

COM413 – Mobile App Development – Workbook Practical 2

Introduction

In weeks 2/3 we worked on creating simple UIs within applications and adding functionality. We created an app which had more than one activity that had a multi-functioning interface: web search, phone call and SMS messaging. So now to move on from where we left off.

We will now create a login/register page and use SQLite to record the users and their data, so that they can log into the app.

Basic:

Create a login page, which will also allow users to register:

- Must use SQLite for this
- View the data using DB Browser

We will now create a login page with a separate Register Activity. Again, we will use SQLite to record the users and their data. But we will use the register button as an OnClickListener to start a new activity called Register, this will open the register layout.

Advanced:

Connecting your app to Firebase

There is a lot of information regarding connecting Firebase to an android app, I will be going over this in the lecture. If you can set up Firebase and connect it to a new project, I will discuss how to use authentication on Friday and next week's Practical we will set-up Firebase Authentication.

Additional:

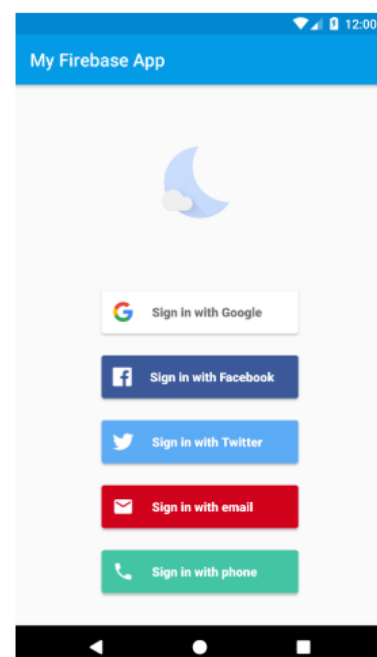
Connecting your Firebase to other login types.

The advanced functionality is to use Twitter or Facebook to Log in to your app.

Easily add sign-in to your Android app with FirebaseUI

[FirebaseUI](#) is a library built on top of the Firebase Authentication SDK that provides drop-in UI flows for use in your app. FirebaseUI provides the following benefits:

- **Multiple Providers** - sign-in flows for email/password, email link, phone authentication, Google Sign-In, Facebook Login, Twitter Login, and GitHub Login.
- **Account Management** - flows to handle account management tasks, such as account creation and password resets.
- **Account Linking** - flows to safely link user accounts across identity providers.
- **Anonymous User Upgrading** - flows to safely upgrade anonymous users.
- **Custom Themes** - customize the look of FirebaseUI to match your app. Also, because FirebaseUI is open source, you can fork the project and customize it exactly to your needs.
- **Smart Lock for Passwords** - automatic integration with [Smart Lock for Passwords](#) for fast cross-device sign-in.



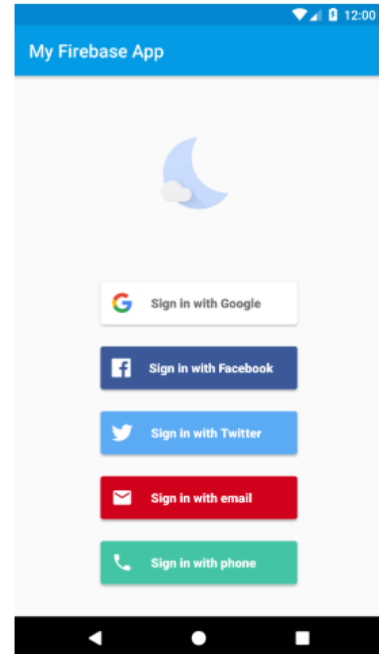
Advanced Functionality:

Firebase

Easily add sign-in to your Android app with FirebaseUI

[FirebaseUI](#) is a library built on top of the Firebase Authentication SDK that provides drop-in UI flows for use in your app. FirebaseUI provides the following benefits:

- **Multiple Providers** - sign-in flows for email/password, email link, phone authentication, Google Sign-In, Facebook Login, Twitter Login, and GitHub Login.
- **Account Management** - flows to handle account management tasks, such as account creation and password resets.
- **Account Linking** - flows to safely link user accounts across identity providers.
- **Anonymous User Upgrading** - flows to safely upgrade anonymous users.
- **Custom Themes** - customize the look of FirebaseUI to match your app. Also, because FirebaseUI is open source, you can fork the project and customize it exactly to your needs.
- **Smart Lock for Passwords** - automatic integration with [Smart Lock for Passwords](#) for fast cross-device sign-in.



Firecasts

Laurence Moroney presents:

Getting started with Authentication on Android



So, what is Firebase:

Firebase allows us to create a database and attach it to our project:

Database Cloud Firestore

Data Rules Indexes Usage

testfirebase-7d239

Users

001

+ Start collection

+ Add document

+ Start collection

Users >

001 >

+ Add field

002

Email: "gm@gmail.com"
Name: "George"
Password: "12345"

Cloud Firestore location: eur3 (europe-west)

It also allows us to create authentication with emails:

