//read_admin_doc_first

News Hour App Doc (v3.0.4)

06-03-21 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."

Introduction

So you are currently on the flutter stable channel with the latest version (2.0.0+). There are some issues with the latest version of flutter with some packages & null safety. So you should downgrade to the previous version to build this app without any problem. To downgrade to a specific version,

1. At first you have to find the **flutter** folder - where you have installed flutter. Because you have to run these following commands from this directory. Now open your terminal (for mac), PowerShell (for windows).

2. Now go to your flutter folder directory. Example, if you have installed flutter in your Desktop directory, then type like this to go to the flutter directory:

   ```
   cd Desktop/flutter
   ```

3. Now run this following command from this **flutter** directory.

```
git checkout 1.22.6
```

After that, run this following command,

```
flutter doctor
```

So, It will take some time and downgrade to this specific version. Please make sure that the flutter version number is 1.22.6.

That's it. Your SDK setup is done.

If you don't have **a mac device** and **an apple developer account**, you can skip iOS setups. Because you need Xcode App which is only available for mac devices and others setup requires an apple developer account.

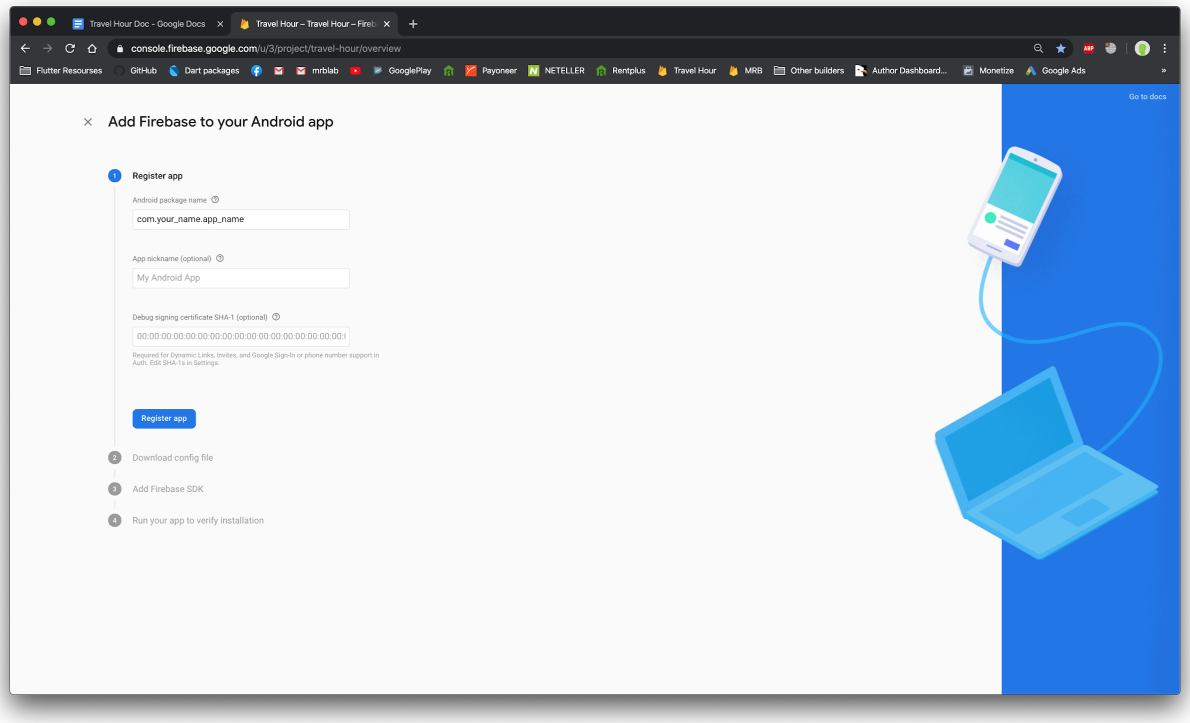
Project Setup

1. Open the **news_app** folder on your IDE (VSCode or Android Studio).
2. Run this following command on the terminal to get all the packages from the server.

```
flutter pub get
```

Firebase Setup for Android

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.

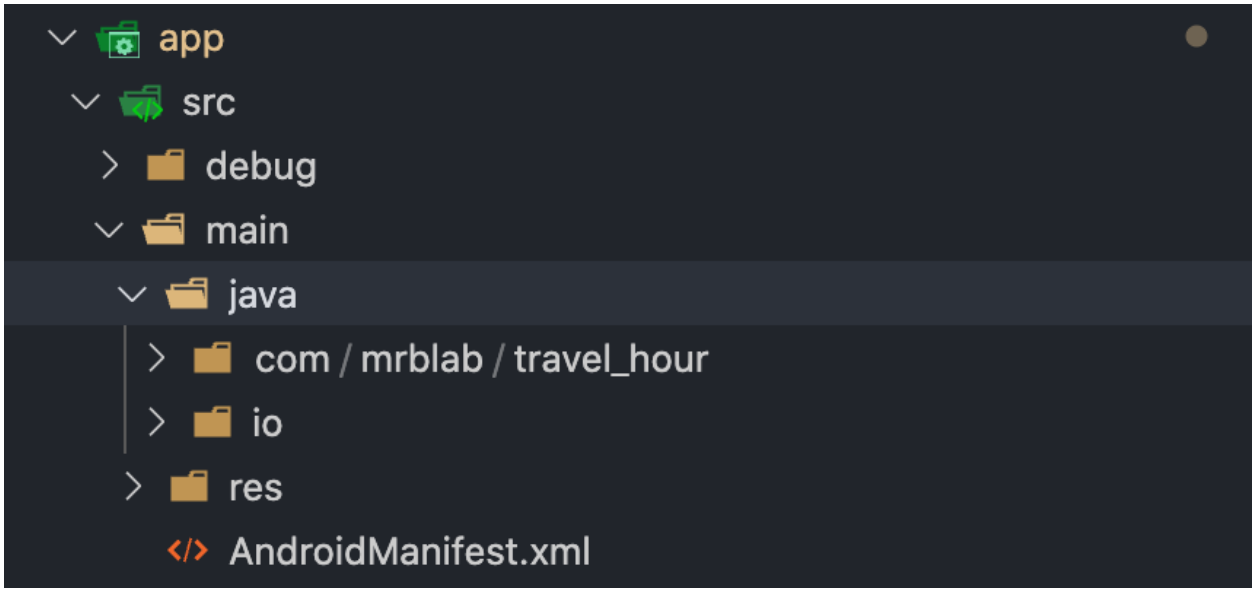
3. Go to your IDE and now you **have to change the package name** of your app. Go to

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

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)

4. Now you have to rename two folders. Go to **android/app/src/main/java** and rename **mrblab** by **your_name** and 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/java** 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 java folder should look like this.



