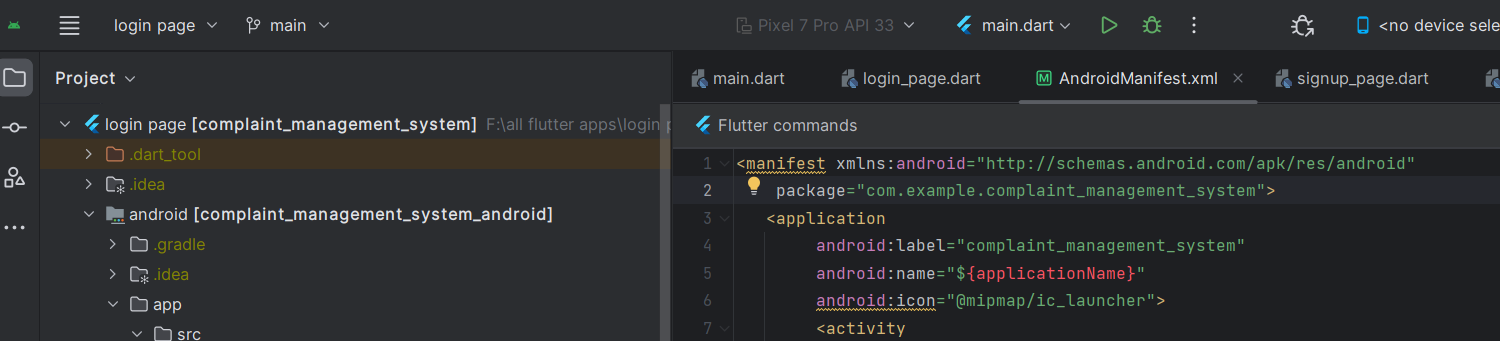
**How we build the login page**

Settings

For android get bundle identifier like below In Android the package name is in the AndroidManifest:



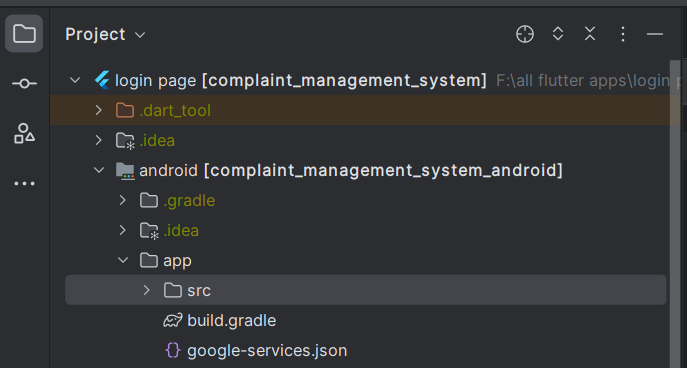
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

...

package="com.example.appname">

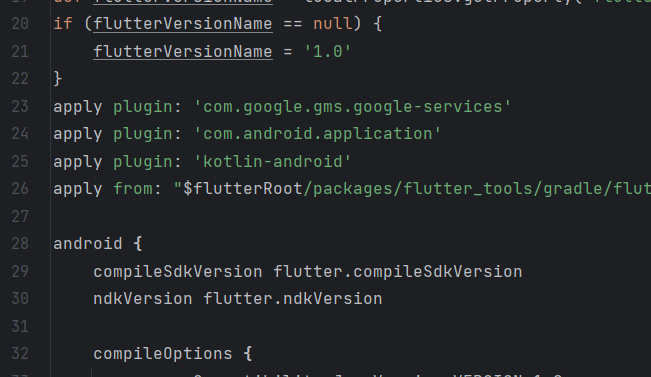
You can change the package name here. It would be the bundle identifier name.

For firebase app, we need several settings. Let's start with Android settings. Go and find android/app/build.gradle file.



In this file we need to add 4 lines here. In the above folder we also add our android setting json file which we get from firebase. It's google-services.json