Authentication












Users Sign-in method Templates Usage

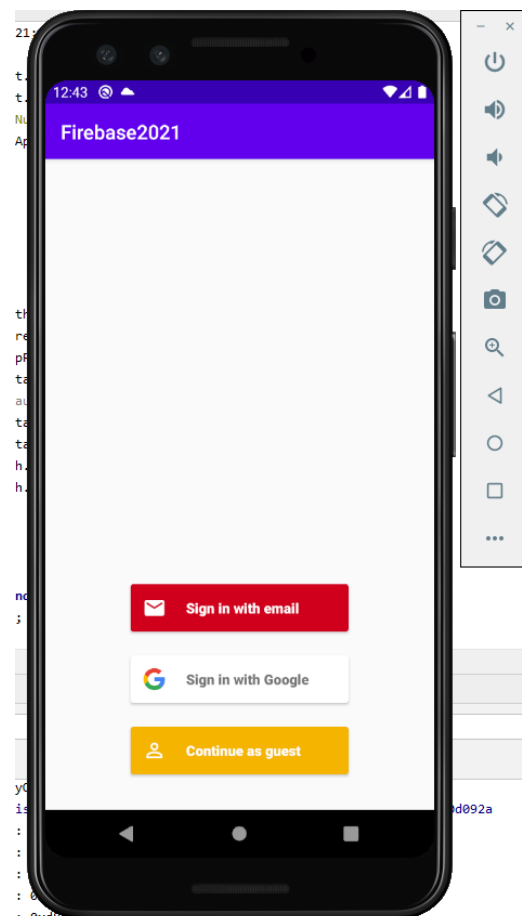
Search by email address, phone number, or user UID

Add user

Identifier	Providers	Created	Signed In	User UID ↑
martin@email.com	📧	Oct 14, 2019	Oct 14, 2019	D5h6JSaHnwQjwJR7e9w9CJALN...
george@email.com	📧	Oct 14, 2019	Oct 15, 2019	dEh7G50rsHXwRzY6ZHa7ySrQms2

Or use other accounts such as Gmail:

Sign-in providers	
Provider	Status
 Email/Password	Enabled
 Phone	Disabled
 Google	Enabled
 Play Games	Disabled
 Game Center Beta	Disabled
 Facebook	Disabled
 Twitter	Disabled
 GitHub	Disabled
 Yahoo	Disabled
 Microsoft	Disabled
 Anonymous	Disabled



First of all go to **Firebase** and use your gmail account to log in.

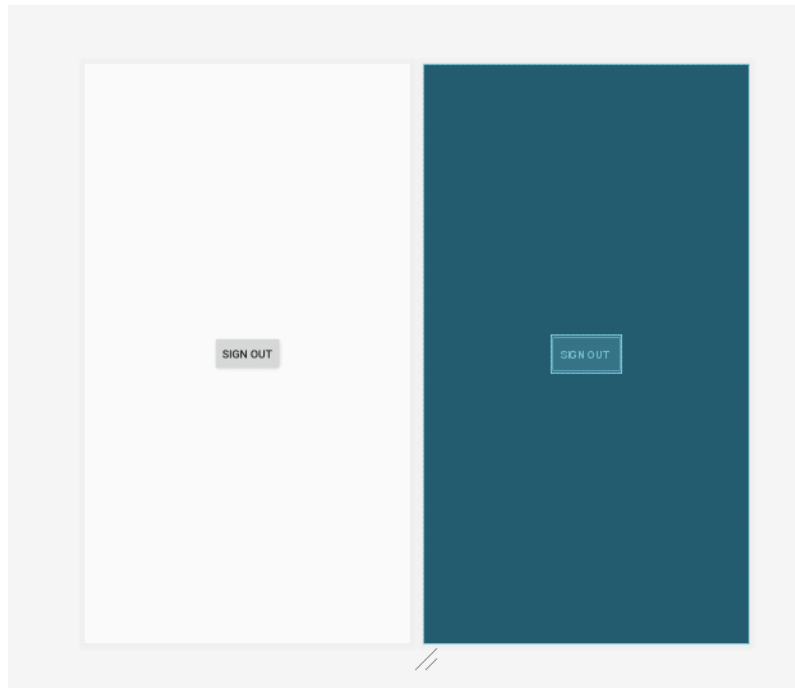
If you don't have a gmail, create one.

We will create our Firebase through Android, this seems to be the best way.

So, getting started with Firebase:

First start with a new project and choose Empty Activity.

We will add a button named btnSignout on the XML layout to the Android project like this:

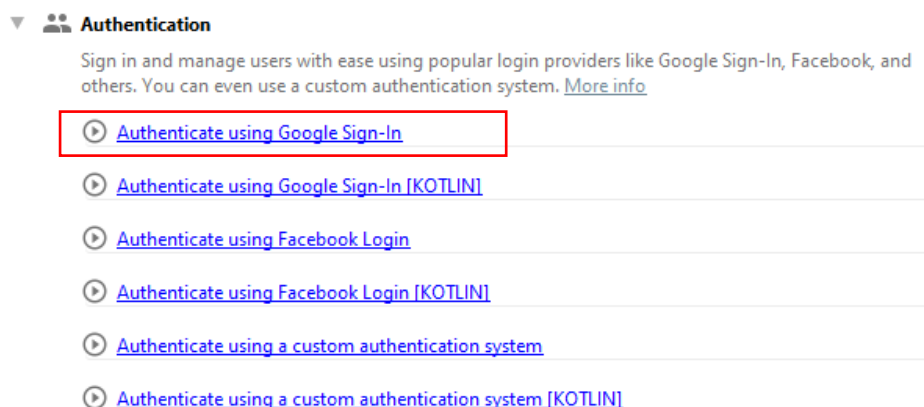


Next, we will create a Firebase account.

