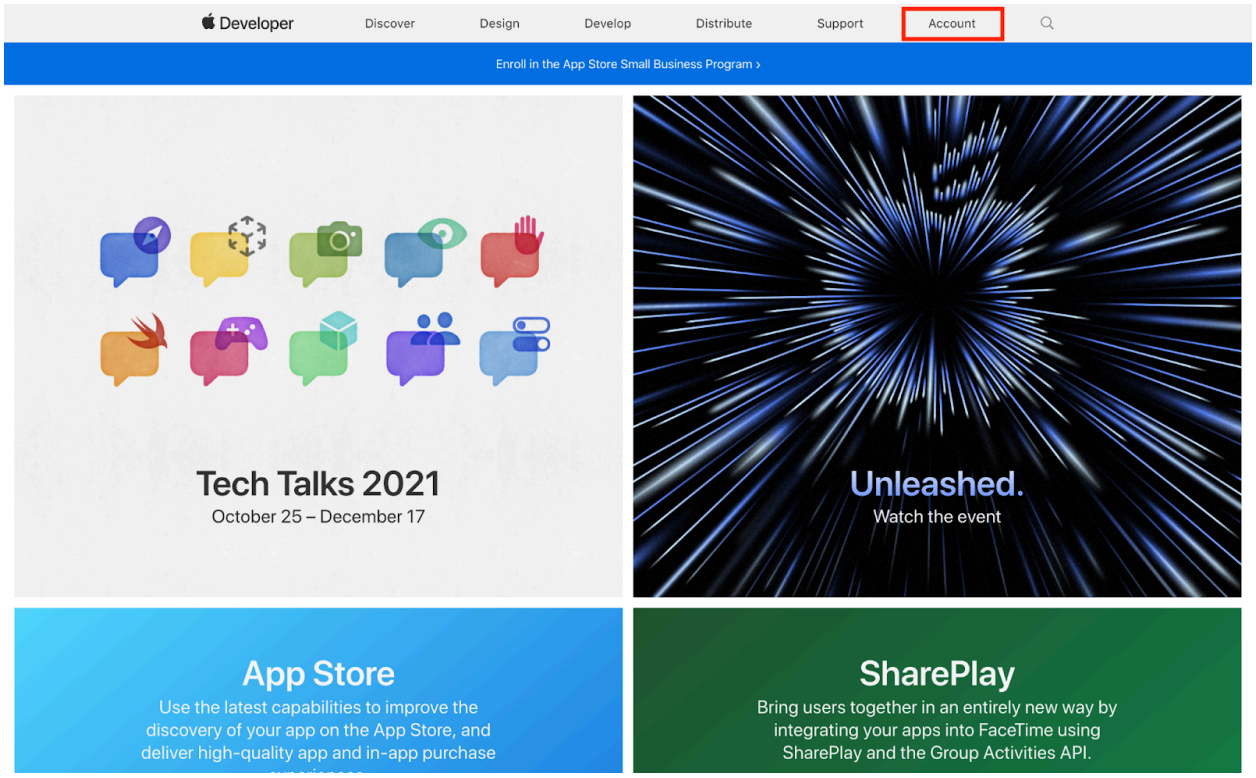


      3. For a detailed overview of app signing, see Create, export, and Delete signing certificates.

xiii. Upload your APNs authentication key

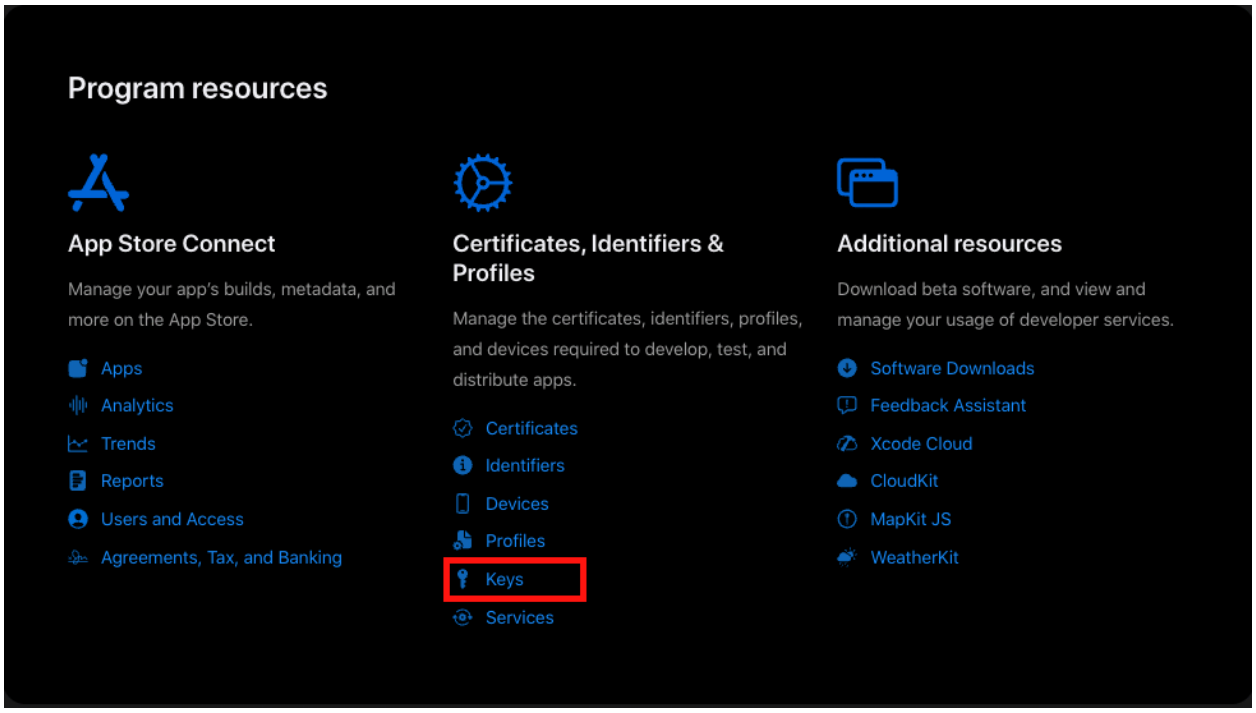
If you don't already have an APNs authentication key, make sure to create one.

