//read_admin_doc_first

Walfy App Doc

28-07-20 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 beta channel. You can build this app from this channel too. If you want to build this app on the flutter stable channel, you can. It's recommended to use the stable channel. If you want to switch on the stable channel, just run **flutter channel stable** and then **flutter upgrade**. That’s it. It will go back to the stable channel just like before. Apple App Store doesn’t accept third party wallpaper apps. So, This app is only for android.

Project Setup

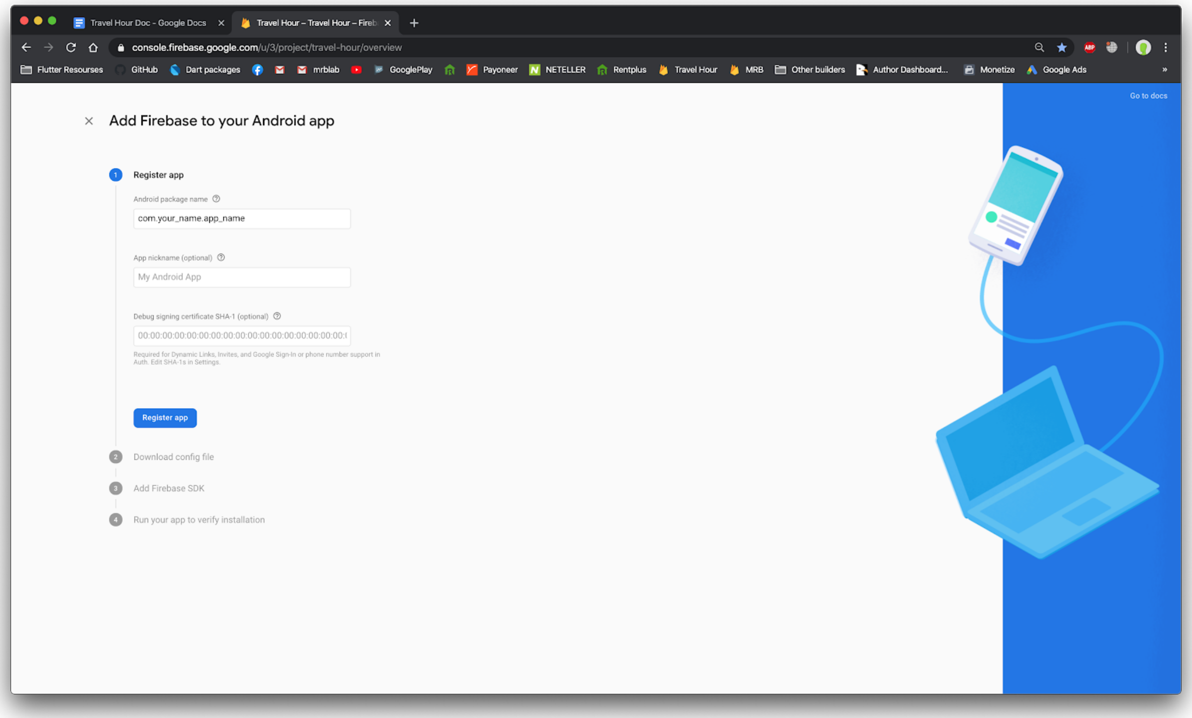
1. Open the **wallpaper_app** folder on your IDE (VSCode or Android Studio).

2. Run this command on terminal to get all the packages.

```
flutter pub get
```

Firebase Setup (Database & Google Sign In)

1. Go to the firebase console > Project overview page. Click on add app and then android icon.



Add Firebase to your Android app

1 Register app

Android package name: com.your_name.app_name

App nickname (optional): My Android App

Debug signing certificate SHA-1 (optional): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

2 Download config file

3 Add Firebase SDK

4 Run your app to verify installation

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/wallpaper_app/MainActivity.java,**
- **android/app/src/profile/AndroidManifest.xml**

files and replace **mrblab.wallpaper_app** by **your_package_name**.

Now you have to rename the some folder by following your package name.

First, go to **android/app/src/main/java** folder and rename **mrblab** by **your_name** and **wallpaper_app** by **app_name**. Then create a new folder inside the **java folder** and rename it as **com** and then move the **your_name** folder inside the **com** folder. After that move the **app_name** folder inside the **your_name** folder. The folders will look like this : **java > com > your_name > app_name.** Make sure the **MainActivity.java is** available inside the **app_name** folder.

4. 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.

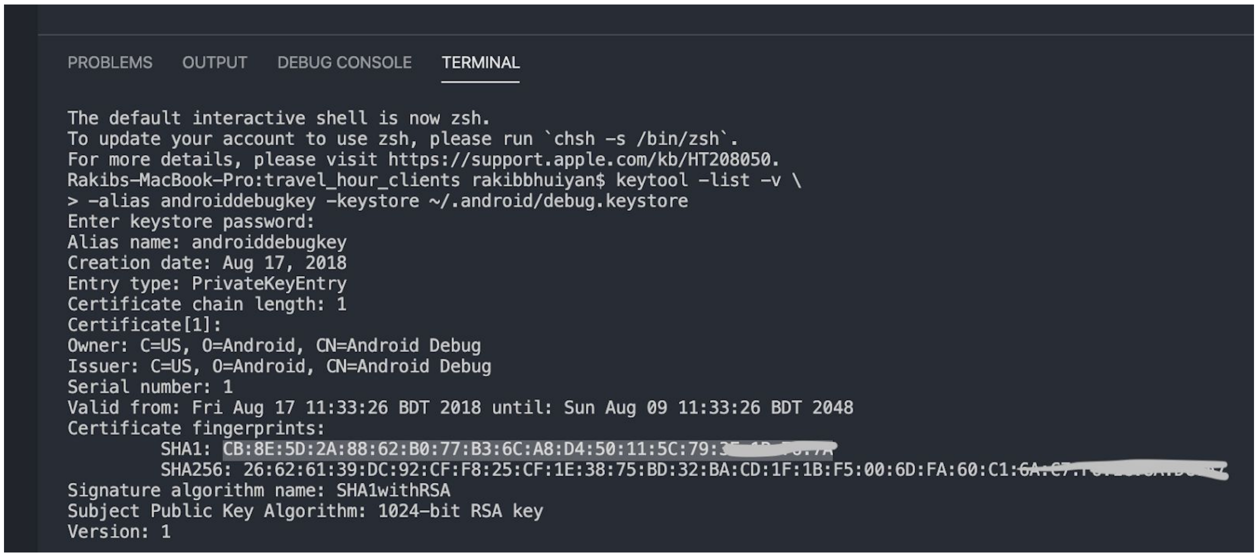
For Mac Users, run

```
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.



```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Rakibs-MacBook-Pro:travel_hour_clients rakibbhuiyan$ keytool -list -v \
> -alias androiddebugkey -keystore ~/.android/debug.keystore
Enter keystore password:
Alias name: androiddebugkey
Creation date: Aug 17, 2018
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: C=US, O=Android, CN=Android Debug
Issuer: C=US, O=Android, CN=Android Debug
Serial number: 1
Valid from: Fri Aug 17 11:33:26 BDT 2018 until: Sun Aug 09 11:33:26 BDT 2048
Certificate fingerprints:
         SHA1: CB:8E:5D:2A:88:62:B0:77:B3:6C:A8:D4:50:11:5C:79:3[redacted]
         SHA256: 26:62:61:39:DC:92:CF:F8:25:CF:1E:38:75:BD:32:BA:CD:1F:1B:F5:00:6D:FA:60:C1:6A[redacted]
Signature algorithm name: SHA1withRSA
Subject Public Key Algorithm: 1024-bit RSA key
Version: 1
```

Copy the SHA1 certificate code and go to Firebase Console>Your Project>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.

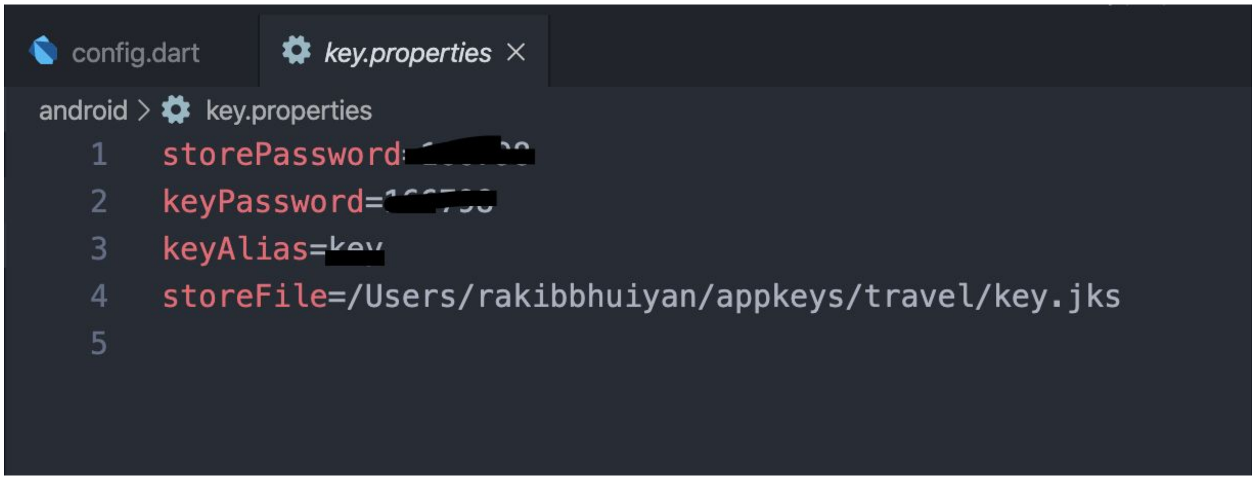
For Mac users, run

```
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**. You can use **key** as alias name. After this, you will get a **.jks** keystore file. Locate this file and paste the file into the **android/app** directory. Then go to **android/key.properties** file and replace the location of the keystore file of yours. Then also replace the **password** and **key alias name** which you have inputted to generate the keystore file.



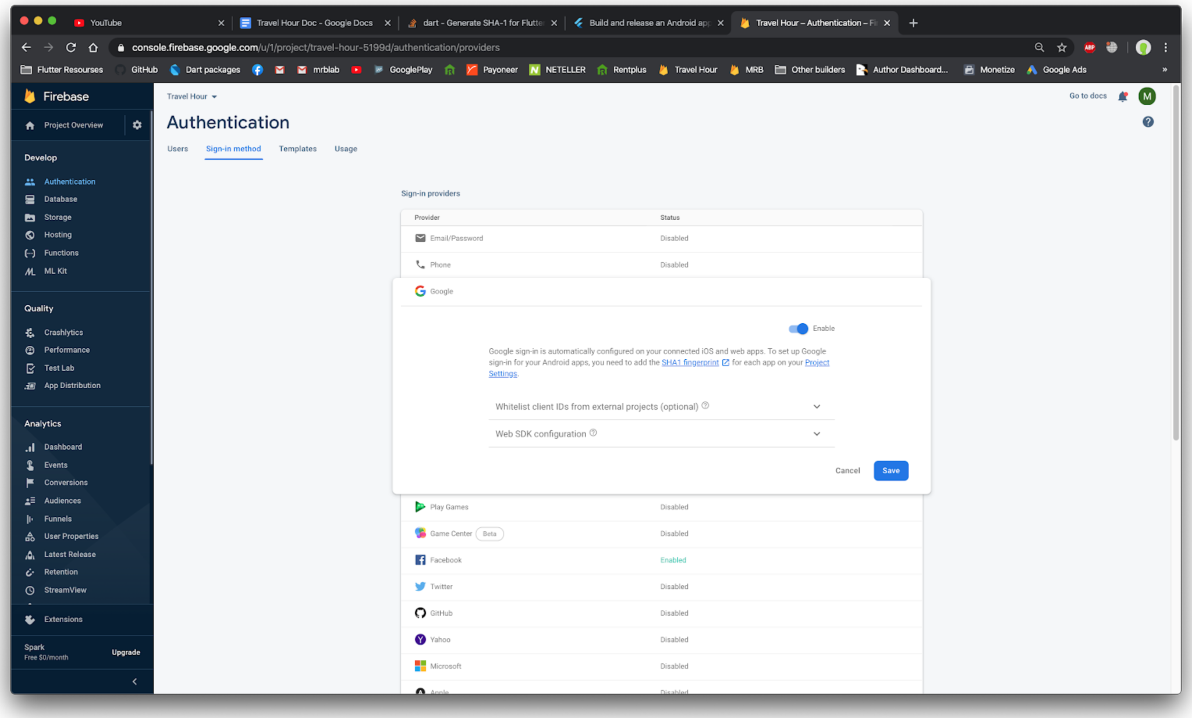
```
storePassword=[redacted]
keyPassword=[redacted]
keyAlias=[redacted]
storeFile=/Users/rakibbhuiyan/appkeys/travel/key.jks
```

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.

5. Now you have to set up **google sign in**. To do that, Go to firebase console>your project>authentication>Sign-in-method and click on **google** and the enable and save it.



Sign-in providers

Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	
Play Games	Disabled
Game Center Beta	Disabled
Facebook	Enabled
Twitter	Disabled
GitHub	Disabled
Yahoo	Disabled
Microsoft	Disabled
Apple	Disabled

Google sign-in is automatically configured on your connected iOS and web apps. To set up Google sign-in for your Android apps, you need to add the SHA1 fingerprint for each app on your Project Settings.

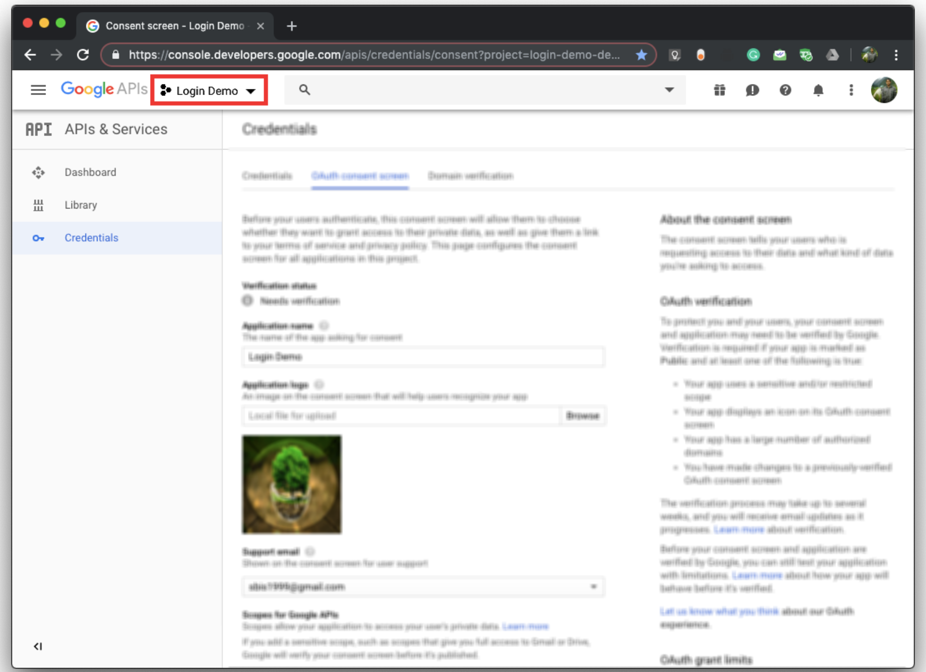
Whitelist client IDs from external projects (optional)

Web SDK configuration

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.



Consent screen - Login Demo

https://console.developers.google.com/apis/credentials/consent?project=login-demo-de...

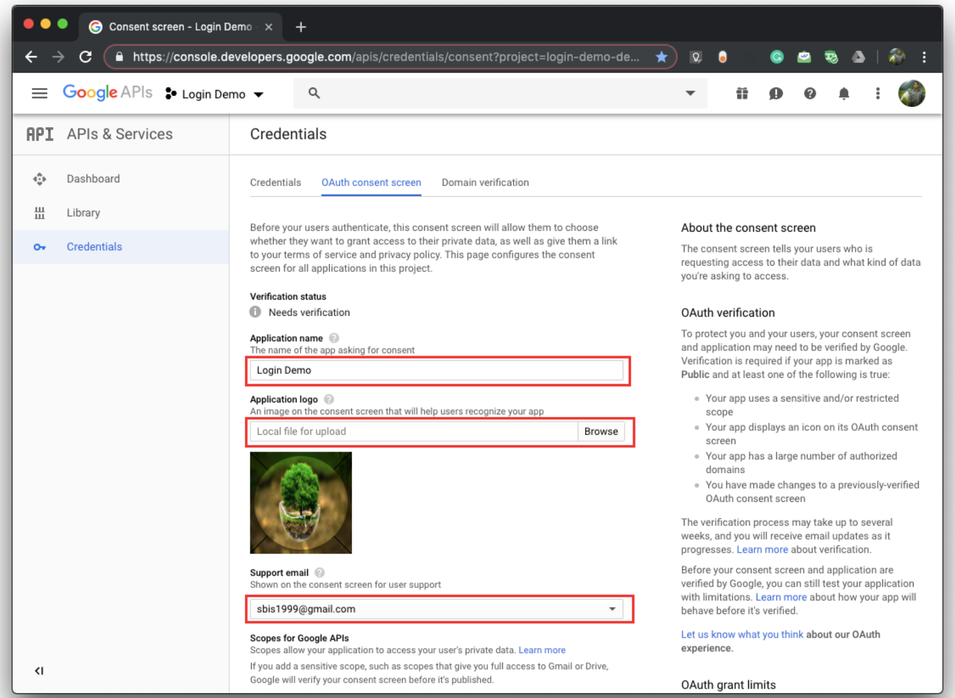
Google APIs Login Demo

APIs & Services

Credentials

9. Go to Credentials → OAuth consent screen tab and start filling the form.

10. Enter “Application name”, “Application logo” & “Support email”.



Consent screen - Login Demo

https://console.developers.google.com/apis/credentials/consent?project=login-demo-de...

Google APIs Login Demo

APIs & Services

Dashboard

Library

Credentials

Credentials

Credentials | OAuth consent screen | Domain verification

Before your users authenticate, this consent screen will allow them to choose whether they want to grant access to their private data, as well as give them a link to your terms of service and privacy policy. This page configures the consent screen for all applications in this project.

Verification status

Needs verification

Application name

The name of the app asking for consent

Login Demo

Application logo

An image on the consent screen that will help users recognize your app

Local file for upload | Browse

Support email

Shown on the consent screen for user support

sbis1999@gmail.com

Scopes for Google APIs

Scopes allow your application to access your user's private data. Learn more

If you add a sensitive scope, such as scopes that give you full access to Gmail or Drive, Google will verify your consent screen before it's published.

About the consent screen

The consent screen tells your users who is requesting access to their data and what kind of data you're asking to access.

OAuth verification

To protect you and your users, your consent screen and application may need to be verified by Google. Verification is required if your app is marked as **Public** and at least one of the following is true:

- Your app uses a sensitive and/or restricted scope
- Your app displays an icon on its OAuth consent screen
- Your app has a large number of authorized domains
- You have made changes to a previously-verified OAuth consent screen

The verification process may take up to several weeks, and you will receive email updates as it progresses. Learn more about verification.

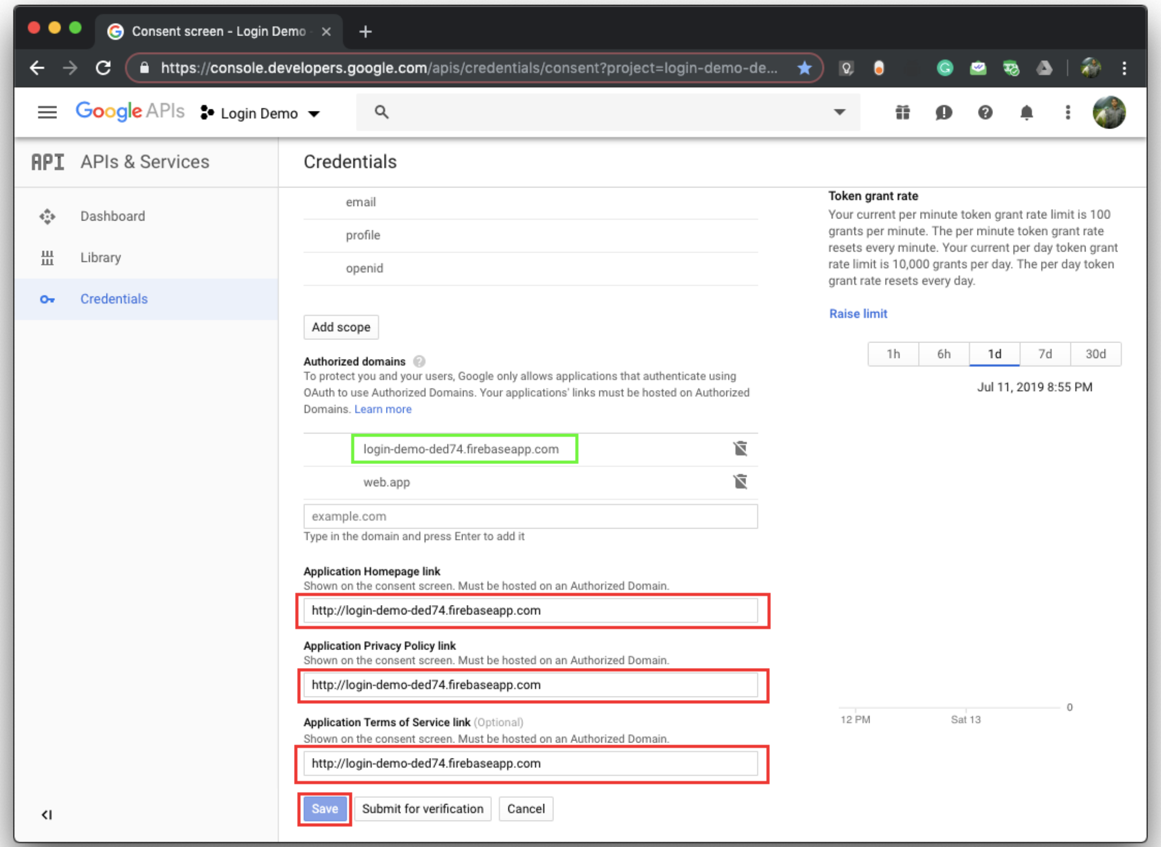
Before your consent screen and application are verified by Google, you can still test your application with limitations. Learn more about how your app will behave before it's verified.