To do this, go to **Tools** and click **Firebase**:

You will open the following:

Open **Authentication** -> **Authenticate using Google Sign-In**:



Email and password authentication

You can use Firebase Authentication to let your users sign in with their email addresses and passwords to your app's password-based accounts. This tutorial helps you set up an email and password system and then learn more about the user.

[Launch in browser](#)

① Connect your app to Firebase

Connect to Firebase

② Add Firebase Authentication to your app

Add Firebase Authentication to your app

To use an authentication provider, you need to enable it in the [Firebase console](#). Go to the Sign in with Email/Password section to enable Email/Password sign-in and any other identity providers.

And Click connect to **Firebase**:

This window will open and ask you to connect to an existing project or create a new project.


Click **Create New** and give it a **name before pressing continue**:

× Create a project (Step 1 of 3)

Let's start with a name for your project[?]

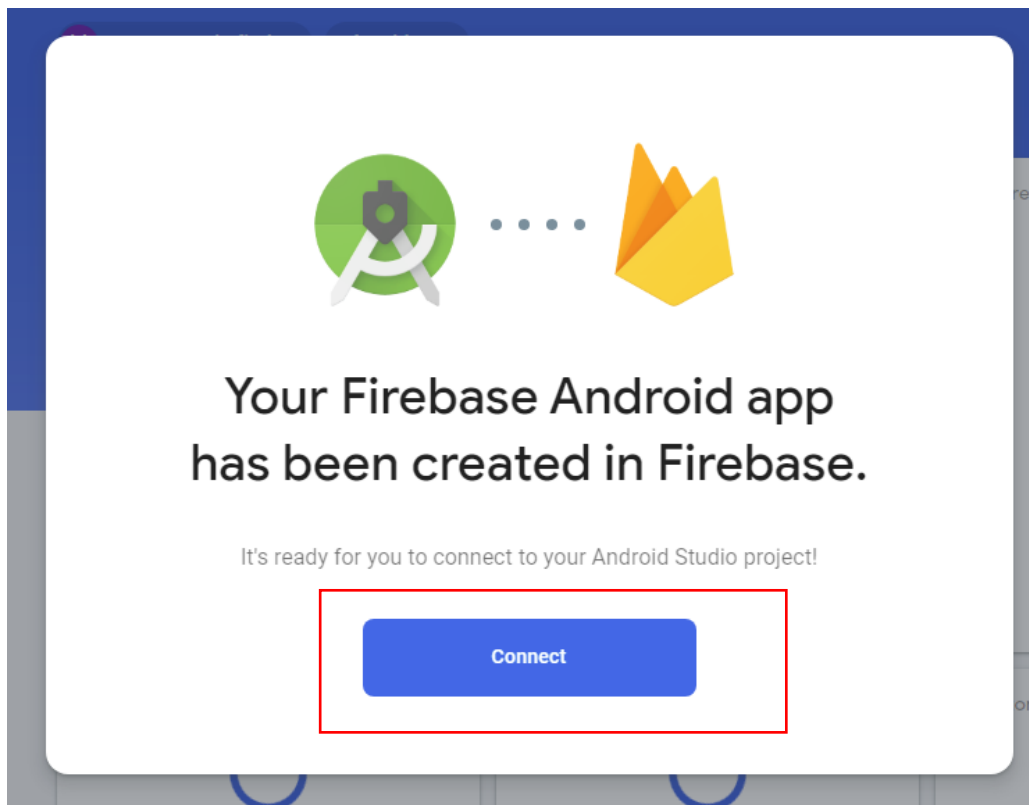
Project name

Firestore2022

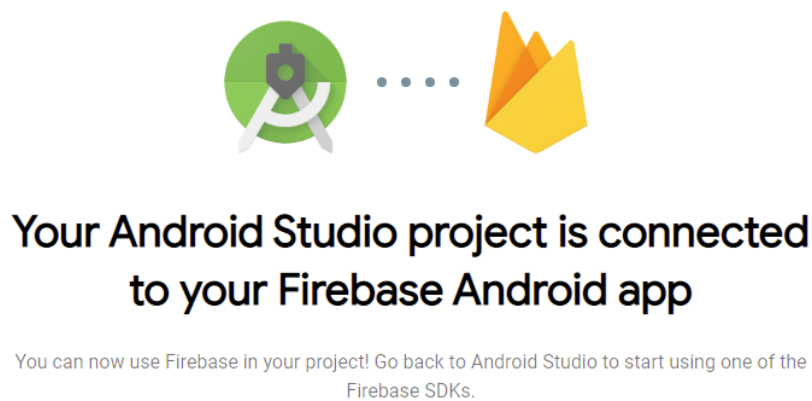
 fir-2022-1c56c

Continue

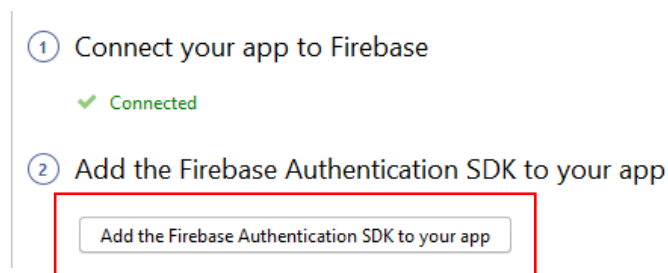
If you go to your **Firebase** account, you should see the new project in your **Firebase** console:



