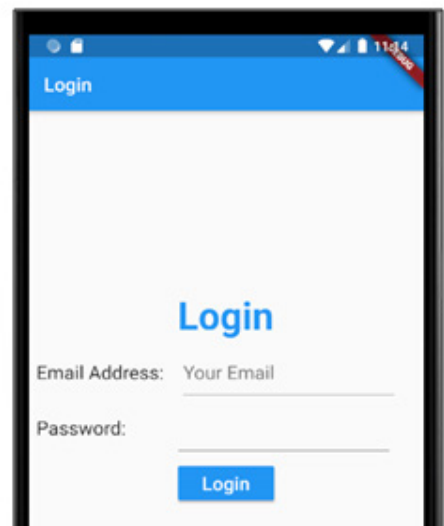
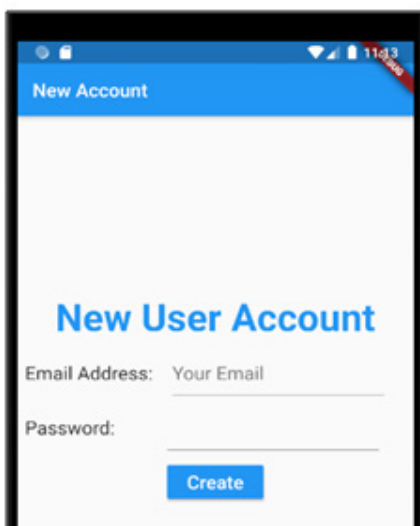
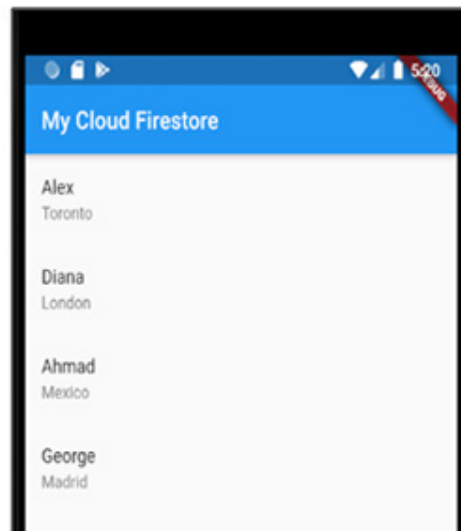


## Lab 9 : Create a User Profile Interface using Firebase

In this lab, you will create a Flutter app that allows the app user to create their accounts (user name & password) to access the app service. You will create an authentication procedure depending on Firebase authentication service.

You will create the startup interface which includes the “**New User Account**” and “**Login**” buttons. When the user taps the “**New User Account**” button he/she will move to the New Account interface which will be used to create the new app user account. Also, if the user taps the **Login** button, he/she can login to the app using the account which he/she has created in the **New Account** interface. To do this, you should configure your Flutter app to use Firebase authentication service.



To achieve the previous lab scenario, perform the following steps:

1- Open **Android Studio**

2- Click **File** → **New** → **New Flutter Project**

3- Select **Flutter Application**, then click **Next**.

4- Type : **lab\_09** for Project Name, and create a new folder : **Lab\_09** for Project Location. Click **Next**.

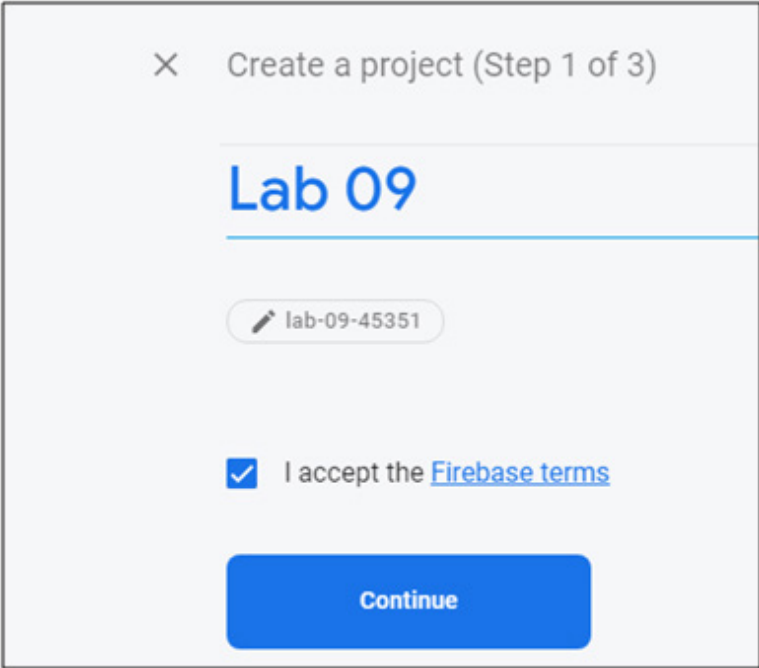
5- Type : **androidatc.com** for Company domain, then click **Finish**

### Configure Your Flutter App to use Firebase Services:

6- Go to : <https://console.firebase.google.com>

7- Sign into Firebase using your Google account.