1. Go to https://developer.apple.com and click Account

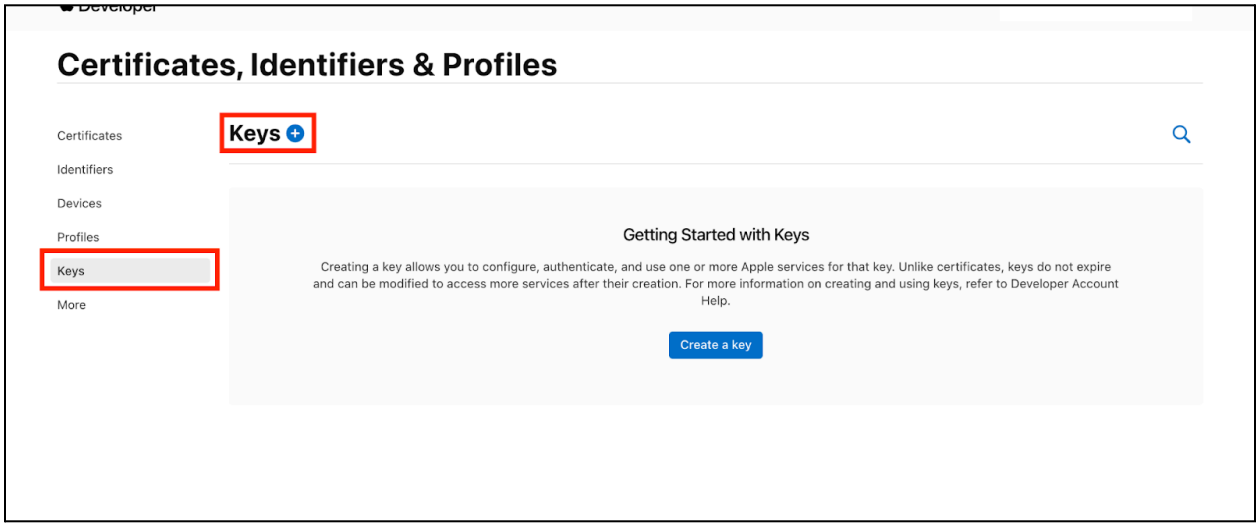


2. Log in with your Apple Developer account

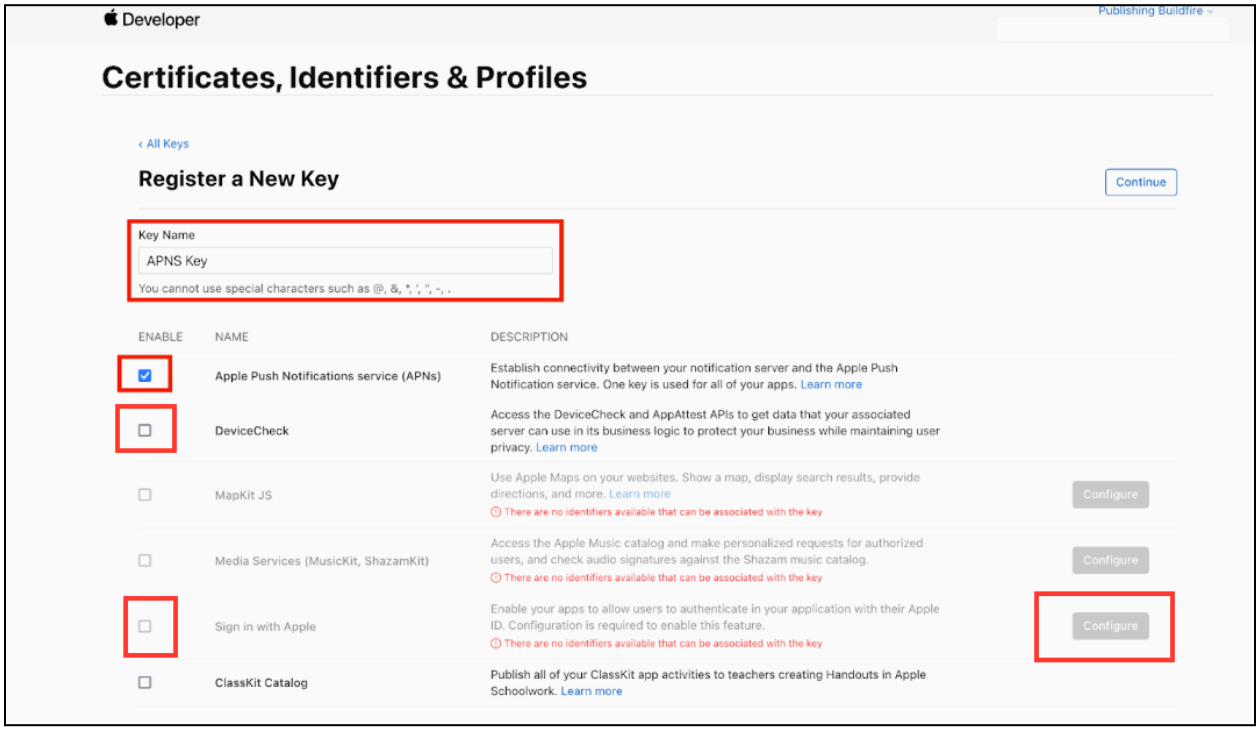
3. Click on Certificates, IDs & Profiles



