//read_admin_doc_first

NewsHour App Doc (v4.0.2)

22-07-21 Last updated

—

It's an offline document. There is an online documentation here. We are highly recommending the online version. Because it will always be updated.”

1. Introduction

In this app setup, make sure you are on the latest version flutter. If you are not, please update the flutter version by running **flutter upgrade** command. If everything is okay, you can follow the further steps. You can use Android Studio or Visual Studio Code.

Note: *A Mac (Apple Desktop/laptop) device and an apple developer account is required for the iOS setup. If you don’t have both of them, you can ignore the iOS steps.*

2. Project Setup

1. Open the **source/news_app** folder on your IDE (VSCode or Android Studio) and wait some moment to load the project.

2. Run this following command on the IDE terminal to clean the whole project first.

```
flutter clean
```

3. Run this following command on the IDE terminal to get all the required packages.

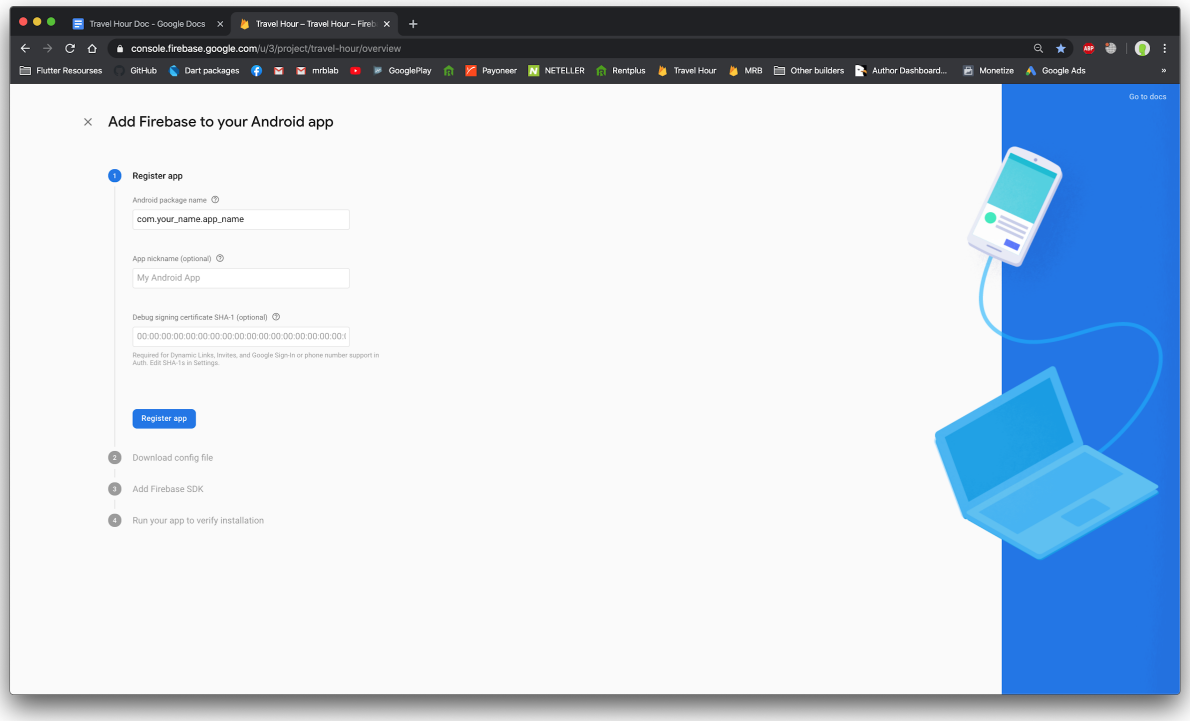
```
flutter pub get
```

Now, your project is ready for the configuration.

3. Firebase Setup for Android

3.1 Package Name Setup on Firebase

1. Go to the **firebase console > Project overview page**. Click on add app and then android icon. Enter your android package name. Your package name should be like **com.your_name.your_app_name** . Like **com.microsoft.skype**. You can use the same package name for android & iOS. iOS doesn't support **underspace** in the package name. So, keep in mind that if you want to use the same package name for both android & iOS.



2. Click on the register app and skip other steps by clicking next.

3.2 Change Package Name for Android

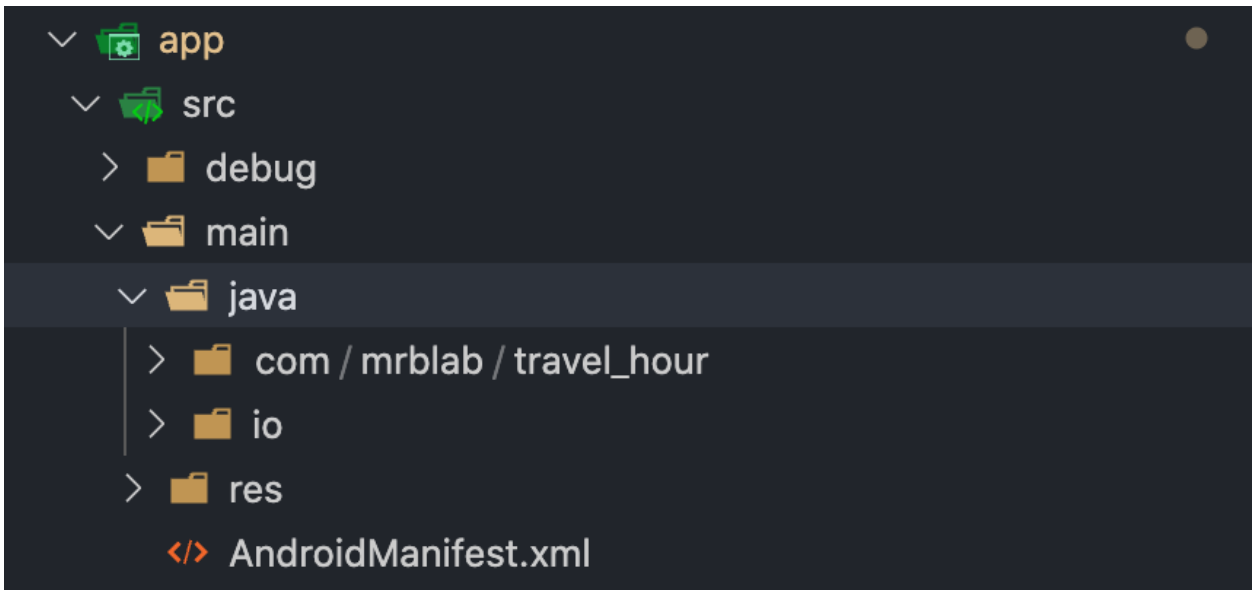
1. Go to your IDE and now you **have to change the package name** of your app. Go to

    - **android>app>build.gradle,**
    - **android/app/src/debug/AndroidManifest.xml,**
    - **android/app/src/main/AndroidManifest.xml,**
    - **android/app/src/main/kotlin/mrblab/news_app/MainActivity.kt,**
    - **android/app/src/profile/AndroidManifest.xml**

    these files and then find & replace **mrblab.news_app** by **your_package_name**. You can use the search option of VSCode for that. (We didn't use **com** in our package name)

2. Now you have to rename two folders. Go to **android/app/src/main/kotlin** and rename **mrblab** by **your_name** and inside this folder rename **news_app** by your **app_name.** Remember this should be according to your android package name.

   So, if you used **com** in the package name, go to this folder **android/app/src/main/kotlin** via your file explorer and make a folder inside it and name it **com** and move the **your_name** folder inside the **com** folder. After that your **kotlin** folder should look like this.



That's it. Your android package name setup is complete.

Now, you need to generate **2 signing certificates** for google sign in feature.

3.3 Generate Debug Certificate

To generate a debug certificate, run this command on your terminal from your app root directory.