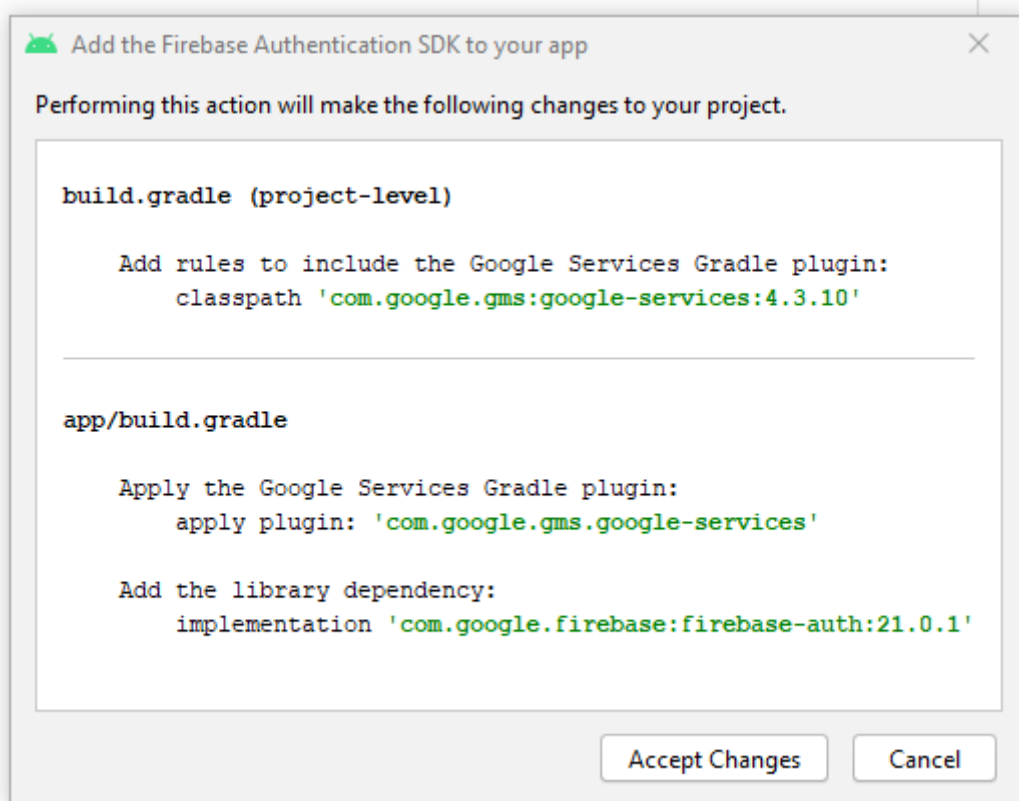
Click Connect:



This will appear in the **Firebase Assistant**, and click **add the Firebase Authentication**:



This will add the following to the build gradle:

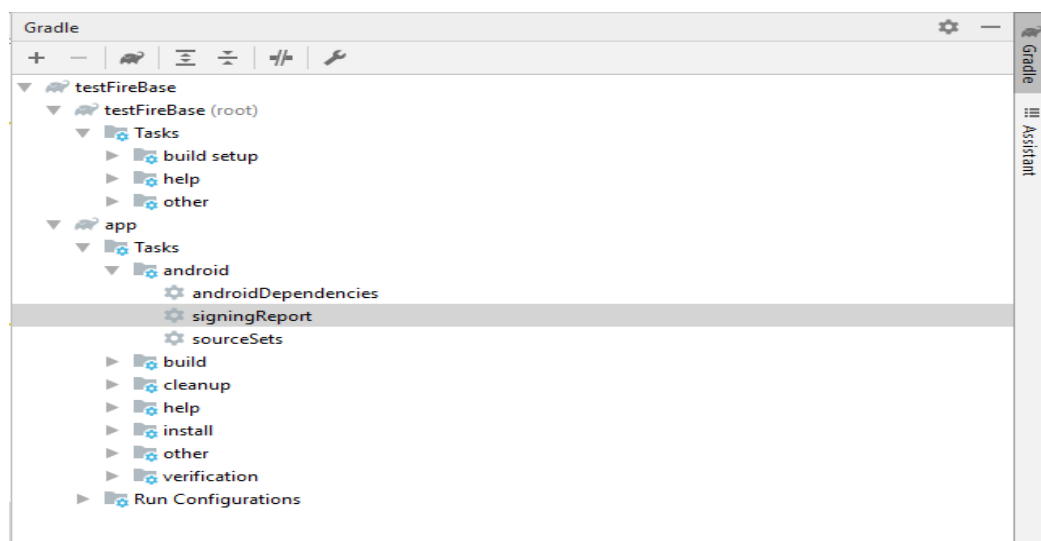


Click Accept changes and move to step 3:

③ Complete your set up in the Firebase console

- If you haven't yet, specify your app's SHA-1 fingerprint. You can do this in your [Project settings](#) in the Firebase console. Refer to [Authenticating Your Client](#) for details on how to get your app's SHA-1 fingerprint.
- To use an authentication provider, you need to enable it for your Firebase project. Enable **Google** (Google Sign-In) in the [Sign-in method](#) tab of the Firebase Authentication section of the Firebase console.