8- Click **Get Started**, then click **Create a project**. Fill out your project name “**Lab 09**” or any other name as illustrated in the following figure. Check **I accept the Firebase terms**, then click **Continue**.



× Create a project (Step 1 of 3)

Lab 09

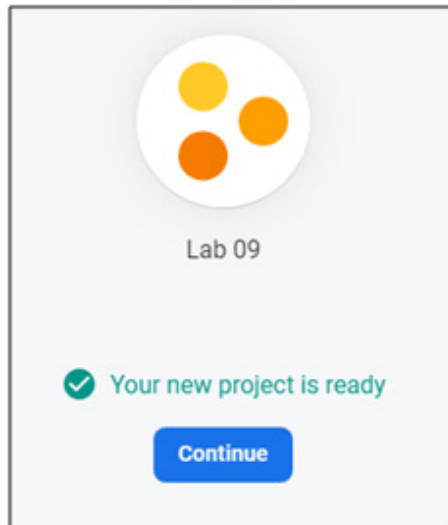
lab-09-45351

☒ I accept the [Firebase terms](#)

Continue

9- Click **Enable Google Analytics for this project** to **disable** this feature because this app is just for testing. Google features are important to get a report about your live apps in the future.

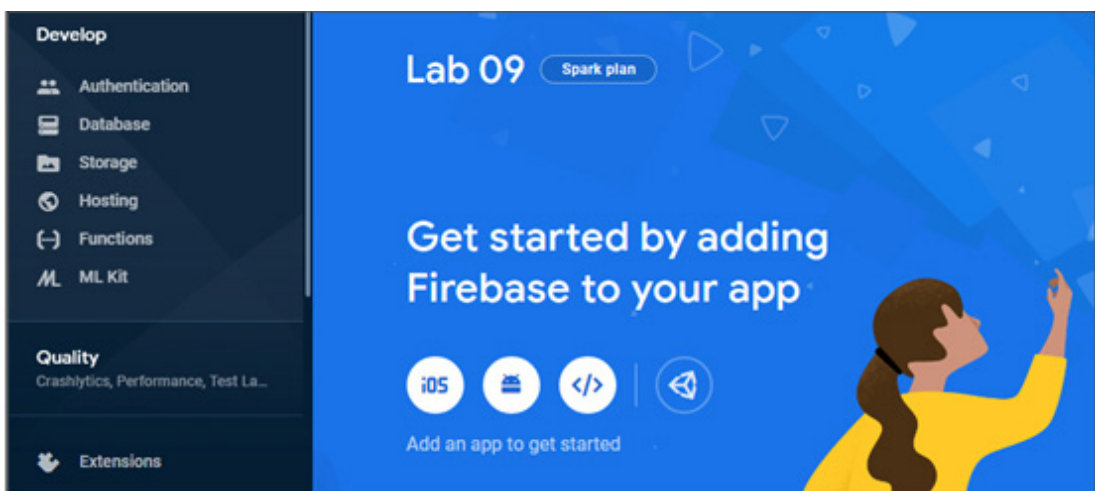
Click **Create Project**. Then within seconds, you should get the following message as illustrated in the following figure. It displays your project that has been created on Google Firebase as illustrated in the following figure. Click **Continue**.



Now, your project has been created on Firebase web site and is ready to configure.

Lesson 9 includes how to add Firebase configuration to iOS and Android files in the Flutter app step by step; however, in this lab you will configure only the Android part only, and if you need more info about iOS, please review the content of the lesson.

10- To add Firebase configurations to your Android files, click on the **Android icon** in the following figure:



11- You should get the following dialog box to register your app. As you will see, you should enter your project name which must be a unique name on Google Play store.

[illegible]

To get your Android project name, go to **Android Studio**, then open the following path:

Project name (**lab\_09\_android**) → **android** → **app** → **build.gradle**

Then, scroll down the file content and as illustrated in the following figure, the **application Id** is : **com.androidatc.lab\_09**

```
defaultConfig {
    // TODO: Specify your own unique Application ID (https://developer.a
    applicationId "com.androidatc.lab_09"
    minSdkVersion 16
    targetSdkVersion 28
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
    testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
}
```

**Copy** this Id and **paste** it on your [Firebase web site](#) for the **Android package name**. Then click the **Register app** button.

