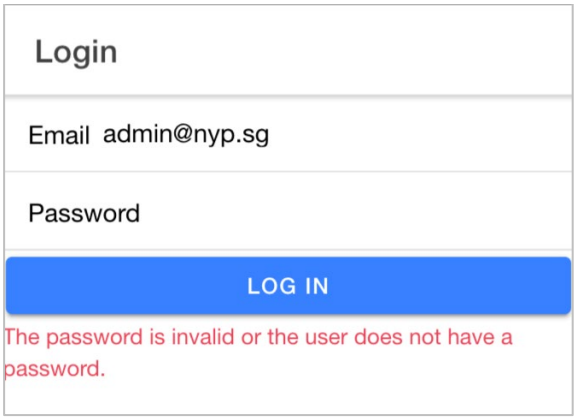
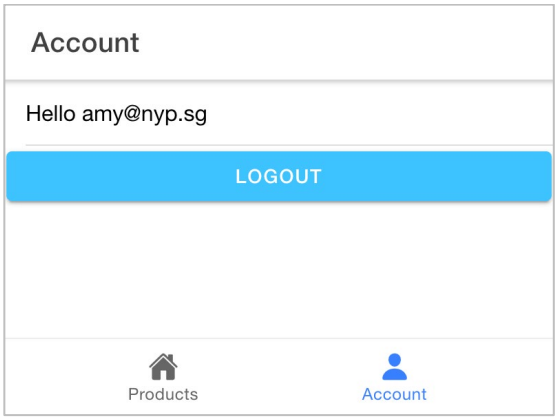
# Practical 8:

# Authentication

**Objectives:**

Perform Authentication using Firebase.



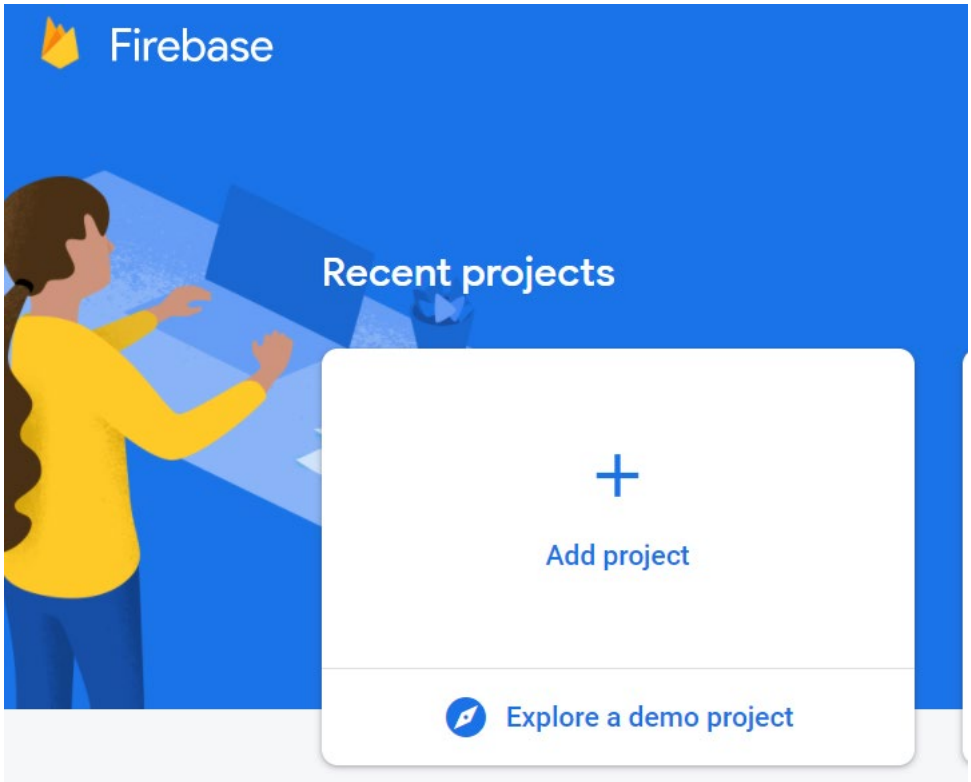

**Tasks:**

1. Set up Firebase
2. Connect to Firebase
3. Observe Authentication State
4. Login
5. Logout
6. Signup
7. Customize Tab Buttons based on User Role