Sometimes Google has trouble connecting the files, to generate an SHA1 key do the following, click Gradle top right hand side and go to **Tasks->android->signingreport**:



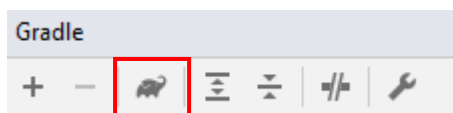
If we click SigningReport we generate our SHA-1 fingerprint:

```
Run: app x C:/Users/se15030621/AndroidStudioProjects/tes... x MainActivity > onActivityResult()
Valid until: Wednesday, 29 September 2049
-----
Variant: debugAndroidTest
Config: debug
Store: C:\Users\se15030621\.android\debug.keystore
Alias: AndroidDebugKey
MD5: 58:C4:77:2D:73:36:38:4E:68:C1:34:83:F9:7A:E3:C5
SHA1: EE:84:92:3A:9B:B1:2F:4E:B1:92:0B:22:89:29:AD:92:31:D2:33:50
SHA-256: D7:A0:5F:49:50:D7:43:37:91:39:EF:30:7C:34:69:4C:A5:94:90:F6:AD:14:84:01:8B:D4:C0:42:73:06:FB:44
Valid until: Wednesday, 29 September 2049
-----
Variant: releaseUnitTest
Config: none
-----

BUILD SUCCESSFUL in 0s
1 actionable task: 1 executed
14:45:21: Task execution finished 'signingReport'.
```

You can also generate this without going to tasks:

Click gradle in the top right and press the little elephant button:

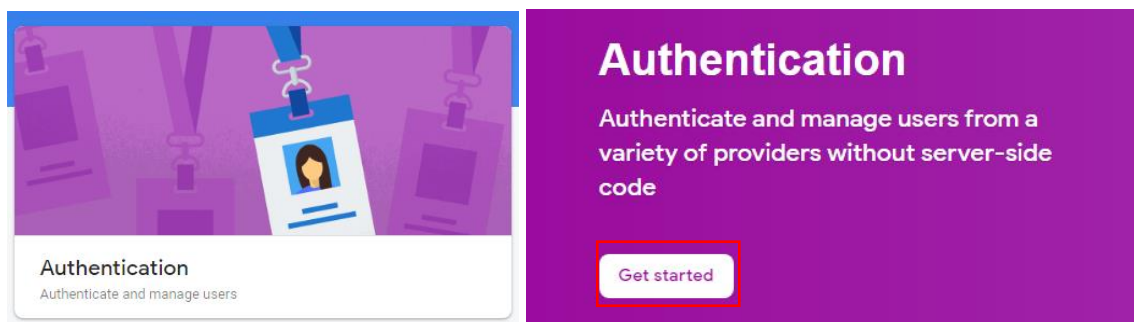


A window appears and will allow you to type: gradle **signingreport**. The Gradle will be autofilled so just write **signingreport** and click **enter**. This will generate your SHA1 key:

```
> Task :app:signingReport
Variant: debugUnitTest
Config: debug
Store: C:\Users\Me\.android\debug.keystore
Alias: AndroidDebugKey
MD5: F4:76:73:AD:49:76:41:7B:05:68:42:31:F9:2F:C3:44
SHA1: 4A:85:94:7E:20:92:57:8E:AC:00:36:E3:5D:0F:BA:A2:69:60:FB:21
SHA-256: 76:13:25:C9:0B:98:2C:60:9E:39:40:A2:13:A1:3C:6E:19:E3:26:86:A1:A4:8C:D7:15:4F:CF:B6:0C:58:78:98
Valid until: Sunday, 18 September 2050
```


Next go to **Firebase Console**.

Click on your **New Firebase Project** and go to **Authentication -> Get Started**:




Firestore authentication:

From here, we can go into the project and find Authentication and enable **Email/Password**, **Google** and **Anonymous**:

 Google Enable

Google sign-in is automatically configured on your connected Apple 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](#).


 Update the [project-level setting](#) below to continue

Project public-facing name ?

project-739339030313

Project support email ?

Not configured

 Please select an email address

Safelist client IDs from external projects (optional) ?

Web SDK configuration ?

Cancel

Save

Google will ask for a project support email address, use the one you used to create the Firebase account. Click Save.



Next, we will see the following:

Download latest configuration file

✕

Enabling Google sign-in for the first time creates new OAuth clients. Download and update your configuration files to make sure Google sign-in works for your apps. See instructions for [Android](#) and [Apple](#).

