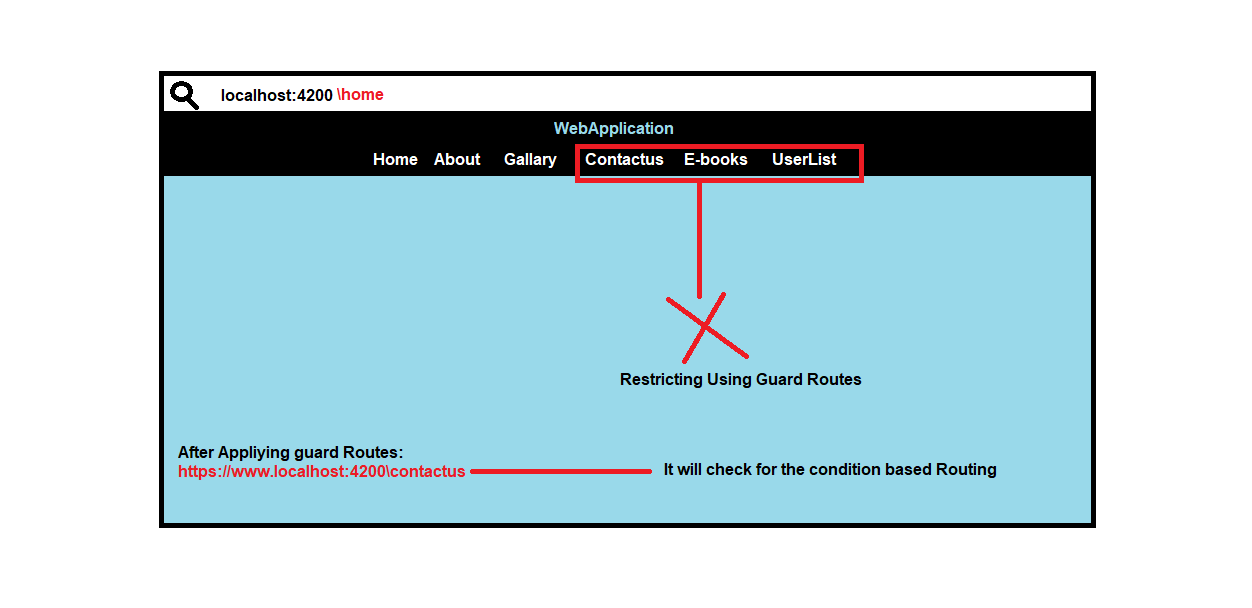
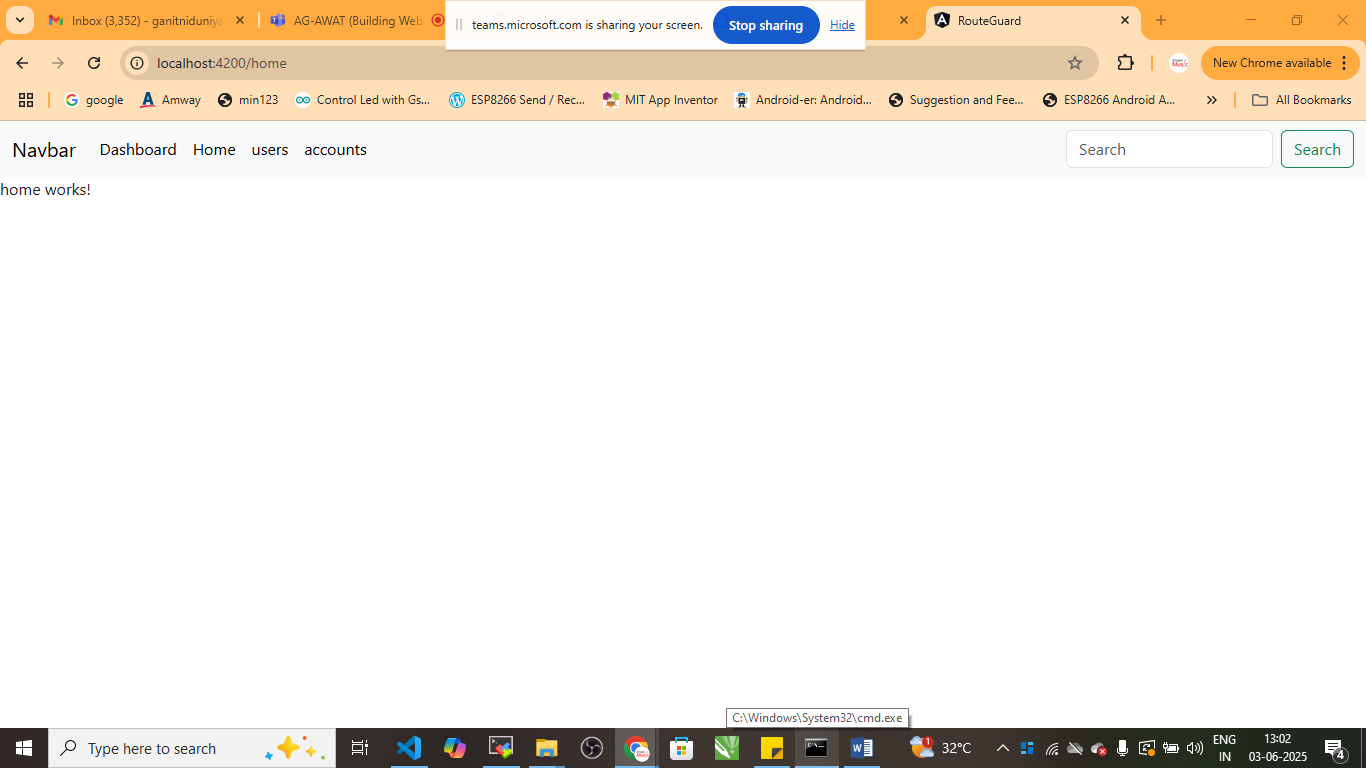