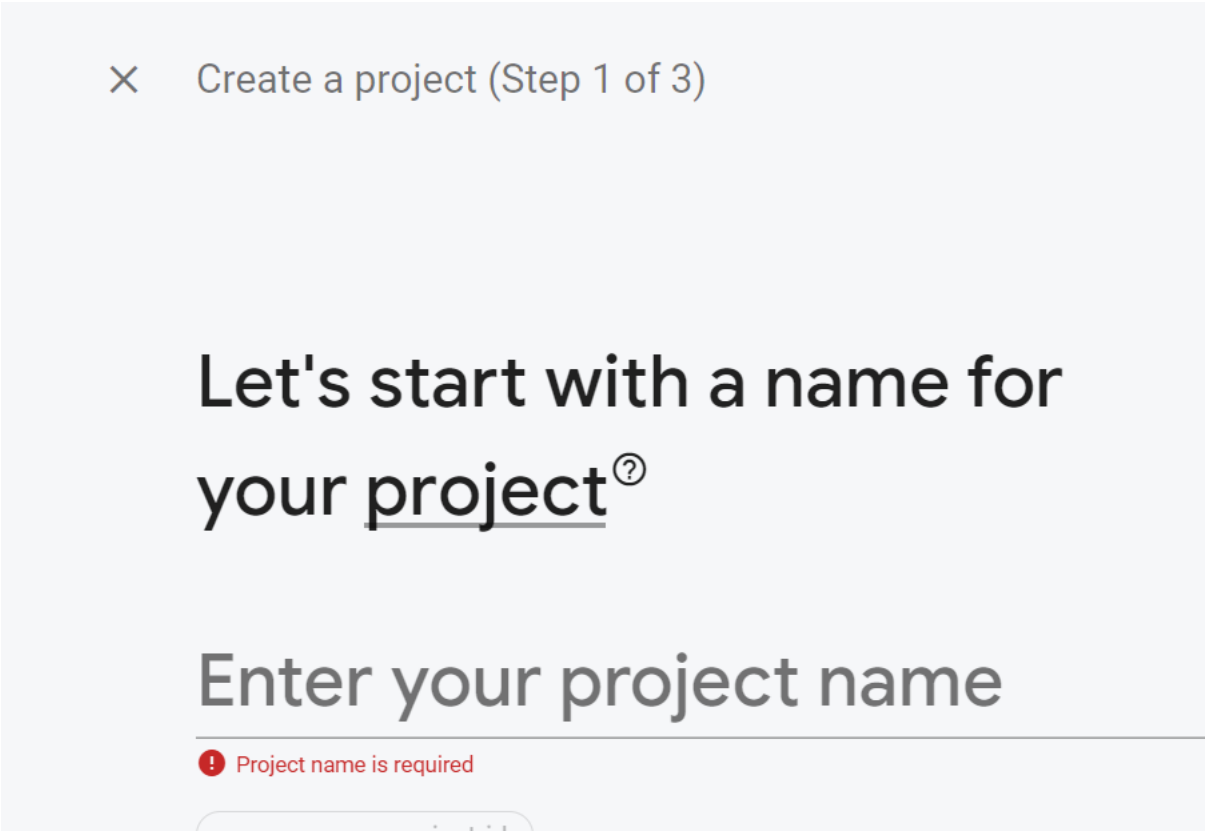
# 1. Set up Firebase

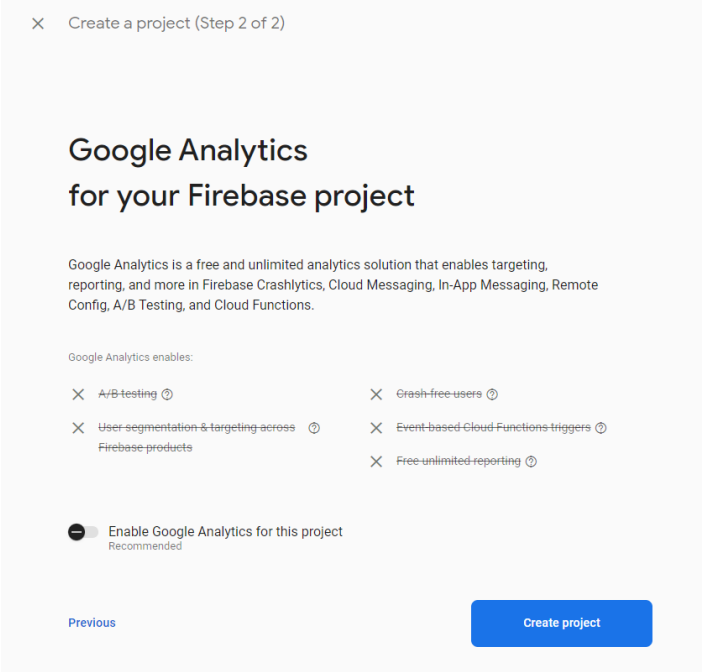
## 1.1. Create Firebase Project

1. Go to https://firebase.google.com.
2. Click **Get Started**.
3. **Sign in** using your Google Account.
4. Select **Add project**.

