# Angular Best Friends

## Module 6 Exercise 6 – Route guards

## Goals

In most of the applications there are different user types and each user type may have or have not access to certain routes. So we need to practice how to implement route guards to make sure that our application meets the needed security requirements

## Steps

1. Go to the “**Module6Exercise6 > initial**” folder
2. Right click it and open he folder in VS Code
3. In the terminal window run **npm install** to install the needed node modules. This step is needed only when you run the application for the first time.
4. We’ll use the terminal to create a new guard. So open the terminal and run the following command:

**Command:** ng g g user/auth

This will add the guard to the userfolder

1. Now in the “user” folder you should see the “**auth.guard.ts**” file with some default generated code.
2. Delete all the code in the file and replace it with the following snippet:

import { Injectable } from '@angular/core';

import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';

import { Observable } from 'rxjs';

import { AuthService } from './auth.service';

@Injectable({

providedIn: 'root'

})

export class AuthGuard implements CanActivate {

constructor(private authService: AuthService,

private router: Router) { }

canActivate(

next: ActivatedRouteSnapshot,

state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

return this.checkLoggedIn(state.url);

}

checkLoggedIn(url: string): boolean {

if (this.authService.isLoggedIn) {

return true;

}

this.authService.redirectUrl = url;

this.router.navigate(['/login']);

return false;

}

}

Some things that we do here. First we import the auth service and inject it into the constructor. We do the same with the Router module. Then we create a checkLoggedIn function that implements some logic to check if a current session is logged in or not. It returns a bool. Then, in the guard itself we simply call this method.

1. Now we need to add the guard to the **product.module.ts** file. Guards are always added to the route definition. Import the created AuthGuard there:

import { AuthGuard } from '../user/auth.guard';

1. Right under “path:”/products”” add the following line:

canActivate: [AuthGuard],

1. Run the application and play around. Right now, if you try to navigate to the product list or any other child path you will be redirected to the login page.