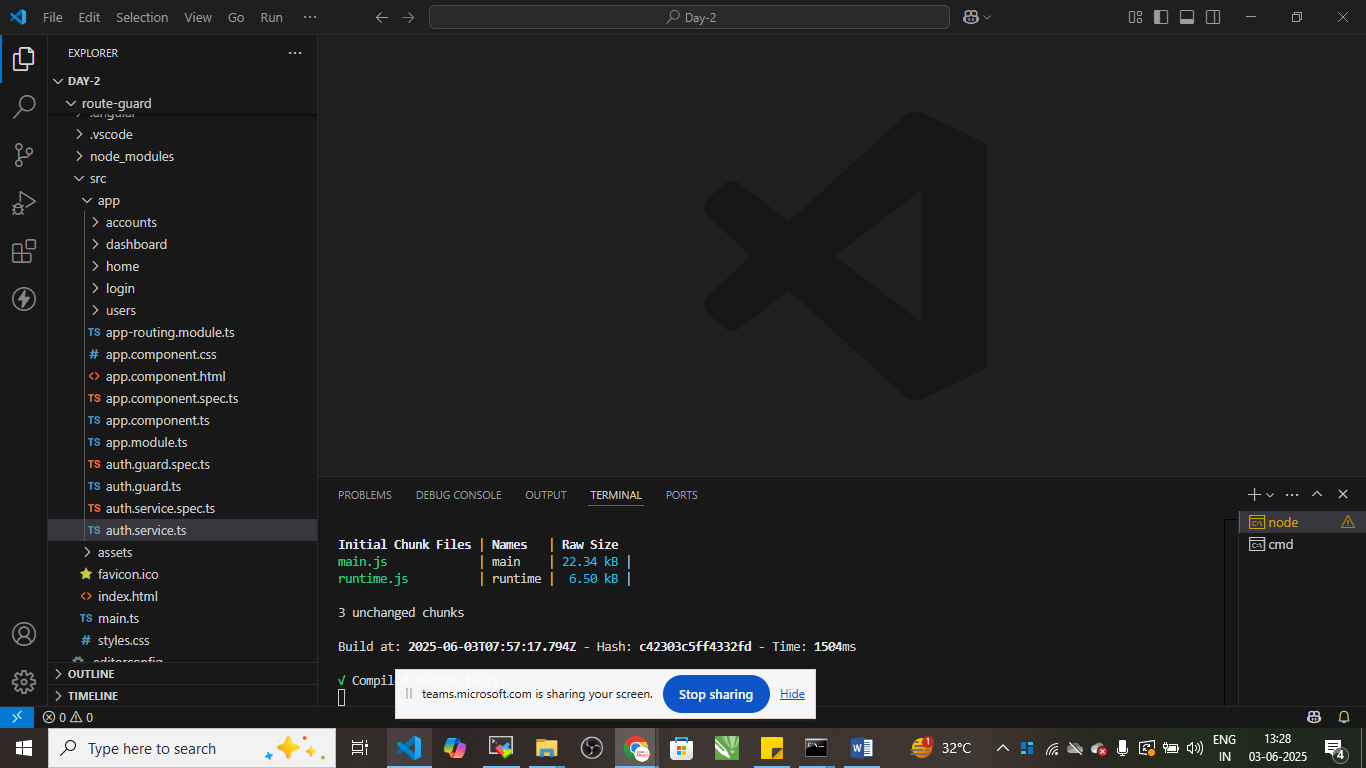
Route Guards