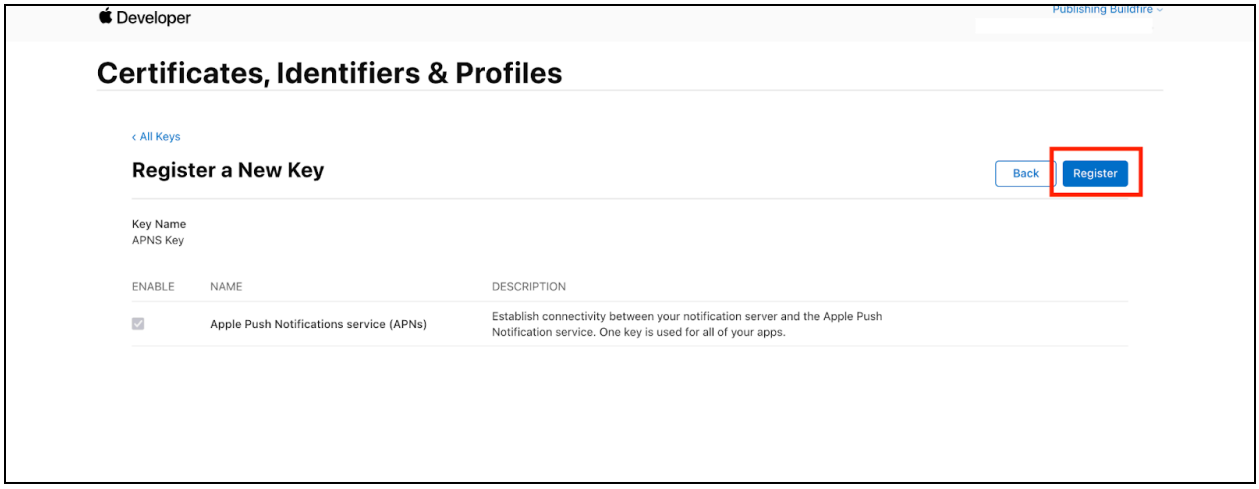
4. Click on Keys and add a new key by clicking on the blue plus (+) icon next to the title Keys.



5. On the next page, enter ‘APNS Key’ in the Key Name field and click the checkbox to enable Apple Push Notifications service (APNs), Device Check and Sign in with Apple. Also configure the Sign in with Apple.



6. Click Register

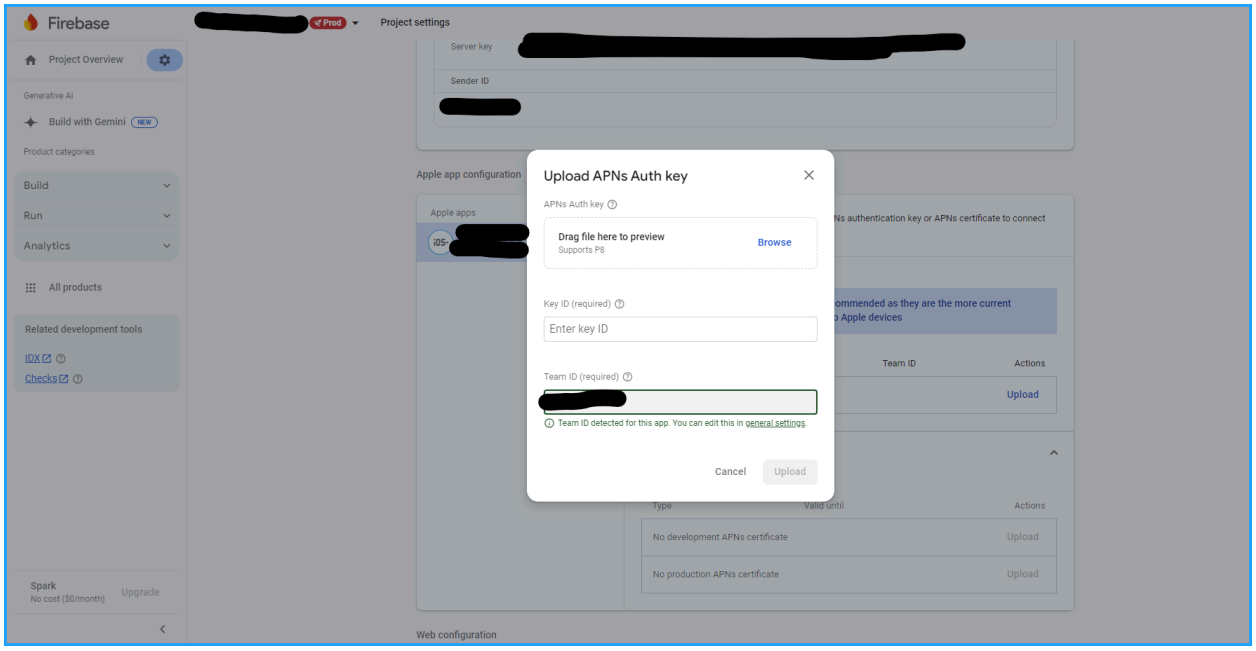


7. Click Download. This will download the APNs key that you will upload to Firebase. Please keep this page open to obtain the Key ID and Team ID for Firebase.