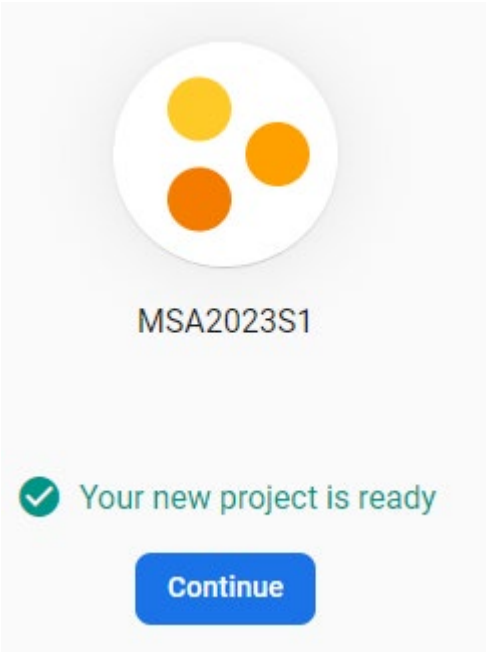



5. Fill in the Project name and location. Click **Create Project.**



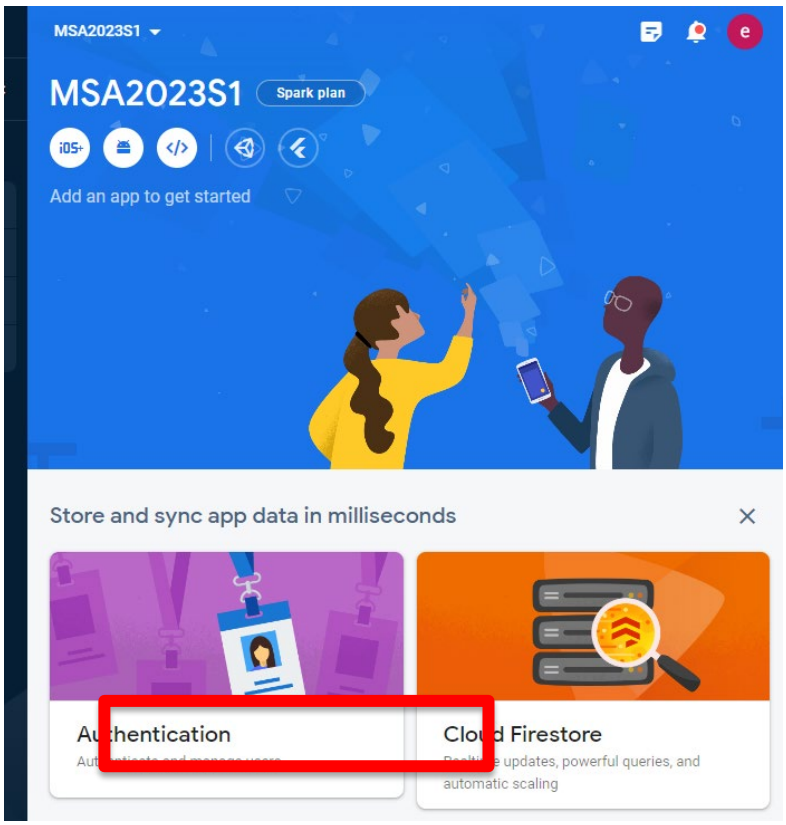

6. Click Create project.

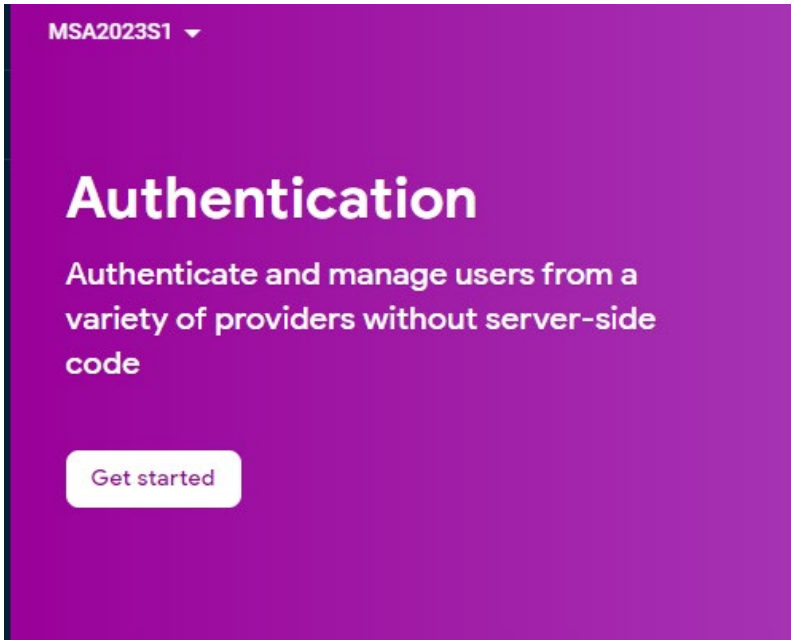



7. Click Continue

## 1.2. Enable Authentication

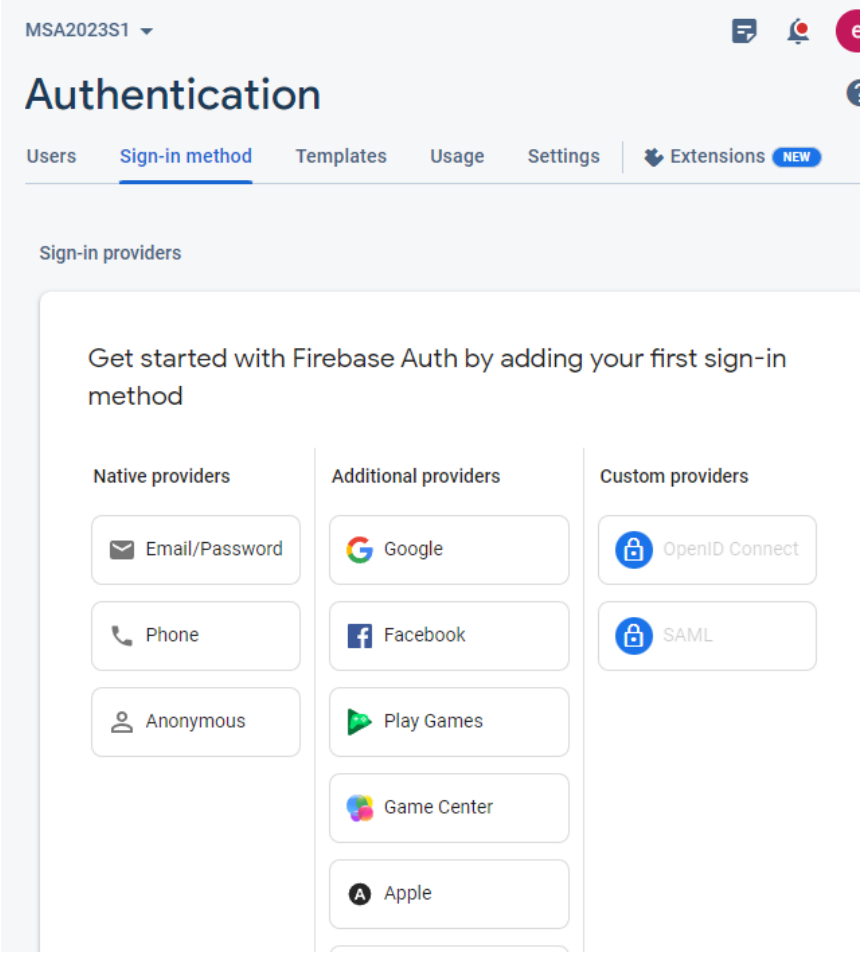
1. Select **Authentication**.



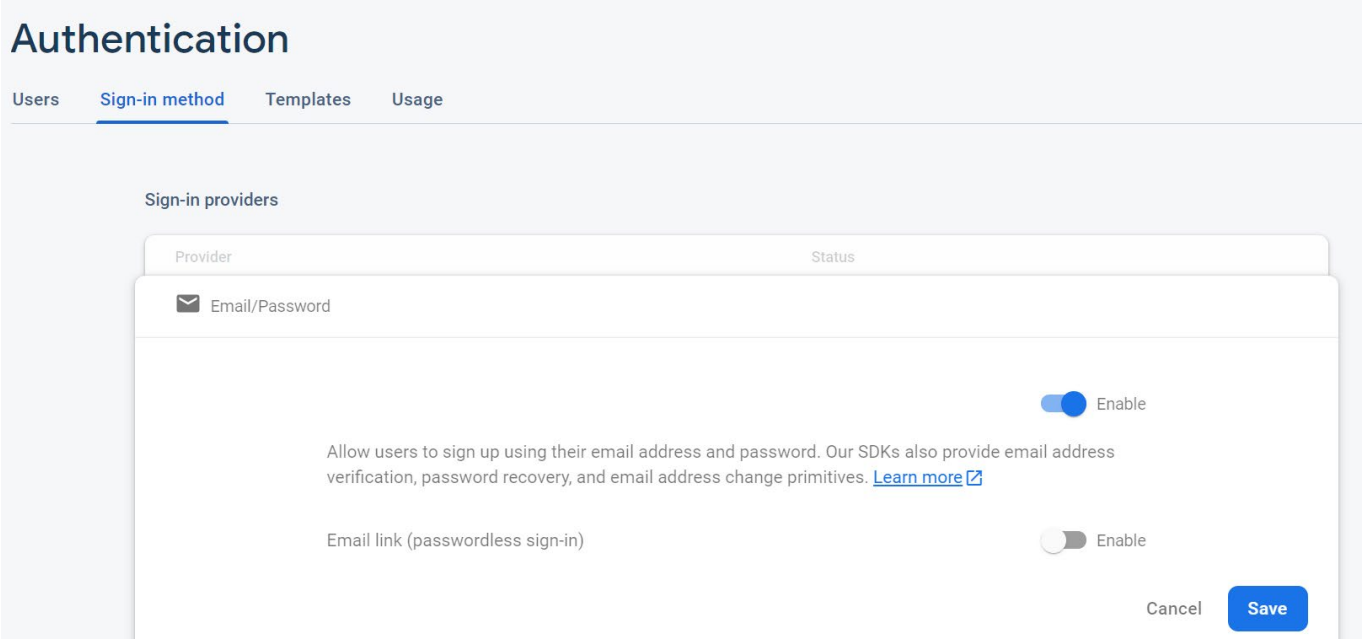



2. Select **Get started**.

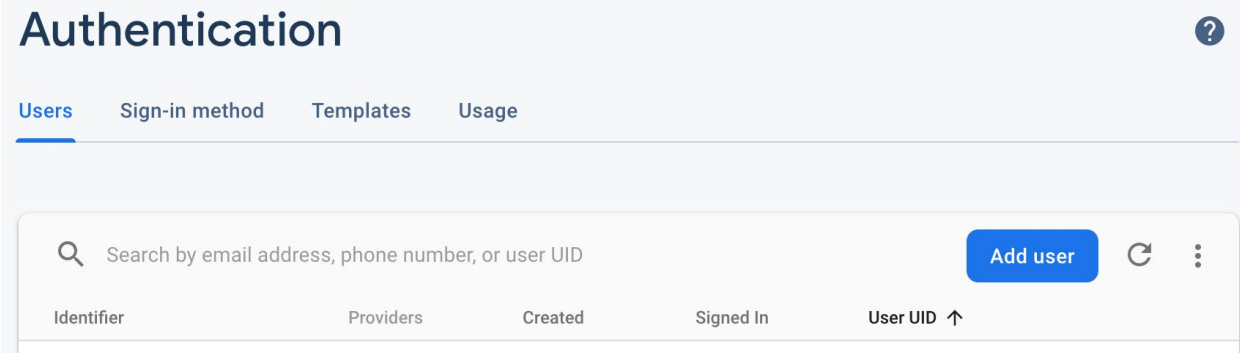
3. Edit **Email/Password**.