* In angular 15 , Route Guards are Used to control access to routes based on conditions like authentication, roles, or unsaved changes. The most common guards includes:
  + - **Can Activate**
    - **CanActivateChild**
    - **CanDeactivate**
    - **Resolve or CanMatch**





* Step:1 Create Multiple components



1. Home : ng g c home
2. Users: ng g c user
3. Login: ng g c login
4. Dashbords: ng g c dashboard
5. Accounts: ng g c accounts

Step:2 Create Navigation

App.component.html

<nav class="navbar navbar-expand-lg bg-body-tertiary">

  <div class="container-fluid">

    <a class="navbar-brand" href="#">Navbar</a>

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">

      <ul class="navbar-nav me-auto mb-2 mb-lg-0">

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="login" href="#">login</a>

        </li>

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="dashboard" href="#">Dashboard</a>

        </li>

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="home" href="#">Home</a>

        </li>

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="users" href="#">users</a>

        </li>

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="accounts" href="#">accounts</a>

        </li>

      </ul>

      <form class="d-flex" role="search">

        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search"/>

        <button class="btn btn-outline-success" type="submit">Search</button>

      </form>

    </div>

  </div>

</nav>

<router-outlet></router-outlet>

Create routes for the same

App.routing.module.ts

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

import { RouterModule, Routes } from '@angular/router';

import { DashboardComponent } from './dashboard/dashboard.component';

import { HomeComponent } from './home/home.component';

import { UsersComponent } from './users/users.component';

import { AccountsComponent } from './accounts/accounts.component';

import { LoginComponent } from './login/login.component';

import { authGuard } from './auth.guard';

const routes: Routes = [

  {path:"login",component:LoginComponent},

  {path:"dashboard",

    component:DashboardComponent,

  },

  {path:"home",component:HomeComponent },

  {path:"users",component:UsersComponent, },

  {path:"accounts",component:AccountsComponent, }

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

Step:3 create Auth Service

* ng g s auth

auth.service.ts

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

@Injectable({

  providedIn: 'root'

})

export class AuthService {

  private loggedIn=false;

  login(){

    this.loggedIn=true;

  }

  logout(){

    this.loggedIn=false;

