Firebase Authentication Integration

This document contains the firebase authentication integration resources on how it is implemented and guide resources that were used to configure it in the project.

CONTENTS:

Overview

The `Firebase Authentication` module of the Firebase SDK is being used within this project to allow a safe and secure login environment for its users.

This means that we do not have to handle our own login credentials and verification, as well as never having to store person information such as passwords anywhere within the application.

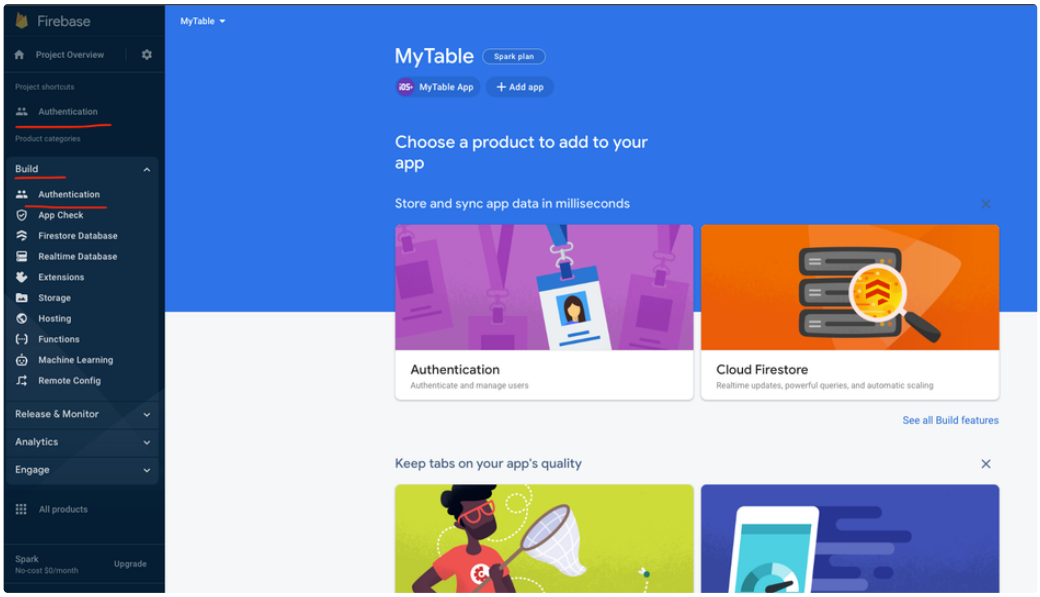
Firebase Authentication Dashboard

The firebase console for this project can be accessed at:

- https://console.firebase.google.com/u/0/project/mytable-12cf3/overview

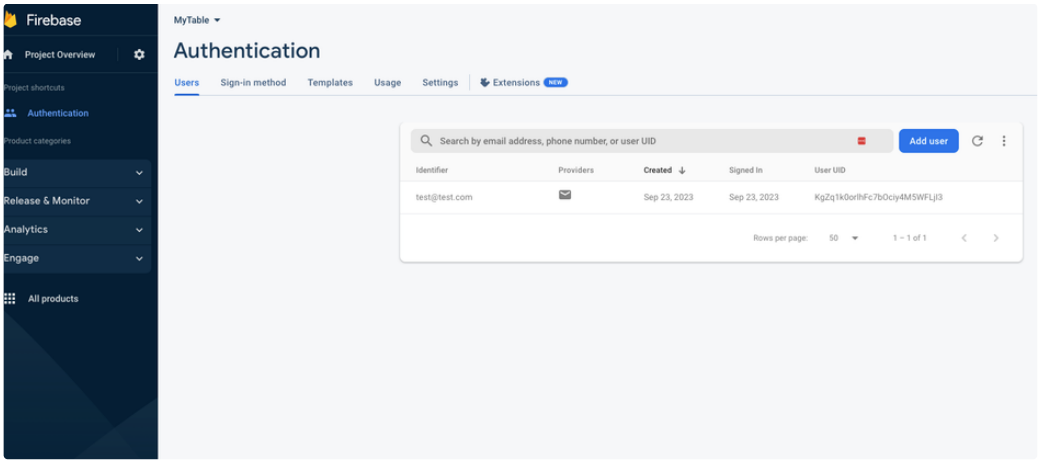
> ⚠ RMIT does not allow for student accounts to access Firebase console, you must use the credentials at the bottom of the document to login and manage the firebase instance

