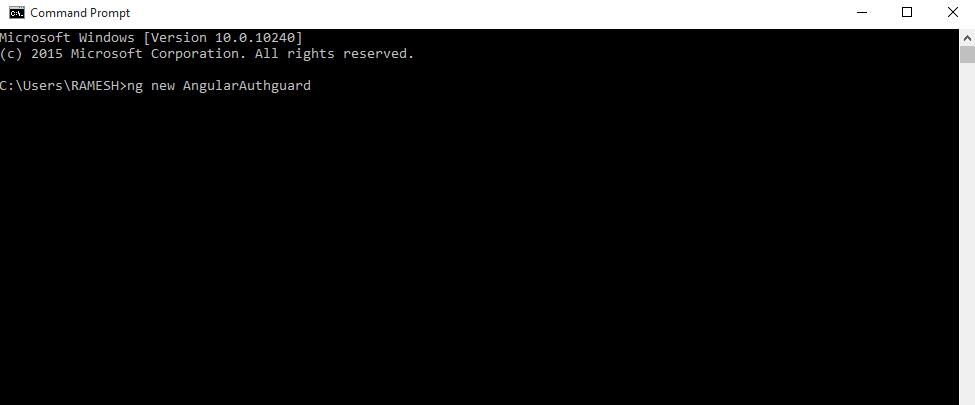
**Introduction**

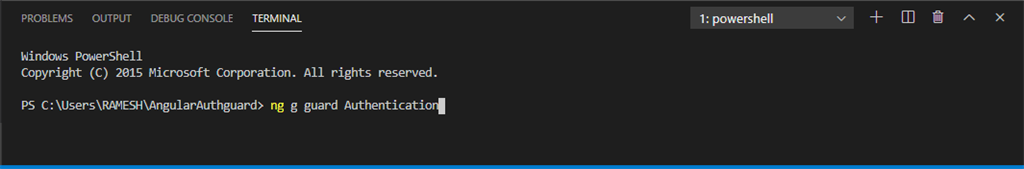
* As we all know that without proper authentication, no one should access the protected information of our web application.
* no one can access sensitive information without proper authentication. This can be achieved by the use of **Auth guard** in Angular.
* Auth-guard makes use of CanActivate interface and it checks for if the user is logged in or not. If it returns true, then the execution for the requested route will continue, and if it returns false, that the requested route will be kicked off and the default route will be shown.
* Guarding routes means whether we can visit a route or not. We can use a route guard in Angular using these simple steps:

**Step 1 - Create an angular app**



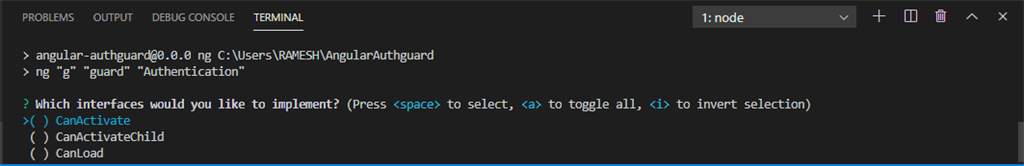
**Step 2**

Create Authguard with the following command, ''ng g guard <Authguardname>"



**Step 3**

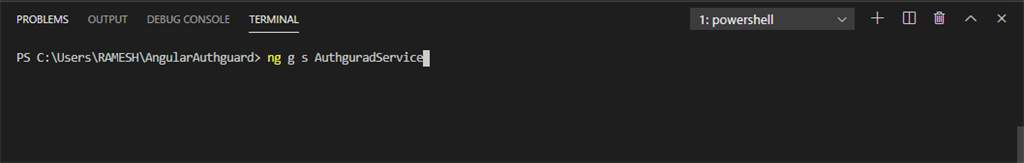
The next step is to choose the CanActivate



CanActivateInterface -Canactivate it is an angular interface.used for users to log in to the application before navigating the route.