For Mac Users, run

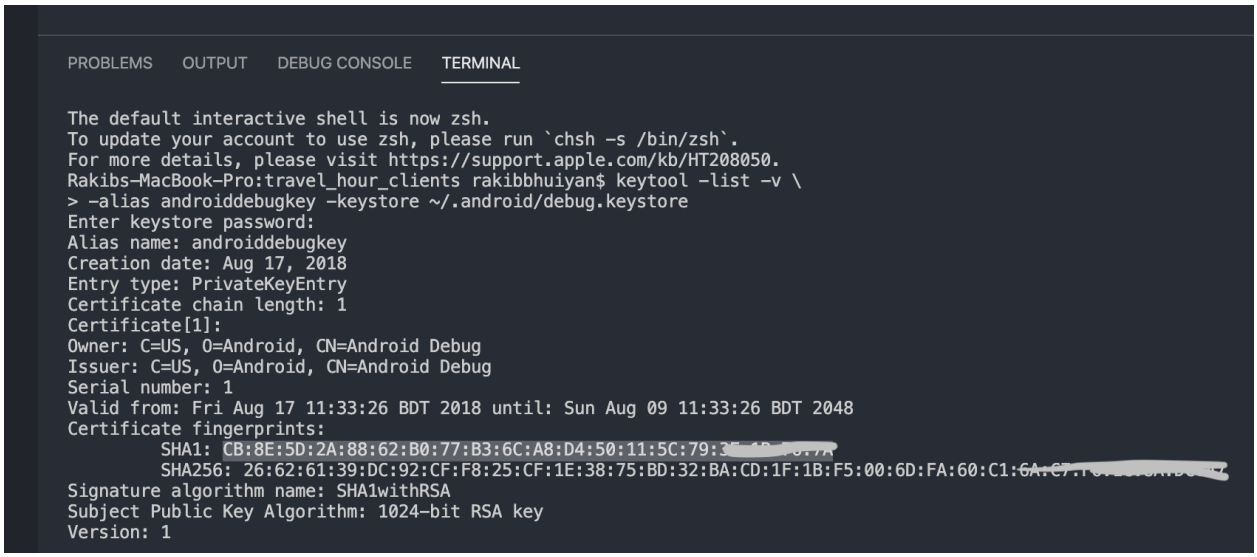
```
keytool -list -v \
-alias androiddebugkey -keystore ~/.android/debug.keystore
```

For Windows users, run

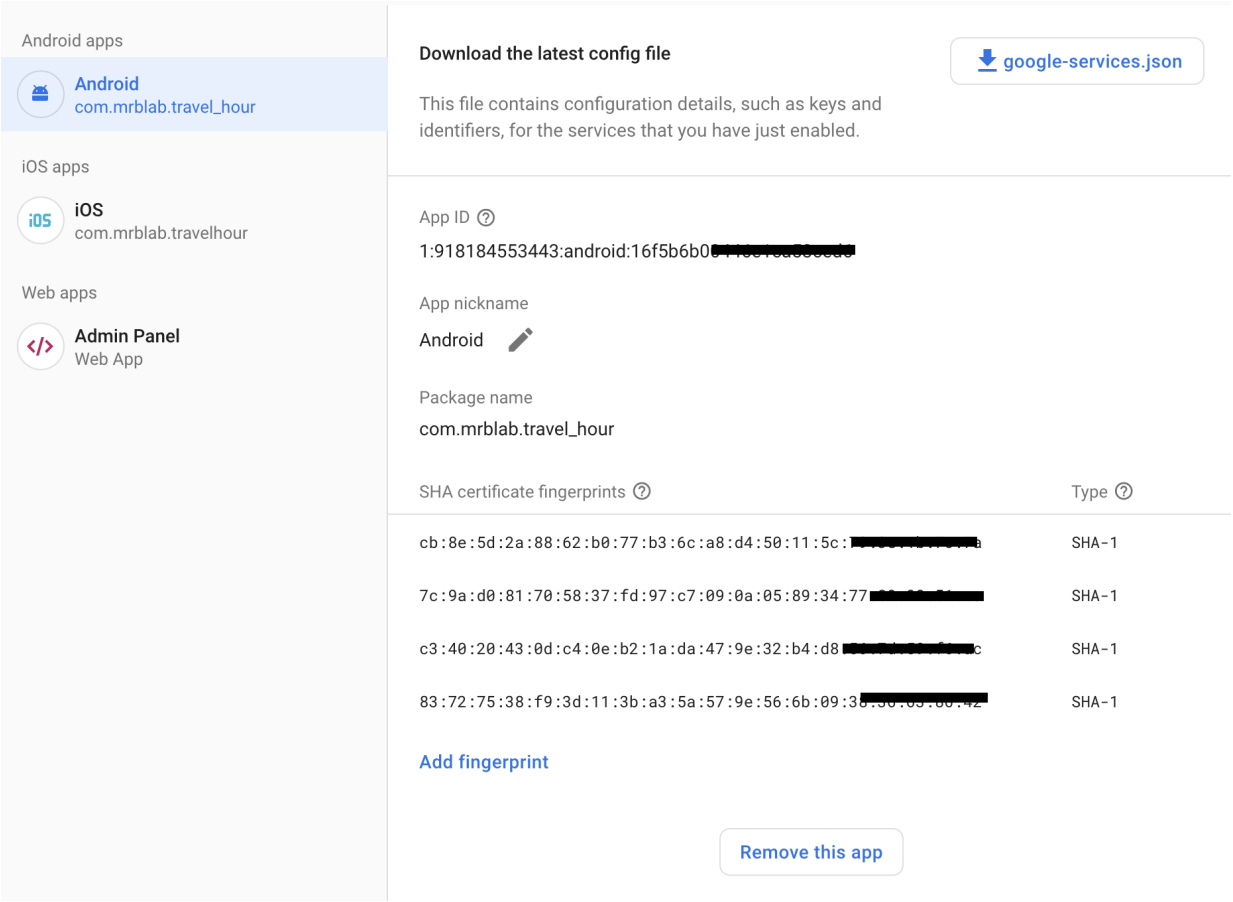
```
keytool -list -v \
-alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore
```

Use **android** as a debug password when the terminal asks for a password.

N.B. *If this command doesn't work, then go to this [link] and copy the debug command from there according to your os.*



Copy the SHA1 certificate code and go to **Firebase Console > Your Project > Project Settings** and click on the **android icon** and then add the **SHA1** code by clicking **add fingerprint** button. Look at the picture below:



3.4 Generate Release Certificate

1. To generate a release certificate, You have to generate a keystore file. To generate a keystore file, run this command below from the root of your project directory.

   For Mac users, run

   ```
   keytool -genkey -v -keystore ~/key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias key
   ```

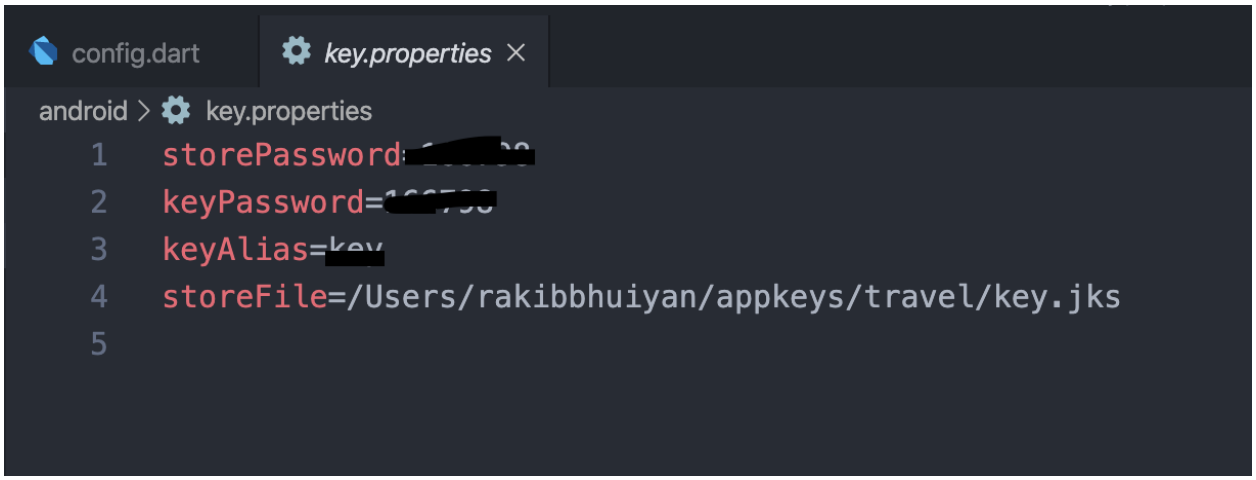
   For Windows users, run

   ```
   keytool -genkey -v -keystore c:/Users/USER_NAME/key.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias key
   ```

2. Enter your details and remember **alias key** name and **password**. After this, you will get a **.jks** keystore file.