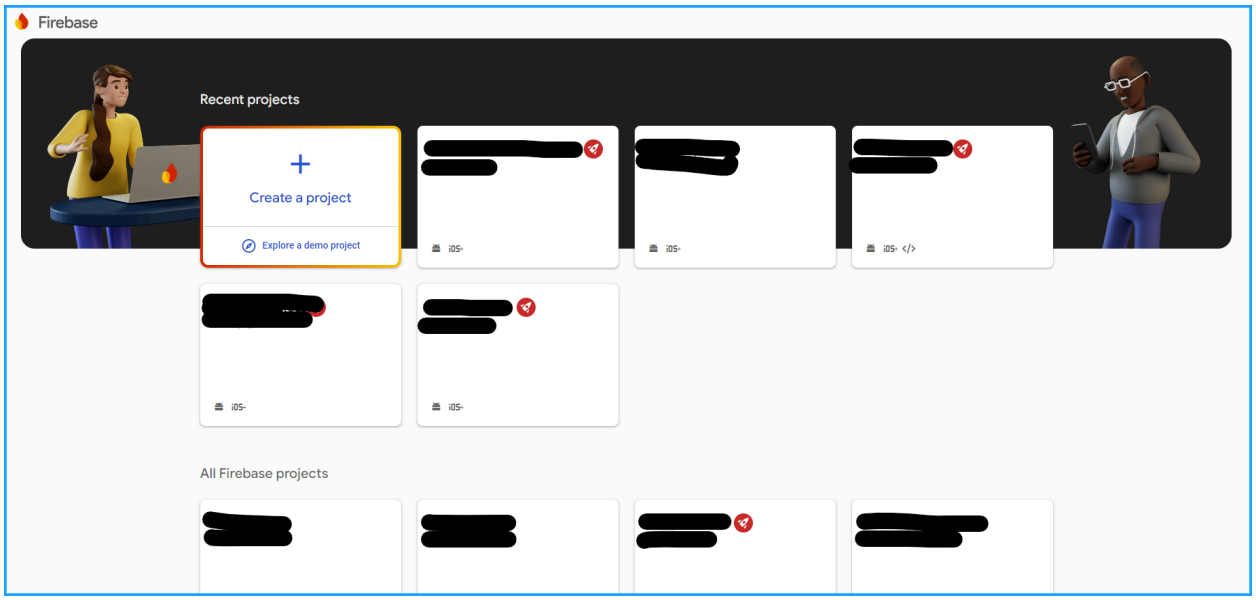
NOTE: Once the key has been downloaded, it cannot be retrieved again.

8. Now that you have the APNS key downloaded, you will need to upload this to Firebase. Open up a new browser tab or window and navigate to https://console.firebase.google.com/

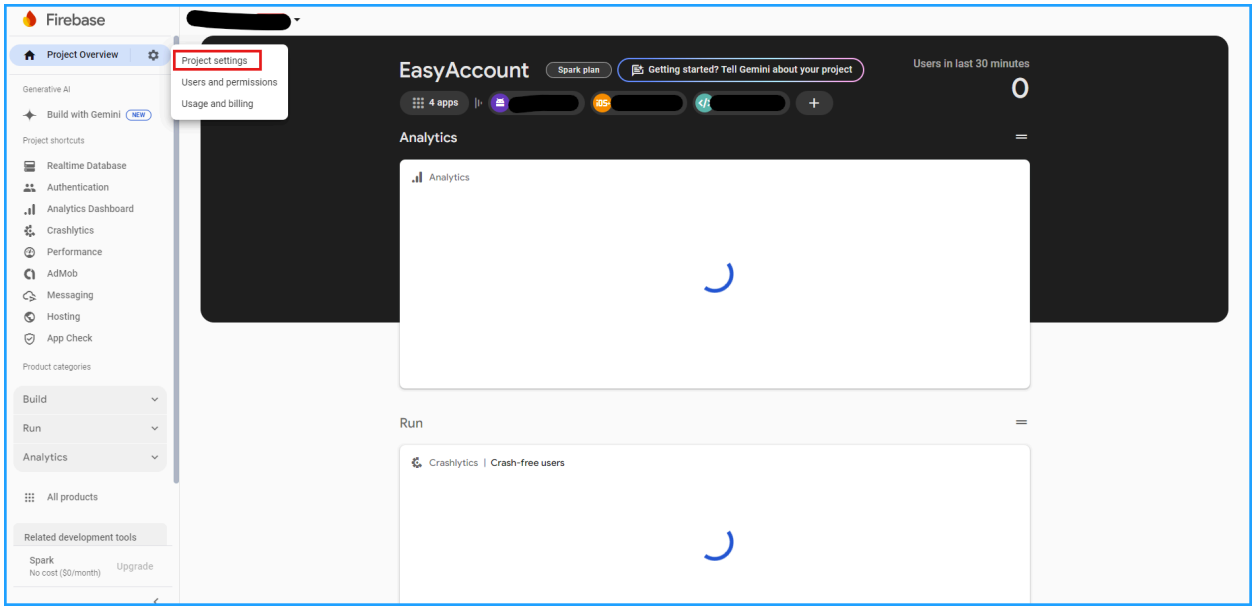
NOTE: The next few steps require you to have an iOS Firebase Certificate. If you have not done this yet, please check out our iOS Firebase Certificate article before continuing.