12- Click **Download google-service.json** button to download the JSON configuration file. Open your download folder and should find this file name : **google-services.json** without any number.

13- Move this file : **google-services.json** from your download folder to the **app** folder using the drag and drop technique. This **app** folder is in **Android Studio** in the following path:

**Project name (lab\_09) → android(lab\_09\_android) → app**, then click **OK**.

14- Now, return to the Firebase web site to complete the setup steps. Click **Next**.

15- In the Add Firebase SDK step, copy the line illustrated in the following figure or click the copy icon.

```
// Add this line
classpath 'com.google.gms:google-services:4.3.3'
```

16- Go to Android Studio, open the **build.gradle** file which is in the following path :

Your Project name → **android(lab\_09\_android) → build.gradle**

Paste this class path within the dependencies braces as illustrated in the following figure:

```
dependencies {  
    classpath 'com.android.tools.build:gradle:3.5.0'  
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"  
    classpath 'com.google.gms:google-services:4.3.3'  
}
```

17- Now, return to the Firebase web site and from the configuration wizard, copy the other two lines :

```
apply plugin: 'com.android.application'
```

```
apply plugin: 'com.google.gms.google-services'
```

go to Android Studio, open the **build.gradle** file which is in the following path:

Your Project name → **android (lab\_09\_android)** → **app** → **build.gradle**

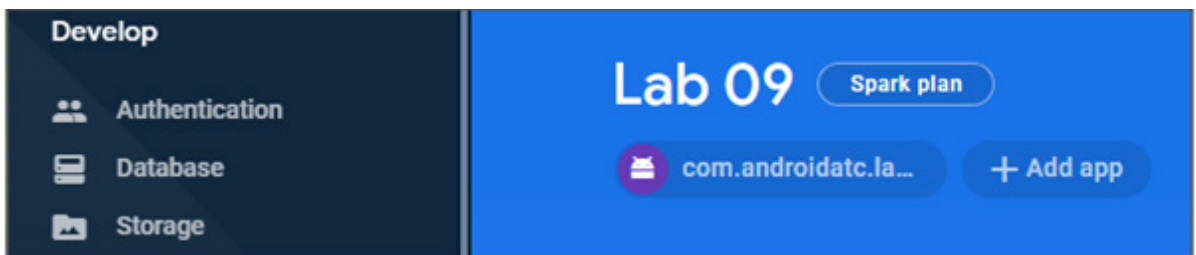
Then, paste the other two lines as separate lines at the end of this file as illustrated in the following figure:

```
dependencies {  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'androidx.test:runner:1.1.1'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'  
}  
  
apply plugin: 'com.android.application'  
apply plugin: 'com.google.gms.google-services'
```

Also, in the same **build.gradle** file add : **multiDexEnabled true** as illustrated in the grey highlighted part of the following configuration:

```
defaultConfig {  
  
    applicationId "com.androidatc.lab_09"  
    minSdkVersion 16  
    targetSdkVersion 28  
    versionCode flutterVersionCode.toInteger()  
    versionName flutterVersionName  
    testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
    multiDexEnabled true  
}
```

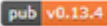
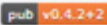
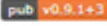
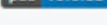

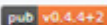


18- Return to the Firebase web site, click **Next**. Then click the **Continue to console** button. You should get the following figure:



19- Now, close all your Gradle files. Be sure that your Android emulator is selected. Stop your app, then run it again to be sure that all changes in your Android Studio Gradle files were applied.

20- The Firebase settings and configurations are always updated. Therefore, to be up to date, you should follow the last setting in the following web link: <https://github.com/FirebaseExtended/flutterfire>

21- Scroll down until you get the following figure:

Available FlutterFire plugins			
Plugin	Version	Firebase feature	Source code
<a href="#">cloud_firestore</a>		Cloud Firestore	<a href="#">cloud_firestore</a>
<a href="#">cloud_functions</a>		Cloud Functions	<a href="#">cloud_functions</a>
<a href="#">firebase_admob</a>		Firebase AdMob	<a href="#">firebase_admob</a>
<a href="#">firebase_analytics</a>		Firebase Analytics	<a href="#">firebase_analytics</a>
<a href="#">firebase_auth</a>		Firebase Authentication	<a href="#">firebase_auth</a>
<a href="#">firebase_core</a>		Firebase Core	<a href="#">firebase_core</a>
<a href="#">firebase_crashlytics</a>		Firebase Crashlytics	<a href="#">firebase_crashlytics</a>
<a href="#">firebase_database</a>		Firebase Realtime Database	<a href="#">firebase_database</a>

22- Now, you should configure your app settings or add Firebase plug-in services to your app by configuring: **pubspec.yaml** file for Firebase authentication and database.