For the first line add like below

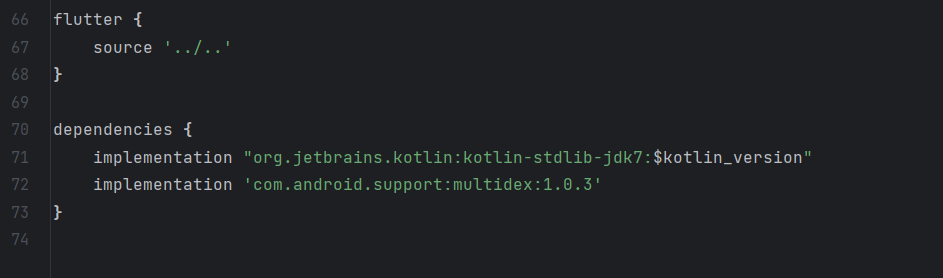


Now add the second and third line

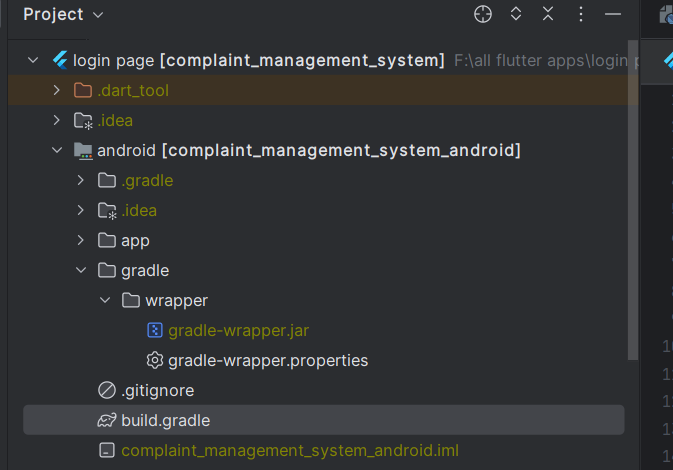
applicationId "com.thecomplaint\_management\_system"  
// You can update the following values to match your application needs.  
// For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-build-configuration.  
minSdkVersion flutter.minSdkVersion  
targetSdkVersion flutter.targetSdkVersion  
versionCode flutterVersionCode.toInteger()  
versionName flutterVersionName  
multiDexEnabled true

If you get error after setting multidex, you remove it. Some sdk throws error after setting this.

Now add the fourth line

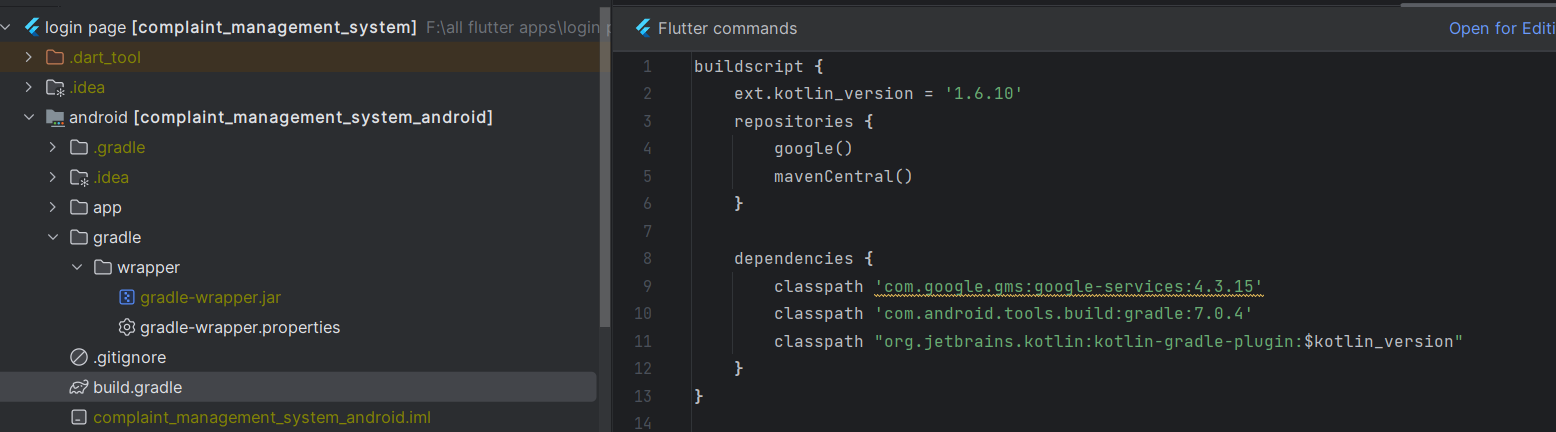


Now we need to add one more line for android setting. That's in android/build.gradle file



Add a line there like below

classpath ‘com.google.gms:google-services:4.3.15’



Getx Authentication

AuthController

We want our AuthController to be availabe thoughout the app. So we will create an AuthController instance that would be accessible from anywhere.

We will create a static instance of the controller

class AuthController extends GetxController{

//with instance AuthController would be able to the app everywhere

static AuthController instance = Get.find();

late Rx<User?> \_user;

FirebaseAuth auth = FirebaseAuth.instance;

......................

}

We created an instance using Get.find() inside the controller, so It will be able everywhere in the app using AuthController.instance

You can initialize your user from firebase and bindStrem with the user, so any changes happen to user, our firebase use would be notified.