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

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

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

For Mac Users, run

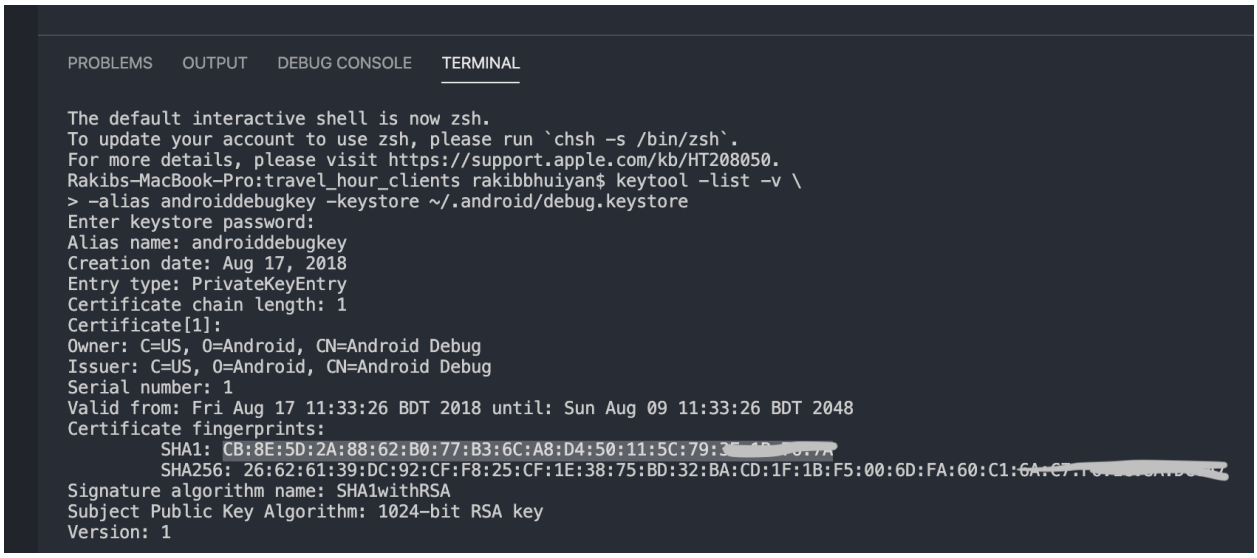
```
keytool -list -v \
-alias androiddebugkey -keystore ~/.android/debug.keystore
```

For Windows users, run

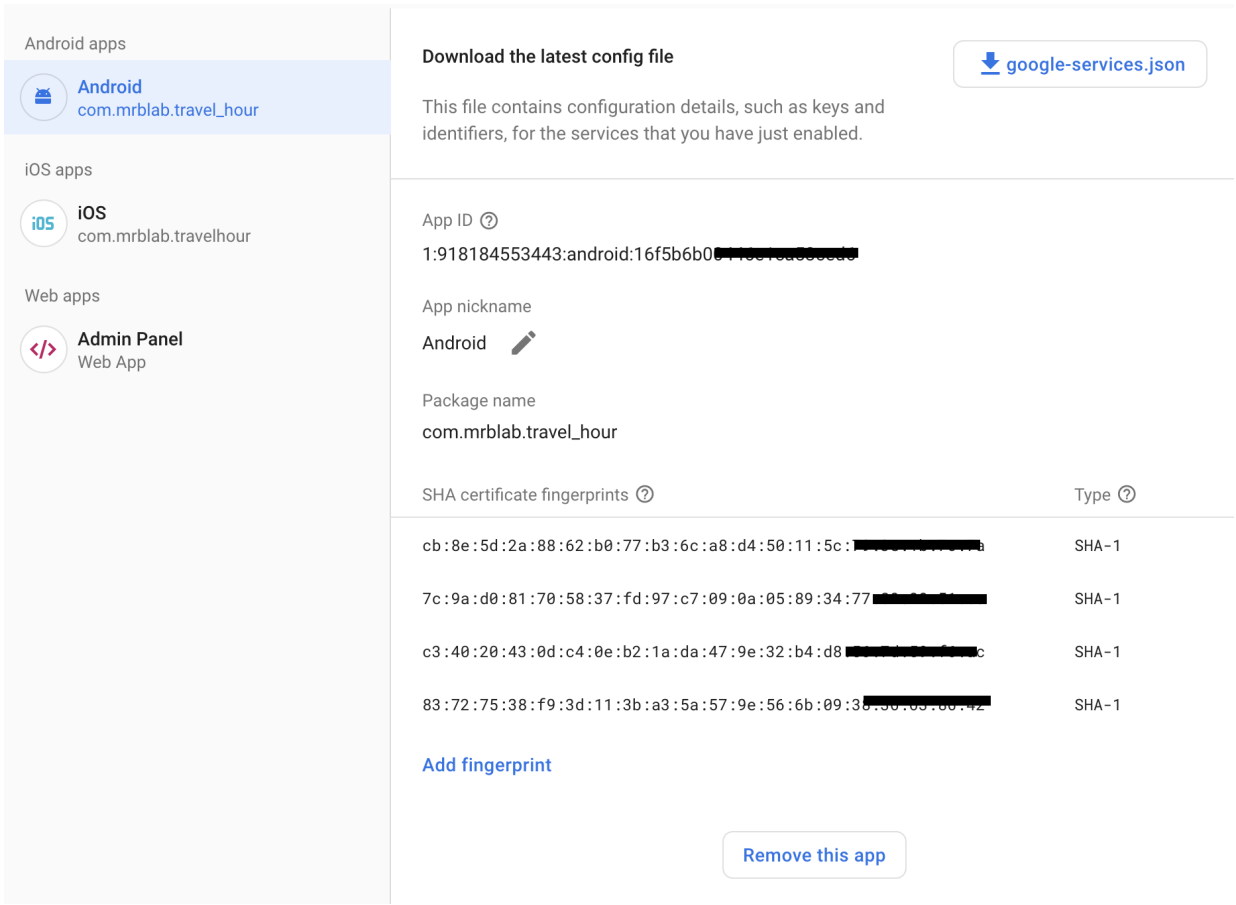
```
keytool -list -v \
-alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore
```

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

N.B. *If this command doesn't work, then go 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.



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.