  }

  isLoggedin():boolean{

    return this.loggedIn;

  }

  constructor() { }

}

Step:4 Create GuardRule

* ng generate guard auth

this will create auth.guard.ts file

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

import { CanActivateFn, Router } from '@angular/router';

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

export const authGuard: CanActivateFn = (route, state) => {

  const authService= inject(AuthService);

  const router=inject(Router)

  if(authService.isLoggedin()){

    return true;

  }

  else{

    return router.createUrlTree(['/login']);

  }

};

After this inject the same in routing

App.routing.module.ts file

const routes: Routes = [

  {path:"login",component:LoginComponent},

  {path:"dashboard",

    component:DashboardComponent,

    canActivate:[authGuard]

  },

  {path:"home",component:HomeComponent},

  {path:"users",component:UsersComponent},

  {path:"accounts",component:AccountsComponent }

];

Goto> login.component.html file

<div class="container">

    <button class="btn btn-warning" (click)="login()">Login</button>

</div>

Goto> login.component.ts file

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

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

import { Router } from '@angular/router';

@Component({

  selector: 'app-login',

  templateUrl: './login.component.html',

  styleUrls: ['./login.component.css']

})

export class LoginComponent {

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

  login(){

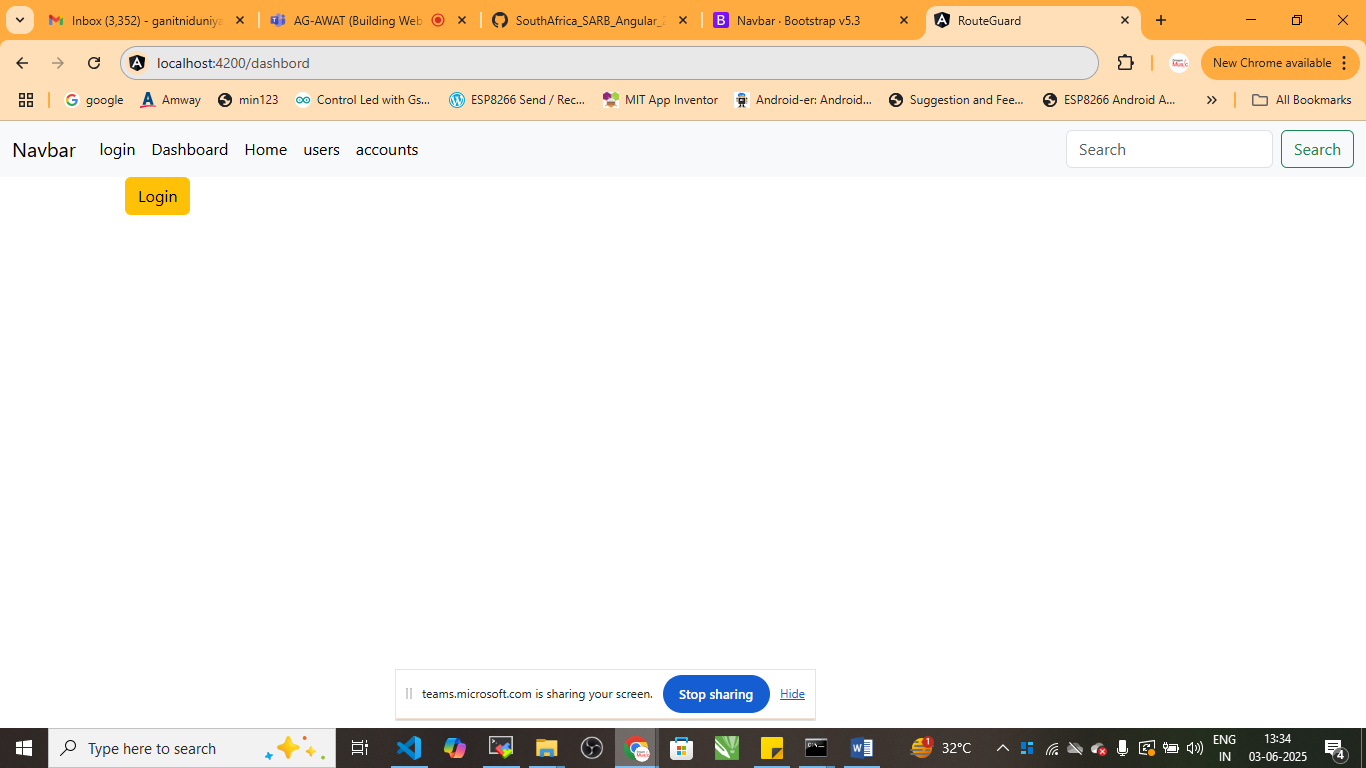
    this.auth.login();

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

  }

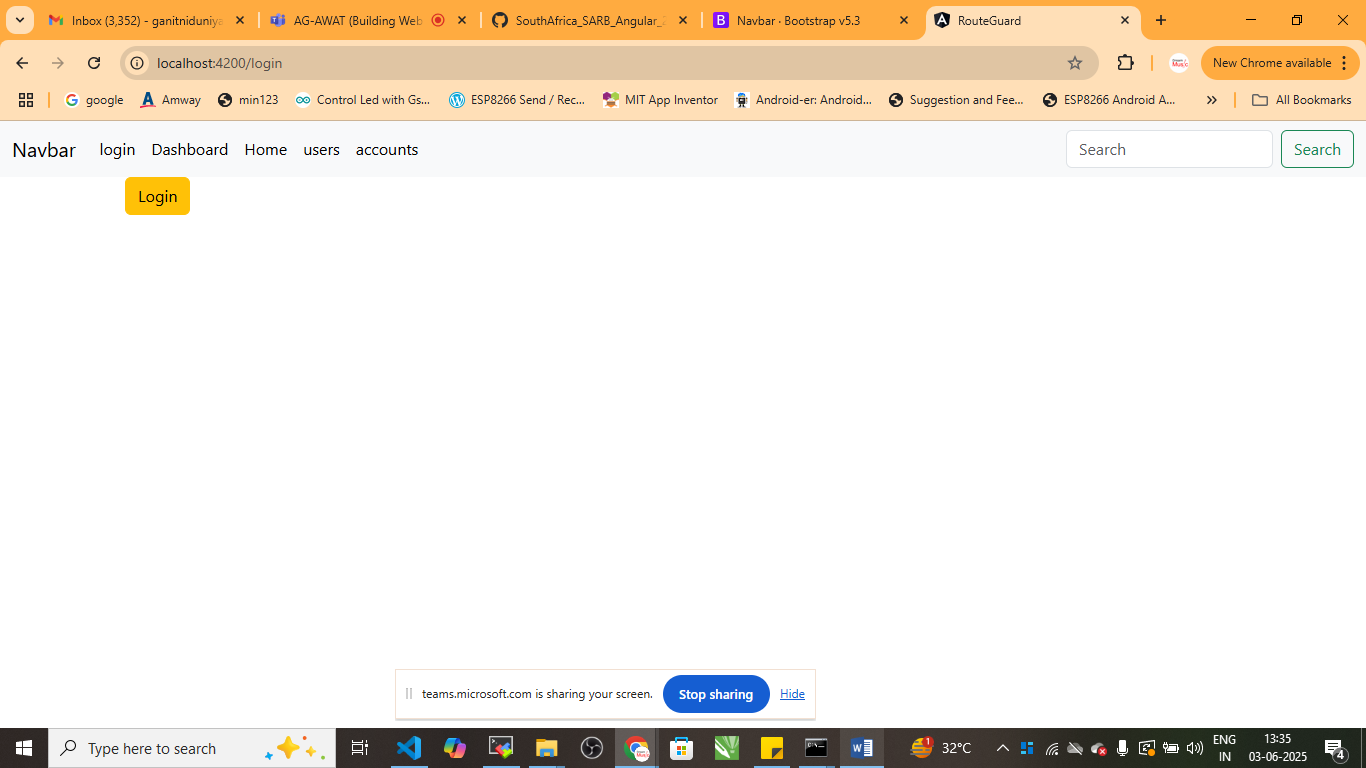
}

Save and check the output



If you will try to access localhost:4200/dashboard

It will redirect you to localhost:4200/login



**Role Based Guard**

Protect /admin route so that only users with role “admin” can access it. Others like “users” should be restricted to / Not- authorized

Create a role:

* ng generate guard role

goto> auth.service.ts file

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

@Injectable({

  providedIn: 'root'

})

export class AuthService {

  private currentUser:{username: string; role: string} | null=null;

  loginAsUser(){

    this.currentUser={username:'jhon',role:'user'};

  }

  loginAsAdmin(){

    this.currentUser={username:'admin',role:'admin'};

  }

  logout(){

    this.currentUser=null;

  }

  isLoggedin():boolean{

    return this.currentUser !==null;

  }

  getRole():string |null {

    return this.currentUser?.role || null;

  }

  constructor() { }

}

role.guard.ts

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

import { CanActivateFn, Router } from '@angular/router';

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

export const roleGuard: CanActivateFn = (route, state) => {

  const authService= inject(AuthService);

  const router=inject(Router);

  const expectedRole= route.data?.['expectedRole'];

  const userRole=authService.getRole();

  if(authService.isLoggedin() && userRole== expectedRole){

    return true;

  }

  else{

    return router.createUrlTree(['/not-authorized']);

  }

};

App.routing.module.ts file

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

import { RouterModule, Routes } from '@angular/router';

import { DashboardComponent } from './dashboard/dashboard.component';