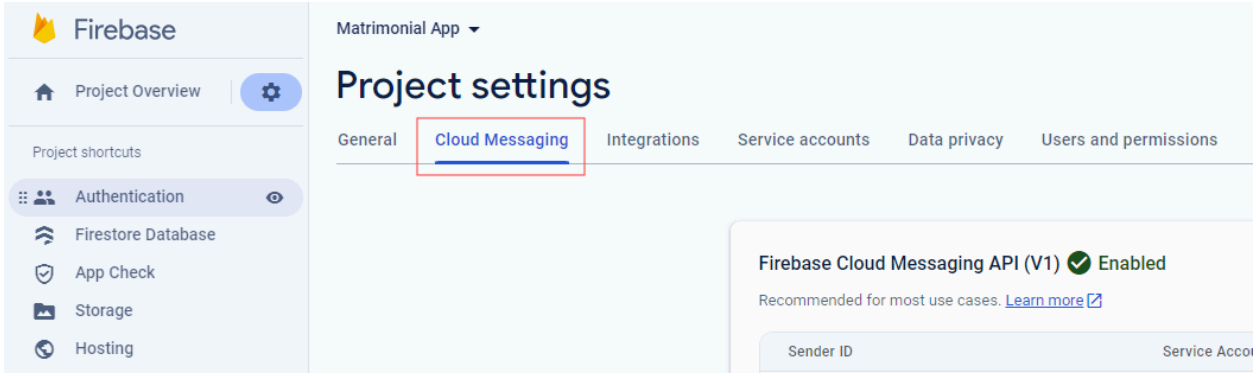
9. Click on your App project

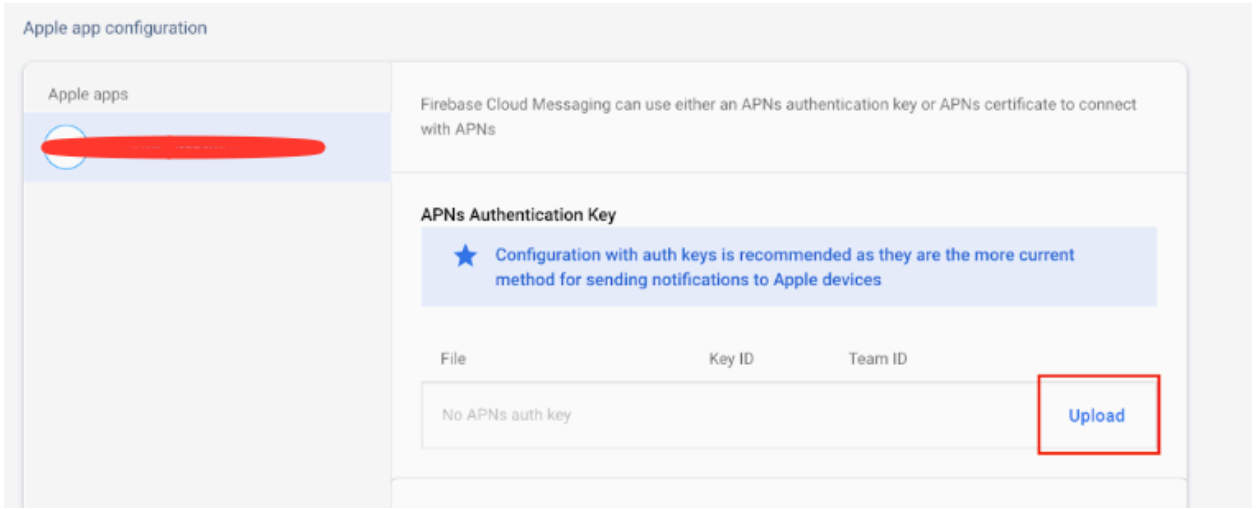


10. On the top left, click on the gear icon on the right side of Project Overview and select Project Settings



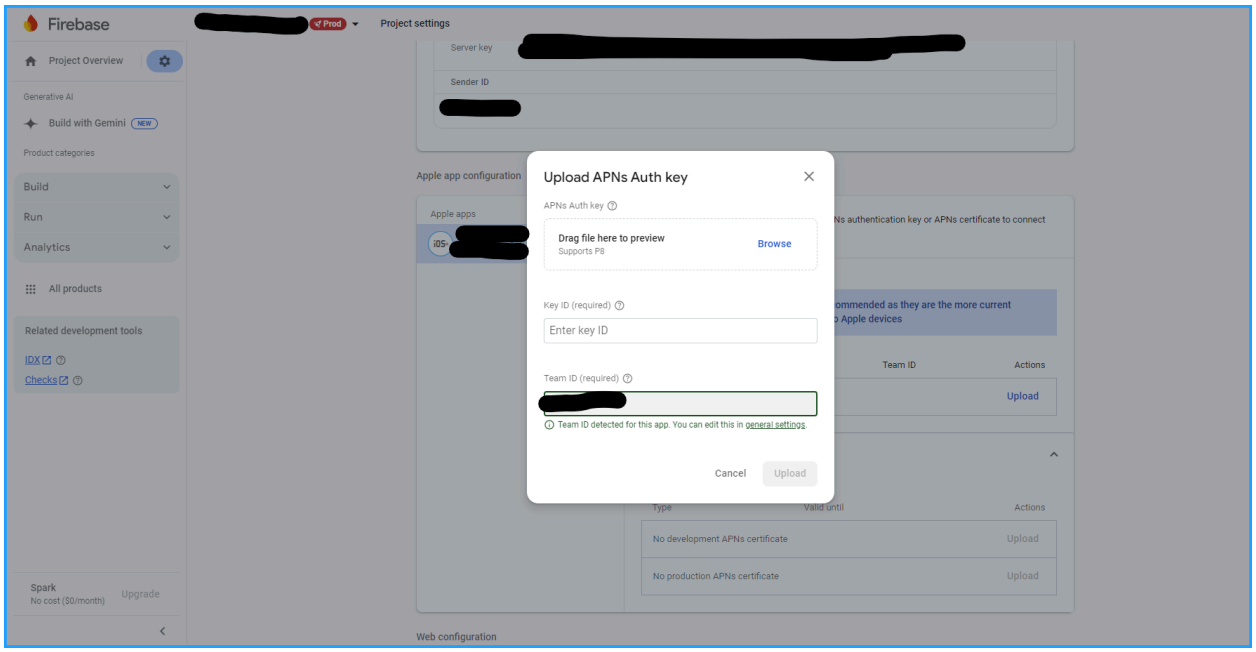
11. Click on the Cloud Messaging tab and in the Apple app configuration section, click Upload





12. Here you will upload the key file by clicking Browse. Select the file that ends with .p8 that was downloaded in the previous steps. The file name will look like this: AuthKey_UN823KU9WJ.p8
13. Now you will have to copy the Key ID and Team ID by going back to your Apple Developer account. The Key ID is located below the key name and the Team ID is located in the top right corner, next to your Apple developer name.