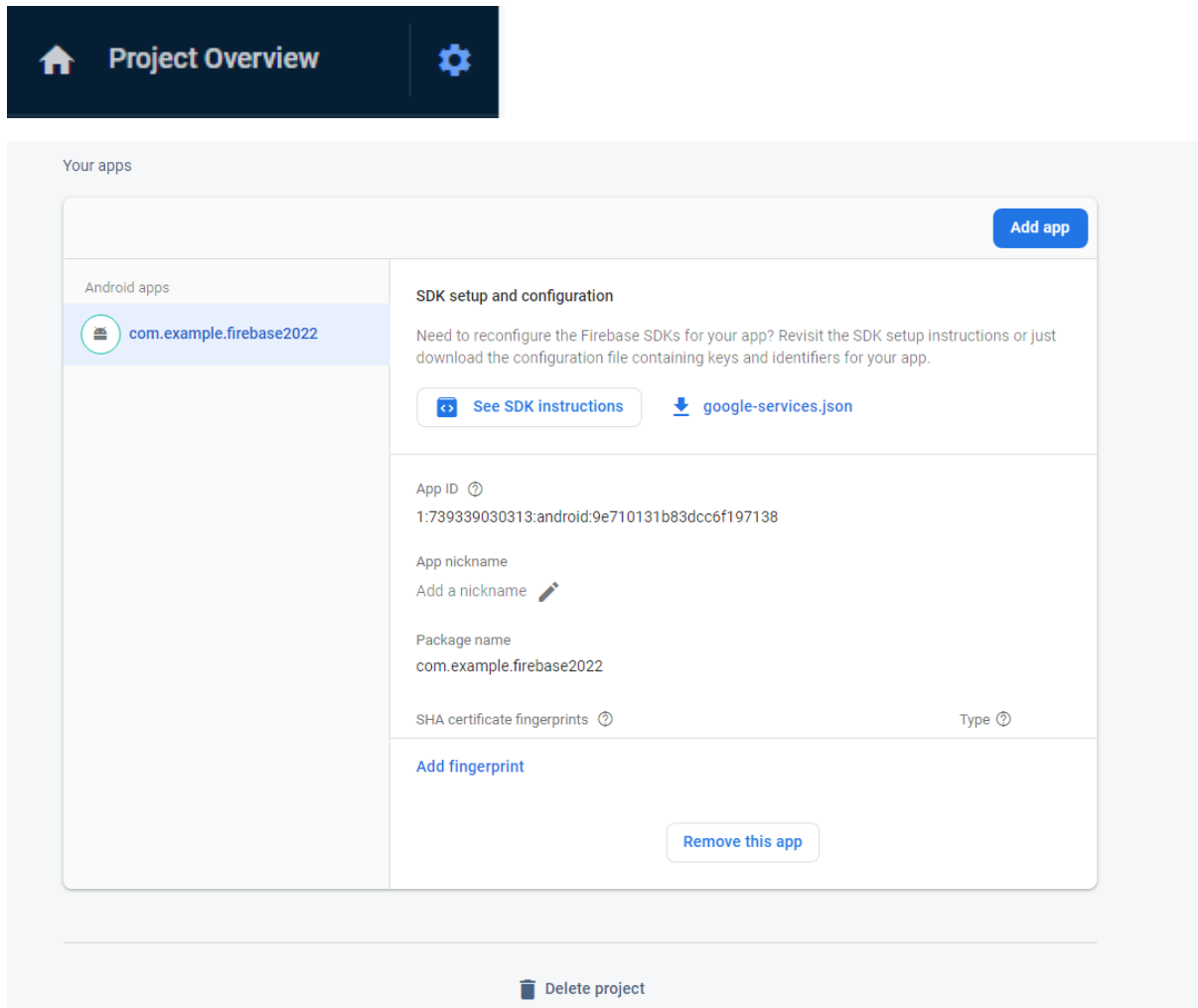
Search

Your apps	Configuration file
 com.example.firebase2022	 Configure SHA-1 certificate and download google-services.json in project settings

Done

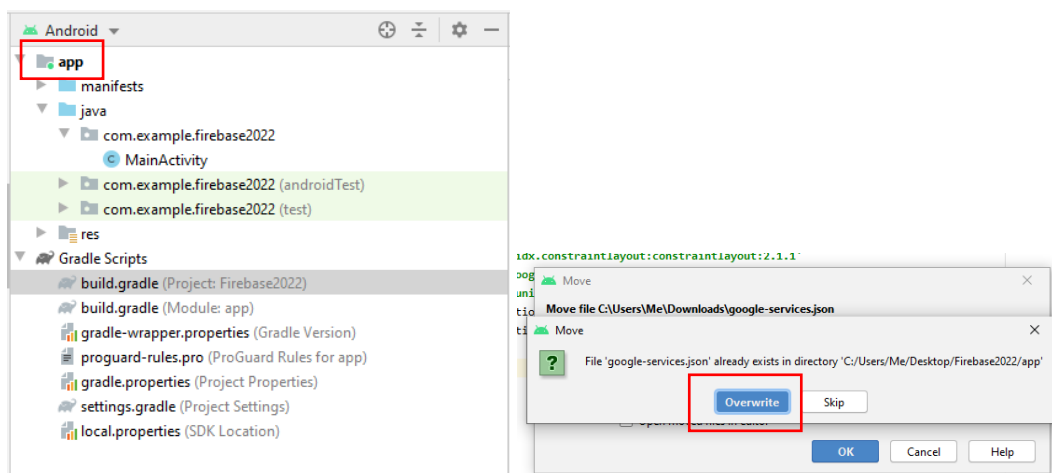
We now need to configure the SHA-1 key in Firebase.

Go to **Project Overview** -> **settings**.



Add your SHA-1 key by clicking **Add fingerprint** and when this is done click the **google-services.json** to download our new config file.