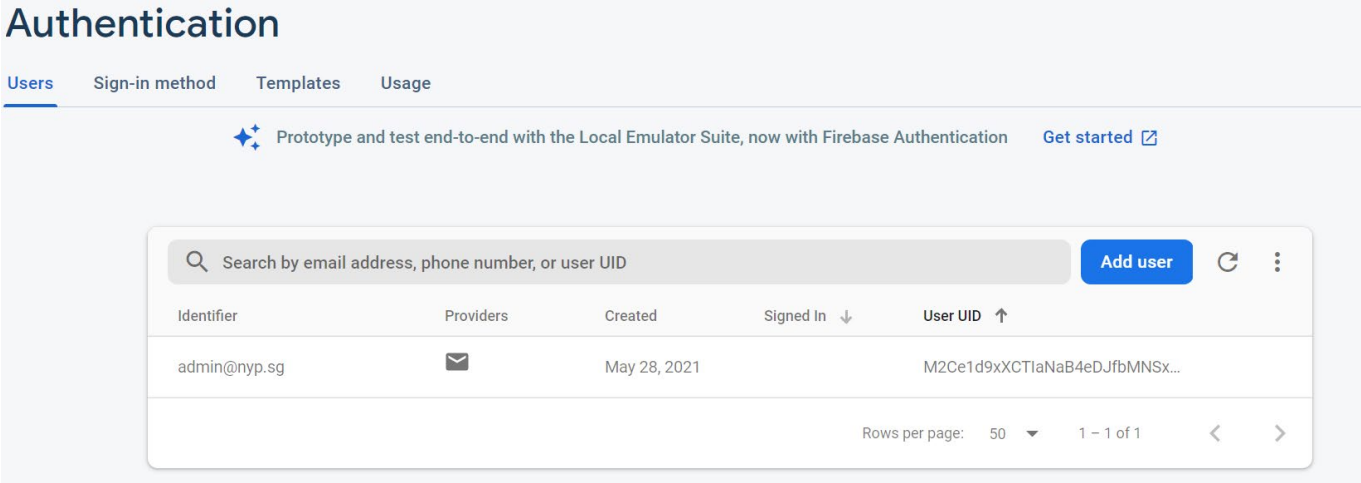
4. **Enable** email/password.
   Select **Save**.



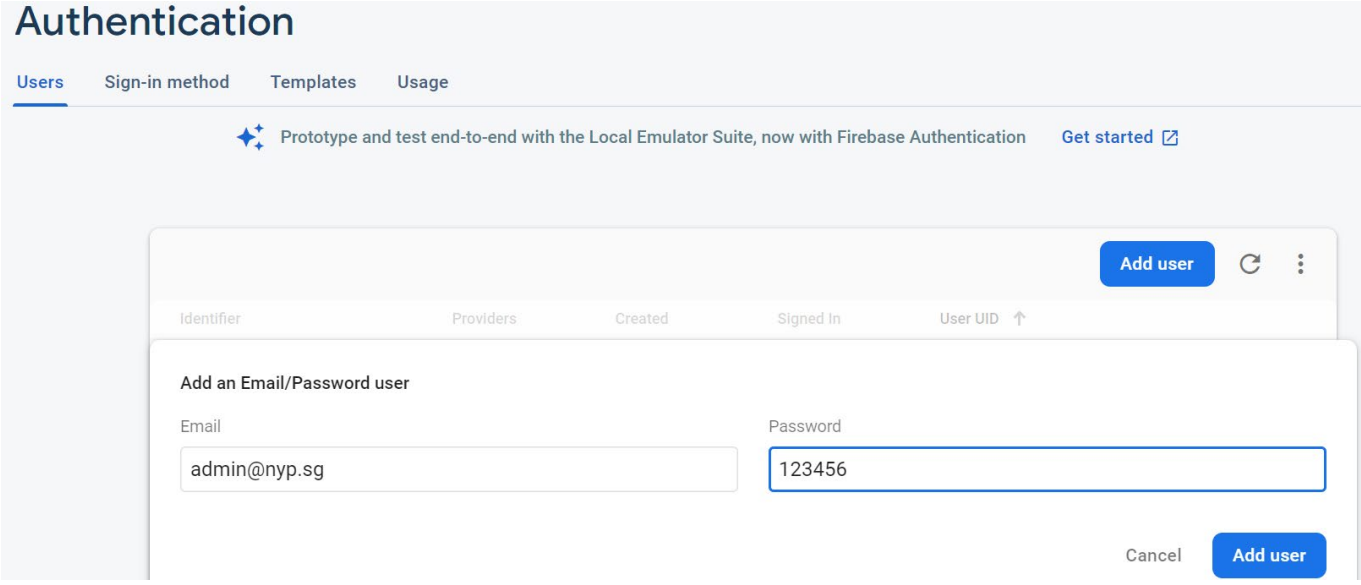

5. Select **Users** tab. Click **Add user**.



6. Fill in the email and password of an administrator account such as admin@nyp.sg. You will need minimum 6 characters as the password.




7.



# 2. Connect to Firebase

## 2.1. Open folder in code editor

1. Open a web code editor such as **Visual Studio Code**.
2. Select **File > Open Folder…** Select *skippyQ* folder.

## 2.2. Install firebase

1. In **Visual Studio Code**, select **View > Integrated Terminal**. You will see the Terminal window appear.
2. Make sure that you are at the *skippyQ* folder. Type the following command to add the **firebase** library to your project.

```
npm install firebase@8.6.3  --save
```

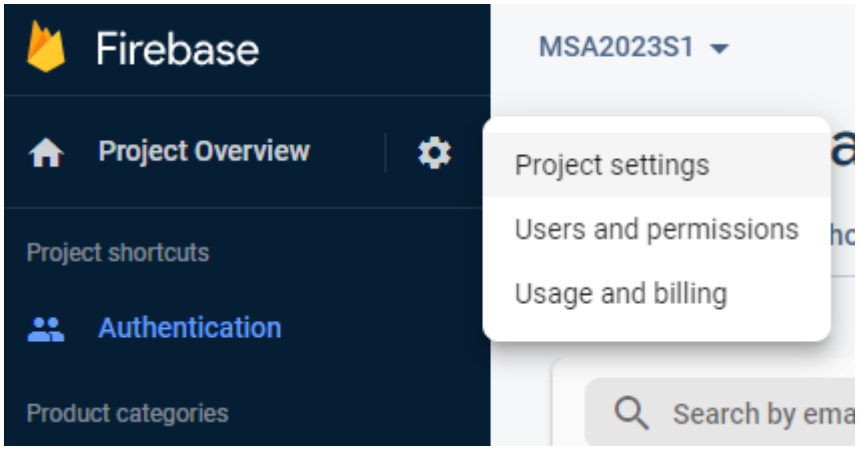
3. 