Once logged in you can view the dashboard, we are only going to be using the firebase `Authentication` module from this dashboard and console with is highlighted below:



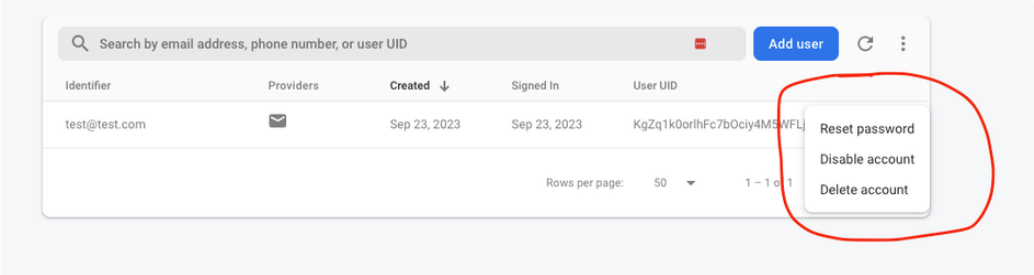
Once selected it will load up the `Users` section, in this example we have only a single user created from the app, but all users will be listed here with their unique details

- identifier => Email address
- Providers => login type
- Created
- Signed in
- User UID => unique identifier used in the application



If you click on the dots next to the specific user you can also edit their details

- Reset password
- Disable account
- Delete account

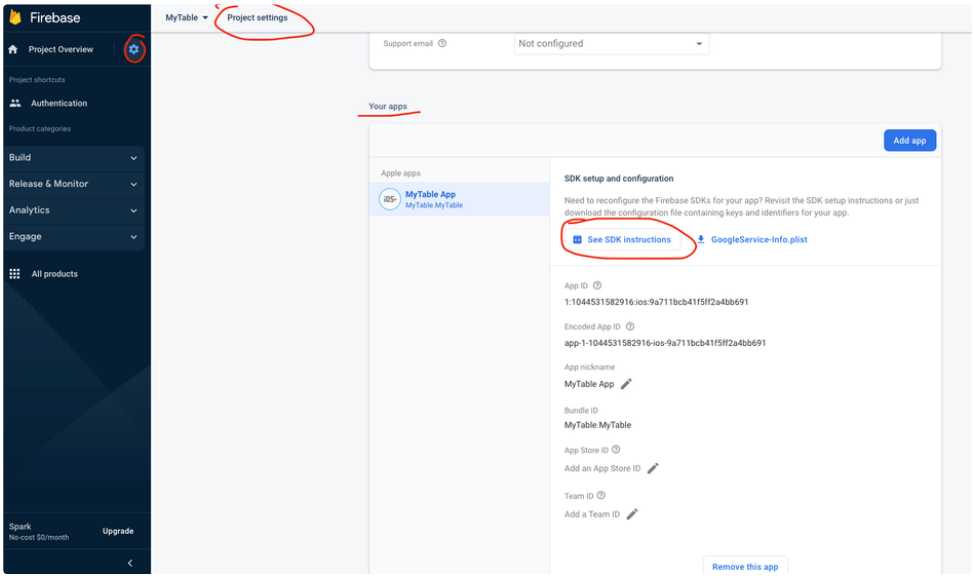


⚠ At the time of writing this, only email authentication has been setup. There is potential to add the google authentication towards the end of the project

Setting up the Authentication Service

Firebase Authentication has it's own `Auth` class that we can use to control the authentication for a user and when a user is successfully authenticated it returns a specific `User` object that we can access and use the `user.uid` to call database content that is user specific.