3. Locate this file and move the file into the **android/app** folder and copy the path by right clicking on the **key.jks** file.

4. Then go to **android/key.properties** file and replace the path of the keystore file of yours. Then also replace the **password** and **key alias name** which you have inputted to generate the keystore file.



Now you can generate a release certificate, To do that,

1. Run with replacing your **alias_name** and **keystore_location**.

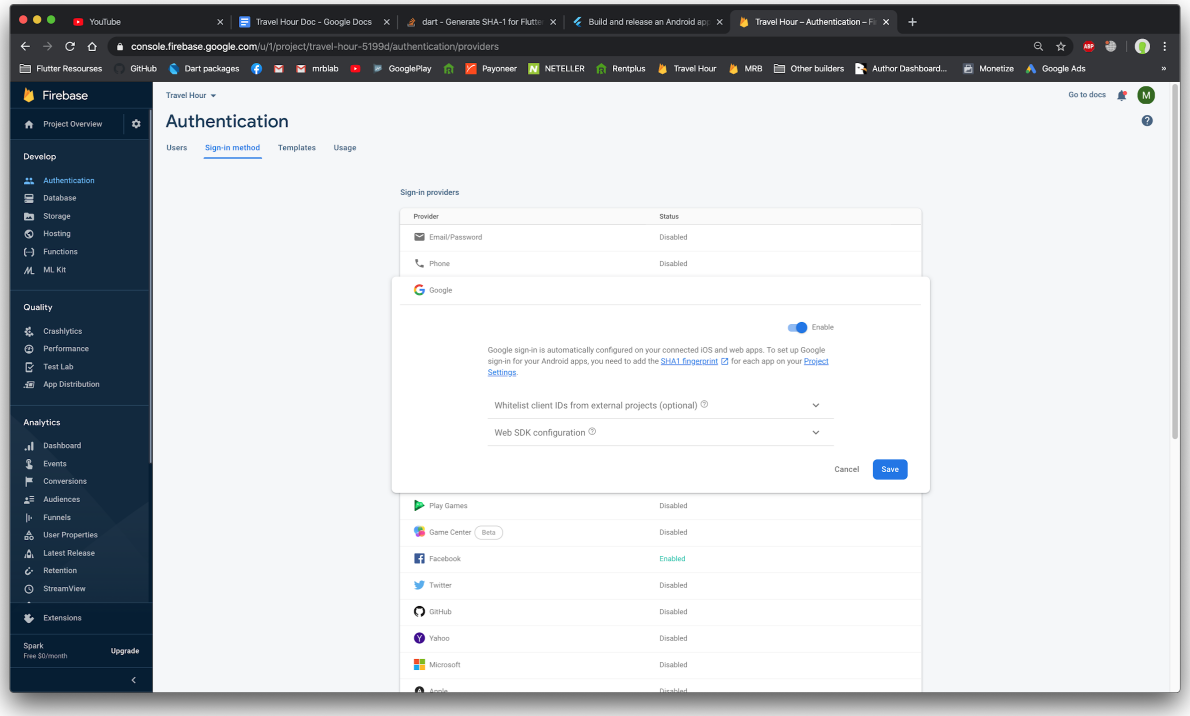
```
keytool -list -v -keystore keystore_location -alias alias_name
```

2. After that you will get a **SHA1** code. Copy that code and add to your firebase console project settings where you previously added a debug **SHA1** code. That's it.

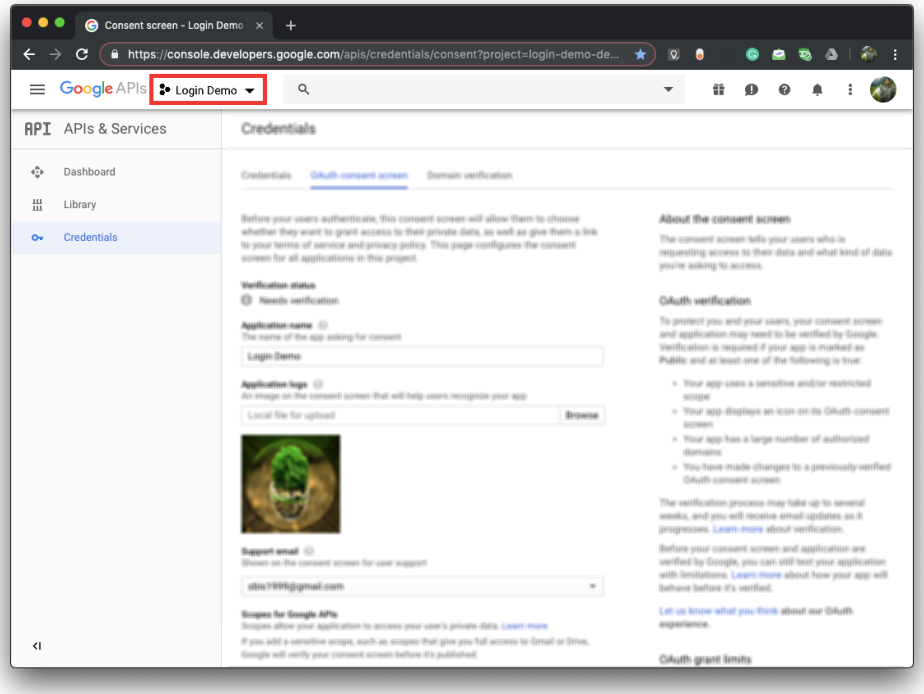
3.5 Google Sign In Setup

Now you have to set up google sign in & email sign in. To do that,

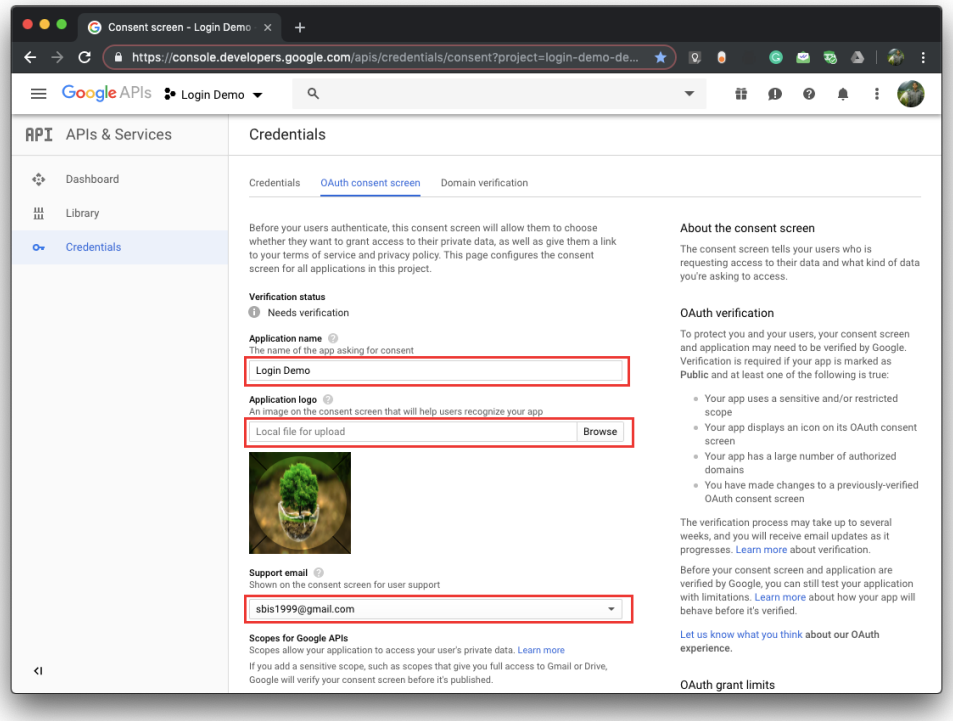
1. Go to **firebase console>your project>authentication>Sign-in-method** and enable both **email/password** and **google** and save it.