Let us know what you think about our OAuth experience.

OAuth grant limits

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.



Credentials

Token grant rate
Your current per minute token grant rate limit is 100 grants per minute. The per minute token grant rate resets every minute. Your current per day token grant rate limit is 10,000 grants per day. The per day token grant rate resets every day.

Authorized domains
To protect you and your users, Google only allows applications that authenticate using OAuth to use Authorized Domains. Your applications' links must be hosted on Authorized Domains. Learn more

login-demo-ded74.firebaseapp.com
web.app
example.com
Type in the domain and press Enter to add it

Application Homepage link
Shown on the consent screen. Must be hosted on an Authorized Domain.
http://login-demo-ded74.firebaseapp.com

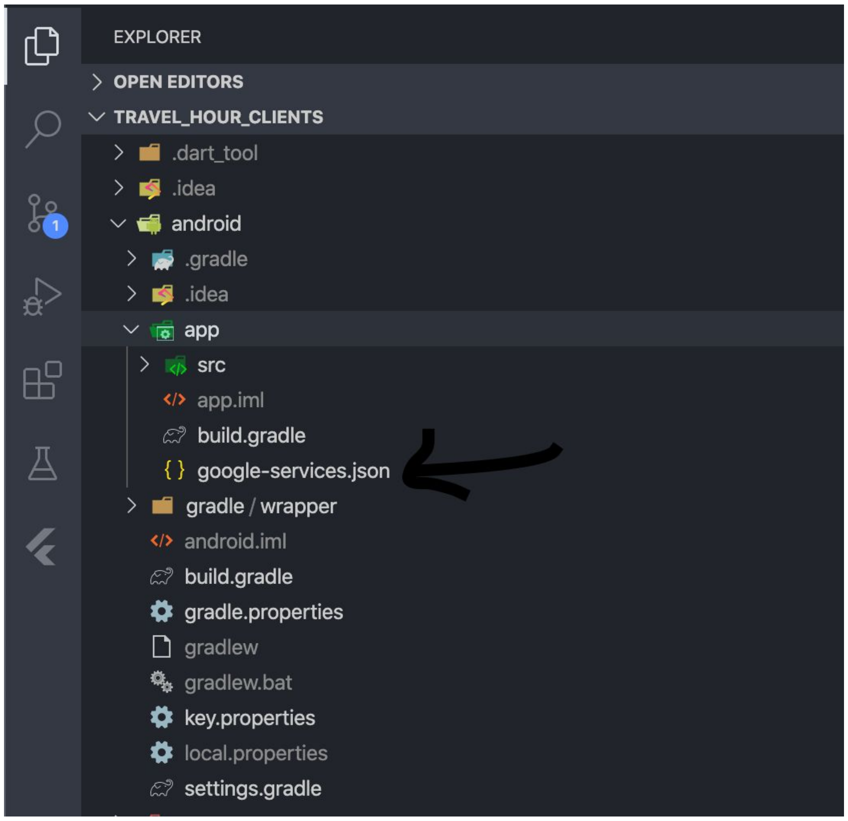
Application Privacy Policy link
Shown on the consent screen. Must be hosted on an Authorized Domain.
http://login-demo-ded74.firebaseapp.com

Application Terms of Service link (Optional)
Shown on the consent screen. Must be hosted on an Authorized Domain.
http://login-demo-ded74.firebaseapp.com

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

14. Now go to Firebase console > project settings and click on **android icon** and download **google-service.json** file.

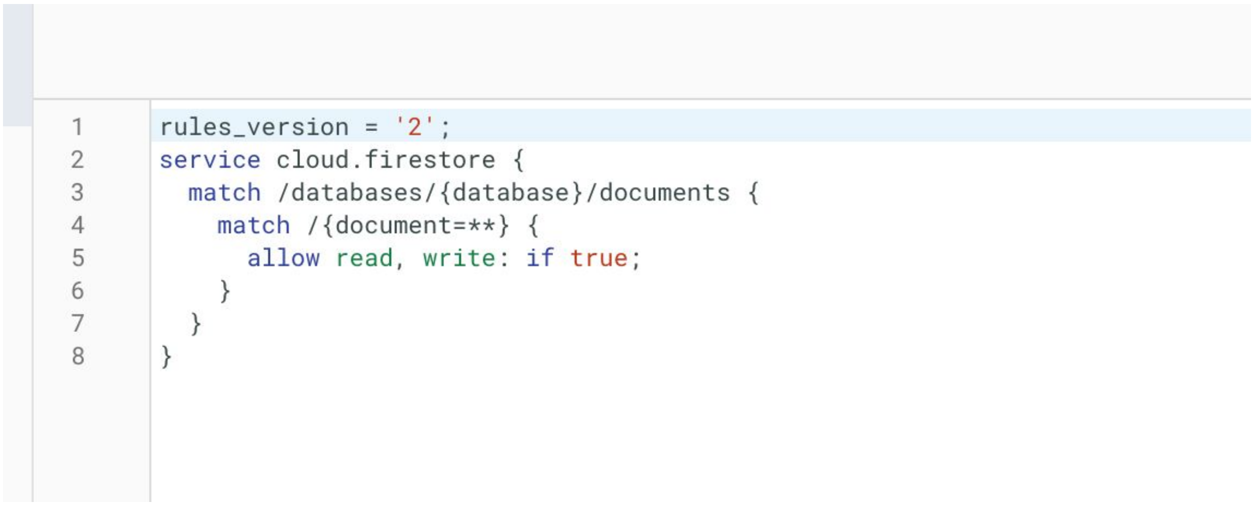
15. Now go to **android/app** directory and paste the **google-service.json** file here.



16. That’s it. Android setup for Google Sign in is complete.

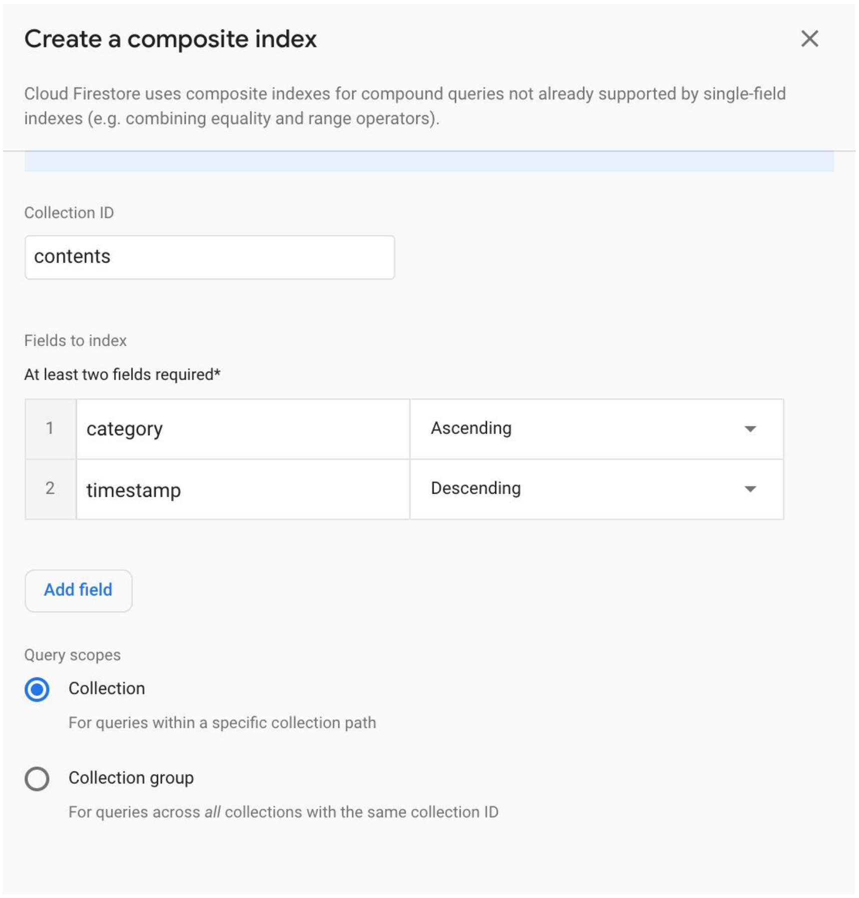
17. Now you have to set up database rules & database indexes.

18. **Database rules setup**: Go to your firebase console > project overview > database > cloud firestore > rules and then edit the rules like this and click publish. That's it.



```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if true;
    }
  }
}
```

19 . **Database Index Setup :** Now go to the right tab (Indexes) of the same page. Click on **Add Index** and then create an index by following the picture below.



Create a composite index

Cloud Firestore uses composite indexes for compound queries not already supported by single-field indexes (e.g. combining equality and range operators).

Collection ID: contents

Fields to index

At least two fields required*

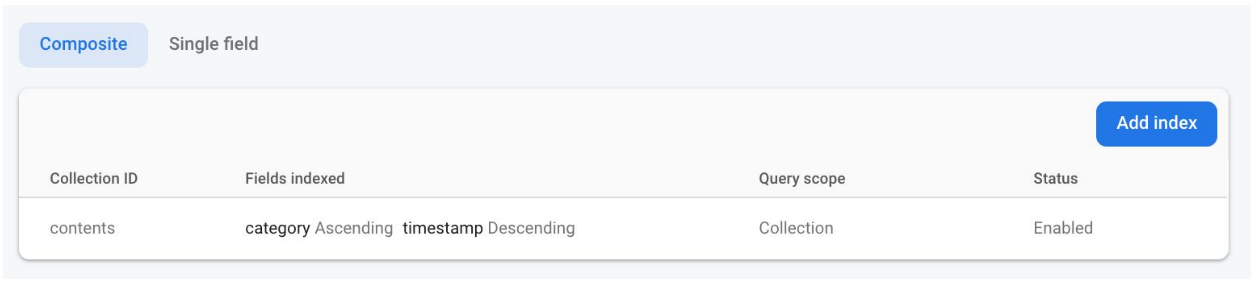
1	category	Ascending
2	timestamp	Descending

Query scopes

[✓ Collection] For queries within a specific collection path

Collection group — For queries across *all* collections with the same collection ID

After setup, the index will look like this picture below.

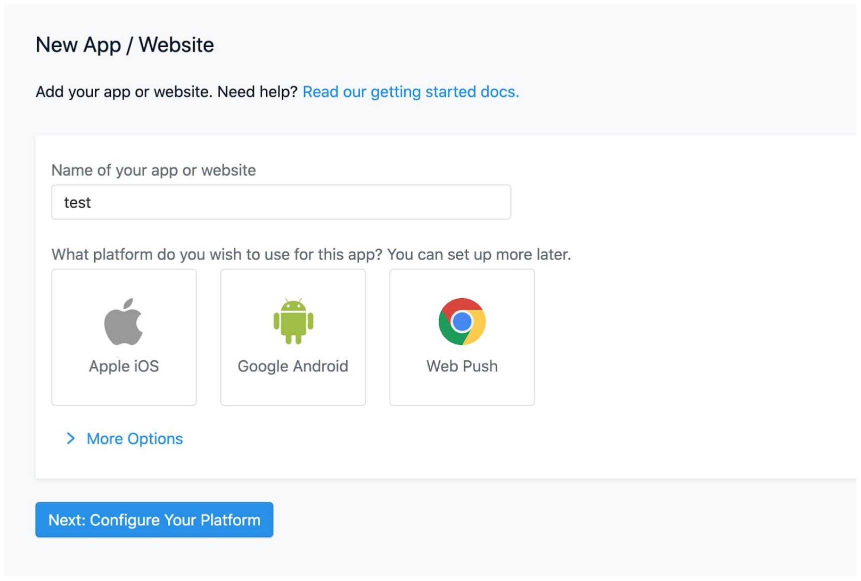


Collection ID	Fields indexed	Query scope	Status
contents	category Ascending timestamp Descending	Collection	Enabled

That’s it. Your database setup is complete.

OneSignal Push Notifications Setup

1. To set up OneSignal Push Notification Service to the app, you need an **App ID** from OneSignal.
2. Go to onesignal.com and create an account and create a project by clicking on the new app/website button.