Click the **cloud\_firestore** plug-in , click the **Installing** tab and copy the dependencies value : `cloud_firestore: ^0.13.4` or the latest update value which you will find at the time you perform this lab. The following figure displays the current `cloud_firestore` configuration:

Readme Changelog Example **Installing** Versions 100

## Use this package as a library

### 1. Depend on it

Add this to your package's pubspec.yaml file:

```
dependencies:  
  cloud_firestore: ^0.13.4
```



23- Open your **pubspec.yaml** file in your Android Studio and paste this value under dependencies.

24- Click **Back** on your web browser toolbar to get the “**Available FlutterFire plugins:**” web page again or go to the web link: <https://github.com/FirebaseExtended/flutterfire> , then click the : **firebase\_auth** , click the **Installing** tab , then copy the existing dependencies value: **firebase\_auth: ^0.15.4**

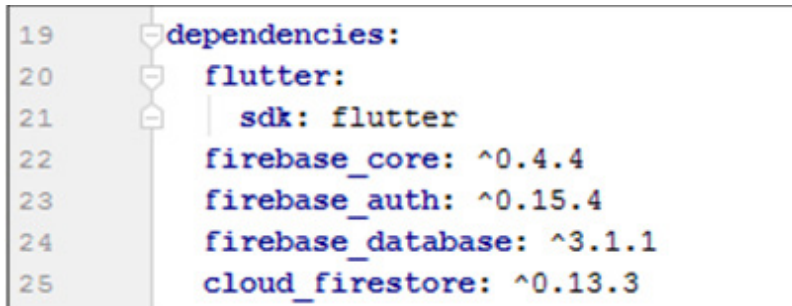
25- Paste this value in your **pubspec.yaml** file under the dependencies.

26- Back on your web browser to the “**Available FlutterFire plugins:**” web page or the web link: <https://github.com/FirebaseExtended/flutterfire>, then click the : **firebase\_database**

27- Click the **Installing** tab, then copy the existing dependencies value:

**firebase\_database: ^3.1.3**

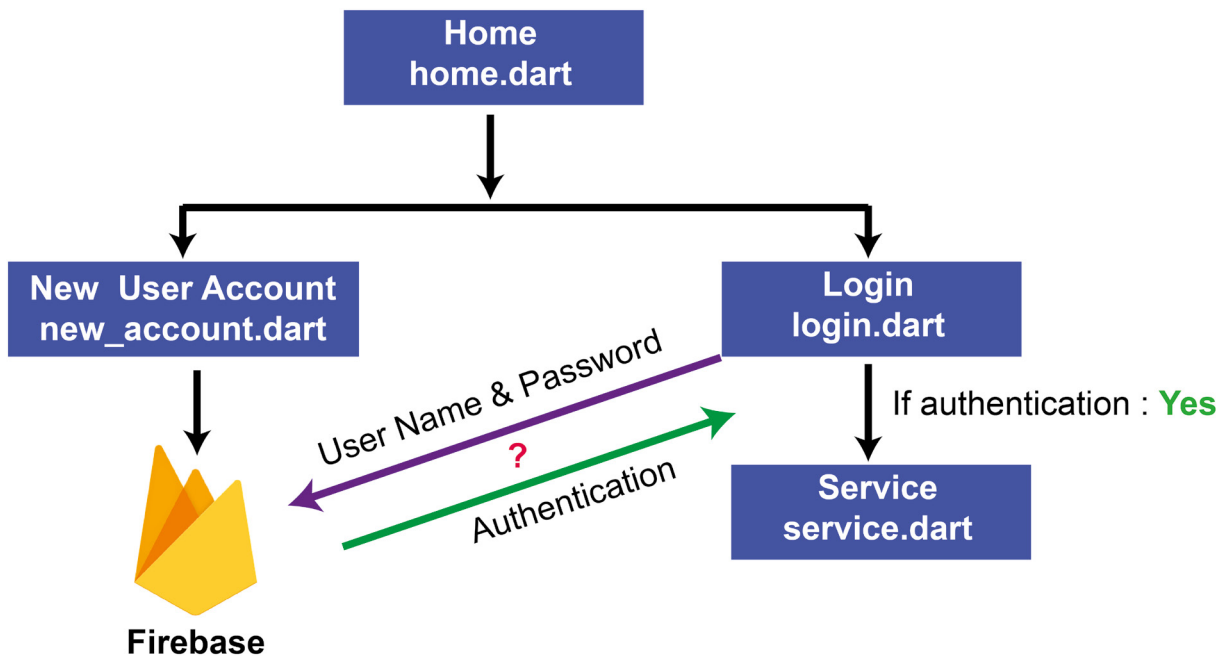
28- Paste this value in your **pubspec.yaml** file under the dependencies. The dependencies of the **pubspec.yaml** file should be as illustrated in the following figure:

A screenshot of a code editor showing the dependencies section of a pubspec.yaml file. The code is as follows:

```
19 dependencies:
20   flutter:
21     sdk: flutter
22   firebase_core: ^0.4.4
23   firebase_auth: ^0.15.4
24   firebase_database: ^3.1.1
25   cloud_firestore: ^0.13.3
```

29- Now, at the top of the **pubspec.yaml** file content, click **Packages Get** to incorporate all of those settings into your Flutter project.

Now, you should create the Flutter app structure as illustrated in the following figure, where the startup interface will be **home.dart** which includes two buttons: The first button (**New User Account**), will forward the app user to (**new\_account.dart**) to create a new user account at Firebase web site. The second button (Login) will forward the app user (**login.dart**) to login using the user name (email address) and password which the app user has created in the “**creating new user**” step. If the user can login successfully, he/she will login to the user profile (**user\_profile.dart**) interface to complete his/her profile by adding an image to his/her profile, and when the app user clicks Save, all new information along with the image will upload to the cloud Firestore database.



20- You should create three new interfaces (Dart files). Right click the **lib** folder and select **New** → **Dart File** . Type **home** for the file name , and then press **Enter** (or Return for Mac).

21- Repeat the previous step to create other three dart files: **new\_account.dart**, **login.dart**, and **user\_profile.dart**

22- Open the **new\_account.dart** file and create the interface which is illustrated in the figure below. This interface contains two text fields to ask the app user to enter his/her username and password. Then click the **Create** button.

The screenshot shows a 'New User Account' form. It has a title 'New User Account' in blue. Below the title, there are two text input fields: 'Email Address: Your Email' and 'Password:'. At the bottom of the form, there is a blue button labeled 'Create'.

The code of **new\_account.dart** will create this interface as follows:

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';

class NewAccount extends StatefulWidget {
  @override
  _NewAccountState createState() => _NewAccountState();
}

class _NewAccountState extends State<NewAccount> {
  String email;
  String password;
  final FirebaseAuth _auth = FirebaseAuth.instance;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('New Account '),
        ),

        body: Padding(
          padding: const EdgeInsets.all(8.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              ListView(
                shrinkWrap: true,
                children: <Widget>[
                  Container(
                    alignment: Alignment.center,
                    child: Text(
                      'New User Account',
                      style: TextStyle(fontSize: 40,
                        color: Colors.blue,
                        fontWeight: FontWeight.bold,)),
                ),
              SizedBox(height: 10.0,)),
            ],
          ),
        ),
      ),
    );
  }
}
```

```
// ***** User Name *****

Row(
  children: [
    Text('Email Address:',
      style: TextStyle(fontSize: 20.0),),
    SizedBox(width: 20.0),

    SizedBox(width: 220.0,

      child: TextField(
        onChanged: (value1) {
          email = value1;},

        style: TextStyle(fontSize: 20, color: Colors.blue),
        keyboardType: TextInputType.emailAddress,
        autocorrect: false,
        cursorColor: Colors.red,
        decoration: InputDecoration(hintText: 'Your Email',),
      ),
    ],
),
SizedBox(height: 10.0,),

// ***** Password *****

Row(
  children: [
    Text('Password:      ',
      style: TextStyle(fontSize: 20.0),),

    SizedBox(width: 20.0,),
    SizedBox(width: 220.0,

      child: TextField(
        onChanged: (value2) {
          password = value2;},

        style: TextStyle(fontSize: 20, color: Colors.blue),
        textInputAction: TextInputAction.done,
```