2. You have to configure some stuff for google sign in. Go to this url.
3. Make sure you are signed in with the same account with which you have created the Firebase project.
4. Also, make sure that on the top-left corner, your project is selected for which you are filling this consent.

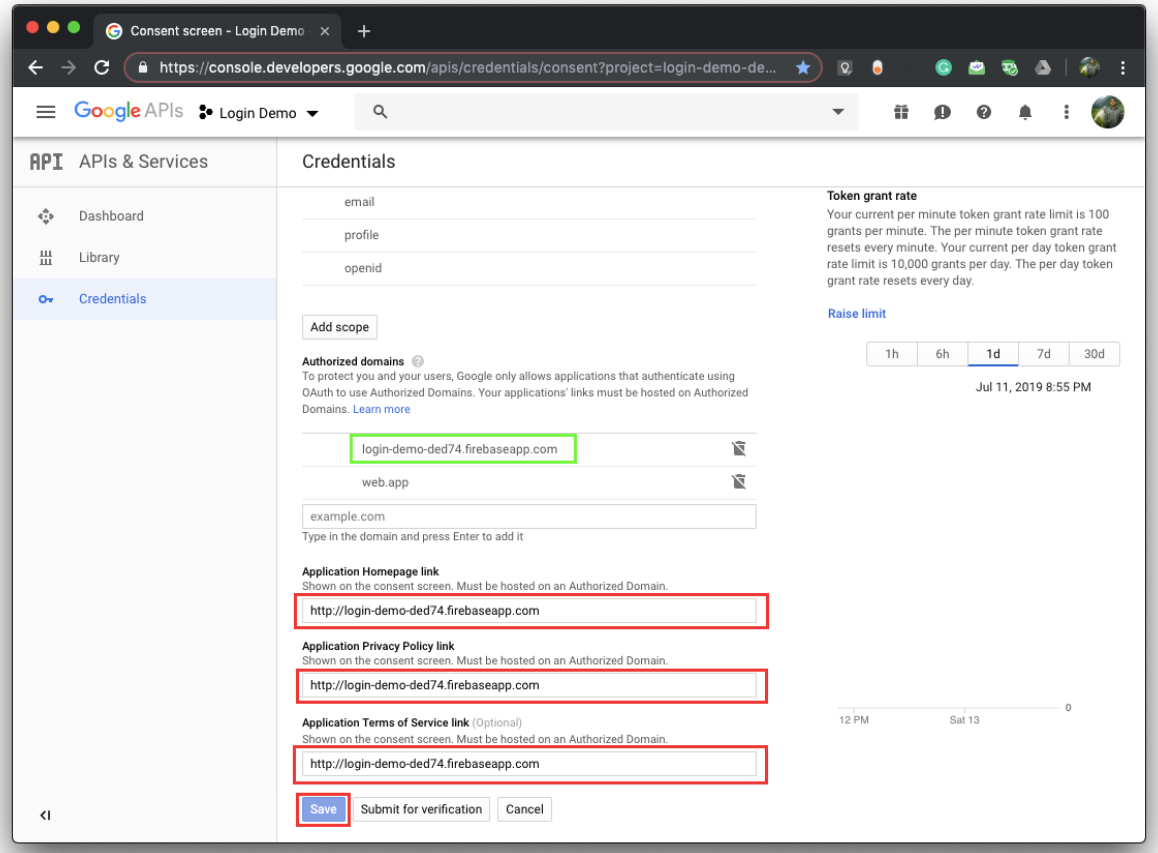


5. Go to Credentials → OAuth consent screen tab and start filling the form.
6. Enter “Application name”, “Application logo” & “Support email”.



7. Then, scroll down and fill in the **Application Homepage link**, **Application Privacy Policy link** and **Application Terms of Services link**.

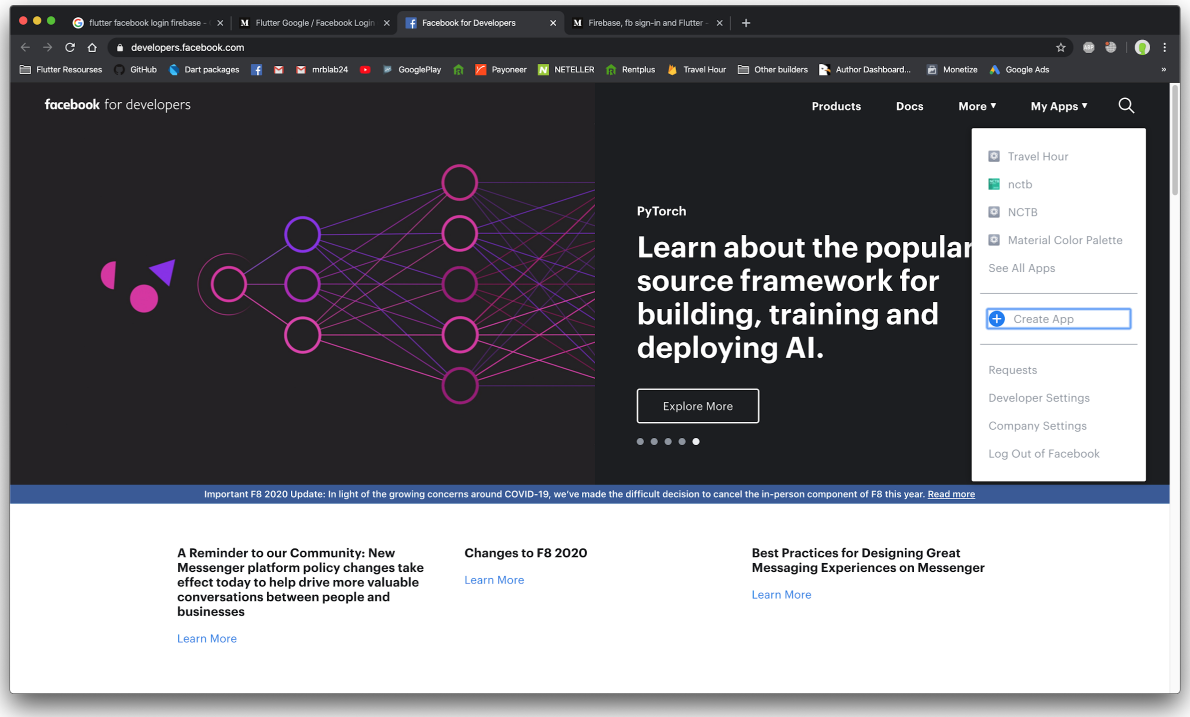
8. In all these places, you have to enter the same link starting with **http://** then your app domain name which I have marked with green below.



9. Click on Save. That's it. You have completed your google signin setup.

3.6 Facebook Login Setup

1. Now you have set up facebook sign in. To do that, Go to this url.
2. Go My apps > Create App.
3. Enter App name and email and go to Dashboard tab.



4. Scroll down on the right pane until you reach '**Add a product**', select **Facebook Login**
5. You will be redirected to the quick start page.
6. Select **Android**. Skip 1 & 2.
7. Enter **your_package_name** in the package option and enter **your_package_name.MainActivity** in the activity option.
8. For the next step, you need to generate two **hash ids**. To do that, run the following commands on the terminal.

