AuthGuard:
In my Angular application, I've utilized `AuthGuard` with the `canActivate` method to protect certain routes. This setup checks if the user is authenticated and, based on their authentication status, either grants access to protected routes or redirects them to the login page.

```
@Injectable({ providedIn: 'root' })
export class AuthGuard implements CanActivate {
  constructor(private backendService: BackendService, private router: Router) {}

  canActivate(
    route: ActivatedRouteSnapshot,
    router: RouterStateSnapshot
  ):
    | boolean
    | UrlTree
    | Promise<boolean | UrlTree>
    | Observable<boolean | UrlTree> {
    return this.backendService.user.pipe(
      take(1),
      map(user => {
        const isAuth = !!user;
        if (isAuth) {
          return true;
        }
        return this.router.createUrlTree(['/login']);
      })
```

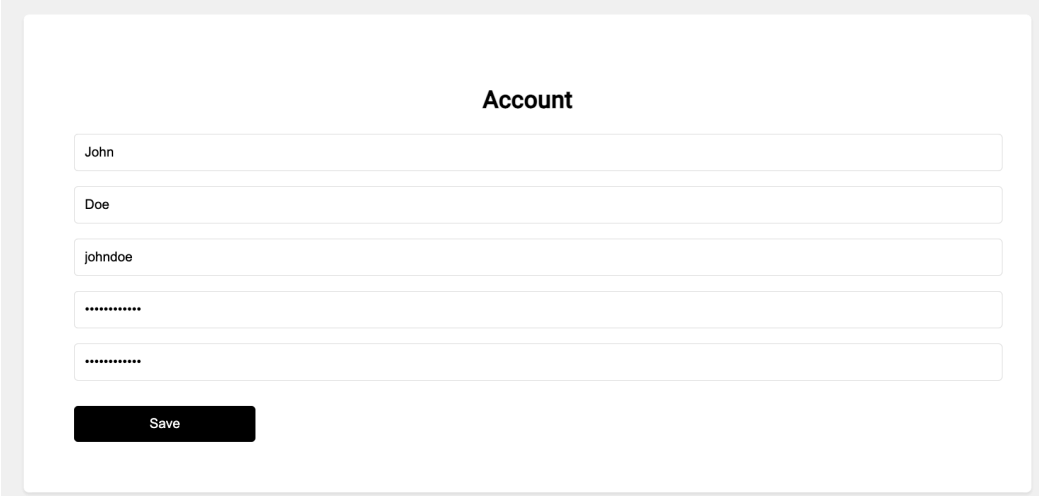
Pre-Populating the Profile Form:

```
export class User {
    constructor(
      public firstName: string,
      public lastName: string,
      public username: string,
      public _token?: string,
      public _tokenExpiry?: Date
    ) { }

    getFullName(): string {
      return `${this.firstName} ${this.lastName}`;
    }

    get token(){
        if (!this._tokenExpiry || new Date() > this._tokenExpiry){
            return null;
        }
        return this._token;
    }
  }
```

In my BackendService, I create a BehaviorSubject to hold the user data. I initialize it with a default value, which could be null or an initial user object, depending on my application's needs. Since BehaviorSubject always holds a value, as soon as my component subscribes to it, it receives the latest user information, which I then use to pre-populate the profile form.

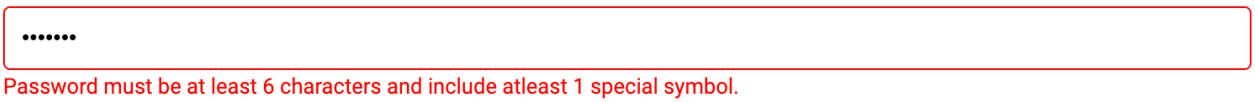


Router:
Used Angular Router to add the following routes:
http://localhost:4200/account
http://localhost:4200/register
http://localhost:4200/login
http://localhost:4200/

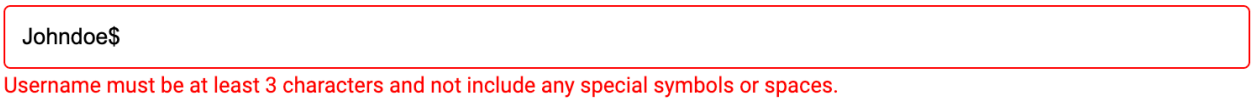
```
const appRoutes: Routes= [
    {path: 'login', component: LoginComponent},
    {path: 'register', component: RegisterComponent},
    {path: 'account', component: AccountComponent, canActivate:[AuthGuard]},
    {path: '', component: HomeComponent},
];
```