```

        autocorrect: false,
      ),
    ),
  ],
),

    SizedBox(height: 10.0,),

// ***** Create Button *****
    Center(
      child: Container(
        width: 100,
        child: RaisedButton(
          color: Colors.blue,
          child: Text("Create",
            style: TextStyle(fontSize: 20, color: Colors.white)),),
        onPressed: () async {
          try {
            final newUser =await _auth.createUserWithEmailAndPassword(
              email: email, password: password);

            if (newUser != null) {
              Navigator.pushNamed(context, 'Home');}}

            catch (e) {
              print(e);}
          },
        ),
      ),
    ),
  ],
),
],
),
),
),
);
}
}

```

23- Open **login.dart** file and add the code which will create a login interface. The app user will use this interface to login using the user name and password which he/she has created in the **create interface** step. The login interface should be as illustrated in the following figure:

The full code of the **login.dart** follows:

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';

class Login extends StatefulWidget {
  @override
  _Login createState() => _Login();
}

class _Login extends State<Login> {
  String email;
  String password;
  final FirebaseAuth _auth = FirebaseAuth.instance;

  TextEditingController _controller = new TextEditingController();
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Login'),
        ),
        body: Padding(
```

```
padding: const EdgeInsets.all(8.0),
child: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    ListView(
      shrinkWrap: true,
      children: <Widget>[
        Container(
          alignment: Alignment.center,
          child: Text(
            'Login',
            style: TextStyle(
              fontSize: 40,
              color: Colors.blue,
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
        SizedBox(height: 10.0,),

        // ***** User Name *****
        Row(
          children: [
            Text('Email Address:',
              style: TextStyle(fontSize: 20.0)),
            SizedBox(width: 20.0),
            SizedBox(width: 220.0,
              child: TextField(
                onChanged: (value1) {
                  email = value1;},
                style: TextStyle(fontSize: 20, color: Colors.blue),
                keyboardType: TextInputType.emailAddress,
                textInputAction: TextInputAction.done,
                autocorrect: false,
                cursorColor: Colors.red,
                decoration: InputDecoration(hintText: 'Your Email'),
              ),
            ),
```

```

    ),
  ],
),

  SizedBox(height: 10.0,),

// ***** Password *****
  Row(
    children: [
      Text('Password: ',
        style: TextStyle(fontSize: 20.0),),

      SizedBox(width: 20.0,),

      SizedBox(width: 220.0,
        child: TextField(
          controller: _controller,
          onChanged: (value2) {
            password = value2;},
          style: TextStyle(fontSize: 20, color: Colors.blue),
          textInputAction: TextInputAction.done,
          autocorrect: false,
        ),
      ),
    ],
  ),

  SizedBox(height: 10.0,),

// ***** Login Button *****
  Center(
    child: Container(
      width: 100,
      child: RaisedButton(
        color: Colors.blue,
        child: Text("Login",

style: TextStyle(fontSize: 20, color: Colors.white),),

        onPressed: () async {
          try {

```



```

        final User = await _auth.signInWithEmailAndPassword(
            email: email, password: password);
        if (User != null) {
            Navigator.pushNamed(context, 'Service');
            _controller.clear();
        }
    }
    catch (e) {
        print(e);
    }
  },
),
),
),
],
),
],
),
),
),
);
}
}

```

Now, you should design the interface which includes the services that the user logged in for. Remember, the purpose of this lab is to practice configuring login and logout of an app interface. Here, we will not focus on the services the user will find after login because there are many services that can be added here, such as creating a user profile, shopping, payment, subscription and others.

24- To create this app interface, open the **user\_profile.dart** file and type the following code:

```

import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';

class Profile extends StatefulWidget {
  @override
  _ProfileState createState() => _ProfileState();
}

```

```
class _ProfileState extends State<Profile> {
  final FirebaseAuth auth = FirebaseAuth.instance;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('User Profile')),
        actions: <Widget>[

          // action button

          IconButton(
            icon: Icon(Icons.exit_to_app),
            onPressed: () {
              auth.signOut();
              Navigator.pop(context);},),],

        body: Padding(
          padding: const EdgeInsets.all(8.0),
          child: Column(
            children: <Widget>[
              Center(
                child: Text('Welcome to Android ATC',
                  style: TextStyle(
                    fontSize: 20.0,),),),
              SizedBox(height: 20.0),

            ],
          ),
        ),
      );
  }
}
```

25- Open **home.dart** file and add the following code which creates the startup interface, which has the navigation buttons to all app interfaces:

```
import 'package:flutter/material.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Firebase Authentication'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              RaisedButton(
                color: Colors.blue,
                child: Text('New User Account',
                  style: TextStyle(fontSize: 20, color: Colors.white),
                ),
                onPressed: () {
                  Navigator.pushNamed(context, 'NewAccount');
                },
              ),
              SizedBox(height: 40.0,),
              RaisedButton(
                color: Colors.blue,
                child: Text('Login',
                  style: TextStyle(fontSize: 20, color: Colors.white),
                ),
                onPressed: () {
                  Navigator.pushNamed(context, 'Login');
                },
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