14. Go back to the Firebase page and copy and paste the Key ID and Team ID. Lastly, click on the Upload button.



i. Configure the Firebase setting to the Project

Go to the **lib\widgets\firebase_option.dart** file

- For Android settings replace your credentials in the android method

```
static const FirebaseOptions android = FirebaseOptions(
  apiKey: "YOUR KEY",
  authDomain: "YOUR FIREBASE PROJECT ID.firebaseapp.com",
  projectId: "YOUR FIREBASE PROJECT ID",
  storageBucket: "YOUR FIREBASE PROJECT ID.appspot.com",
  messagingSenderId: "YOUR MESSAGE SENDER ID",
  appId: "YOUR FIREBASE APP ID",
);
```

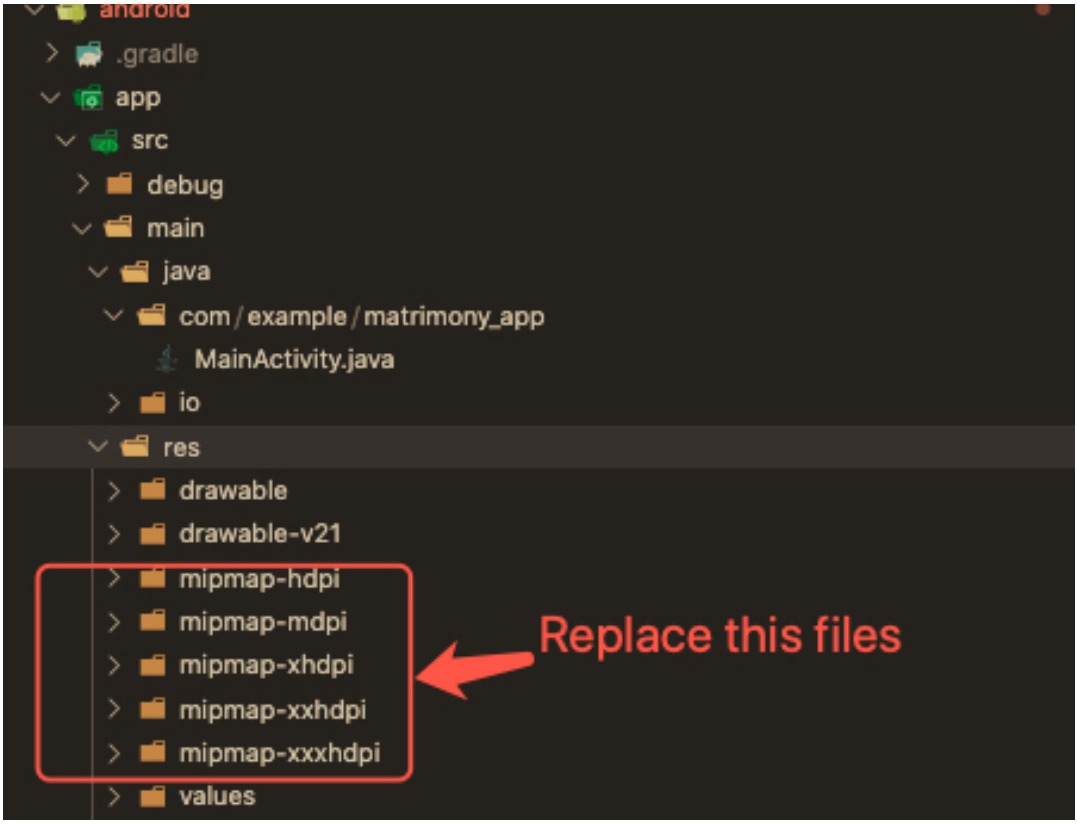
- For iOS settings replace your credentials in the ios method

```
static const FirebaseOptions ios = FirebaseOptions(
  apiKey: "YOUR KEY",
  authDomain: "YOUR FIREBASE PROJECT ID.firebaseapp.com",
  projectId: "YOUR FIREBASE PROJECT ID",
  storageBucket: "YOUR FIREBASE PROJECT ID.appspot.com",
  messagingSenderId: "YOUR MESSAGE SENDER ID",
  appId: "YOUR FIREBASE APP ID",
  androidClientId: 'YOUR ANDROID CLIENT ID',
  iosClientId: 'YOUR Ios CLIENT ID',
  iosBundleId: 'YOUR Ios BUNDLE ID',
);
```

i. Change App Icon

ii. For Android

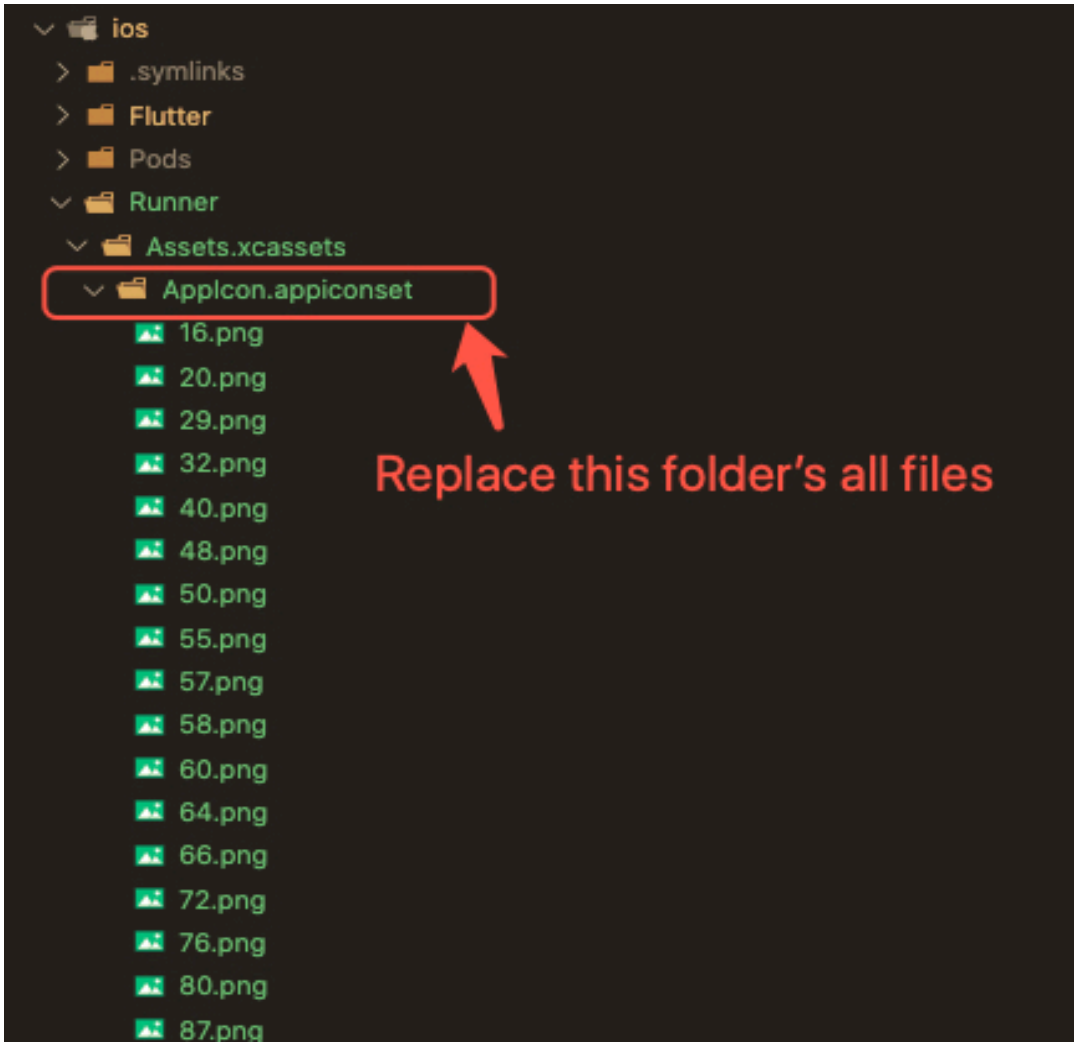
Replace the icons in the **android\app\src\main\res** folder as shown in the below image.