New App / Website

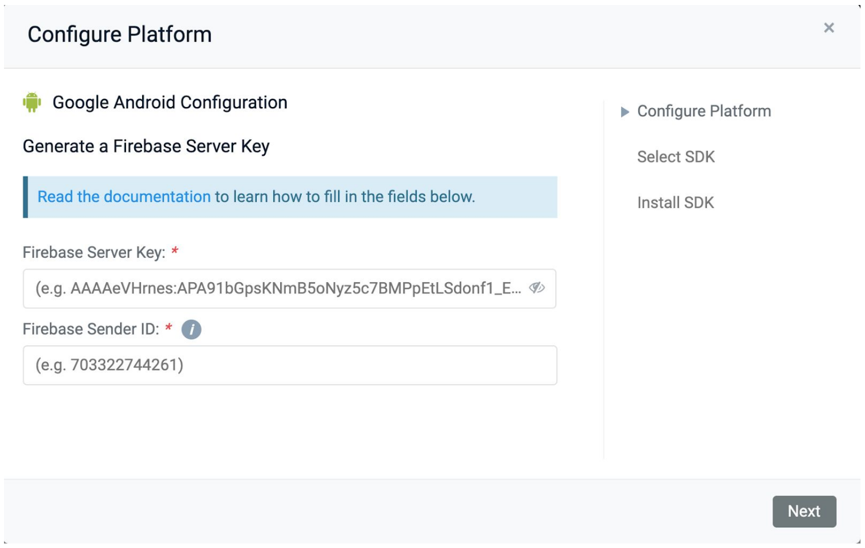
Add your app or website. Need help? Read our getting started docs.

Name of your app or website

test

What platform do you wish to use for this app? You can set up more later.

3. Enter your **app name** and then select **Google Android.**
4. And then you need a **Web API Key** and a **Sender ID**.


Configure Platform

Google Android Configuration

Generate a Firebase Server Key

Read the documentation to learn how to fill in the fields below.

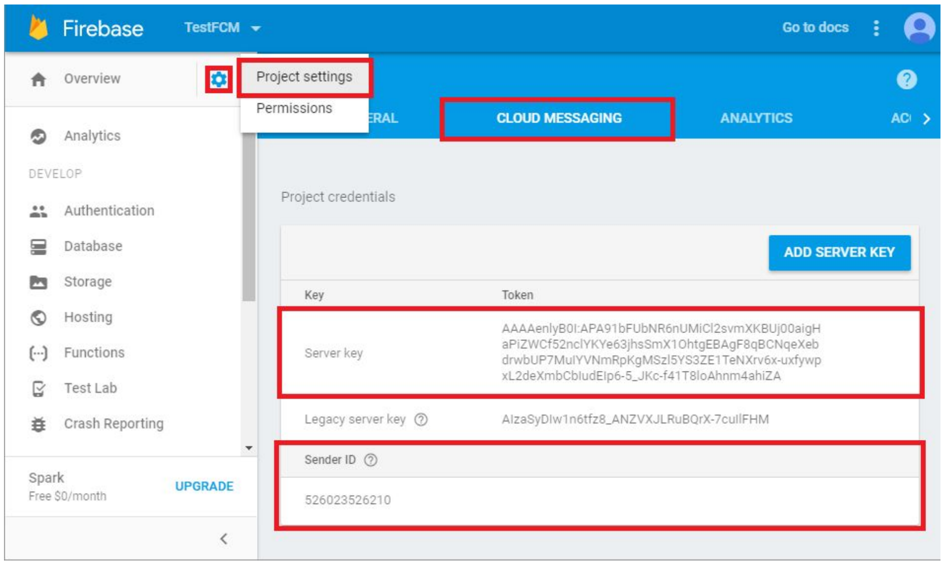
Firebase Server Key: *

(e.g. AAAAeVHrnes:APA91bGpsKNmB5oNyz5c7BMPpEtLSdonf1_E...

Firebase Sender ID: *

(e.g. 703322744261)

5. You will get them from your firebase project. Go to your firebase project > project settings > cloud messaging and copy them and input both **Server Key** and **Sender ID** in the Onesignal and save them and complete the setup.



Project credentials

Key	Token
Server key	AAAAenlyB0I:APA91bFUbNR6nUMiCl2svmXKBUj00aigHaPiZWCf52nclYKYe63jhsSmX1OhtgEBAgF8qBCNqeXebdrwbUP7MulYVNmRpKgMSzl5YS3ZE1TeNXrv6x-uxfywpxL2deXmbCbludElp6-5_JKc-f41T8loAhnm4ahiZA
Legacy server key	AIzaSyDIw1n6tfz8_ANZVXJLRuBQrX-7cullFHM

Sender ID

526023526210

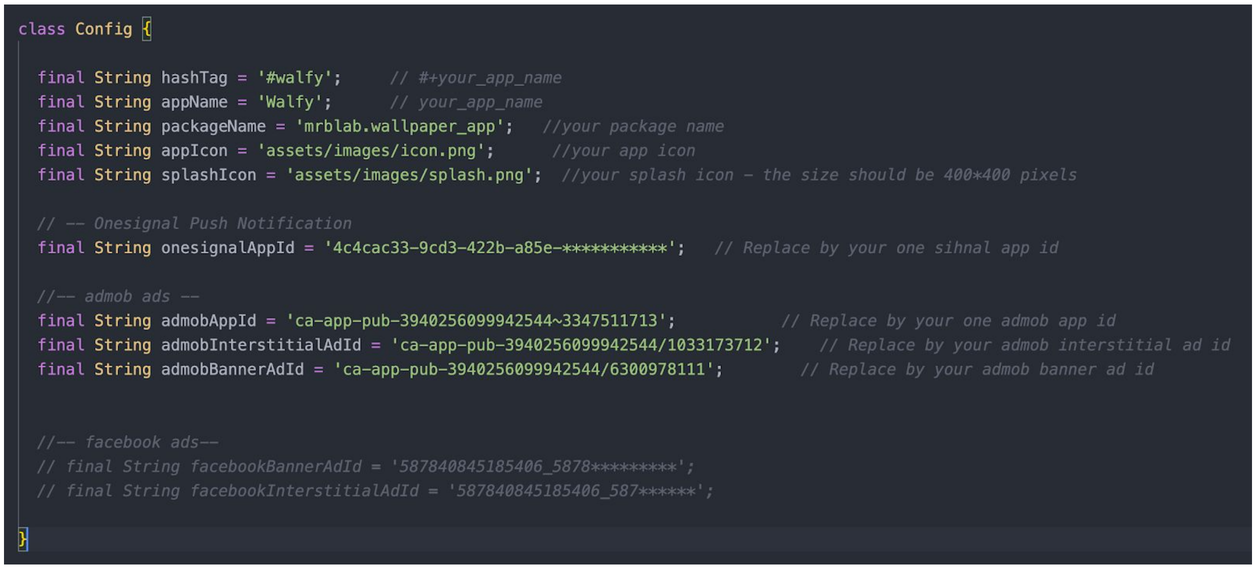
6. You will get an **App ID** from Onesignal. Copy that **App ID** and go to,

   lib/models/config.dart file and replace the Onesignal App ID. That's it. Your OneSignal Push Notification Setup is complete.

Admob & Other Setup

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. Admob ads applied by default. So, if you want to add admob ads, you just have to put your ads id. That's it. 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.

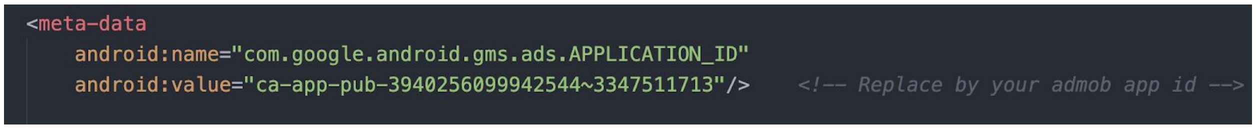
1. Go to **lib/models/config.dart** file and change all of your details.



```dart
class Config {

  final String hashTag = '#walfy';     // #+your_app_name
  final String appName = 'Walfy';      // your_app_name
  final String packageName = 'mrblab.wallpaper_app';   //your package name
  final String appIcon = 'assets/images/icon.png';      //your app icon
  final String splashIcon = 'assets/images/splash.png';  //your splash icon - the size should be 400*400 pixels

  // -- Onesignal Push Notification
  final String onesignalAppId = '4c4cac33-9cd3-422b-a85e-***********';   // Replace by your one sihnal app id

  //-- admob ads --
  final String admobAppId = 'ca-app-pub-3940256099942544~3347511713';           // Replace by your one admob app id
  final String admobInterstitialAdId = 'ca-app-pub-3940256099942544/1033173712';    // Replace by your admob interstitial ad id
  final String admobBannerAdId = 'ca-app-pub-3940256099942544/6300978111';        // Replace by your admob banner ad id


  //-- facebook ads--
  // final String facebookBannerAdId = '587840845185406_5878*********';
  // final String facebookInterstitialAdId = '587840845185406_587******';

}
```

2. Change your **App name, hashtag name, package name, admob app id** and **interstitial ad id**. We have used admob test ids for testing purposes. You can use these too. But make sure you have changed these before publishing your app to the play store.

3. Now go to **android/app/src/main/AndroidManifest.xml** file and replace by your admob app id. You can use this given id for testing purposes. But make sure you have changed these before publishing your app to the play store.



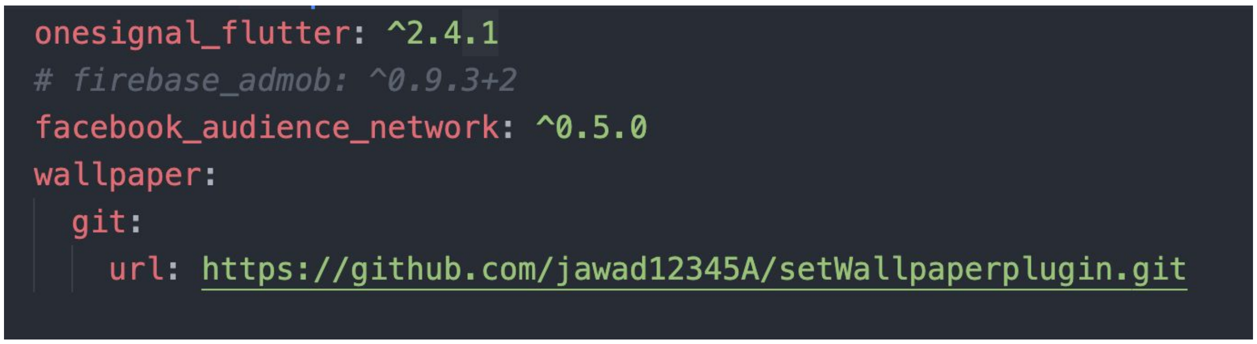
```xml
<meta-data
    android:name="com.google.android.gms.ads.APPLICATION_ID"
    android:value="ca-app-pub-3940256099942544~3347511713"/>    <!-- Replace by your admob app id -->
```

That's it. Your Admob setup is complete.

Facebook Ads Setup

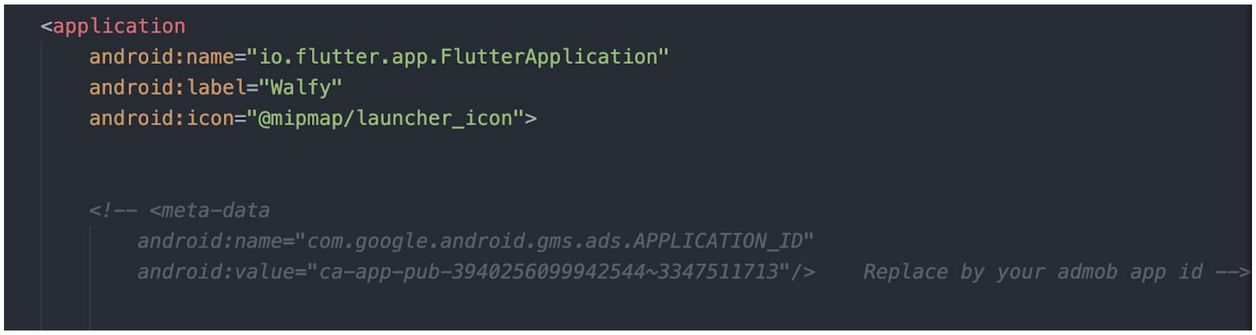
Skip this step if you added admob ads already. Admob ads are added by default in this app. So, Before applying facebook ads, you have to disable admob ads first. To do that,

1. Go to pubspec.yml and disable firebase_admob and enable facebook_audience network. To disable/enable any line or method, just select the line/method and press **cmd + /** (for Mac)**, ctrl + /** (for Windows). And then run **flutter pub get** to the terminal.



```
onesignal_flutter: ^2.4.1
# firebase_admob: ^0.9.3+2
facebook_audience_network: ^0.5.0
wallpaper:
  git:
    url: https://github.com/jawad12345A/setWallpaperplugin.git
```

2. Go to **android/app/src/main/AndroidManifest.xml** and disable the following metadata.

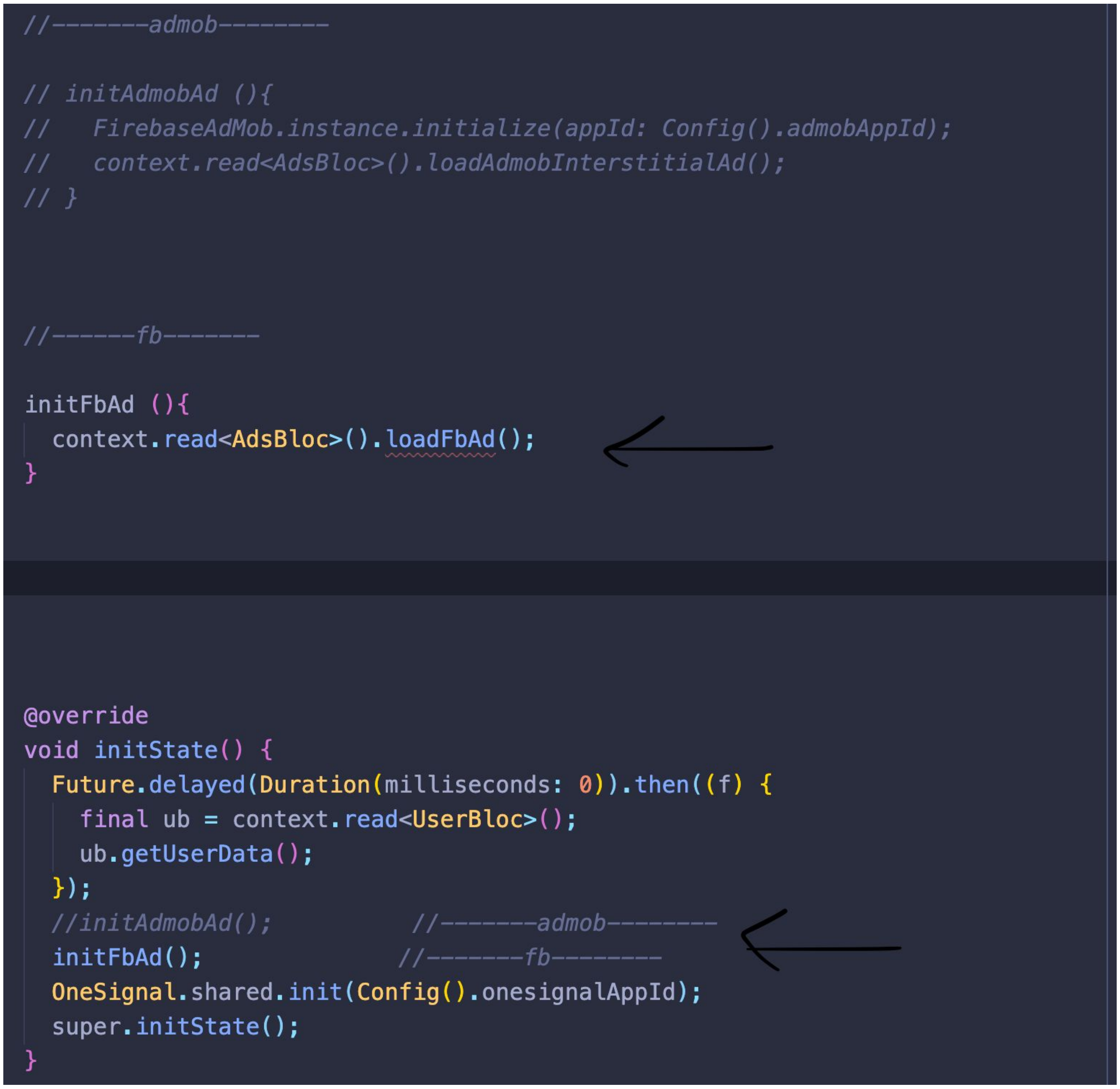


```
<application
    android:name="io.flutter.app.FlutterApplication"
    android:label="Walfy"
    android:icon="@mipmap/launcher_icon">


    <!-- <meta-data
        android:name="com.google.android.gms.ads.APPLICATION_ID"
        android:value="ca-app-pub-3940256099942544~3347511713"/>    Replace by your admob app id -->
```

3. Now go to **lib/blocs/ads_bloc.dart** and disable the whole admob part and enable the whole fb part and then go to the bottom of the page and you will see a dispose method. Here, just disable the **disposeAdmobInterstitialAd()** method and enable the **destroyFbAd()** method.

4. Now go to **lib/models/config.dart** and enable facebook ad section and replace by **ads ids** which you will get from facebook monetization manager.

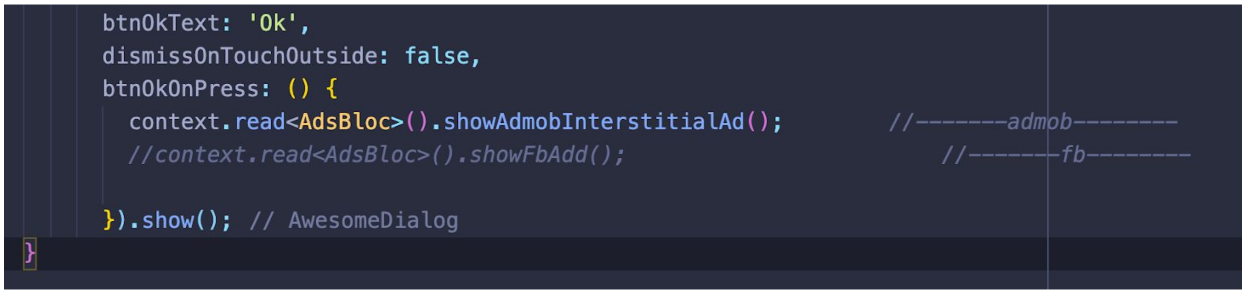
5. Now go to **lib/pages/home.dart** file and disable **initAdmobAd** and enable **initFbAd** in both places.



```
//--------admob---------

// initAdmobAd (){
//   FirebaseAdMob.instance.initialize(appId: Config().admobAppId);
//   context.read<AdsBloc>().loadAdmobInterstitialAd();
// }



//-------fb--------

initFbAd (){
  context.read<AdsBloc>().loadFbAd();
}
```

```
@override
void initState() {
  Future.delayed(Duration(milliseconds: 0)).then((f) {
    final ub = context.read<UserBloc>();
    ub.getUserData();
  });
  //initAdmobAd();           //--------admob---------
  initFbAd();               //--------fb---------
  OneSignal.shared.init(Config().onesignalAppId);
  super.initState();
}
```

6. Now go to **lib/pages/details.dart** file and disable the admob part and enable the fb part.

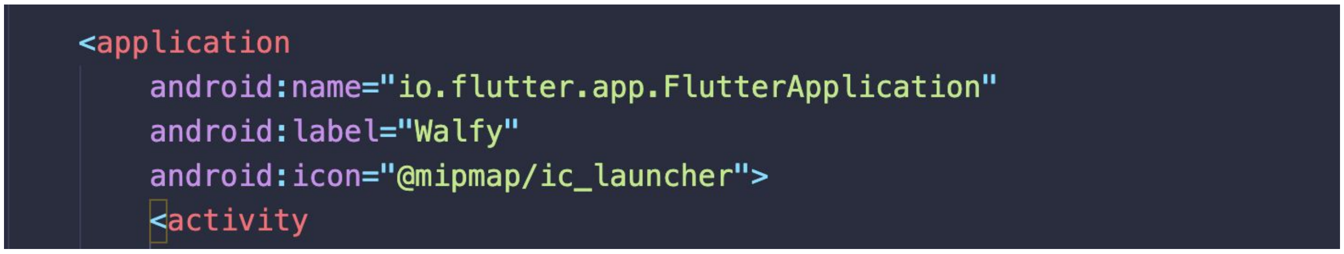


```
      btnOkText: 'Ok',
      dismissOnTouchOutside: false,
      btnOkOnPress: () {
        context.read<AdsBloc>().showAdmobInterstitialAd();        //--------admob---------
        //context.read<AdsBloc>().showFbAdd();                        //--------fb---------

      }).show(); // AwesomeDialog
}
```

That's it. Your Facebook ads setup is complete.

Change App Name

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



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

Change Splash

1. Go to the asset folder and delete the default icon (splash.png).
2. Now upload your splash icon as png in the asset folder and rename it to **splash.png.** (Your splash image should be 400*400 pixels and in png format)
3. Now run the following command on the terminal,

```
Flutter pub get
```

And then,

```
flutter pub pub run flutter_native_splash:create
```

4. That's it. For more info about changing splash icons, visit this site .

Change 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 asset 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.

```
flutter clean
```

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

```
flutter run
```

You will see the preview of the debug version. To get the release version, run the following command :

```
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.

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 envato market. Thanks

MRB Lab

Contact: mrblab24@gmail.com