```

        ),
    ),
);
}
}

```

25- Open **main.dart** file, **delete** all the code and add the following code which makes the **home.dart** file interface the start up interface when you run this app. This file includes all the route navigation:

```

import 'login.dart';
import 'new_account.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'home.dart';
import 'user_profile.dart';

void main() => runApp(MyApp());

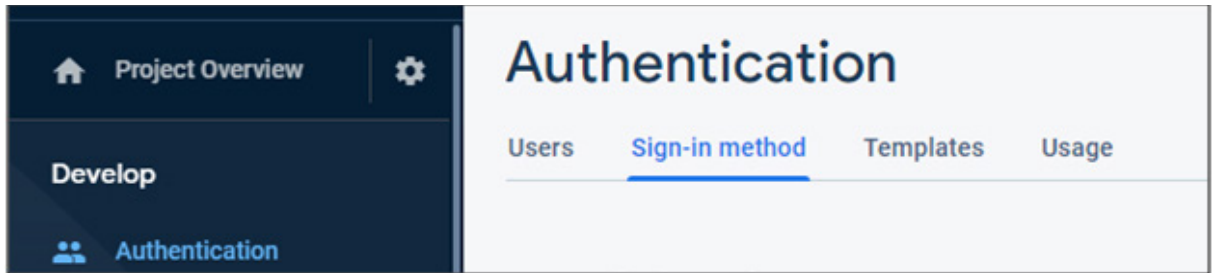
class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Home(),