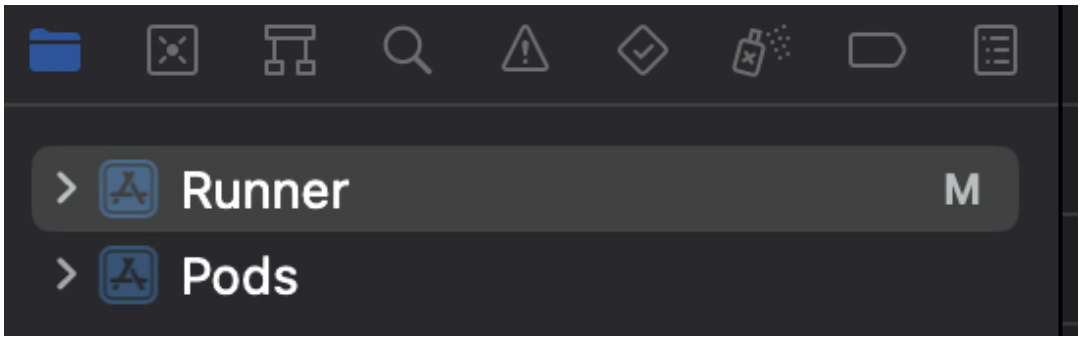
iii. For iOS

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 left side of the XCode window

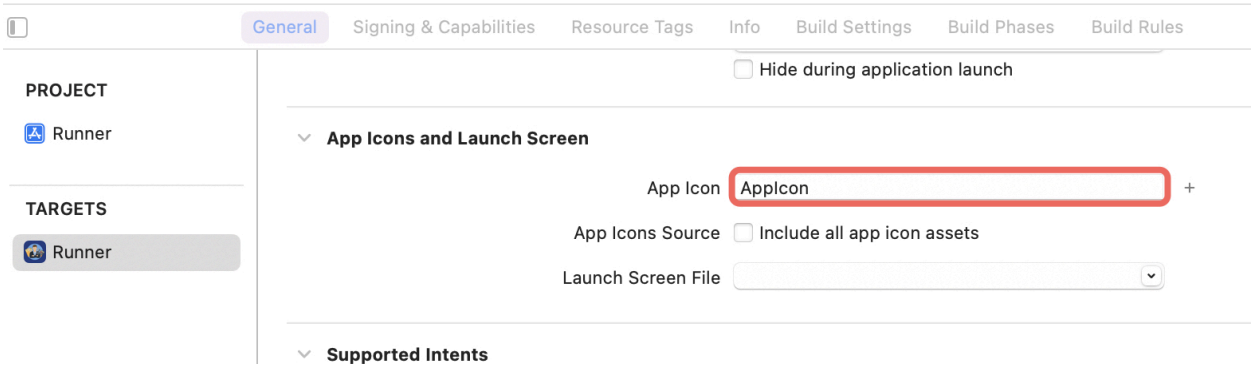


    c. Select Runner.

d. Select Target runner

e. Go to App Icons And Launch Images

f. Click the right arrow button of the app icons 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/

j. Build Release for Android

i. Open Project in VS Code

ii. In Terminal Execute the below commands

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

iii. After making the release, to generate the release bundle Execute the below command