Dynamic Navigation Bar to reflect the authentication state

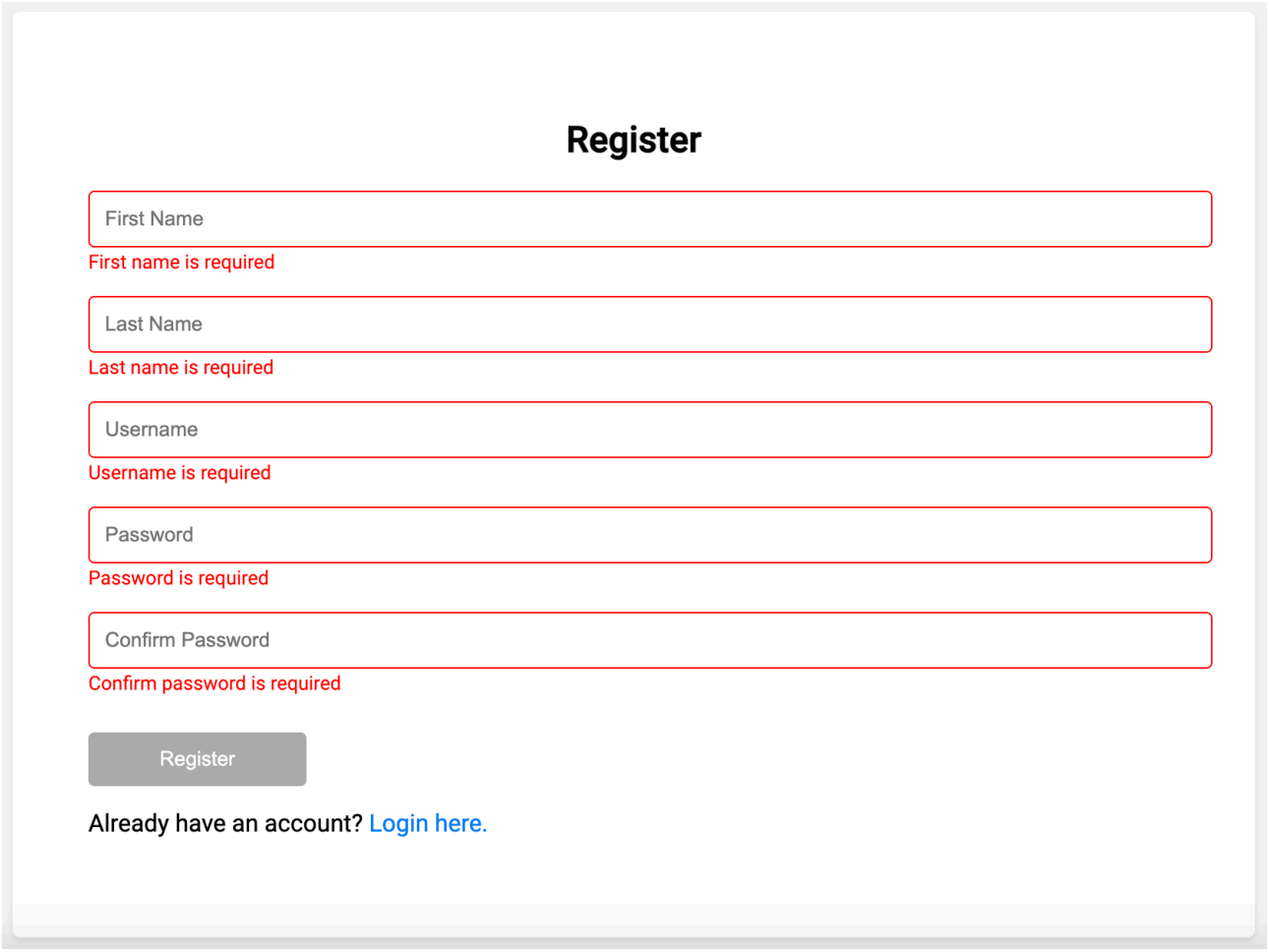
Validation:
Password Validation

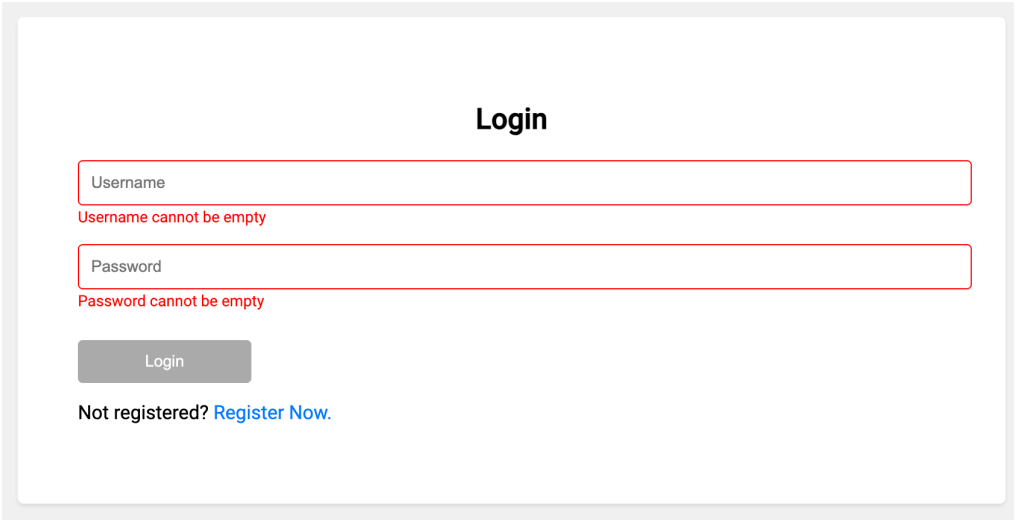


UserName Validation:

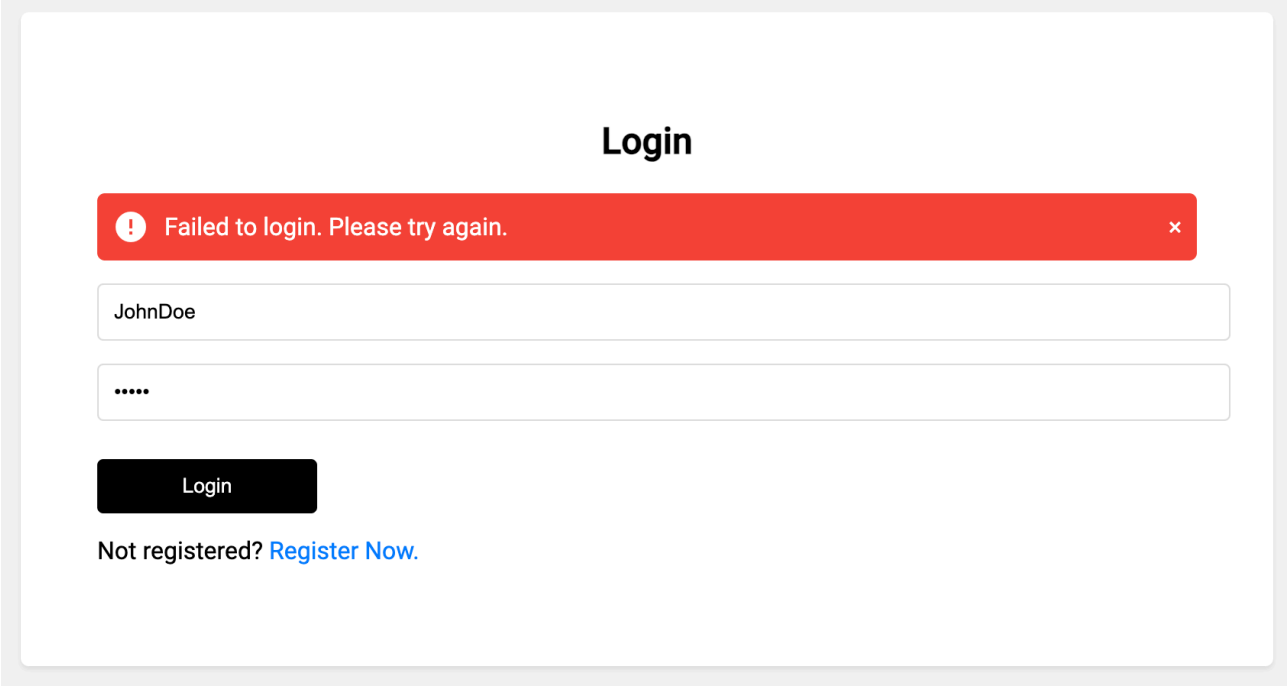


Confirm Password Validation:


Register
First Name
First name is required
Last Name
Last name is required
Username
Username is required
Password
Password is required
Confirm Password
Confirm password is required
Register
Already have an account? Login here.


Login
Username
Username cannot be empty
Password
Password cannot be empty
Login
Not registered? Register Now.

Error Handling:




```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { User } from './user.model';
import { BehaviorSubject } from 'rxjs';
import { tap } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class BackendService {
  private backendUrl = 'http://localhost:3000'; // Replace with your actual backend URL
  user = new BehaviorSubject<User | null>(null)
  constructor(private http: HttpClient) { }

  //  username ,frist name, last name, password, confirmpassword and token
  saveData(data: any) {
    return this.http.post(`${this.backendUrl}/save`, data); // Adjust endpoint as necessary
  }

  // username nad password
  login(credentials: { username: string, password: string }) {
    return this.http.post(`${this.backendUrl}/login`, credentials);
  }

  // username ,frist name, last name, password, confirmpassword
  register(userInfo: { firstName: string; lastName: string; username: string; password: string; }) {
    return this.http.post(`${this.backendUrl}/register`, userInfo).pipe(
      tap((response: any) => {
        console.log(response);
        if (response) {
          const user = new User(
            response.firstName,
            response.lastName,
            response.username,
```

Connection to the backend: