Advanced Mobile Application Development
Week 15: Android and Firebase Authentication

Firebase provides sign-in flows for email/password, email link, phone authentication, Google Sign-In, Facebook Login, Twitter Login, and GitHub Login.
FirebaseUI Auth provides a drop-in auth solution that handles the UI flows for signing in users with all the methods

Google Authentication
https://firebase.google.com/docs/auth/android/google-signin
To implement authentication there's some settings you must change first.

Firebase Console
In your Firebase console go into Authentication and in the Sign-in tab edit Google authentication so it's enabled.
Confirm that in your project's settings you have your app hooked up to your database.
Then in Database go to Rules and change the rules to require authentication.

```
{
   "rules": {
      ".read": "auth != null",
      ".write": "auth != null"
   }
}
```

Note that once you've changed the rules if you run your app without implementing authentication you won't see any data.

Recipes app
Add the Firebase auth SDKs to your app by adding the following dependencies to your app Gradle file.
`app/build.gradle`

```
implementation 'com.google.firebase:firebase-auth:16.2.1'
implementation 'com.google.android.gms:play-services-auth:16.0.1'
```

Now we're ready to implement authentication.
I've added Google authentication to the Recipes app (Recipes auth).

Create a new empty activity called SignInActivity
Check Generate Layout File
Check Launcher Activity
Check Backwards Compatibility

Since we just created a launcher activity if you go into the AndroidManifest file you'll see we now have two activities with the category launcher. If you run it this way two launcher icons will get created so for MainActivity you should remove the category and action.

```
<intent-filter>
   <action android:name="android.intent.action.MAIN" />

   <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

In activity_sign_in I added an image to res/drawable that I'll use on my sign-in page.
Add an ImageView and set the resource to the image you just added.
Add a google sign-in button in the XML (I couldn't figure out how to do this in design mode).

```
<com.google.android.gms.common.SignInButton
    android:id="@+id/login_with_google"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Change the width if you don't want it the width of the screen.
I also made the height a little larger than wrap_content gave me.
Add needed constraints.

All authentication logic will be in SignInActivity.java
Create the class data members we'll need. It doesn't matter what value you give RC_SIGN_IN, we're just using a constant so it's easy to check.

```
private GoogleSignInClient googleSignInClient;
private static final int RC_SIGN_IN = 9001;
private FirebaseAuth firebaseAuth;
private SignInButton signInButton;
```

Update onCreate() to configure Google sign-in.

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

    //get the shared instance of the FirebaseAuth object
    firebaseAuth = FirebaseAuth.getInstance();

    //configure Google sign-in
    GoogleSignInOptions options = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken(getString(R.string.default_web_client_id))
            .build();

    //creates a new instance of GoogleSignInClient
    googleSignInClient = GoogleSignIn.getClient(this, options);

    signInButton = findViewById(R.id.login_with_google);
    signInButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            signIn();
        }
    });
}
```

In the GoogleSignInOptions object you can add requestEmail() to request the user's email as well. signIn() will give you an error because we need to create that method.

```
private void signIn() {
    //gets an Intent to start the Google sign-in flow
    Intent signInIntent = googleSignInClient.getSignInIntent();
    //start Google sign-in intent. onActivityResult() will be called with the requestCode when it's finished
    startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

Starting the intent prompts the user to select a Google account to sign in with. After the user signs in, you can get a GoogleSignInAccount object for the user in the activity's onActivityResult method.

```
//called when Google sign-in finishes
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    //result returned from launching the Intent from GoogleSignInApi.getSignInIntent()
    if (requestCode == RC_SIGN_IN) {
        //returns a completed Task containing an error or GoogleSignInAccount object
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        try {
            //Google Sign In was successful
            //returns Google account
            GoogleSignInAccount account = task.getResult(ApiException.class);
            firebaseAuthWithGoogle(account);
        } catch (ApiException e) {
            // Google Sign In failed
            Log.e("LOGIN", "Google sign in failed", e);
            Toast.makeText(this, "Login Unsuccessful", Toast.LENGTH_SHORT)
                .show();
        }
    }
}
```

The GoogleSignInAccount object contains information about the signed-in user, such as the user's name. firebaseAuthWithGoogle() will give you an error because we need to create that method.

```
private void firebaseAuthWithGoogle(GoogleSignInAccount account){
    String idToken = account.getIdToken();
    final String name = account.getDisplayName();

    //get Firebase credential
    AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);

    //Firebase authentication
    firebaseAuth.signInWithCredential(credential).addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
```

```
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            Log.d("SIGNIN", "signInWithCredential:onComplete:" + task.isSuccessful());
            if (task.isSuccessful()) {
                Toast.makeText(SignInActivity.this, "Login successful " + name,
Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(SignInActivity.this, RecipeMainActivity.class);
                startActivity(intent);
            }else {
                Log.e("SIGNIN", "signInWithCredential" + task.getException());
                Toast.makeText(SignInActivity.this, "Authentication failed.",
Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

At this point you should be able to run it, log in, and then go to the main recipes page.

Note that I'm using the newer GoogleSignInClient as outlined in the Android Firebase Google authentication documentation. The older GoogleApiClient is used in many tutorials and will still work. You can read about the differences and why GoogleSignInClient is suggested in this Android Developers blog post https://android-developers.googleblog.com/2017/11/moving-past-googleapiclient_21.html.

It would be nice to check to see if the user is currently signed in when initializing your Activity which you can do in onStart(). I have this commented out in my example, and I suggest you do the same while testing, or once you're logged in you won't see the login activity and won't be able to log in.

```
@Override
protected void onStart() {
    super.onStart();

    //check to see if the user is already logged in
    if(firebaseAuth.getCurrentUser()!=null){
        startActivity(new Intent(getApplicationContext(),RecipeMainActivity.class));
    }
}
```

This is a minimal example to introduce you to Firebase authentication on Android.
If you provide a logout button you can have it call the signOut() method.
Firebase also has an AuthStateListener that's called when there is a change in the authentication state.
If different accounts have different data in Firebase you'd need to organize your database differently so recipes are specific to a user.