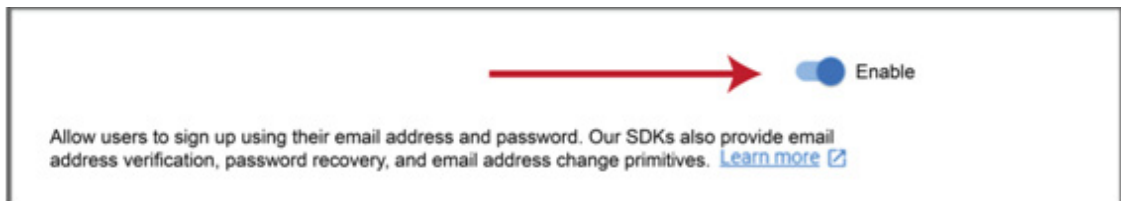
      routes: {
        'Login': (context) => Login(),
        'NewAccount': (context) => NewAccount(),
        'Profile': (context) => Profile(),
        'Home': (context) => Home(),
      },
    );
  }
}

```

26- Before creating any user account on Firebase, you should open your Firebase web site again and on the left side of your web browser under the Develop console, click **Authentication**. Then click the **Sign-in method** tab as illustrated in the following figure:



27 - Click **Email/Password**, click the **Enable** switch button, then click **Save** as illustrated in the following figure:



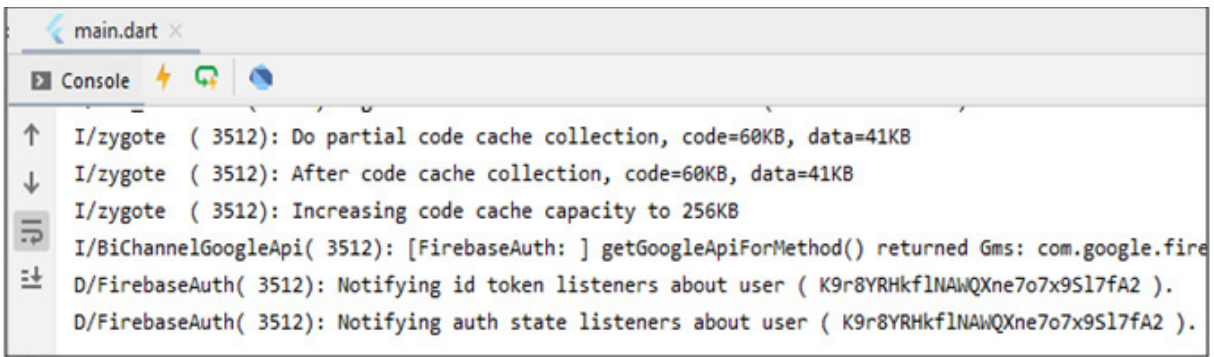
28- Stop your emulator, click **Run** your app, then click : **New User Account** button to move to the **new\_account.dart** interface.

**Important Note:** In the login or create the user account interface, make sure not to add a space before or after your email address. Also, don't press the **Tab** key to move from the email address text field to the password login since there will be an extra space (extra character) added to your email address. This extra character will result in an error message in your Android Studio run console such as : (ERROR\_INVALID\_EMAIL, the email address is badly formatted., null).

Type your email address and any password (at least 6 characters), then tap the **Create** button as illustrated in the following figure:

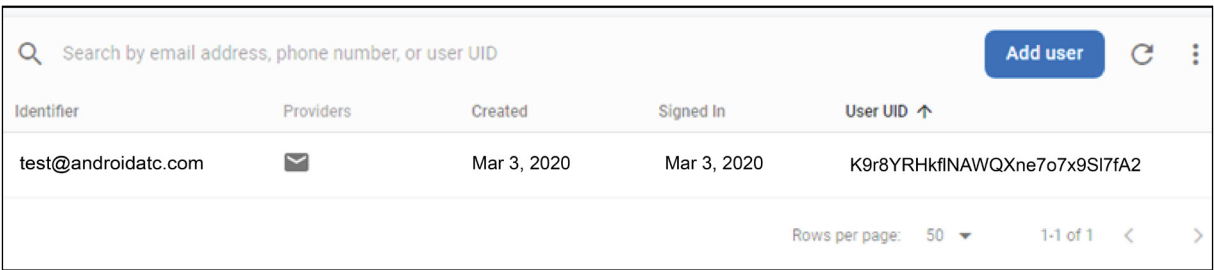
 The image shows a 'New User Account' form. It has a title 'New User Account' in blue. Below the title are two input fields: 'Email Address:' with the value 'test@androidatc.com' and 'Password:' with the value '123456'. At the bottom of the form is a blue button labeled 'Create'.

You should get the following result in the Run console:

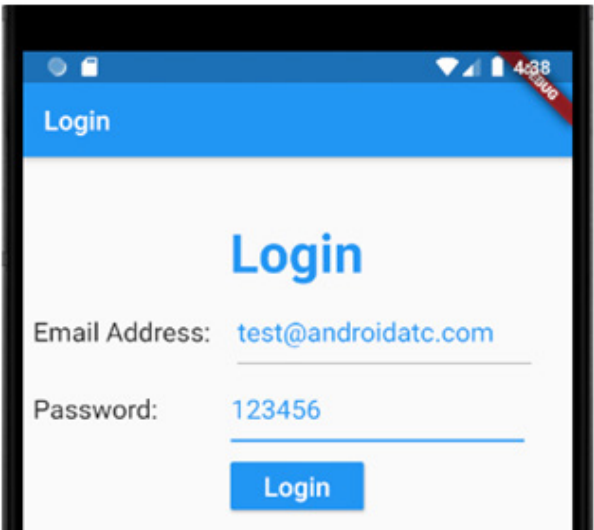


29- Back to your Firebase web site , and in the **Authentication** part, click the **Users** tab where you should find that there is a new user account that has been created as illustrated in the following figure:

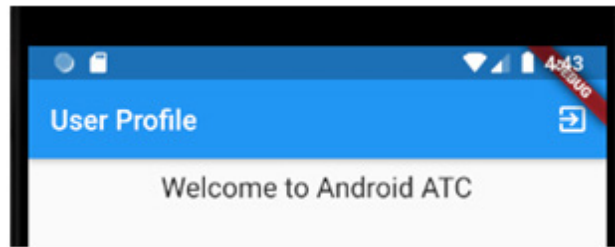
Click **Refresh** if you did not find a new user account.



30- Tap the **Login** button. Enter the user name (email) and the password for the account which you have created in the previous step (without pressing Tab key when you move from the email address Text Field to the Password Text Field), then tap the **Login** button as illustrated in the following figure:



You will get to the content of the **user\_profile.dart** file as illustrated in the following figure:



31- In the right corner of the title bar of the **user\_profile.dart** interface, tap the exit icon to logout of this interface. Then you will go back to the Login interface.

This is the end of the lab. However, please note the following:

I- The explanation of all the previous steps is available in the example of this lesson.

II- Try to complete this lab by adding an extra part to create a user profile interface. For example, try to add an image to the Firebase storage or add the user information such as full name and address to the cloud Firestore.

III- You may use the lab source files to import these lab files to your Android Studio IDE, and test how this lab works, but you should configure your Firebase web account to be compatible with this lab code. Also, add your Firebase **google-services.json** file to your Android Studio app folder instead of the existing file.