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

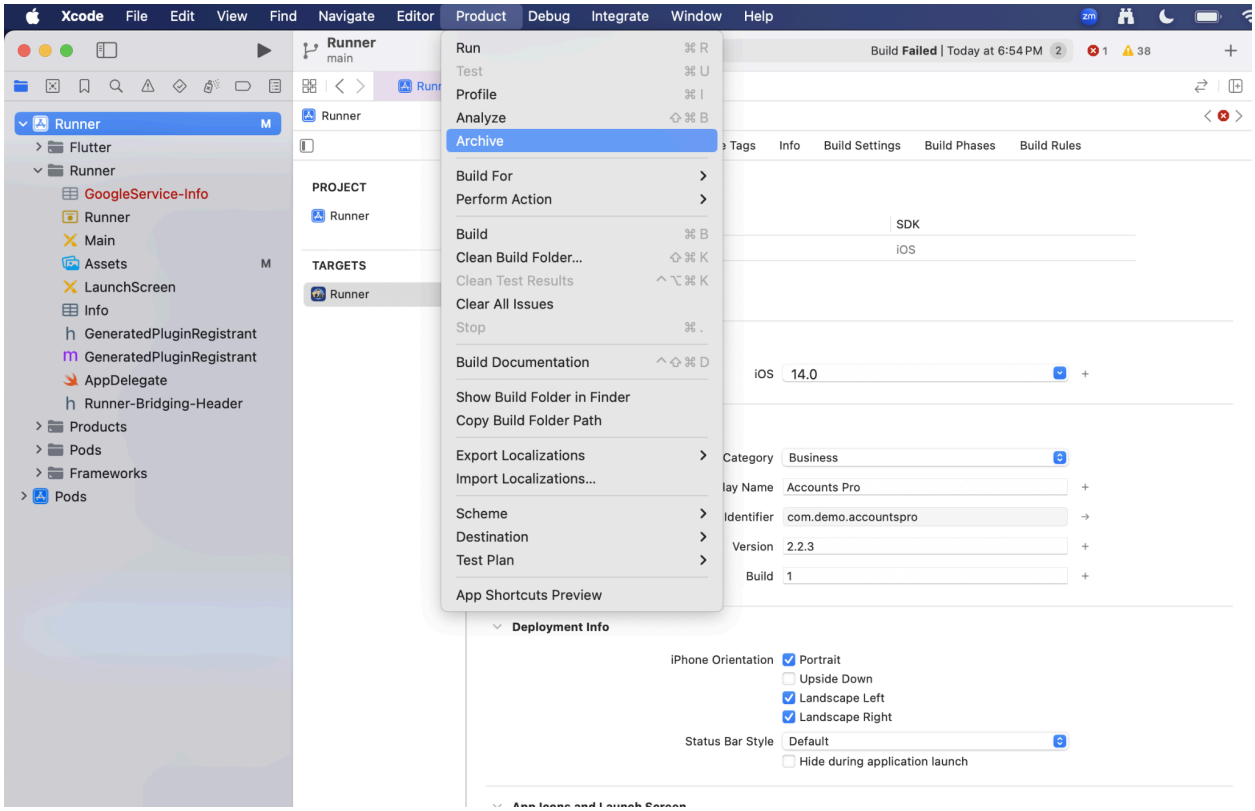
iv. Get the APK from the below path

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

k. Build Release for iOS

i. Open Project in XCode

ii. 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

I. Other Options for the Advanced User

i. Paths to the images used in the app

Images	Path	Screen Path
Business Rule	assets/images/generatedBy.png	lib\models\businessLayer\businessRule.dart
About Us Screen	assets/images/logo.png	lib\screens\aboutScreen.dart
Forgot Password Screen	assets/images/logo.png	lib\screens\forgotPasswordScreen.dart
Local Auth Screen	assets/images/logo.png	lib\screens\localAuthScreen.dart
Login Screen	assets/images/logo.png	lib\screens\newLoginScreen.dart
	assets/images/appleicon.png	
	assets/images/googleicon.png	
	assets/images/facebookicon.png	
OTP Verification screen	assets/images/logo.png	lib\screens\otpVerificationScreen.dart
Splash screen	assets/images/logo.png	lib\screens\splashScreen.dart

Poppins	assets/fonts/Poppins-Bold.ttf
	assets/fonts/Poppins-Regular.ttf
	assets/fonts/Poppins-SemiBold.ttf
	assets/fonts/Poppins-Medium.ttf
	assets/fonts/Poppins-Light.ttf
	assets/fonts/Poppins-Thine.ttf

ii. 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	0xFF223d82
Scaffold Background Color	0XFFFAFAFA
Icon Color	0XFF2B2F38
Elevated Button - background color	0xFF223d82
App Bar - foreground color	0XFFFAFAFA
App Bar - color	0XFFFAFAFA
App Bar - action icon color	0XFF2B2F38
App Bar - icon color	0XFF2B2F38
Card - color	0XFFFFFFFF
Checkbox - check color	0XFF2B2F38
Checkbox - fill color	transparent
Radio - fill color	0XFF2B2F38

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

Package Name - Version	Description
package_info_plus: ^8.0.0	For getting information about the application package
shared_preferences: ^2.2.2	For storing important data like user session
url_launcher: ^6.2.2	For launching types of url
material_design_icons_flutter: 7.0.7296	For icons
http: ^1.1.2	For making api requests
email_validator: ^3.0.0	For validate email address
font_awesome_flutter: ^10.6.0	For icons
image_cropper: ^8.0.2	For crop image feature
ribbon_widget: ^1.0.5	For showing ribbon on widget
printing: ^5.13.2	For print/share documents
local_auth: ^2.3.0	For access to facial and biometric data to provide a lock app feature

badges: 3.1.2	For show counts on filter icons
encrypt: 5.0.3	For generate hash string
device_info_plus: ^10.1.0	For getting information about device like os, model, version
fl_chart: ^0.68.0	For showing beautiful charts in the app
flutter_randomcolor: ^1.0.14	For getting random color
fast_contacts: ^4.0.0	For accessing phone directory of the device
multi_image_picker_plus: ^0.0.4	For select multiple images
permission_handler: ^11.1.0	For asking device/system permission like access memory, contacts
google_sign_in: ^6.2.1	For implement sign in using google
flutter_facebook_auth: ^7.1.0	For implement sign in using facebook
sign_in_with_apple: ^6.1.2	For implement sign in using apple
share_plus: ^10.0.0	For sharing info on other platform like social media, email
connectivity_plus: ^6.0.3	For checking internet connectivity status
flutter_inappwebview: ^6.0.0	For showing web views in the app
in_app_update: ^4.2.2	For checking and updating android app
flutter_image_compress: ^2.1.0	For reduce image size
firebase_analytics: ^11.2.0	For tracking issues and bugs on Firebase Crashlytics
get_ip_address: ^0.0.7	For getting device ip address
cached_network_image: ^3.3.1	For showing images from network
pinput: 5.0.0	For showing OTP text box
flutter_cached_pdfview: ^0.4.2	For showing generated pdf preview
get: ^4.6.6	For state management

USEFUL LINKS

- To set up NodeJS with Typescript 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 10 October 2024.