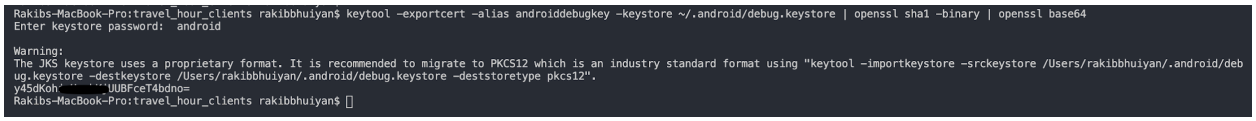
For Mac Users,

```
keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore | openssl sha1 -binary | openssl base64
```

For Window users,

```
keytool -exportcert -alias androiddebugkey -keystore "C:\Users\USERNAME\.android\debug.keystore" | "PATH_TO_OPENSSL_LIBRARY\bin\openssl" sha1 -binary | "PATH_TO_OPENSSL_LIBRARY\bin\openssl" base64
```

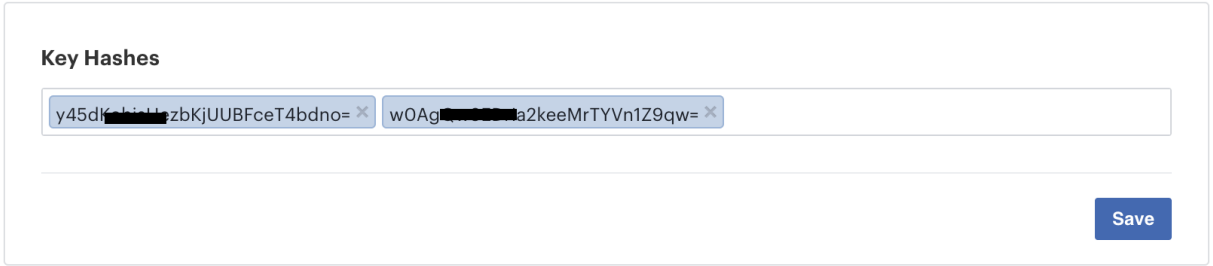
9. Use **android** as a debug password. After that you will get a **hash id** like this.



10. For release hash id, run this following command by replacing your **alias key name** and **keystore location**. You can get these from your **android/key.properties** file.

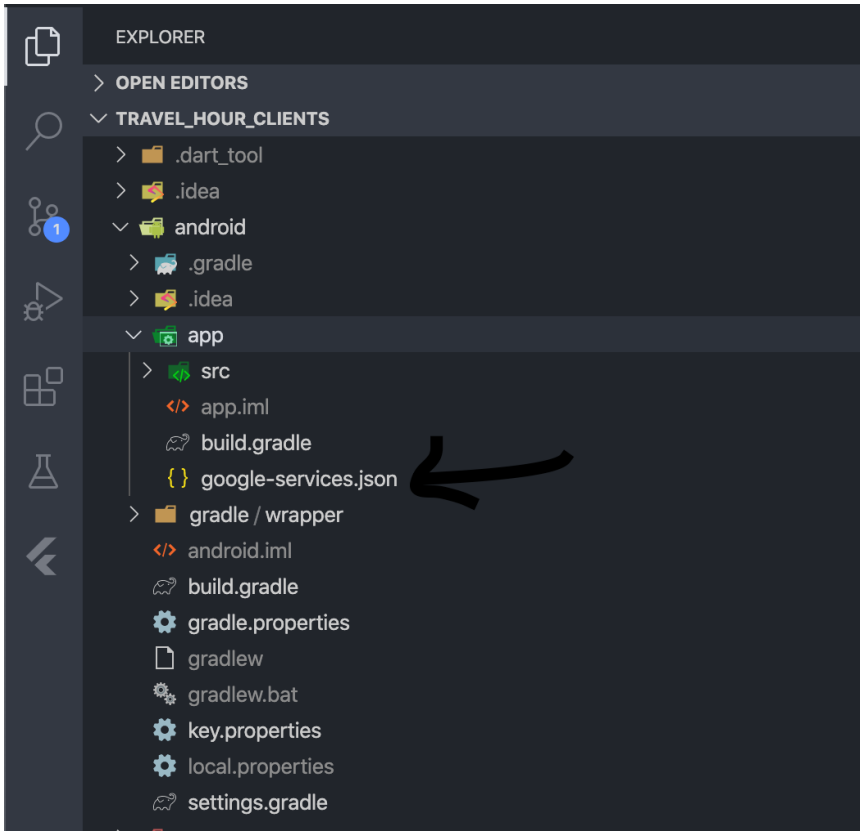
```
keytool -exportcert -alias YOUR_RELEASE_KEY_ALIAS -keystore YOUR_RELEASE_KEY_PATH | openssl sha1 -binary | openssl base64
```

11. After that you will get another **hash id**. Now, copy and paste them in the next steps of the facebook developer site. Like this,



12. Skip all the steps by clicking next.

13. Now go to **settings** tab & copy both **app id** and app **secret key**.
14. Now go to firebase console > your project > authentication > Sign-in-method and click on **facebook**, enable it and paste both app id and app secret key and save it.
15. Now go to project settings and click on the **android icon** and download **google-service.json** file.
16. Now go to **android/app** directory and paste the **google-service.json** file here.



17. Now go to **android/app/src/main/res/values/strings.xml** this directory and change **the app name**, **app_id** and **fb+app_id**.