For Macs, add `sudo`

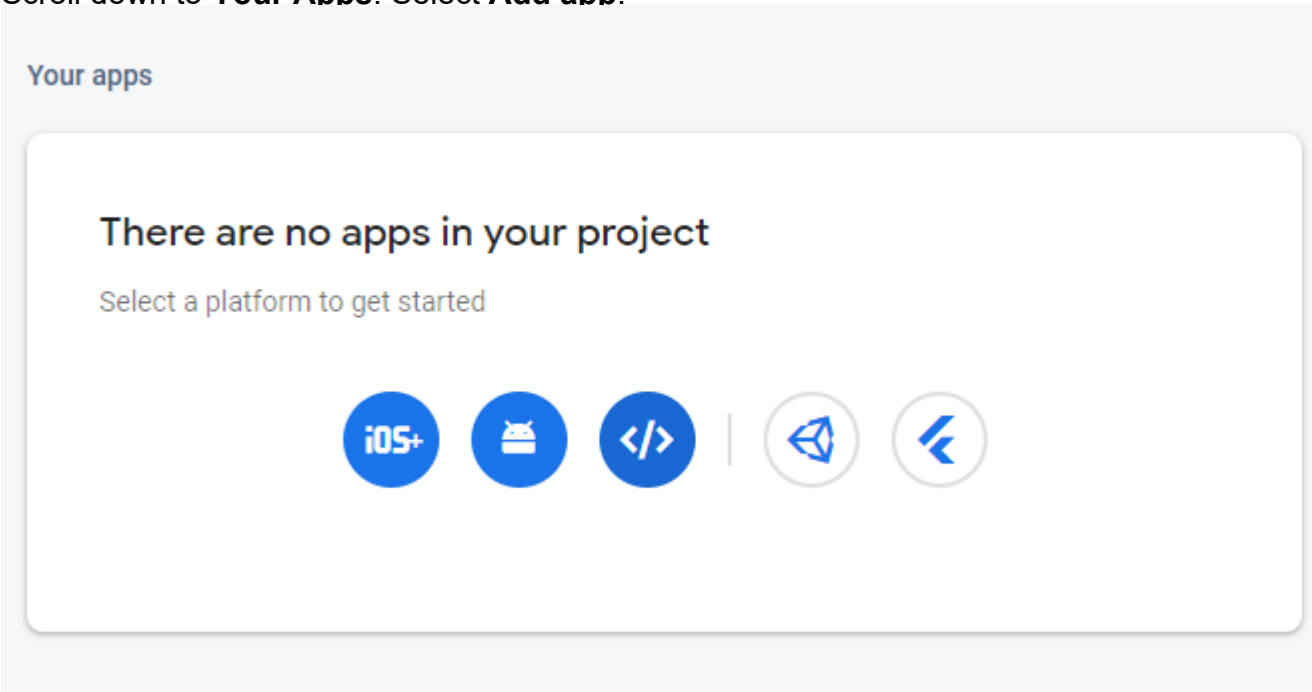
```
sudo npm install firebase@8.6.3 --save
```

## 2.3. Set up Firebase config

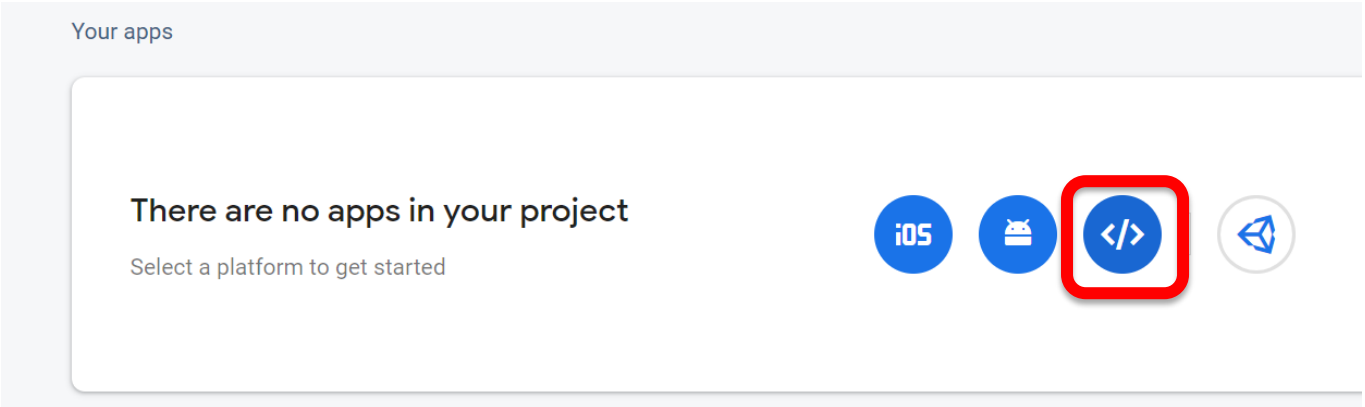
1. Go back to your **Firebase console** opened in browser. Select the **gear icon > Project Settings**.



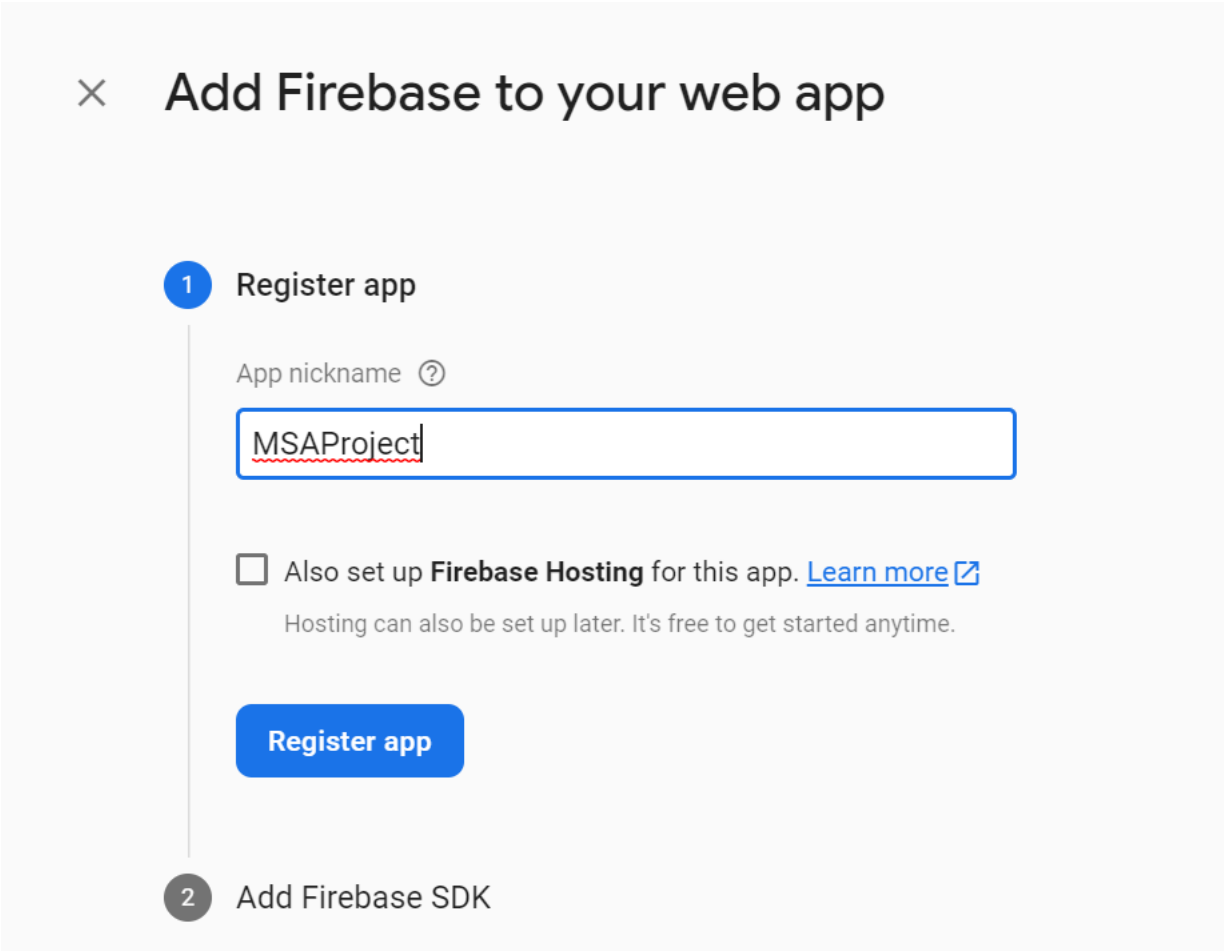
2. Scroll down to **Your Apps**. Select **Add app**.