Drag this file from **Downloads** into android studio, and copy to **app**, then **Overwrite** the previous version of the file:



Next, we will use the **Firestore** walkthrough found here:

<https://firebase.google.com/docs/auth/android/google-signin>

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:29.0.0')

    // Declare the dependency for the Firebase Authentication library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebase:firebase-auth'

    // Also declare the dependency for the Google Play services library and specify its version
    implementation 'com.google.android.gms:play-services-auth:19.2.0'
}
```

Add these dependencies to your **build.gradle** and **sync**:

```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    implementation 'com.google.firebase:firebase-auth:21.0.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:29.0.0')

    // Declare the dependency for the Firebase Authentication library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebase:firebase-auth'

    // Also declare the dependency for the Google Play services library and specify its version
    implementation 'com.google.android.gms:play-services-auth:19.2.0'
}
```

Next, go to this link:

<https://firebase.google.com/docs/auth/android/firebaseui>

This will give you a walkthrough and give you the code to implement the default Google login to your app. Add the first line to your dependencies:

```
dependencies {
    // ...

    implementation 'com.firebaseui:firebase-ui-auth:7.2.0'

    // Required only if Facebook login support is required
    // Find the latest Facebook SDK releases here: https://goo.gl/Ce5L94
    implementation 'com.facebook.android:facebook-android-sdk:8.x'
}
```

```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    implementation 'com.google.firebase:firebase-auth:21.0.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:29.0.0')

    // Declare the dependency for the Firebase Authentication Library
    // When using the BoM, you don't specify versions in Firebase Library dependencies
    implementation 'com.google.firebase:firebase-auth'
    implementation 'com.firebaseui:firebase-ui-auth:7.2.0'

    // Also declare the dependency for the Google Play services Library and specify its version
    implementation 'com.google.android.gms:play-services-auth:19.2.0'
}
```

At the top of the file add the following:

```
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
```

It should look like this:

```
android {
    compileSdkVersion 30
    buildToolsVersion "30.0.2"

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    defaultConfig {
        applicationId "com.example.firebase2022"
        minSdkVersion 29
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
}
```

That's our dependencies set up.

On to the **code**.

I use both links so be careful of the walkthrough you use here:

First, we want to add 2 global variables:

```
public class MainActivity extends AppCompatActivity {  
    private FirebaseAuth mAuth;  
    Button SignOut;
```

Under this we want to use a new activity lifecycle function called `onStart()`:

```
@Override  
public void onStart() {  
    super.onStart();  
    // Check if user is signed in (non-null) and update UI accordingly.  
    FirebaseUser currentUser = mAuth.getCurrentUser();  
}
```

This will get the current user when the app begins.

We then add a new `ActivityResultLauncher`. This will handle our logins to the app.

```
private final ActivityResultLauncher<Intent> signInLauncher = registerForActivityResult(  
    new FirebaseAuthUIActivityResultContract(),  
    new ActivityResultCallback<FirebaseAuthUIAuthenticationResult>() {  
        @Override  
        public void onActivityResult(FirebaseAuthUIAuthenticationResult result) {  
            onSignInResult(result);  
        }  
    }  
);
```

And then we get to our `onCreate()`:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    mAuth = FirebaseAuth.getInstance();  
    createSignInIntent();  
  
    SignOut = (Button)findViewById(R.id.btn_signOut);  
    SignOut.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) { signOut(); }  
    });  
}
```

Get the firebaseAuth to grab the current user and set up our button with an onClickListener which calls a function signOut(); All together it looks like this:

```
public class MainActivity extends AppCompatActivity {
    private FirebaseAuth mAuth;
    Button SignOut;

    @Override
    public void onStart() {
        super.onStart();
        // Check if user is signed in (non-null) and update UI accordingly.
        FirebaseUser currentUser = mAuth.getCurrentUser();
    }

    private final ActivityResultLauncher<Intent> signInLauncher = registerForActivityResult(
        new FirebaseAuthUIActivityResultContract(),
        new ActivityResultCallback<FirebaseAuthUIAuthenticationResult>() {
            @Override
            public void onActivityResult(FirebaseAuthUIAuthenticationResult result) {
                onSignInResult(result);
            }
        }
    );

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mAuth = FirebaseAuth.getInstance();
        createSignInIntent();

        SignOut = (Button)findViewById(R.id.btn_signOut);
        SignOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { signOut(); }
        });
    }
}
```

Now, under our onCreate() we will create 3 functions.

CreateSignInIntent:

```
public void createSignInIntent() {  
    // Choose authentication providers  
    List<AuthUI.IdpConfig> providers = Arrays.asList(  
        new AuthUI.IdpConfig.EmailBuilder().build(),  
        new AuthUI.IdpConfig.GoogleBuilder().build(),  
        new AuthUI.IdpConfig.AnonymousBuilder().build());  
  
    // Create and launch sign-in intent  
    Intent signInIntent = AuthUI.getInstance().AuthUI  
        .createSignInIntentBuilder().AuthUI.SignInIntentBuilder  
        .setAvailableProviders(providers)  
        .build();  
    signInLauncher.launch(signInIntent);  
}
```

onSignInResult:

```
private void onSignInResult(FirebaseAuthUIAuthenticationResult result) {  
    IdpResponse response = result.getIdpResponse();  
    if (result.getResultCode() == RESULT_OK) {  
        //Successfully signed in  
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();  
        Toast.makeText(context, this, user.getEmail(), Toast.LENGTH_SHORT).show();  
    } else {  
        // Sign in failed. If response is null the user canceled the  
        Toast.makeText(context, this, text: "Failed Login", Toast.LENGTH_SHORT).show();  
    }  
}
```

signOut:

```
private void signOut() {  
    AuthUI.getInstance()  
        .signOut(context, this)  
        .addOnCompleteListener(new OnCompleteListener<Void>() {  
            public void onComplete(@NonNull Task<Void> task) { createSignInIntent(); }  
        });  
}
```


