ATLS 4320: Advanced Mobile Application Development
Week 7: iOS and Firebase Authentication

Firebase supports authentication using email/password, google, facebook, twitter, or github. They even offer a pre-built UI you can use if you're supporting all of these.
https://firebase.google.com/docs/auth/ios/google-signin
We're going to implement authentication using Google. This requires the following pods in the Podfile:
pod 'Firebase/Auth'
pod 'GoogleSignIn'

(if you needed to add these you need to do pod install again after you add them).

In the Firebase console go into Authentication Sign-in Method and enable Google.
Then go into Database | Rules that we had changed to be wide open, and change it back so users must be authenticated.

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

More info on rules https://firebase.google.com/docs/database/security/quickstart

Back in your app go into the `GoogleService-Info.plist` configuration file, and look for the `REVERSED_CLIENT_ID` key. Copy the value of that key.
Go into your target's Info tab and expand the URL Types section and add a new URL scheme and paste the REVERSED_CLIENT_ID key into the URL Schemes box. Leave the other fields blank.
Repeat this with your Bundle Identifier that you can get in the General tab. Add another URL type with that value as the URL scheme.

In AppDelegate.swift import the needed frameworks

```
import Firebase
import GoogleSignIn
```

Configure the the FirebaseApp object and initialize sign-in in application(_:didFinishLaunchingWithOptions:)

```
// Initialize sign-in
GIDSignIn.sharedInstance().clientID = FirebaseApp.app()?.options.clientID
```

Implement the application:openURL:options: method of your app delegate. The method should call the handleURL method of the GIDSignIn instance, which will properly handle the URL that your application receives at the end of the authentication process. (a different method is needed for iOS 8 and earlier so if you're supporting iOS8 or earlier before this method add `@available(iOS 9.0, *)`)

```
    func application(_ application: UIApplication, open url: URL, options:
[UIApplication.OpenURLOptionsKey : Any])
        -> Bool {
            return GIDSignIn.sharedInstance().handle(url,
sourceApplication:options[UIApplication.OpenURLOptionsKey.sourceApplication]
as? String, annotation: [:])
    }
```

Create a new Cocoa Touch class called LoginViewController and subclass UIViewController.
In your storyboard drag out a new View Controller and make its class LoginViewController.
Make it the Initial View Controller.
We could change the view to be the GIDSignInButton class but we can't control its layout using autoresizing or autolayout in interface builder, so we'll create the button programmatically.

In the LoginViewController import the needed frameworks

```
import Firebase
import GoogleSignIn
```

Adopt the needed the GIDSignInDelegate and GIDSignInUIDelegate protocol.

```
class LoginViewController: UIViewController, GIDSignInDelegate,
GIDSignInUIDelegate
```

The GIDSignInDelegate and GIDSignInUIDelegate protocols handle Google sign in and out
https://developers.google.com/identity/sign-in/ios/api/protocol_g_i_d_sign_in_delegate-p
https://developers.google.com/identity/sign-in/ios/api/protocol_g_i_d_sign_in_u_i_delegate-p

You can fix the error and it will add the required method.

In viewDidLoad() assign the GIDSignIn delegate and ui delegate and then define and add the Google sign in button.

```
        GIDSignIn.sharedInstance().uiDelegate = self
        GIDSignIn.sharedInstance().delegate = self

        //define Google sign in button
        let googleSignInButton = GIDSignInButton()
        googleSignInButton.style = .wide
        googleSignInButton.center = view.center
        view.addSubview(googleSignInButton)
```

The GIDSignInButton class lets you choose a style and colorScheme for the button.
https://developers.google.com/identity/sign-in/ios/api/interface_g_i_d_sign_in_button

Now we have to implement the GIDSignInDelegate method to handle sign-in. Users might log in using different methods but they pass the authentication to Firebase so all users are logged in with Firebase.

```
    func sign(_ signIn: GIDSignIn!, didSignInFor user: GIDGoogleUser!,
withError error: Error?) {
        if let error = error {
            print(error.localizedDescription)
            return
        }
        guard let authentication = user.authentication
            else {
                return
        }
        let credential = GoogleAuthProvider.credential(withIDToken:
authentication.idToken,
                                                       accessToken:
authentication.accessToken)
```

```
        Auth.auth().signInAndRetrieveData(with: credential, completion:
{(user, error) in
            if let error = error {
                print(error.localizedDescription)
                return
            }
            print("User logged in with Google")
        })
    }
```

To handle logging out you would implement `func sign(_ signIn: GIDSignIn!, didDisconnectWith user: GIDGoogleUser!, withError error: Error!)`

The first time you log in it will take you to Google to sign in. Right now we just go back to the login screen. In your Firebase console go to Authentication and look at Users and you should see yourself. The app will remember you have logged in and you won't be prompted again.

Now once you log in let's have our app go to our RecipeTableViewController.
Back in the storyboard create a segue from the login view controller(yellow circle icon) to the recipe table view's navigation controller and chose Manual Segue Present Modally. Give it the identifier "showRecipes".

Back in LoginViewController let's use an alert to tell the user they're logged in and then in the completion block segue to our RecipeTableViewController.

Update sign(_:didSignInFor: withError:)

```
//create a UIAlertController object
            let alert=UIAlertController(title: "Firebase", message: "Welcome
to Firebase " + (user?.user.displayName)!, preferredStyle:
UIAlertController.Style.alert)

            //create a UIAlertAction object for the button
            let okAction=UIAlertAction(title: "OK", style:
UIAlertAction.Style.default, handler: {action in
                //perform segue
                self.performSegue(withIdentifier: "showRecipes", sender:
nil)
            })
            alert.addAction(okAction)
            self.present(alert, animated: true, completion: nil)
```

Now when you run the app and click the button to sign in you should get an alert that tells you you're logged in and when you click OK it takes you to the RecipeTableViewController.