config.dart strings.xml

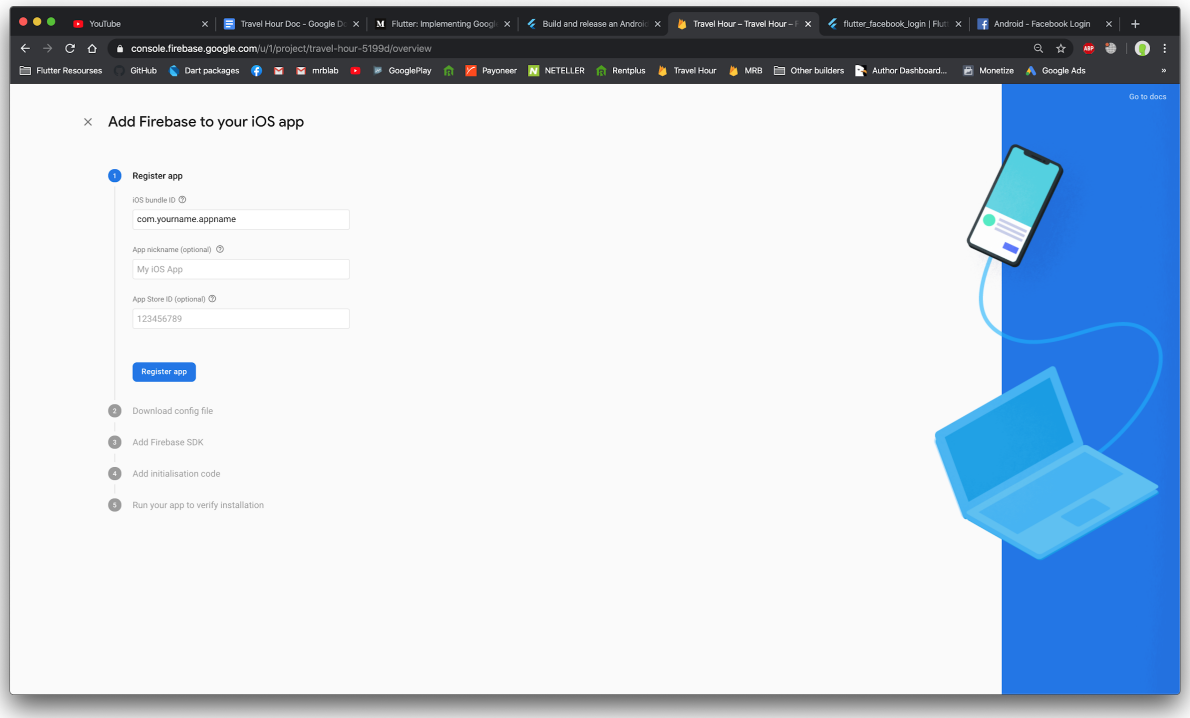
android > app > src > main > res > values > strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Travel Hour</string>  //app name that you created on developer.facebook.com
    <string name="facebook_app_id">000000000</string>  //app id
    <string name="fb_login_protocol_scheme">fb000000000</string>  //fb+app id
</resources>
```

That's it. Android setup for Firebase database, Google Sign in & Facebook login setup is complete.

4. Firebase Setup for iOS

1. Go to the firebase console > Project overview page. Click on **add app** and then **android icon**. Enter **your package name**. You can use the same package name that you have used for android.



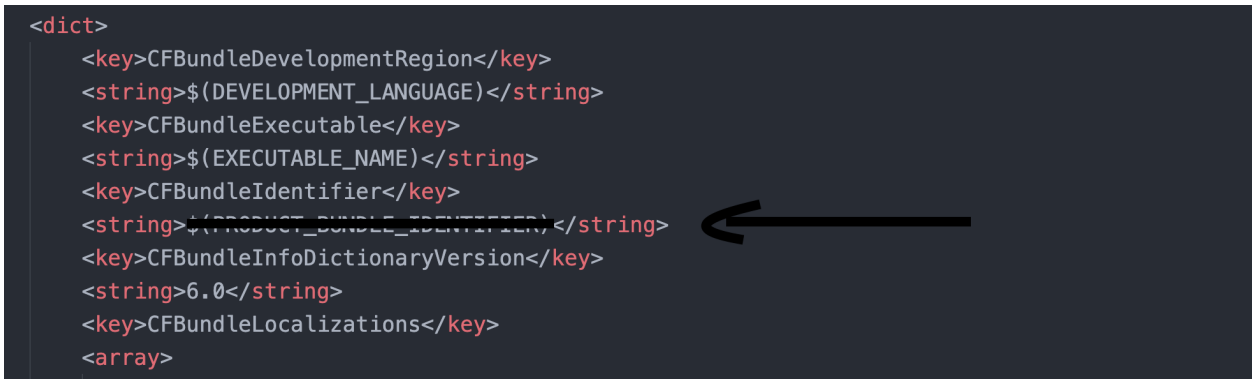
2. Click on the **register app** and skip others by clicking next.
3. Now go to project settings and click on ios and download the **GoogleService-info.plist** file.
4. Then go to **ios/Runner** directory and paste the file here.
5. Now, Open the **iOS folder** on Xcode by right clicking on the **iOS** folder from VSCode or Android Studio and go to the **runner** folder and move the **GoogleService-info.plist** file here. You will get a popup and click yes or confirm the popup message.
6. Now, open the **GoogleService-info.plist** file from your IDE or from Xcode and copy the **REVERSED_CLIENT_ID**. (See the picture below)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>CLIENT_ID</key>
    <string>918184553443-fl09a8l241nk6eslp4c5lcoaalkjj8d3.apps.googleusercontent.com</string>
    <key>REVERSED_CLIENT_ID</key>
    <string>com.googleusercontent.apps.918184553443-fl09a8l241nk6eslp[redacted]</string>
    <key>ANDROID_CLIENT_ID</key>
    <string>918184553443-1tal16hutlomnip48g42k3am93re1j7a.apps.googleusercontent.com</string>
    <key>API_KEY</key>
    <string>AIzaSyBdctrSAqjNZFTjh_WIP83Mtcksq0qb5Zg</string>
    <key>GCM_SENDER_ID</key>
    <string>918184553443</string>
    <key>PLIST_VERSION</key>
    <string>1</string>
    <key>BUNDLE_ID</key>
    <string>com.mrblab.travelhour</string>
    <key>PROJECT_ID</key>
    <string>travel-hour-5199d</string>
    <key>STORAGE_BUCKET</key>
    <string>travel-hour-5199d.appspot.com</string>
    <key>IS_ADS_ENABLED</key>
    <false></false>
```

7. Go to **ios/Runner/Info.plist** file and replace the **REVERSED_CLIENT_ID** here.

```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleTypeRole</key>
        <string>Editor</string>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>fb544514846502023</string>
            <string>com.googleusercontent.apps.989933527697-mjaatabg904p6lihd[redacted]</string>
        </array>
    </dict>
</array>
```

8. Now you have to change the iOS package name. To do that, again go to **ios/Runner/Info.plist** file and replace **CFBundleIdentifier** value with your iOS package name. (See the picture below)



That's it., Your Firebase & Google Sign In for iOS setup is complete.

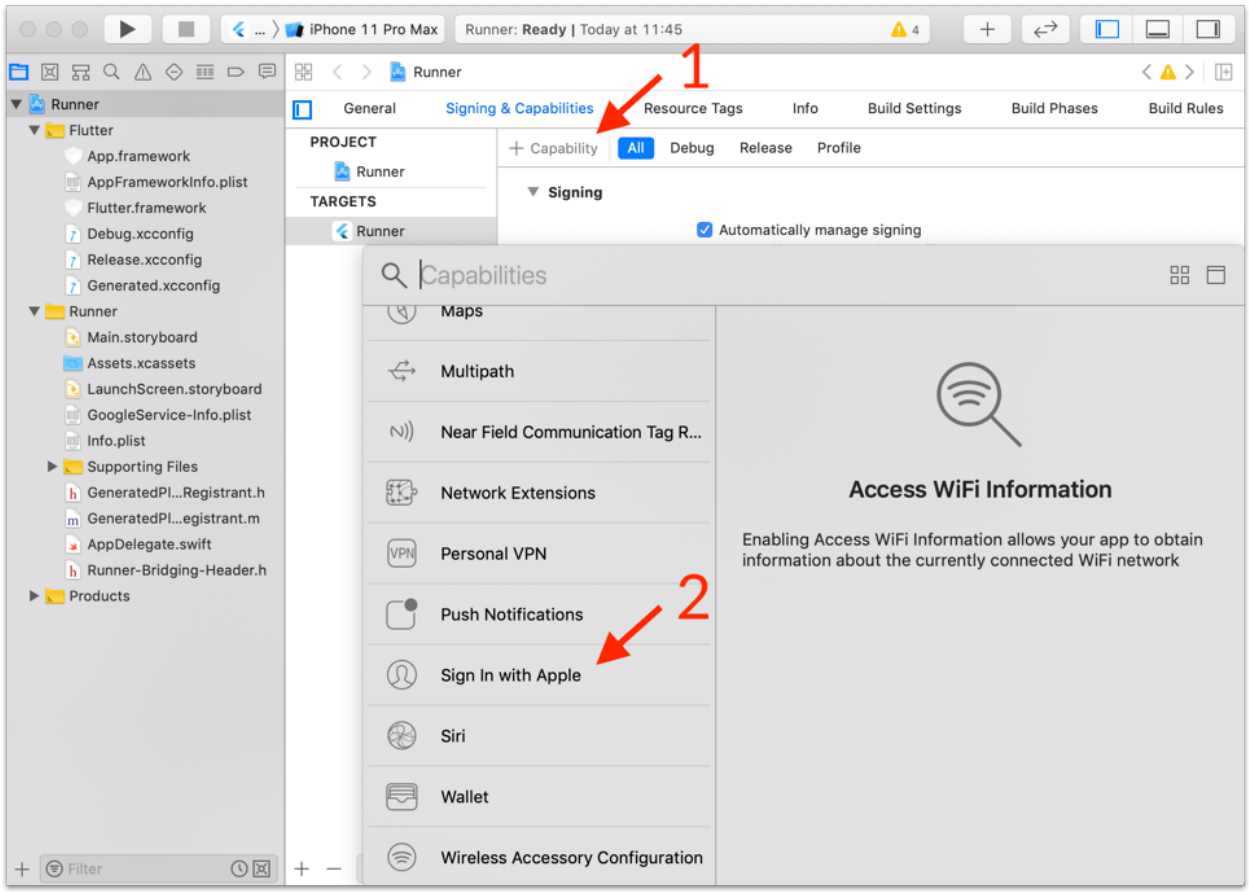
4.1 Facebook Login for iOS Setup

1. Now go to developers.facebook.com again and navigate to your project and click facebook login > quick setup > ios icon.
2. Skip 1 & go to step 2.
3. Enter **your package name** in the **bundle ID** option & skip others by clicking next.
4. That's it. iOS setup for Facebook Sign In is complete.

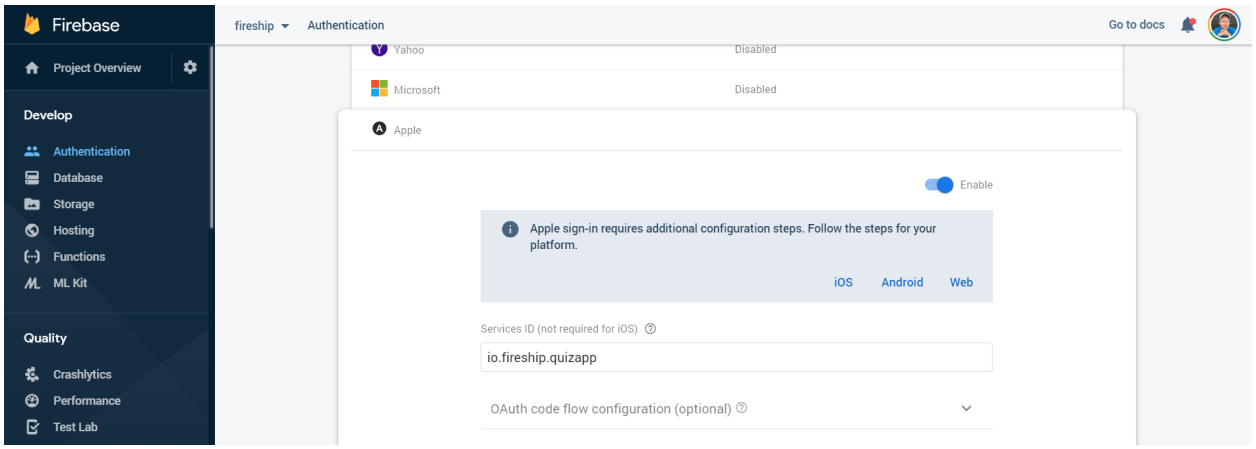
4.2 Apple Login Setup (Only for iOS)

To do that, you need an paid apple developer account and xcode app on your mac.

1. From your IDE, Right click on the **ios folder** and click on **open on xcode** and then go to **runner > sign in & capability** tab.



2. Add **Sign in with Apple**. We already did this in the project. If this is not available in the project, do this by yourself.
3. No go to your **firebase console > your project > authentication page** and enable apple sign in option. You don't have to put anything in the text fields.



That's it. One more thing, when you configure Firebase push notification for iOS in the next step, make sure you have also tik on the **Sign In with Apple** option in the identifier on apple developer page.

That's it. Your database setup is complete.

5. Firebase Push Notifications Setup

1. For Android, You don't have to do anything. We already integrated the procedures in the project.

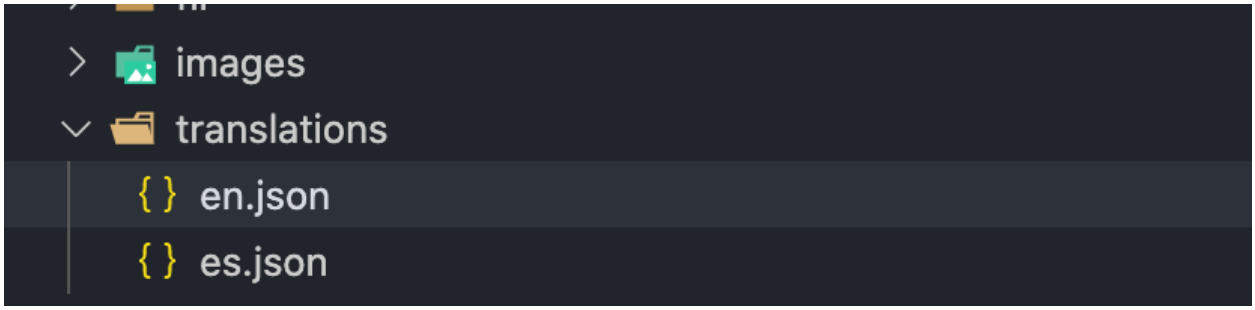
2. For iOS, Go to this link and follow the instructions. This is a well written doc from Flutter Team.

6. Multi-language setup

You can skip this setup now. This is not a mandatory setup to run this app.

So, we have used 3 languages in this app. English, Spanish & Arabic. You can do as much as you can. We are assuming that you want to add your own country language. You need to know about your two letter language code. Like, English language code is **en** and Spanish language code is **es** . You can search for your language code on google.

1. First go to the **assets/translations** folder from your IDE. Add a .json file here with **your_language_code.json** name. Now go to **assets/translations/en.json** file and copy everything from this file and paste to **your_language_code.json** file.



2. Now, Rename the all right side strings. Do not edit left side strings. These are the keys. Look at the **es.json** file and you will understand what to do.

3. Now go to **lib/main.dart** file and add your language code to **supportedLocals**.

```
  runApp(
    EasyLocalization(
      supportedLocales: [Locale('en'), Locale('es'), Locale('ar')],
      path: 'assets/translations',
      fallbackLocale: Locale('en'),
      startLocale: Locale('en'),
      useOnlyLangCode: true,
      child: MyApp(),
    ) // EasyLocalization
  );

}
```

Your code will be look like this :

```
supportedLocales: [Locale('en'), Locale('es'), Locale('ar'), Locale('your_language_code')],
```

4. You can edit the **startLocale** by replacing **en** by **your_language_code** if you want to add your default language to your language.

5. Now go to **lib/config/config.dart** and add your **Language name** at the bottom of the list by adding a comma**.**

```
//Language Setup

final List<String> languages = [
  'English',
  'Spanish',
  'Arabic'
];
```

6. Now go to **lib/widgets/language.dart** file and enable the disable lines by removing slashes and rename **your_language_code** and **language_name** that you added in the list..

```
Widget _itemList (d, index){
  return Column(
    children: [
      ListTile(
        leading: Icon(Icons.language),
        title: Text(d),
        onTap: () async{
          if(d == 'English'){
            context.locale = Locale('en');
          }
          else if(d == 'Spanish'){
            context.locale = Locale('es');
          }
          else if (d == 'Arabic'){
            context.locale = Locale('ar');
          }
          // else if(d == 'your_language_name'){
          //   context.locale = Locale('your_language_code');
          // }
          Navigator.pop(context);
        },
```

7. For Android, you don't have to do anything.
8. For iOS, go to **ios/Runner/Info.plist** and add your language code in a string.

```
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleLocalizations</key>
    <array>
        <string>ar</string>
        <string>en</string>
        <string>es</string>
    </array>
```

9. Add **<string>your_language_code</string>** inside the array.
10. That's it. Your multi-language setup is complete. You can add as many languages as you want by following these steps.

11. To remove any language, first go to the **asset/translations** folder and delete the language_code.json file that you want. (You shouldn't delete the **en.json** file because this is the default language). Now go to **lib/main.dart and** remove the locale from the **supportedLocale** line and then go to **lib/widgets/language.dart** file and remove the else if section of language code that you want to delete and finally go to **ios/Runner/Info.plist** and remove the string. That's it.

7. Ads Setup

You can skip this setup for now and can configure later. We have enabled admob ads by default with test ad unit ids. You can enable/disable ads from the admin panel.

So, We have used both admob and facebook ads in this app. But you can use only one at a time. Either admob or facebook ads. We recommend you to use admob ads because they provide higher revenue than any other ads networks. Use facebook ads if only your admob account is suspended. By the way, that was just a suggestion from us. The choice is yours.

We have used both **Interstitial** ads and **banner** ads. ***Interstitial ads will show when the video loaded successfully and banner ads will show at the bottom of the image article screen.***

1. You can control ads from the admin panel. We have added an option to turn off/on ads at any time you want. Don't enable ads on the admin panel if you are not enabled ads in the app.

2. We have applied ads into 2 screens.

   Article_details.dart - banner ad ,

   video_article_details.dart - interstitial ad.

7.1 Admob Setup

We have used admob ads by default with test unit ids. To test the ads, you should test with the test unit ids. Before releasing the app for production, make sure you have changed the app ids and ad unit ids with yours. Admob is applied by default. So you don't have to do anything for admob.

Follow these steps for production (release):

1. For **Android**, Go to **android/app/src/main/AndroidManifest.xml** file and replace with your **admob app id** of yours which you will get from your admob account. You can still use this app id for testing purposes. Make sure you have replaced your app id before releasing the app in the play/app store.

```
78      <!---admob section-->
79
80      <meta-data
81          android:name="com.google.android.gms.ads.APPLICATION_ID"
82          android:value="ca-app-pub-3940256099942544~3347511713"/>
83
```

2. For **iOS**, go to **ios/Runner/Info.plist** file and replace your admob app id of ios.

```
<key>GADApplicationIdentifier</key>
<string>admob app id</string>
```

3. Go to **lib/config/ad_config.dart** file and replace **admob app ids & ad unit ids** with yours**.** Ignore the iOS ids if you don't want to make the app for iOS.

```
//-- Admob Ads --
static const String admobAppIdAndroid = 'ca-app-pub-3940256099942544~3347511713';
static const String admobAppIdiOS = 'ca-app-pub-3940256099942544~1458002511';

static const String admobInterstitialAdUnitIdAndroid = 'ca-app-pub-3940256099942544/1033173712';
static const String admobInterstitialAdUnitIdiOS = 'ca-app-pub-3940256099942544/4411468910';

static const String admobBannerAdUnitIdAndroid = 'ca-app-pub-3940256099942544/6300978111';
static const String admobBannerAdUnitIdiOS = 'ca-app-pub-3940256099942544/2934735716';
```

That's it. Your admob setup is complete.

7.2 Facebook Ads Setup

Ignore this if you have added admob.

1. First, you have to place your ad ids. Go to the **lib/config/ad_config.dart** file and replace the fb ad unit ids with yours.

```
//-- Fb Ads --
static const String fbInterstitialAdUnitIdAndroid = '544514846502**************';
static const String fbInterstitialAdUnitIdiOS = '544514846502023_702**************';

static const String fbBannerAdUnitIdAndroid = '544514846502023_70**************';
static const String fbBannerAdUnitIdiOS = '544514846502023_7**************';
```

2. Now you have to disable admob procedures and enable fb procedures. Go to **lib blocs/ad_bloc.dart** file and disable where admob is quoted and enable where fb is quoted. You can disable any procedures or line by adding two slash (**//**) at the left and enable by removing two slash (**//**) from the left. Look at the picture below.

```
//enable only one
Future initiateAds ()async{
  await MobileAds.instance.initialize();  //admob
  //await FacebookAudienceNetwork.init();  //fb
}


//enbale only one
void loadAds (){
  createInterstitialAdAdmob();  //admob
  //createInterstitialAdFb(); //fb
}


//enbale only one
@override
void dispose() {
  interstitialAdAdmob?.dispose();  //admob
  //disposefbInterstitial();       //fb
  super.dispose();
}

```

3. You have enabled fb interstitial ads and now you have to add fb banner ads. Go to the lib/pages/article_details.dart file and disable the admob line and enable the fb line. Look at the picture below.

```
            |
            |   // -- Banner ads --
            |
            |---context.watch<AdsBloc>().bannerAdEnabled == false ? Container()
            |___: BannerAdAdmob()   //admob
                //: BannerAdFb()    //fb
          ],
        ), // Column
      )); // SafeArea // Scaffold
  }

```

That's it. Your facebook ad setup is complete. Keep in mind that you can't test ads on emulators for fb ads. You have to use a real device.

8. Category Setup

You have to add 4 categories here which you added in the admin panel. ***This is mandatory for UI purposes.*** Now go to **lib/config/config.dart** file and add your own 4 categories that you have added on the admin panel previously. Make sure the category names are the same.

```
36
37  //initial categories - 4 only (Hard Coded : which are added already on your admin panel)
38
39  final List initialCategories = [
40    'Entertainment',
41    'Sports',
42    'Politics',
43    'Travel'
44  ];
45
```

9. Other Setup

1. Go to **lib/models/config.dart** file and change all of your details.
2. Change App name,
3. Support Email,
4. Privacy policy url (Ignore this for now if you are not going to release now).
5. Your website url (same as before)
6. iOS app ID (Only for iOS and you can ignore this for now)
7. Change fb page url, youtube channel url and twitter url with yours.

```dart
final String appName = 'NewsHour';
final String splashIcon = 'assets/images/splash.png';
final String supportEmail = 'mrblab24@gmail.com';
final String privacyPolicyUrl = 'https://www.mrb-lab.com/privacy-policy';
final String ourWebsiteUrl = 'https://www.mrb-lab.com';
final String iOSAppId = '000000';


//social links
static const String facebookPageUrl = 'https://www.facebook.com/mrblab24';
static const String youtubeChannelUrl = 'https://www.youtube.com/channel/UCnNr2eppWVVo-NpRIy1ra7A';
static const String twitterUrl = 'https://twitter.com/FlutterDev';

//app theme color
final Color appColor = Colors.deepPurpleAccent;



//Intro images
final String introImage1 = 'assets/images/news1.png';
final String introImage2 = 'assets/images/news6.png';
final String introImage3 = 'assets/images/news7.png';

//animation files
final String doneAsset = 'assets/animation_files/done.json';
```

10. Change Splash Icon

To change the **splash icon**, you have to upload your own splash icon. The icon should be in the **.png** format and make sure you have renamed it to **splash** . Go to the **lib/assets/images** folder and drop the icon here and replace it with our icon. That's it.

11. Change App Theme Color

Go to the **lib/config/config.dart** file and edit the **appThemeColor**. We have used deepPurpleAcent color.You can use any color and the changes will happen in whole app.

12. Change App Name for Android

1. Go to **android/app/src/main/AndroidManifest.xml** file and Change your app name.

Also go to **lib/utils/app_name.dart** and change the app name.

```
<application
    android:name="io.flutter.app.FlutterApplication"
    android:label="Travel Hour"
    android:icon="@mipmap/ic_launcher">
```

13. Change App Name for iOS

1. Go to **ios/Runner/Info.plist file** and Change your app name.

```
<dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>$(DEVELOPMENT_LANGUAGE)</string>
    <key>CFBundleExecutable</key>
    <string>$(EXECUTABLE_NAME)</string>
    <key>CFBundleIdentifier</key>
    <string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>Travel Hour</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleShortVersionString</key>
    <string>$(FLUTTER_BUILD_NAME)</string>
```

14. Change the App Icon

1. Go to the asset folder and delete the default icon **(icon.png**).
2. Now upload your app icon as png in the **assets/images** folder and rename it to **icon.png**
3. Now run the following command on the terminal,

```
flutter pub get
flutter pub run flutter_launcher_icons:main
```

That's it.. For more info, visit this site.

So Your Setup is 100% complete now.

15. Run the app

Run this command on the terminal.

```
flutter clean
```

And After that run the following command to run this app on your physical or emulator devices.

```
flutter run
```

Test if everything is okay or not.

16. Release the Android App on Google Play Store

You have done all the things that are required for android release. To Test the release android app, run the following command on the terminal.

```
flutter build apk --split-per-abi
```

You will get 3 apk files from the **build/app/output/apk/release** folder. You can test the **v7** version of the apk file. If you want to publish the app in the google play store, don't upload any of the following files. Use an **appbundle** file which is recommended by Google. To generate an appbundle, run the following command on terminal :

```
flutter build appbundle
```

After that, you will get an **.aab** file in the **build/app/output/appbundle/release** folder.

Now you can upload this .aab file to the google play store.

17. To Release the iOS app on App Store

Follow the official doc from flutter team here.

That’s it. We know that you are so tired right now. Take some rest. Everything is complete now.

If you love our work then don't forget to submit a review on codecanyon market.

Thanks

MRB Lab

Contact: mrblab24@gmail.com