@override

void onReady(){

super.onReady();

\_user = Rx<User?>(auth.currentUser);

\_user.bindStream(auth.userChanges());

}

You also need to bind \_user with our app lifecycle. So that, any time user logs in or log out or navigate to a different page, app would know it. This is the actualy auth controller.

For this we will use ever worker function in Getx. This function ever() takes a listener and a callback.  Before you call ever() worker function, you need to call \_user.bindStream(auth.userChanges());

@override

void onReady(){

super.onReady();

\_user = Rx<User?>(auth.currentUser);

\_user.bindStream(auth.userChanges());

//worker function

ever(\_user, initialScreen);

}

[Getx worker function ever()](https://learnflutter.co/getx-worker-in-detail-ever-function/)

With this auth controller would be able to listen to any changes,

Sign in and Sign out Method

Since we have an access to FirebaseAuth instance, we would be able to call the FirebaseAuth properties and fields. We will use

//Sign up or register method

await auth.createUserWithEmailAndPassword(email: email, password: password);

//Sign in or login method

await auth.signInWithEmailAndPassword(email: email, password: password);

Here await and async keywords are quiet important. Otherwise you will get error. See the complete controller

import 'package:firebase\_app/login\_page.dart';

import 'package:firebase\_app/welcome\_page.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:get/get.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

class AuthController extends GetxController{

//AuthController.intsance..

static AuthController instance = Get.find();

//email, password, name...

late Rx<User?> \_user;

FirebaseAuth auth = FirebaseAuth.instance;

@override

void onReady(){

super.onReady();

\_user = Rx<User?>(auth.currentUser);

//our user would be notified

\_user.bindStream(auth.userChanges());

ever(\_user, \_initialScreen);

}

\_initialScreen(User? user){

if(user==null){

print("login page");

Get.offAll(()=>LoginPage());

}else{

Get.offAll(()=>WelcomePage(email:user.email!));

}

}

void register(String email, password)async{

try{

await auth.createUserWithEmailAndPassword(email: email, password: password);

}catch(e){

Get.snackbar("About User", "User message",

backgroundColor: Colors.redAccent,

snackPosition: SnackPosition.BOTTOM,

titleText: Text(

"Account creation failed",

style: TextStyle(

color: Colors.white

),

),

messageText: Text(

e.toString(),

style: TextStyle(

color: Colors.white

)

)

);

}

}

void login(String email, password)async{

try{

await auth.signInWithEmailAndPassword(email: email, password: password);

}catch(e){

Get.snackbar("About Login", "Login message",

backgroundColor: Colors.redAccent,

snackPosition: SnackPosition.BOTTOM,

titleText: Text(

"Login failed",

style: TextStyle(

color: Colors.white

),

),

messageText: Text(

e.toString(),

style: TextStyle(

color: Colors.white

)

)

);

}

}

void logOut()async{

await auth.signOut();

}

}

Injecting Dependencies

You need to inject this AuthController to the app, as we talked early, because this is a dependency for our app. Our app needs some initialization. First we will wait for firebase to be initialized and then we will inject the auth controller. This happens within the main function of our app.

Future<void> main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp().then((value) => Get.put(AuthController()));

runApp(MyApp());

}

Here we inject AuthController using Get.put() . Now auth controller would be available to the app.

View

Now it's time to call register and login method from our view

GestureDetector(

onTap: (){

AuthController.instance.register(emailController.text.trim(), passwordController.text.trim());

}

GestureDetector(

onTap: (){

AuthController.instance.login(emailController.text.trim(), passwordController.text.trim());

}

Firebase app errors and solution

No signature of method error

\* What went wrong: A problem occurred evaluating project ':app'. > No signature of method:

This happended due to multiDexEnabled. You need to set it to true.

multiDexEnabled true .

If you spell is wrong you will get the error as well.

The architecture of the login system consists of 3 separate application pages: the ***Login Page***, the ***Signup Page***, the ***Welcome Page.***

1. ***Login Page*** - the landing page of the application

* The page will have fields for the user to enter their login credentials, which usually consist of their email address and password.

2. ***Sign-up Page*** - allows creation of new users

* If the user doesn't have an account, they will be directed to the sign-up page after clicking on the 'Sign Up' button on the login page.
* Once the user enters their information, they can submit the form to create a new account.
* The sign-up page will ask the user to enter their personal information such as their name, email address, and password.
* The page will also have validation checks to ensure that the email address is valid and the password is at least six characters long.
* If the user already has an account, they can click on the 'Sign In' button on the login page to be directed to the sign-in page.
* The sign-in page will have fields for the user to enter their email address and password to log into their account.

1. ***Welcome Page*** - redirected here when user has successfully logged in

* The user can then proceed further to the Home page or else move back to the sign-in page from this page.