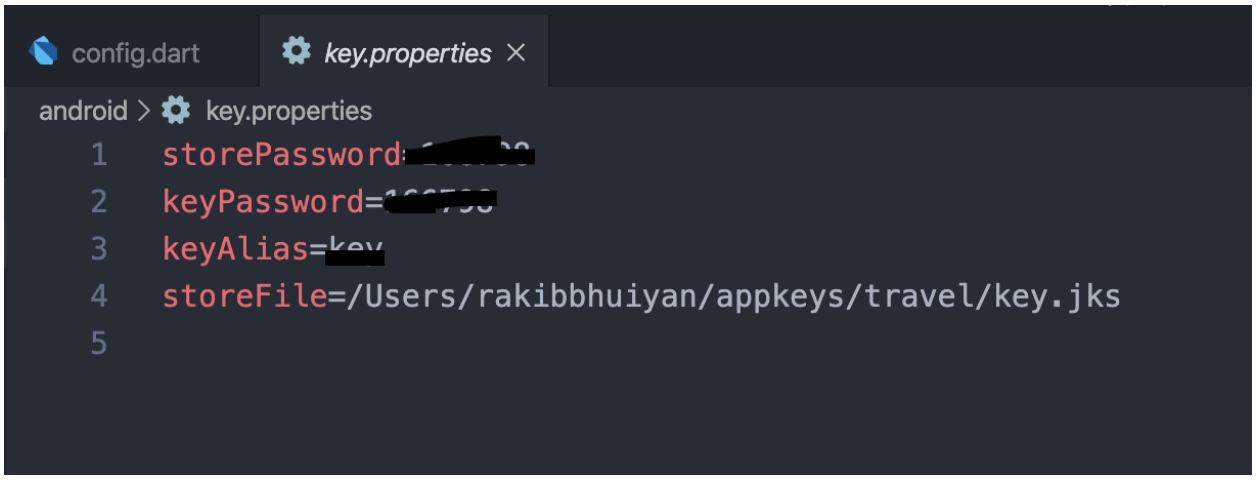
For Mac users, run

```
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
```

Enter your details and remember **alias key** name and **password**. After this, you will get a **.jks** keystore file. Locate this file and move the file into the **android/app** folder and copy the path by right clicking on the **key.jks** file 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.



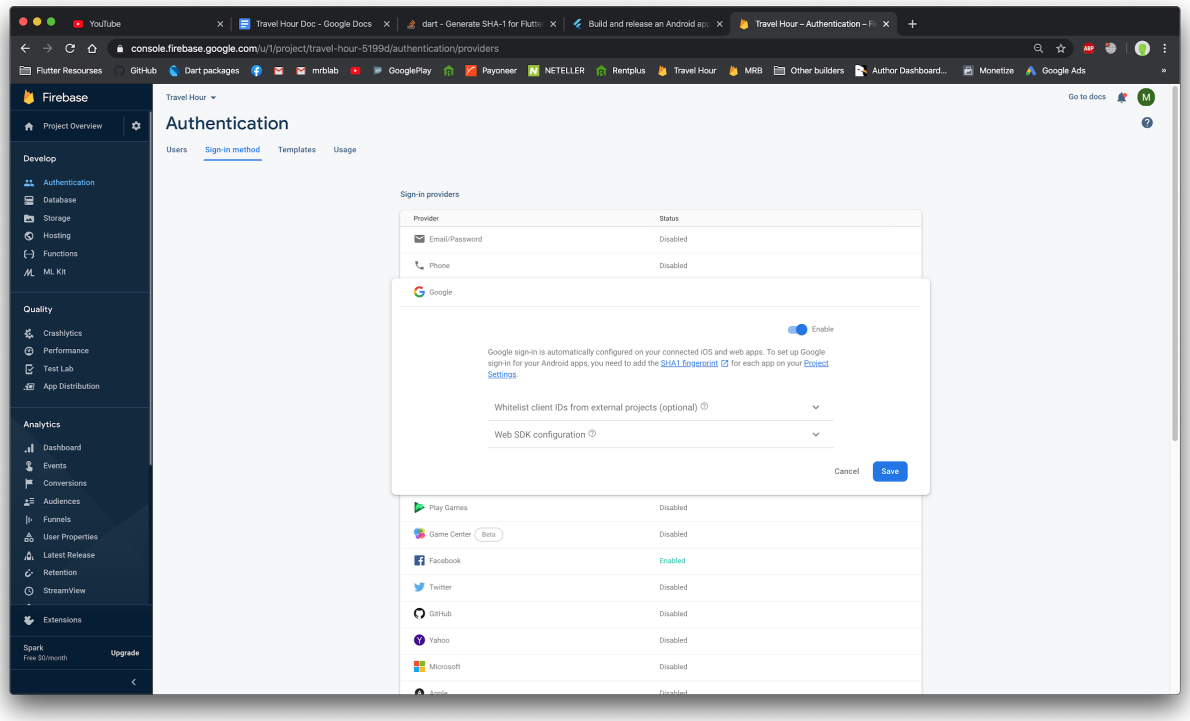
Now you can generate a release certificate, To do that, run with replacing your **alias_name** and **keystore_location**.

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

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.

Google Sign In Setup :

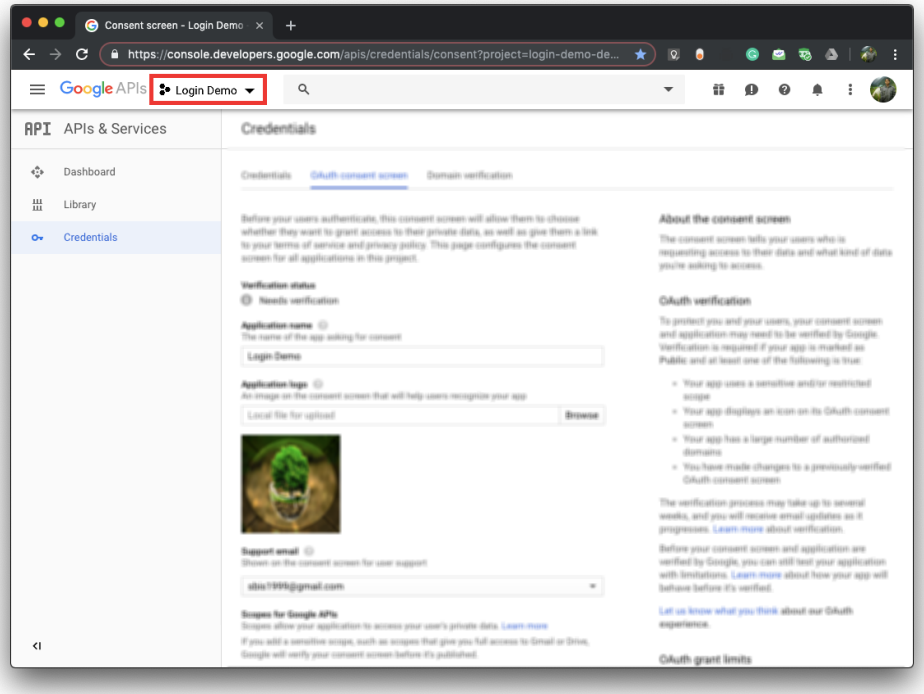
5. Now you have to set up google sign in & email sign in. To do that, Go to **firebase console>your project>authentication>Sign-in-method** and enable both **email/password** and **google** and save it.



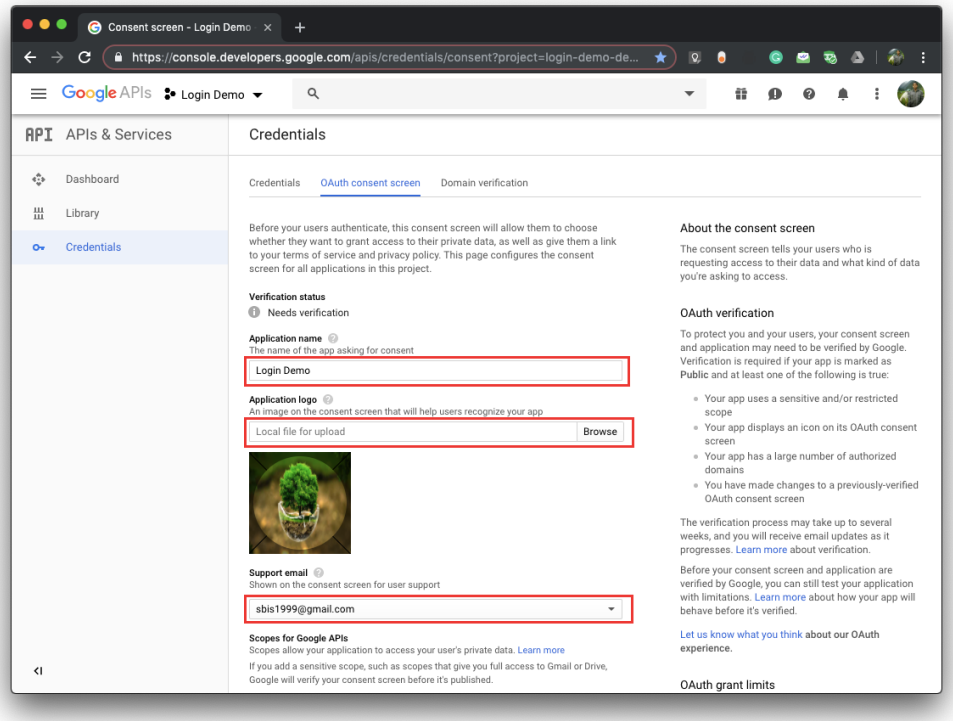
6. You have to configure some stuff for google sign in. Go to this [url](#).

7. Make sure you are signed in with the same account with which you have created the Firebase project.

8. Also, make sure that on the top-left corner, your project is selected for which you are filling this consent.

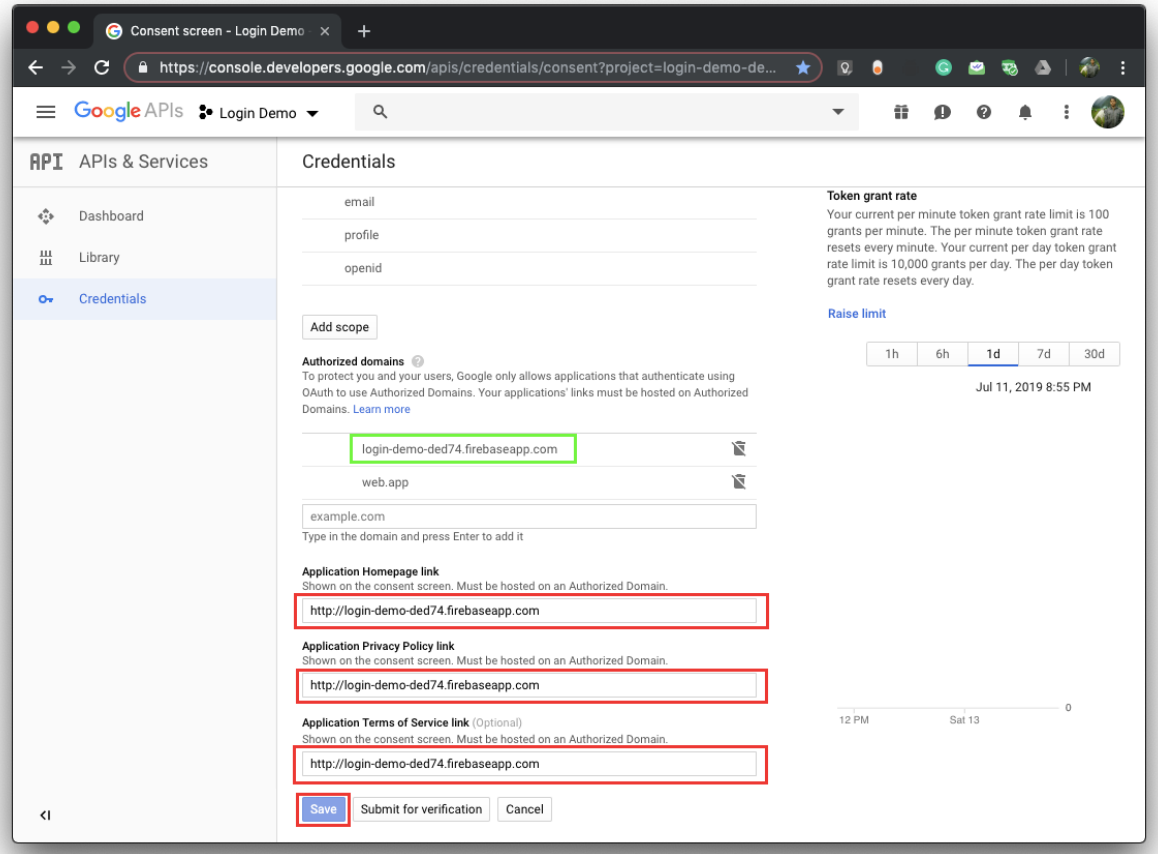


9. Go to Credentials → OAuth consent screen tab and start filling the form.
10. Enter “Application name”, “Application logo” & “Support email”.



11. Then, scroll down and fill the “Application Homepage link”, “Application Privacy Policy link” and “Application Terms of Services link”.