3. Select **Web**




4. Click **Register app**

5. 


6. Copy the codes INSIDE the `<script>` tag.

2 Add Firebase SDK

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.6.3/firebase-app.js"></scri

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyCrclJ8dvSBBF3ZTwK1S0cKrVJU3mPnroE",
    authDomain: "msaproject-79797.firebaseapp.com",
    projectId: "msaproject-79797",
    storageBucket: "msaproject-79797.appspot.com",
    messagingSenderId: "736659417745",
    appId: "1:736659417745:web:f278ceb7f6b26705bbed30"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

7. Open *src > app > app.component.ts*.
   add `contructor()` method.
   Initialize firebase by *filling in firebaseConfig with that copied earlier*.

**app.component.ts**

```
  constructor() {

    // Your web app's Firebase configuration
      var firebaseConfig = {
        apiKey: "AIzaSyCrclJ8dvSBBF3ZTwK1S0cKrVJU3mPnroE",
        authDomain: "msaproject-79797.firebaseapp.com",
        projectId: "msaproject-79797",
        storageBucket: "msaproject-79797.appspot.com",
        messagingSenderId: "736659417745",
        appId: "1:736659417745:web:f278ceb7f6b26705bbed30"
      };
      // Initialize Firebase
      firebase.initializeApp(firebaseConfig);
  });
}
```

6. Import `firebase`

**app.component.ts**

```
import firebase from 'firebase';
```

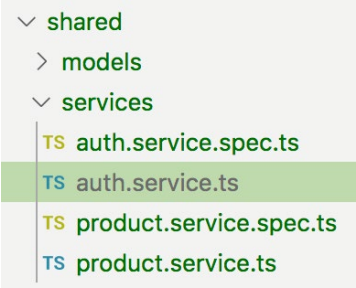
## 2.4. Create Authentication Service

Let's keep our firebase code in a service in case we change our mind and wish to use another service provider to manage user authentication.

1. Type the following command to generate ***Auth Service***.

```
ionic generate service shared/services/auth
```

2.




2. Open *src > app > shared > services > auth.service.ts*.
Import `firebase` and `firebase/auth`

**auth.service.ts**

```
import firebase from 'firebase';
import 'firebase/auth';
```

3.

# 3. Observe Authentication State

We want to observe the authentication state by keeping track of login and logout events.

1. Open *src > app > shared > services > auth.service.ts*.
   Add `observeAuthState()` method.

**auth.service.ts**

```
  observeAuthState(func: firebase.Observer<any, Error> | ((a: firebase.User | null) => any)
)
 {
    return firebase.auth().onAuthStateChanged(func);
  }
```

2. Open *src > app > tab3 > tab3.page.ts*.
   Inject ***AuthService***.
   Use **Quick Fix** to auto add the import.

**tab3.page.ts**

```
constructor(private modalController: ModalController,
            private authService: AuthService) { … }
```

3.

3. Store the user email when logged in by calling the service's `observeAuthState()` method.

**tab3.page.ts**

```
export class Tab3Page {
  userEmail: any;

  constructor(private modalController: ModalController,
              private authService: AuthService) {

    this.authService.observeAuthState(user => {
      // User is logged in
      if (user) {
         this.userEmail = user.email;
      }
      // User has logged out
      else {
        this.userEmail = undefined;
      }

    });
  }
…
```

4.

4. Open *src > app > tab3 > tab3.page.html*.
Next, we will display the user's email and **Logout** button if logged in.
If not logged in, we will display the **Login** button.

**tab3.page.html**

```
…
<ion-content [fullscreen]="true">
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">Account</ion-title>
    </ion-toolbar>
  </ion-header>

  <div *ngIf="userEmail === undefined">
    <ion-item>
      You are not logged in
    </ion-item>
    <ion-button expand="block" color="secondary" (click)=login()>Login</ion-button>
    <ion-button expand="block" color="secondary" fill="clear" (click)=signup()>Or Create an
Account</ion-button>
  </div>

  <div *ngIf="userEmail !== undefined">
    <ion-item>
      Hello {{userEmail}}
    </ion-item>
    <ion-button expand="block" color="secondary">Logout</ion-button>
  </div>

</ion-content>
```

5.

# 4. Login

1. Open *src > app > shared > services > auth.service.ts*.
   Add `login()` method.

**auth.service.ts**

```
login(email: string, password: string) {
  return firebase.auth().signInWithEmailAndPassword(email, password);
}
```

2.

2. Open *src > app > shared > login > login.page.ts*.
   Inject ***AuthService***.
   Use **Quick Fix** to auto add the import.

**login.page.ts**

```
  constructor(private modalController: ModalController,
              private authService: AuthService) { … }
```

Issue "ionic serve" again to recompile

3. In **Login Page**, add an `loginError` property.
   We want to be able to display an error to the user if he could not login such as password do not match.

**login.page.ts**

```
export class LoginPage implements OnInit {
  loginForm: FormGroup;
  loginError: string= "";
…
```

4.

4. In **Login Page**, find the `login()` method.
   Call the service's `login()` method.
   We wait for the login promise using `then()`. When the promise is fulfilled, it indicates successful login and we dismiss the modal.
   If there is an error, we will display the error while keeping the modal open.

**login.page.ts**

```
  login() {
    this.authService.login(
      this.loginForm.value.email, this.loginForm.value.password)
    .then(
      user => this.modalController.dismiss()
    )
    .catch(
      error => this.loginError = error.message
    );
  }
```

5.

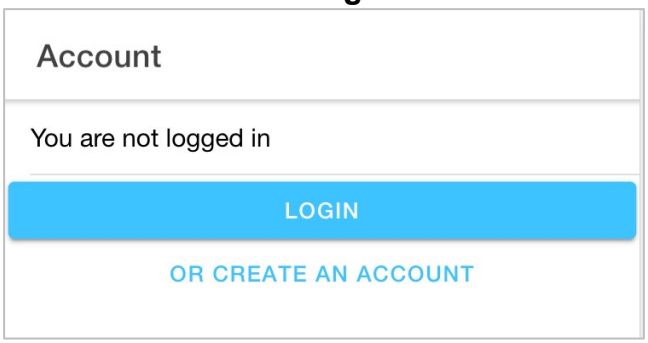
5. Open *src > app > shared > login > login.page.html*.
   Display the `loginError` below the form.