For any of the authentication to work we need to have the Firebase SDJK installed and configured in the project first. *Please see the Firebase setup guide on how to do this.*



ℹ For detailed instructions on setting up the SDK in SwiftUI, in the project in firebase, select `settings` (the cog icon) and then `Project settings`

From there scroll down to the `Your apps` section and click on `See SDK instructions` and it has a step by step process for SwiftUI setup

Steps:

1. Make sure the Firebase SDK is installed and configured as per above steps in the settings
2. Create the AuthenticationModel

AuthenticationModel.swift Example

```swift
import Foundation
import FirebaseAuth

// This will contain the login model for authentication specifically when connecting to an
```

```
// external resource for authentication
class AuthenticationModel: ObservableObject {

    var user: User? {
        didSet {
            objectWillChange.send()
        }
    }

    func listenToAuthState() {
        Auth.auth().addStateDidChangeListener { [weak self] _, user in
            guard let self = self else {
                return
            }
            self.user = user
        }
    }

    func signUp(
        email: String,
        password: String,
        completion: @escaping (Result<User?, Error>) -> Void
    ) {

        Auth.auth().createUser(withEmail: email, password: password) {
            (result, error) in
            if let error = error {
                // if there is an error log the error
                // TODO: add error monitoring
                print(error.localizedDescription)

                completion(.failure(error))
            } else {
                // Update and add new user details to the store
                self.user = result?.user

                // TODO: map and retrieve user data after signup

                print("signup success")
                completion(.success(result?.user))
            }
        }
    }

    func signIn(
        email: String,
        password: String,
        completion: @escaping (Result<User?, Error>) -> Void
    ) {

        Auth.auth().signIn(withEmail: email, password: password) {
            (result, error) in
            if let error = error {
                // if there is an error log the error
                // TODO: add error monitoring
                print(error.localizedDescription)

                completion(.failure(error))
```

```
            } else {
                self.user = result?.user

                // TODO: map and retrieve user data after signin

                print("signin success")
                completion(.success(result?.user))
            }
        }
    }

    func signOut() {
        // TODO: Clean up how this works

        // Signout from authentication
        try? Auth.auth().signOut()
        // clear the user
        self.user = nil

        // TODO: clear and remove user data from model but keep in store
    }
}

```

3. In the main app (**MyTableApp.swift**) add the `environmentObject` so that way it can be accessed in all areas of the application

a.

4. Import the environment variable into the required files to access it globally i.e. in `ContentView` access it by using `@EnvironmentObject var authModel: AuthenticationModel`
5. Add the listener from the `AuthenticationModel` to the app i.e.

a.

6. Use the `authModel` in a component by calling the `@EnvironmentObject` and using it's functions within the required areas i.e. for signup:

```swift
// Use firebase authentication to signin and
authModel.signUp(email: email, password: password) { result in
    switch result {
    case .success(let user):
        print("signup success \(String(describing: user?.uid))")
        self.profileModel.profile.firstName = firstName
        self.profileModel.profile.lastName = lastName
        self.profileModel.profile.username = userId
        self.profileModel.profile.password = password
        self.profileModel.profile.phone = phoneNumber
        self.profileModel.profile.email = email
        self.profileModel.profile.isLoggedIn = true
    case .failure(let error):
        signupResponse = error.localizedDescription
    }
}
```

a.

b. This does the signup using email and password, on success it loads the profileModel and also returns the `user` value within `authModel` which we can use to check if the state is logged in or not

c. On an `error` it can return a string of the error response from the server, and in this example we are saving it to `singupResponse` so that it can be displayed to user.

d. Some error examples can be:
   i. Server error
   ii. email user already exists
   iii. password doesn't match
   iv. etc

Resources

Setup Firebase in SwiftUI:

- Setup Firebase with SwiftUI in Xcode 13

Authentication and login example guides:

Using firebase authentication need to create specific classes to handle transitions

- How do I rerender my view in swiftUI after a user logged in with Google on Firebase?
- https://medium.com/swift-productions/swiftui-firebase-auth-listener-user-signup-manage-fae2294e8192

Example for catching success error from singup

- Firebase Auth - SwiftUI Advanced Handbook - Design+Code