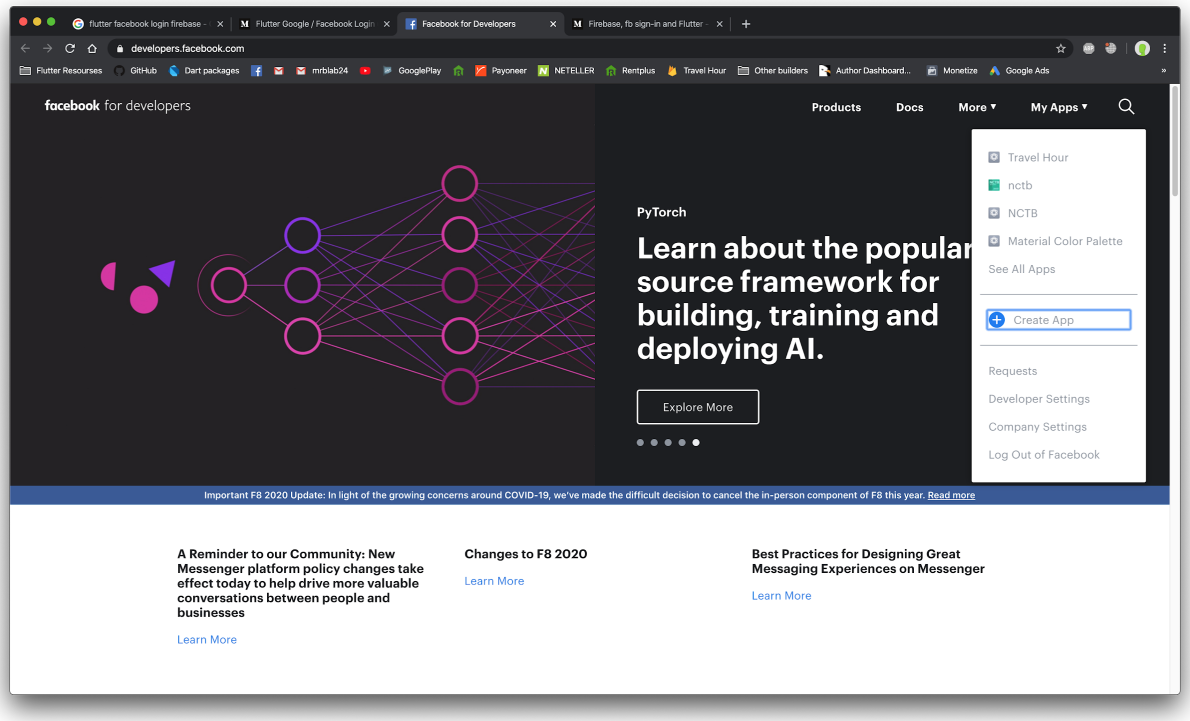
12. In all these places, you have to enter the same link starting with **http://** then your app domain name which I have marked with green below.



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

Facebook Sign In Setup :

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



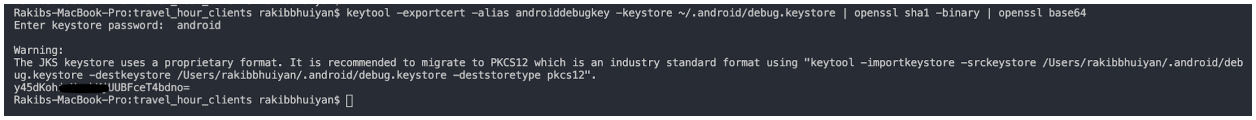
17. Scroll down on the right pane until you reach '**Add a product**', select **Facebook Login**
18. You will be redirected to the quick start page.
19. Select **Android**. Skip 1 & 2.
20. Enter **your_package_name** in the package option and enter **your_package_name.MainActivity** in the activity option.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

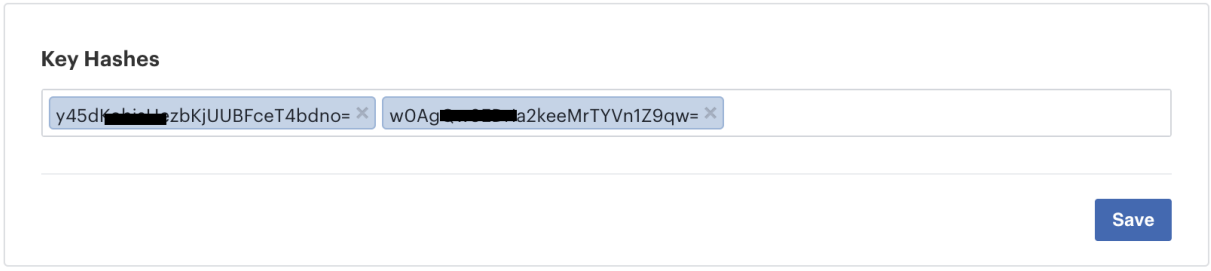
Use **android** as a debug password. After that you will get a **hash id** like this.



