You can implement role-based navigation restrictions in your Angular project using \*\*Angular Guards\*\* (`CanActivate` and `CanActivateChild`). Follow these steps:

### \*\*1. Define a Mock Authentication Service\*\*

Create an `AuthService` to store and provide user role information.

#### \*\*auth.service.ts\*\*

```typescript

Import { Injectable } from ‘@angular/core’;

@Injectable({

providedIn: ‘root’

})

Export class AuthService {

Private userRole: string = ‘’; // Default role

Constructor() {}

setRole(role: string) {

this.userRole = role;

}

getRole(): string {

return this.userRole;

}

isAuthenticated(): boolean {

return this.userRole !== ‘’;

}

}

```

### \*\*2. Implement Role-Based Route Guards\*\*

Create two guards: `RoleGuard` (`CanActivate`) and `ChildRoleGuard` (`CanActivateChild`).

#### \*\*role.guard.ts\*\*

```typescript

Import { Injectable } from ‘@angular/core’;

Import { CanActivate, Router } from ‘@angular/router’;

Import { AuthService } from ‘../services/auth.service’;

@Injectable({

providedIn: ‘root’

})

Export class RoleGuard implements CanActivate {

Constructor(private authService: AuthService, private router: Router) {}

canActivate(): boolean {

if (this.authService.isAuthenticated()) {

return true;

} else {

This.router.navigate([‘/403’]); // Redirect to Forbidden page

Return false;

}

}

}

```

#### \*\*child-role.guard.ts\*\*

```typescript

Import { Injectable } from ‘@angular/core’;

Import { CanActivateChild, Router } from ‘@angular/router’;

Import { AuthService } from ‘../services/auth.service’;

@Injectable({

providedIn: ‘root’

})

Export class ChildRoleGuard implements CanActivateChild {

Constructor(private authService: AuthService, private router: Router) {}

canActivateChild(): boolean {

const role = this.authService.getRole();

if (role === ‘Admin’ || role === ‘Manager’) {

return true;

} else {

This.router.navigate([‘/403’]); // Redirect to Forbidden page

Return false;

}

}

}

```

### \*\*3. Define Routes with Guards\*\*

Modify your `app-routing.module.ts` to apply guards.

#### \*\*app-routing.module.ts\*\*

```typescript

Import { NgModule } from ‘@angular/core’;

Import { RouterModule, Routes } from ‘@angular/router’;

Import { RoleGuard } from ‘./guards/role.guard’;

Import { ChildRoleGuard } from ‘./guards/child-role.guard’;

Import { LoginComponent } from ‘./login/login.component’;

Import { OrderManagementComponent } from ‘./ordermanagement/ordermanagement.component’;

Import { UserManagementComponent } from ‘./usermanagement/usermanagement.component’;

Import { ProductManagementComponent } from ‘./productmanagement/productmanagement.component’;

Import { ForbiddenComponent } from ‘./forbidden/forbidden.component’;

Const routes: Routes = [

{ path: ‘’, component: LoginComponent },

{

Path: ‘ordermanagement’,

Component: OrderManagementComponent,

canActivate: [RoleGuard]

},

{

Path: ‘usermanagement’,

Component: UserManagementComponent,

canActivate: [RoleGuard],

canActivateChild: [ChildRoleGuard]

},

{

Path: ‘productmanagement’,

Component: ProductManagementComponent,

canActivate: [RoleGuard]

},

{ path: ‘403’, component: ForbiddenComponent },

{ path: ‘\*\*’, redirectTo: ‘’ } // Redirect unknown paths to login

];

@NgModule({

Imports: [RouterModule.forRoot(routes)],

Exports: [RouterModule]

})

Export class AppRoutingModule { }

```

### \*\*4. Handle Login and Role Assignment\*\*

Modify your `login.component.ts` to set user roles.

#### \*\*login.component.ts\*\*

```typescript

Import { Component } from ‘@angular/core’;

Import { Router } from ‘@angular/router’;

Import { AuthService } from ‘../services/auth.service’;

@Component({

Selector: ‘app-login’,

templateUrl: ‘./login.component.html’,

styleUrls: [‘./login.component.css’]

})

Export class LoginComponent {

selectedRole: string = ‘Admin’; // Default role

constructor(private authService: AuthService, private router: Router) {}

login() {

this.authService.setRole(this.selectedRole);

this.router.navigate([‘/ordermanagement’]); // Redirect to dashboard

}

}

```

#### \*\*login.component.html\*\*

```html

<select [(ngModel)]=”selectedRole”>

<option value=”Admin”>Admin</option>

<option value=”Manager”>Manager</option>

<option value=”Customer Support”>Customer Support</option>

</select>

<button (click)=”login()”>Login</button>

```

### \*\*5. Create a Forbidden Page\*\*

Create a `ForbiddenComponent` to show a \*\*403 Forbidden\*\* message.

#### \*\*forbidden.component.ts\*\*

```typescript

Import { Component } from ‘@angular/core’;

@Component({

Selector: ‘app-forbidden’,

Template: `<h2>403 Forbidden</h2><p>You do not have access to this page.</p>`,

styleUrls: [‘./forbidden.component.css’]

})

Export class ForbiddenComponent {}

```

### \*\*How It Works\*\*

1. \*\*User logs in\*\* and selects a role.

2. \*\*Role is stored\*\* in `AuthService`.

3. \*\*Route Guards check role\*\*:

- `RoleGuard` blocks unauthorized users.

- `ChildRoleGuard` restricts access to `usermanagement` based on role.

4. \*\*Unauthorized users are redirected\*\* to the \*\*403 Forbidden page\*\*.

This implementation ensures only authorized users can access the right sections of your \*\*E-Commerce Admin Panel\*\*.

Would you like any modifications?