All together this should look like the following:

```
public class MainActivity extends AppCompatActivity {
    private FirebaseAuth mAuth;
    Button SignOut;

    @Override
    public void onStart() {
        super.onStart();
        // Check if user is signed in (non-null) and update UI accordingly.
        FirebaseUser currentUser = mAuth.getCurrentUser();
    }

    private final ActivityResultLauncher<Intent> signInLauncher = registerForActivityResult(
        new FirebaseAuthUIActivityResultContract(),
        new ActivityResultCallback<FirebaseAuthUIAuthenticationResult>() {
            @Override
            public void onActivityResult(FirebaseAuthUIAuthenticationResult result) {
                onSignInResult(result);
            }
        }
    );

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mAuth = FirebaseAuth.getInstance();
        createSignInIntent();

        SignOut = (Button)findViewById(R.id.btn_signOut);
        SignOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { signOut(); }
        });
    }

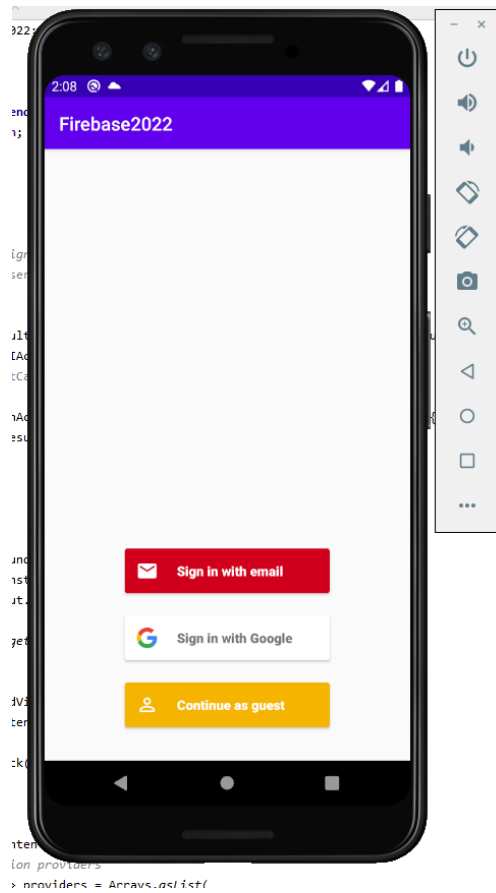
    public void createSignInIntent() {
        // Choose authentication providers
        List<AuthUI.IdpConfig> providers = Arrays.asList(
            new AuthUI.IdpConfig.EmailBuilder().build(),
            new AuthUI.IdpConfig.GoogleBuilder().build(),
            new AuthUI.IdpConfig.AnonymousBuilder().build());

        // Create and launch sign-in intent
        Intent signInIntent = AuthUI.getInstance().AuthUI
            .createSignInIntentBuilder().AuthUI.SignInIntentBuilder
            .setAvailableProviders(providers)
            .build();
        signInLauncher.launch(signInIntent);
    }

    private void onSignInResult(FirebaseAuthUIAuthenticationResult result) {
        IdpResponse response = result.getIdpResponse();
        if (result.getResultCode() == RESULT_OK) {
            //Successfully signed in
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
            Toast.makeText( context: this, user.getEmail(), Toast.LENGTH_SHORT).show();
        } else {
            // Sign in failed. If response is null the user canceled the
            Toast.makeText( context: this, text: "Failed Login", Toast.LENGTH_SHORT).show();
        }
    }

    private void signOut() {
        AuthUI.getInstance()
            .signOut( context: this)
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                public void onComplete(@NonNull Task<Void> task) { createSignInIntent(); }
            });
    }
}
```

Now if we run our project we should get the following:



We can now sign in and our Authentication page on the Firebase Console will have updated:

Search by email address, phone number, or user UID					Add user	↺	⋮
Identifier	Providers	Created ↓	Signed In	User UID			
(anonymous)	👤	Nov 9, 2021	Nov 9, 2021	DPsJN304k7YVPagnGJsOzpacVC...			
georgemrtn5@gmail.com	🌐	Nov 9, 2021	Nov 9, 2021	NoDv8e1QYQQb300xZWokrUGHB...			
george@mail.com	✉️	Nov 9, 2021	Nov 9, 2021	BLeVirtF5HQqFD40QWIFmwcEGVr1			
Rows per page: 50					1 - 3 of 3	⏪	⏩

Additional:

Connecting your Firebase to other login types.

The advanced functionality is to use Twitter or Facebook to Log in to your app.

This one is for you to explore other app logins. They can be slightly tricky but they're good to know, you might have to know Windows Command line, but there are plenty of tutorials on YouTube.