22. 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
```

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



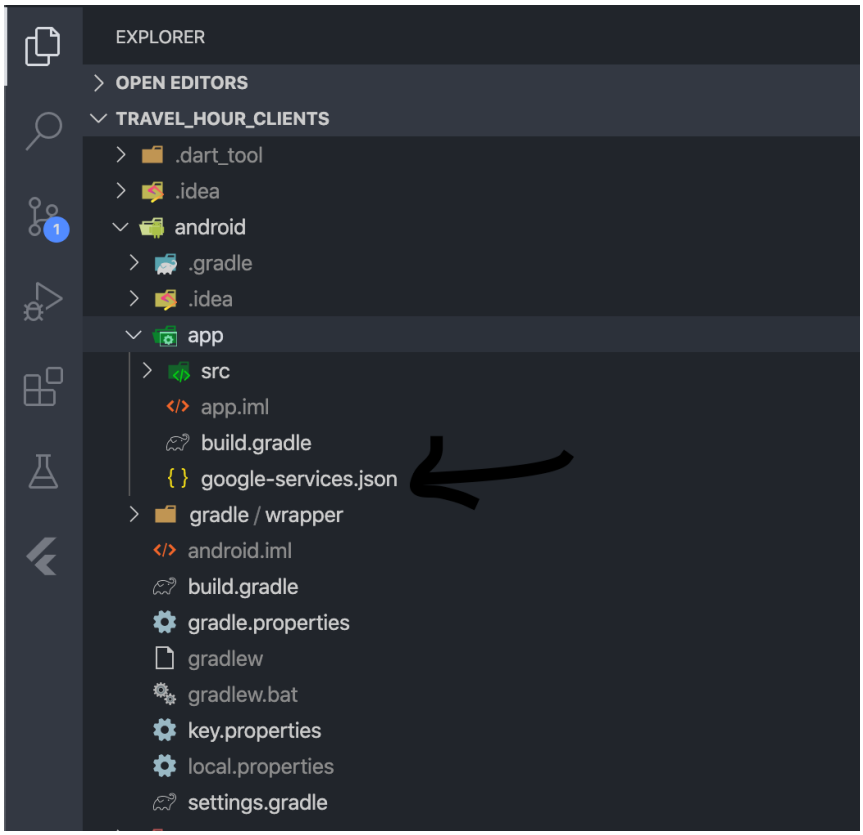
24. Skip all the steps by clicking next.

25. Now go to **settings** tab & copy both **app id** and app **secret key**.

26. 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.

27. Now go to project settings and click on **android icon** and download **google-service.json** file.

28. Now go to **android/app** directory and paste the **google-service.json** file here.

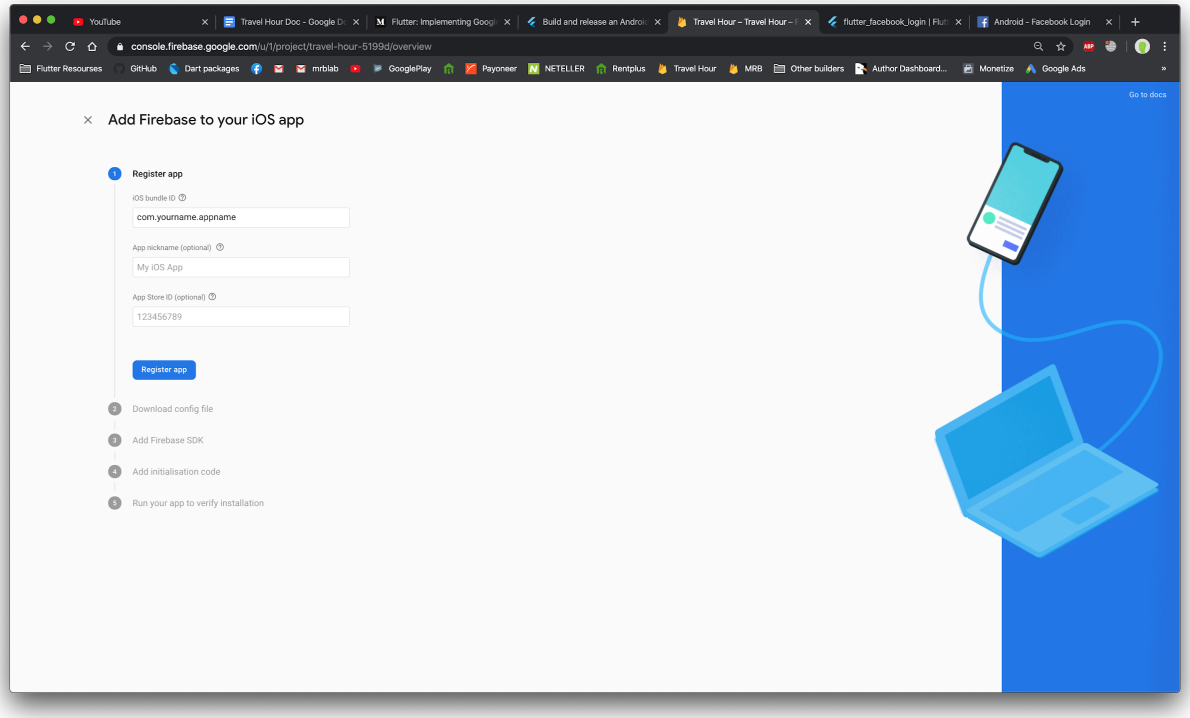


29. Now go to **android/app/src/main/res/values/strings.xml** this directory and change **the app name**, **app_id** and **fb+app_id**.

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

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. The go to **ios/Runner** directory and paste the file here.

5. Now, Open **iOS folder** on Xcode by right clicking on iOS folder from VSCode or Android Studio and go to runner folder and move the **GoogleService-info.plist** file here. You will get a popup and click yes or confirm the popup message.

5. 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>
```

6. 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>
```

7. 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)