import { HomeComponent } from './home/home.component';

import { UsersComponent } from './users/users.component';

import { AccountsComponent } from './accounts/accounts.component';

import { LoginComponent } from './login/login.component';

import { authGuard } from './auth.guard';

import { AdminComponent } from './admin/admin.component';

import { roleGuard } from './role.guard';

import { NotAuthorizedComponent } from './not-authorized/not-authorized.component';

const routes: Routes = [

  {path:"login",component:LoginComponent},

  {path:"dashboard",

    component:DashboardComponent,

    canActivate:[authGuard]

  },

  {path:"home",component:HomeComponent,canActivate:[authGuard]},

  {path:"users",component:UsersComponent,canActivate:[authGuard]},

  {path:"accounts",component:AccountsComponent,canActivate:[authGuard]},

  {path:"admin",

    component:AdminComponent,

    canActivate:[roleGuard],

    data:{ecpectedRole:'admin'}

  },

  {path:'not-authorized',component:NotAuthorizedComponent}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

generate not-authorized component

* ng g c not-authorized

<h3>You Are Not Authorised to use This</h3>

Login.component.ts file

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

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

import { Router } from '@angular/router';

@Component({

  selector: 'app-login',

  templateUrl: './login.component.html',

  styleUrls: ['./login.component.css']

})

export class LoginComponent {

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

  // login(){

  //   this.auth.login();

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

  // }

  loginAsUser(){

    this.auth.loginAsUser();

    this.router.navigate(['/']);

  }

  loginAsAdmin(){

    this.auth.loginAsAdmin();

    this.router.navigate(['/admin']);