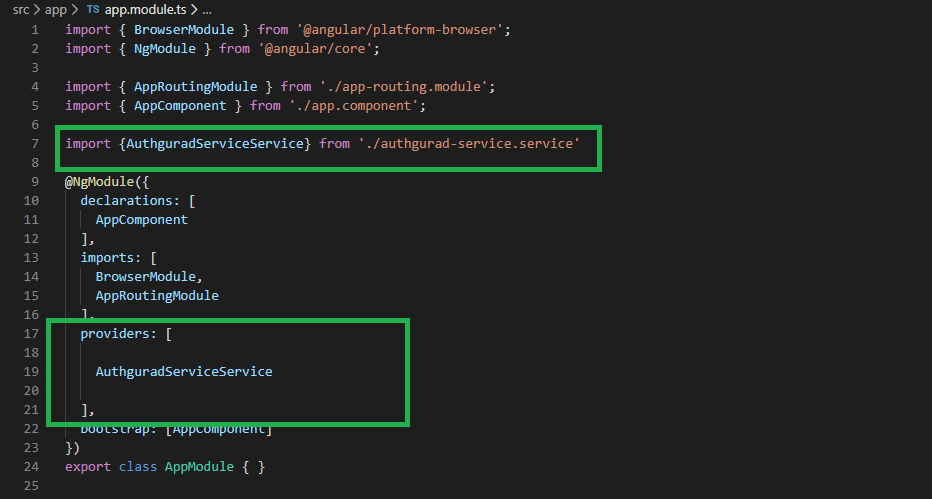
**Step 4**

Create a service using the following command, “ng g s <myService>”.



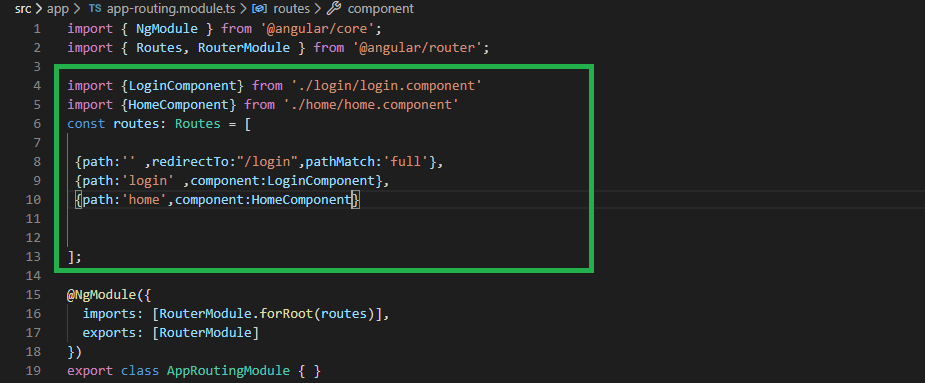
**Step 5**

Once you create the service file, you can add appmodule.ts with the following code



**Step 6 - Routing**

Let's create the component and add the routing.ts file, given below :



**Step 7**

Open the authentication.guard.ts file and change the code to what is given below:

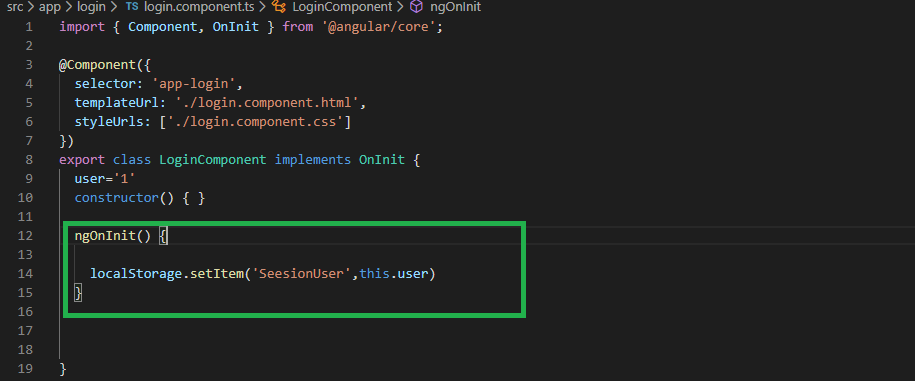


Boolean - Represents value in two states: true or false.

**Step 8**

Store the user session details in your login page and call the ngOnInt, for giving the below code

1. ngOnInit() {
2. localStorage.setItem('SeesionUser',**this**.user)
3. }



**Step 9**

Open AuthguardService.ts file get your local storage data

1. **import** { Injectable } from '@angular/core';
3. @Injectable({
4. providedIn: 'root'
5. })
6. **export** **class** AuthguradServiceService {
7. constructor() {
8. }
9. gettoken(){
10. **return** !!localStorage.getItem("SeesionUser");
11. }
12. }

**Step 10**

The canActivate() method from AuthGuard returns true only when the route can be navigated. In the case of a false value, navigation can be redirected to the login page.

1. constructor(**private** Authguardservice: AuthguradServiceService, **private** router: Router) {}
2. canActivate(): **boolean** {
3. **if** (!**this**.Authguardservice.gettoken()) {
4. **this**.router.navigateByUrl("/login");
5. }
6. **return** **this**.Authguardservice.gettoken();
7. }

**Step 11**

Open the app.component.html file. You can add the routeroutlet following the below code

1. <router-outlet></router-outlet>

**Step 12**

Use canActivate property of Route interface to guard the route and assign service class implementing CanActivate interface, for example, AuthGuard. Now find the canActivate property used in route declarations.