**login.page.html**

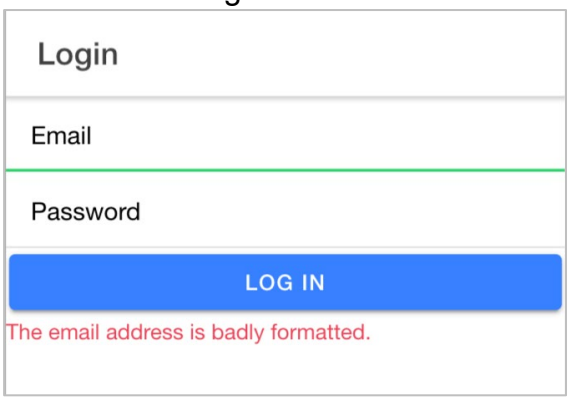
```
  </form>

  <ion-label color='danger'>
    <p>{{loginError}}</p>
  </ion-label>

</ion-content>
```

6.

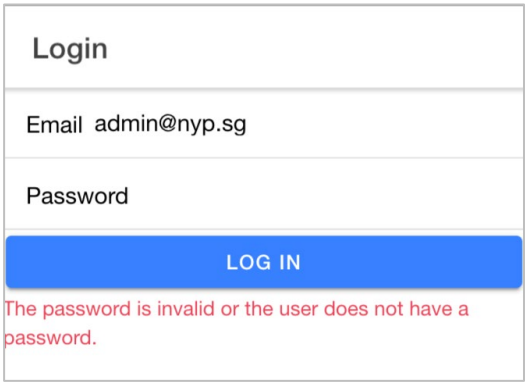
6. Run the app in your browser using ionic serve.
7. Notice that **Account Page** indicates that *'You are not logged in'*.



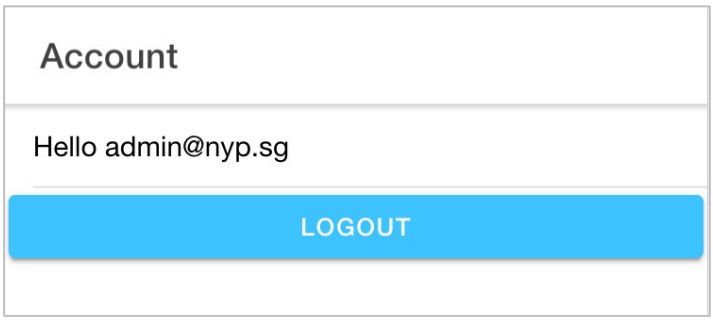

8. Click **Login** and try to login without email.
   An error message shows.




9. Try login without password or wrong password.
   An error message shows.



10. Try login with the correct email and password.
This email should be the one you have created in Firebase Authentication in the first Task.
The modal should dismiss and you are brought back to the **Account Page** showing your user email.




# 5. Logout

1. Open *src > app > shared > services > auth.service.ts*.
Add `logout()` method.

**auth.service.ts**

```
logout() {
  return firebase.auth().signOut();
}
```

2. Open *src > app > tab3 > tab3.page.ts*.
Add `logout()` method to call ***AuthService*** `logout()` method.

**tab3.page.ts**

```
logout() {
  this.authService.logout();
}
```

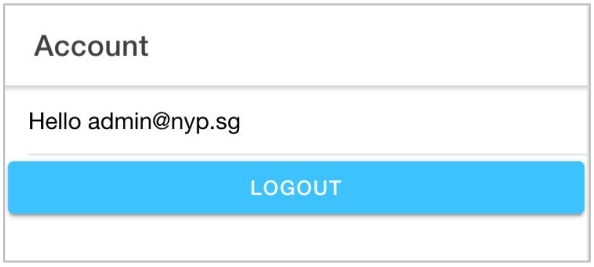
3. Open *src > app > tab3 > tab3.page.html*.
Find the **Logout** button. When clicked, call the `logout()` method.

**tab3.page.html**

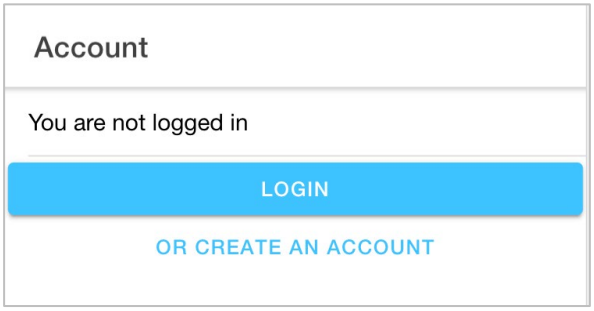
```
<ion-button expand="block" color="secondary" (click)=logout()>
  Logout
</ion-button>
```

4.

4. Go to **Account Page**. Click **Logout**.




5. You will see that the page changes to show the **Login** button.




# 6. Signup

1. Open *src > app > shared > services > auth.service.ts*. Add `signup()` method.

**auth.service.ts**

```
signup(email: string, password: string) {
  return firebase.auth().createUserWithEmailAndPassword(
    email, password);
}
```

2. Open *src > app > shared > signup > signup.page.ts*.
   Inject ***AuthService***.
   Use **Quick Fix** to auto add the import.

**signup.page.ts**

```
constructor(private modalController: ModalController,
            private authService: AuthService) { … }
```

Issue "ionic serve" again to recompile

3. In **Signup Page**, add a `signupError` property.
   We want to be able to display an error to the user if he could not signup such as password does not have minimum 6 characters.

**signup.page.ts**

```
export class SignupPage implements OnInit {
  signupForm: FormGroup;
  signupError: string ="";
…
```

4. In **Signup Page**, find the `signup()` method.
   Call the service's `signup ()` method.
   We wait for the signup promise using `then()`. When the promise is fulfilled, it indicates successful signup and we dismiss the modal.
   If there is an error, we will display the error while keeping the modal open.

**signup.page.ts**

```
signup() {
  this.authService.signup(
    this.signupForm.value.email,
    this.signupForm.value.password)
    .then(user => this.modalController.dismiss()
    )
    .catch(
      error => this.signupError = error.message
    );
}
```

5.

5. Open *src > app > shared > signup > signup.page.html*.
   Display the `signupError` below the form.

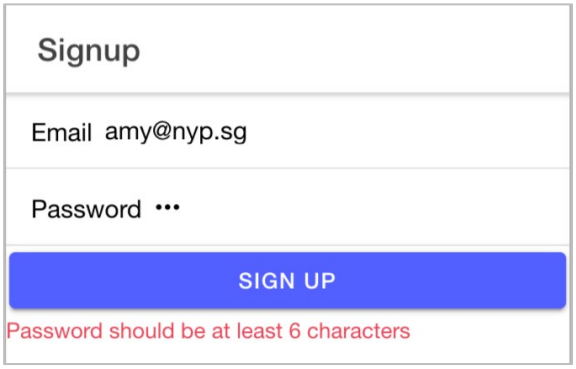
**signup.page.html**

```
…
  </form>

  <ion-label color='danger'>
    <p>{{signupError}}</p>
  </ion-label>

</ion-content>
```

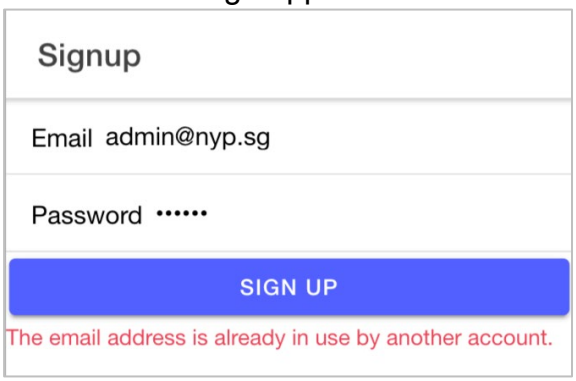
6.