```
<dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>$(DEVELOPMENT_LANGUAGE)</string>
    <key>CFBundleExecutable</key>
    <string>$(EXECUTABLE_NAME)</string>
    <key>CFBundleIdentifier</key>
    <string>[redacted]</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleLocalizations</key>
    <array>
```

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

Facebook Sign In for iOS :

8. Now go to developers.facebook.com again and navigate to your project and click facebook login > quick setup > ios icon.

9. Skip 1 & go to step 2.

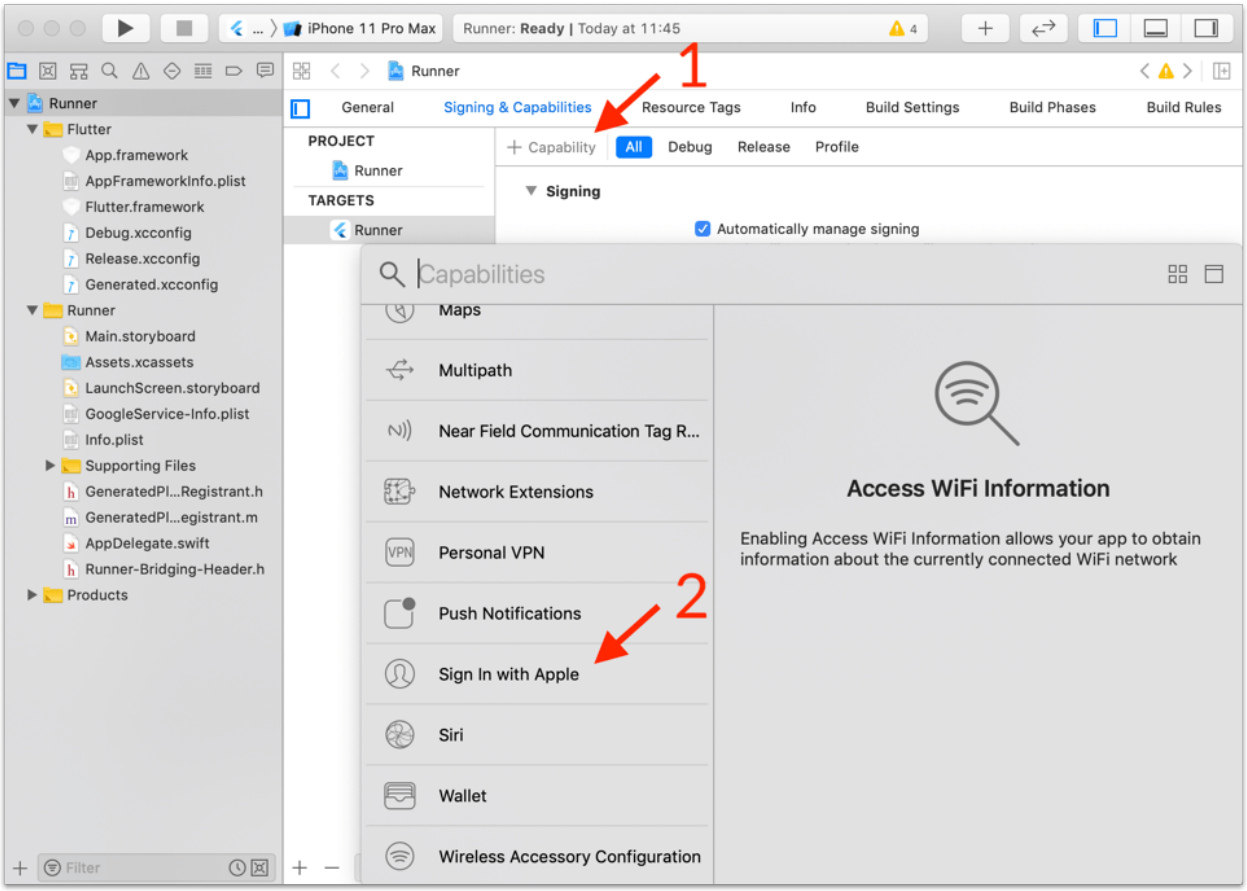
10.Enter **your package name** in the **bundle ID** option & skip others by clicking next.

11. That's it. iOS setup for Facebook Sign In is complete.

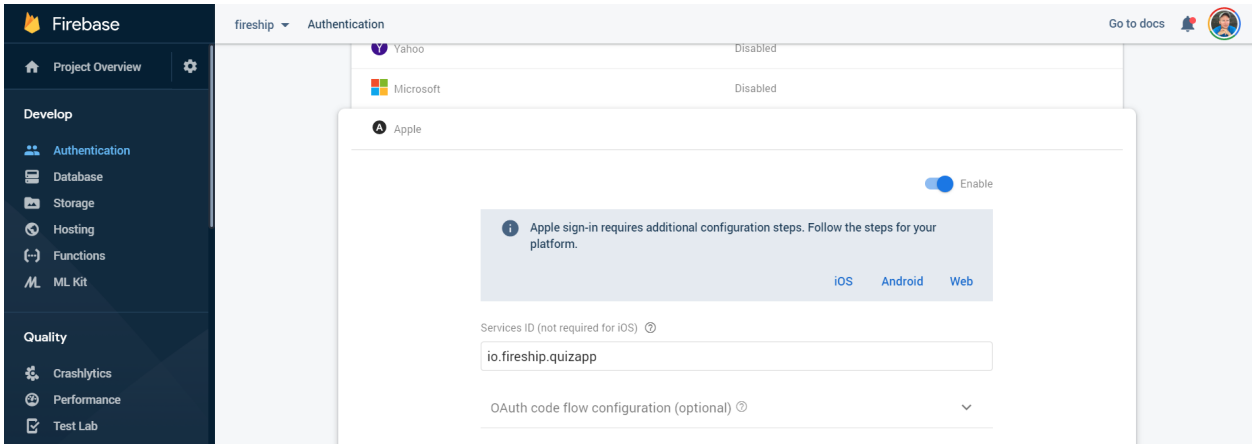
Sign In with Apple (Only for iOS)

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.



4. 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.

Firebase Push Notifications Setup :

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

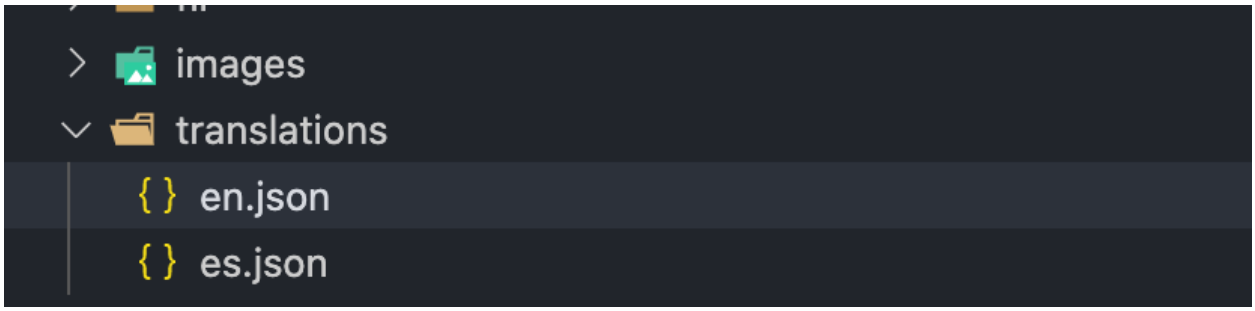
2. For iOS, Go to this link and follow the instructions. This is a well written doc from Flutter Team.

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 as much 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. No 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. No 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 list by adding 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 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.

Ads Setup :

You can skip this setup for now and can configure later. We disabled ads by default.

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 main screens.

   Article_details.dart - interstitial ad ,

   video_article_details.dart - banner ad.

Admob Setup :

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 add these lines at the bottom inside </dict> by replacing your admob app id.

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

3. Go to **lib/config/config.dart** file and replace **admob ids.**
4. Go to pubspec.yaml file and enable **firebase_admob** package by removing # and run **flutter pub get** command on the terminal.
5. Now go to **lib/pages/home.dart** file and enable admob function by removing // from the back.

```
@override
 void initState() {
   super.initState();
   Future.delayed(Duration(milliseconds: 0))
   .then((value) async{
     final adb = context.read<AdsBloc>();
     await context.read<NotificationBloc>().initFirebasePushNotification(context)
     .then((value) => context.read<NotificationBloc>().handleNotificationlength())
     .then((value) => adb.checkAdsEnable())
     .then((value)async{
       if(adb.interstitialAd == true || adb.bannerAd == true){
         //FirebaseAdMob.instance.initialize(appId: Config().admobAppId);  //admob
         //FacebookAudienceNetwork.init();  //fb


       }
     });
   });
 }
```

6. Now go to **lib/pages/video_article_details.dart** file and enable the admob section. See the pictures below.

```
386
387   //Admob ads -- START
388
389
390   // InterstitialAd _admobInterstitialAd;
391
392   // static const MobileAdTargetingInfo targetingInfo = MobileAdTargetingInfo(
393   //   nonPersonalizedAds: true,
394   // );
395
396
397   // InterstitialAd createAdmobInterstitialAd() {
398   //   return InterstitialAd(
399   //     adUnitId: Platform.isAndroid ? Config().admobInterstitialAdIdAndroid : Config().admobInterstitialAdIdiOS,
400   //     targetingInfo: targetingInfo,
401   //     listener: (MobileAdEvent event) {
402   //       print("InterstitialAd event $event");
403   //     },
404   //   );
405   // }
406
407
408   // initiateAdmobInterstitial (){
409   //   if(mounted){
410   //     if(context.read<AdsBloc>().interstitialAd == true){
411   //       adInitiated = true;
412   //       _admobInterstitialAd = createAdmobInterstitialAd()..load()..show();
413   //     }
414   //   }
415   // }
416
417   // disposeAdmobInterstitial (){
418   //   if(adInitiated != null){
419   //     _admobInterstitialAd?.dispose();
420   //   }
421   // }
422
423   //admob ads -- END
424
```

Select the code and press **command + /** for mac & **control + /** for windows to enable the code.

7. Again in this file, enable these lines.