6. Preview your app in the browser.
   Try Sign up with invalid Password.

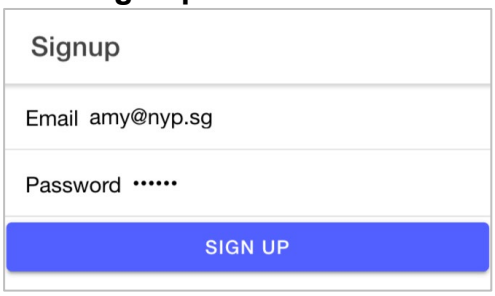
An error message appears.



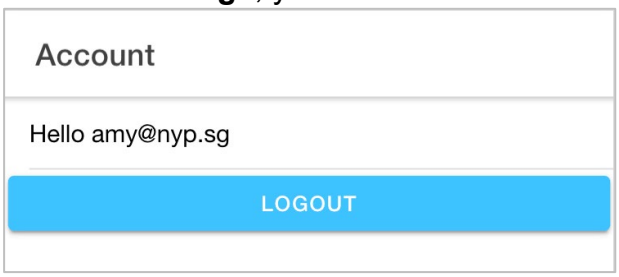

7. Try Sign up with an existing Email.
An error message appears.



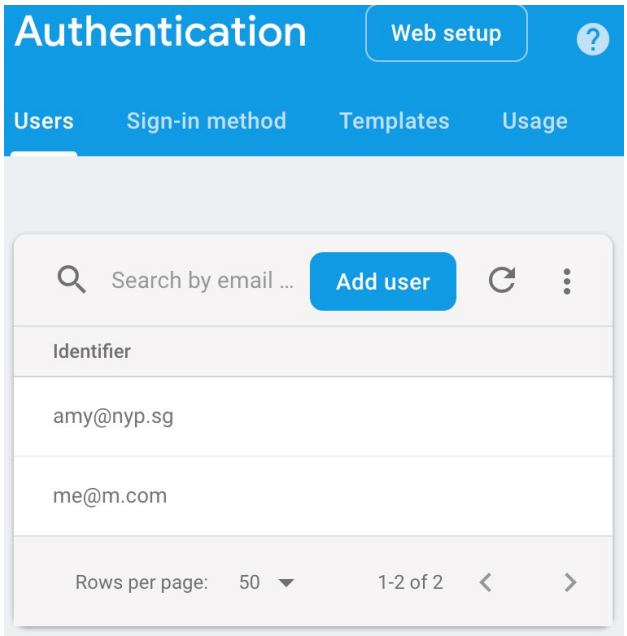

8. Type valid email and password.
Click **Sign up**.




9. In **Account Page**, you will see that the new user is auto logged in after sign up.



10. Go to **Firebase console** and verify that the new user is created.




# 7. *(Optional)* Customize Tab Buttons based on User Role

Only an administrator has the right to add, edit and delete products in the **Catalogue Page**.

In this task, we are going to show the **Catalogue** Tab Button only if the user is an admin, i.e. logs in with the email admin@nyp.sg. Refer to the admin account you have created in the first task when you enable Firebase Authentication.

1. Open *src > app > tabs > tabs.page.html*.
   Show **Catalogue** Tab Button `tab2` only if the user is an admin.

**tabs.page.html**

```
<ion-tab-button tab="tab2" *ngIf="isAdmin">
  <ion-icon name="folder"></ion-icon>
  <ion-label>Catalogue</ion-label>
</ion-tab-button>
```

2.

2. Open *src > app > tabs > tabs.page.ts*.
   Add the `isAdmin` property initialized to *false*.

**tabs.page.ts**

```
export class TabsPage {
  isAdmin = false;
  …
```

3. Inject ***AuthService***.
   Use **Quick Fix** to auto add the import.

**tabs.page.ts**

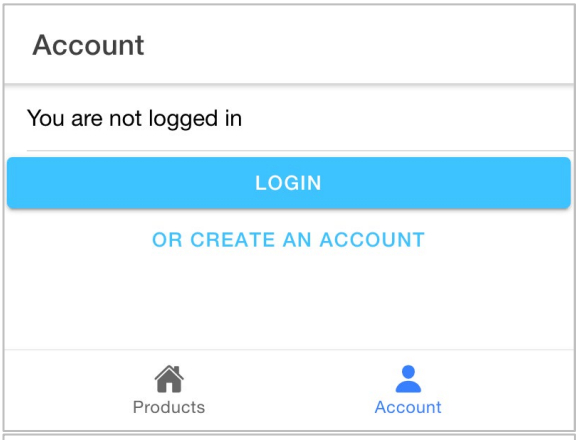
```
constructor(private authService: AuthService) { }
```

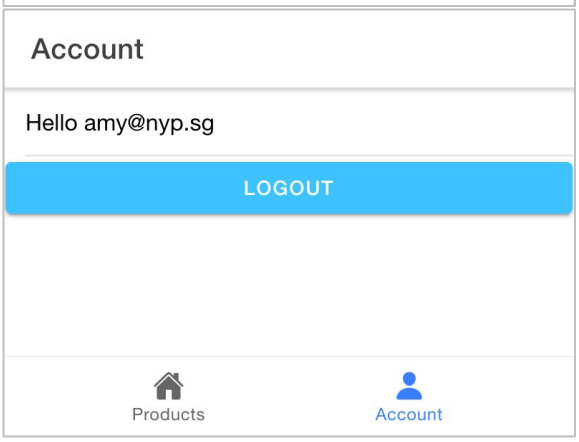
4. Observe the authentication state changes. Set `isAdmin` to *true* only if the user's email is *'admin@nyp.sg'*.

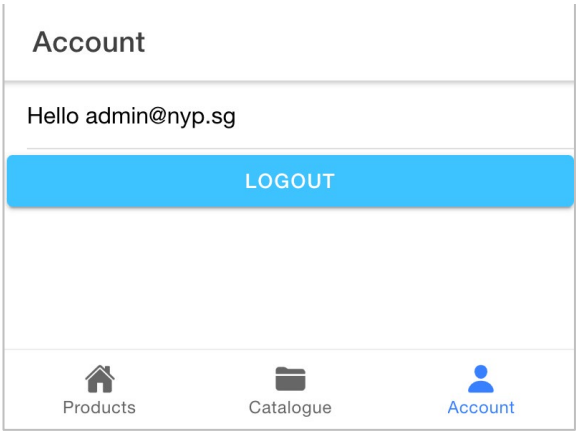
**tabs.page.ts**

```
constructor(private authService: AuthService) {
  this.authService.observeAuthState(user => {
    // User is logged in as administrator
    // For simplicity, there is only one fixed admin
    // Further enhancement would be to save the user role in Database
    if (user && user.email == 'admin@nyp.sg') {
      this.isAdmin = true;
    }
    // User has logged out or is NOT administrator
    else {
      this.isAdmin = false;
    }
  });
}
```

5. Preview your app in the browser.




*~ End ~*