```
6
7   @override
8   void initState() {
9     super.initState();
10    //initiateAdmobInterstitial();  //admob
11    //initFbInterstitialAd();    //fb
12
13    initYoutube();
14    Future.delayed(Duration(milliseconds: 100)).then((value) {
15      setState(() {
16        rightPaddingValue = 10;
17      });
18    });
19  }
20
21  @override
22  void dispose() {
23    _controller.dispose();
24    //disposeAdmobInterstitial(); //admob
25    //disposefbInterstitial();  //fb
26
27    super.dispose();
28  }
29
```

8. Now go to **lib/pages/article_details.dart** and do the same. Only enable where the admob is noted.

```
//admob --START
// BannerAd _admobBannerAd;

//  static const MobileAdTargetingInfo targetingInfo = MobileAdTargetingInfo(
//    nonPersonalizedAds: true,
// );


// BannerAd createAdmobBannerAd() {
//   return BannerAd(
//     adUnitId: Platform.isAndroid ? Config().admobBannerAdIdAndroid : Config().admobBannerAdIdiOS,
//     size: AdSize.banner,
//     targetingInfo: targetingInfo,
//     listener: (MobileAdEvent event) {
//       print("BannerAd event $event");
//     },
//   );
// }


// initiateAdmobBanner (){
//   if(mounted){
//     if(context.read<AdsBloc>().bannerAd == true){
//       adInitiated = true;
//       _admobBannerAd = createAdmobBannerAd()..load()..show(
//       anchorOffset: 0.0,
//       anchorType: AnchorType.bottom
//   );
//     }
//   }
// }

// disposeAdmobbanner (){
//   if(adInitiated != null){
//     _admobBannerAd?.dispose();
//   }
// }

//Admob -- END
```

```
                label: Text('comments',
                        style:
                            TextStyle(color: Colors.black87)) // Text
                    .tr(),
                onPressed: () {
                  //disposeAdmobbanner(); //admob
                  nextScreen(context,
                      CommentsPage(timestamp: d.timestamp));
                },
              ) // FlatButton.icon
            ], // <Widget>[]
          ), // Row
```

```
@override
void initState() {
  super.initState();
  //initiateAdmobBanner();  //admob
  Future.delayed(Duration(milliseconds: 100)).then((value) {
    setState(() {
      rightPaddingValue = 10;
    });
  });
}

@override
void dispose() {
  //disposeAdmobbanner(); //admob
  super.dispose();
}
```

That's it. Your Admob setup is complete.

Facebook Ads Setup :

Ignore this if you have added admob.

Before we do this, we need to inform you that iOS won't support facebook ads in this version. We have found some problems and issues with the fb ads package and decided to disable this for iOS. So you have to enable the **facebook_audience_network** package first from the **pubspec.yaml** file.

1. First, as we previously said that you need to remove admob first to apply fb ads on your app. To do that, go to pubspec.yaml file and remove **firebase_admob** line enable **facebook_audience_network** and then run **flutter pub get** on the terminal.

2. Now, go to the **android/app/src/main/AndroidManifest.xml** and remove these 3 lines.

```
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
```

3. Now go to **lib/config/config.dart** file and replace the values fb ads.

4. Now go to **lib/pages/home.dart** file and enable fb function by removing // from the back.

```
@override
 void initState() {
   super.initState();
   Future.delayed(Duration(milliseconds: 0))
   .then((value) async{
     final adb = context.read<AdsBloc>();
     await context.read<NotificationBloc>().initFirebasePushNotification(context)
     .then((value) => context.read<NotificationBloc>().handleNotificationlength())
     .then((value) => adb.checkAdsEnable())
     .then((value)async{
       if(adb.interstitialAd == true || adb.bannerAd == true){
         //FirebaseAdMob.instance.initialize(appId: Config().admobAppId);  //admob
         //FacebookAudienceNetwork.init();  //fb
```

5. Now go to **lib/models/fb_banner.dart** file and enable everything in this file by selecting all the code and pressing **Command+/** for mac & **Control+/** for windows. Import the packages.

6. Now go to **lib/pages/article_details.dart** and **lib/pages/video_article_details.dart** and enable fb procedures. (only fb not admob- see the pictures below)

```
//banner ad fb

//context.watch<AdsBloc>().bannerAd != true
//:BannerAdsFb().fbBanner
```

```
//fb ads -- START

// initFbInterstitialAd() {
//   if (mounted) {
//     if (context.read<AdsBloc>().interstitialAd == true) {
//       adInitiated = true;
//       FacebookInterstitialAd.loadInterstitialAd(
//         placementId: Platform.isAndroid ? Config().fbInterstitalAdIDAndroid : Config().fbInterstitalAdIDiOS,
//         listener: (result, value) {
//           print('fb=--------------------$result + $value');
//           if (result == InterstitialAdResult.LOADED){
//             FacebookInterstitialAd.showInterstitialAd();
//           }
//           if (result == InterstitialAdResult.ERROR && value["invalidated"] == true) {
//             print('fb interstitial : $result');

//           }
//         },
//       );

//     }
//   }
// }




// disposefbInterstitial()async {
//   if(adInitiated != null){
//     FacebookInterstitialAd.destroyInterstitialAd();
//   }
// }


//fb ads -- END
```

```
@override
void initState() {
  super.initState();
  //initiateAdmobInterstitial();  //admob
  //initFbInterstitialAd();    //fb

  initYoutube();
  Future.delayed(Duration(milliseconds: 100)).then((value) {
    setState(() {
      rightPaddingValue = 10;
    });
  });
}

@override
void dispose() {
  _controller.dispose();
  //disposeAdmobInterstitial(); //admob
  //disposefbInterstitial();  //fb

  super.dispose();
}
```

7. That’s it. Your fb ads setup is successful.

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.

```
//initial categories - 4 only (Hard Coded : which are added already on your admin panel)

final List initialCategories = [
  'Entertainment',
  'Sports',
  'Politics',
  'Travel'
];

```

Other Setup

1. Go to **lib/models/config.dart** file and change all of your details.
2. Change App name,
3. 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 **lib/assets/images** folder and drop the icon here and replace with our icon.
4. Support Email,
5. Privacy policy url (Ignore this for now if you are not going to release now).
6. Your website url (same as before)
7. iOS app ID (Only for iOS and you can ignore this for now)

```
final String appName = 'NewsHour';
final splashIcon = 'assets/splash.png';
final String supportEmail = 'mrblab24@gmail.com';
final String privacyPolicyUrl = 'https://docs.google.com/document/d/e/2PACX-1vQdPJa9hY4jNUICjM-nwMdiDp6vwZQXxR_irFizRTZXfus01h
final String ourWebsiteUrl = 'https://codecanyon.net/user/mrblab24/portfolio';
final String iOSAppId = '00000000';

final String doneAsset = 'assets/done.json';
final Color appColor = Colors.deepPurpleAccent;



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

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">
```

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>
```

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. Now run this following command to clean the project.

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.

To 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.

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