  }

}

Login.component.html

<div class="container">

    <!-- <button class="btn btn-warning" (click)="login()">Login</button> -->

    <button class="btn btn-primary" (click)="loginAsUser()">User</button>

    <button class="btn btn-success" (click)="loginAsAdmin()">Admin</button>

</div>

Generate component admin

* ng g c admin

<h1>Welcome Admin</h1>

**Topic 2: CanActivateChild**

Its is used when you want to protect child routing in angular auth guard

Step:1 create child routing

* ng g c child-a
* ng g c child-b

Step: 2 Create Navigation Link in HomeComponent

<div class="container">

    <div class="row">

        <a routerLink="child-a"> Child-A</a>

    </div>

    <div class="row">

        <a routerLink="child-b"> Child-B</a>

    </div>

</div>

<router-outlet></router-outlet>

Step:3 import canActivateChildFn in auth.guard.ts file

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

import { CanActivateFn,CanActivateChildFn, Router } from '@angular/router';

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

export const authGuard: CanActivateFn = (route, state) => {

  const authService= inject(AuthService);

  const router=inject(Router)

  if(authService.isLoggedin()){

    return true;

  }

  else{

    return router.createUrlTree(['/login']);

  }

};

export const authChildGuard: CanActivateChildFn =()=>{

  const auth= inject(AuthService);

  const router=inject(Router);

  return auth.isLoggedin() || router.createUrlTree(['/login']);

}

Step:4 add method to auth.service.ts

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

@Injectable({

  providedIn: 'root'

})

export class AuthService {

  private loggedIn=false;

  private currentUser:{username: string; role: string} | null=null;

  loginAsUser(){

    this.currentUser={username:'jhon',role:'user'};

  }

  loginAsAdmin(){

    this.currentUser={username:'admin',role:'admin'};

  }

  logout(){

    this.currentUser=null;

     this.loggedIn=false;

  }

  isLoggedin():boolean{

    return this.currentUser !==null && this.loggedIn;

  }

  getRole():string |null {

    return this.currentUser?.role || null;

  }

  login(){

    this.loggedIn=true

  }

}

Step:5 Create Parent and Child Routing in app.routing.module.ts file

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

import { RouterModule, Routes } from '@angular/router';

import { DashboardComponent } from './dashboard/dashboard.component';

import { HomeComponent } from './home/home.component';

import { UsersComponent } from './users/users.component';

import { AccountsComponent } from './accounts/accounts.component';

import { LoginComponent } from './login/login.component';

import { authChildGuard, authGuard } from './auth.guard';

import { AdminComponent } from './admin/admin.component';

import { roleGuard } from './role.guard';

import { NotAuthorizedComponent } from './not-authorized/not-authorized.component';

import { ChildAComponent } from './child-a/child-a.component';

import { ChildBComponent } from './child-b/child-b.component';

const routes: Routes = [

  {path:"login",component:LoginComponent},

  {path:"dashboard",

    component:DashboardComponent,

    canActivate:[authGuard]

  },

  {

    path:"home",

    component:HomeComponent,

    canActivateChild:[authChildGuard],

    children:[

      {path:"child-a",component:ChildAComponent},

      {path:"child-b",component:ChildBComponent}

    ]

  },

  {path:"users",component:UsersComponent},

  {path:"accounts",component:AccountsComponent},

  {

    path: 'admin',

    component: AdminComponent,

    canActivate: [roleGuard],

    data: { expectedRole: 'admin' }

  },

  { path: 'not-authorized', component: NotAuthorizedComponent }

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

goto> login.component.ts file

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

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

import { Router } from '@angular/router';

@Component({

  selector: 'app-login',

  templateUrl: './login.component.html',

  styleUrls: ['./login.component.css']

})

export class LoginComponent {

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

  loginAsUser() {

    this.authService.loginAsUser();

    this.router.navigate(['/']);

  }

  loginAsAdmin() {

    this.authService.loginAsAdmin();

    this.router.navigate(['/admin']);

  }

  loginChild(){

    this.authService.login();

    this.router.navigate(['/home/child-a'])

  }

}

Login.component.html

<div class="container">

    <!-- <button class="btn btn-warning" (click)="login()">Login</button> -->

    <h2>Login Page</h2>

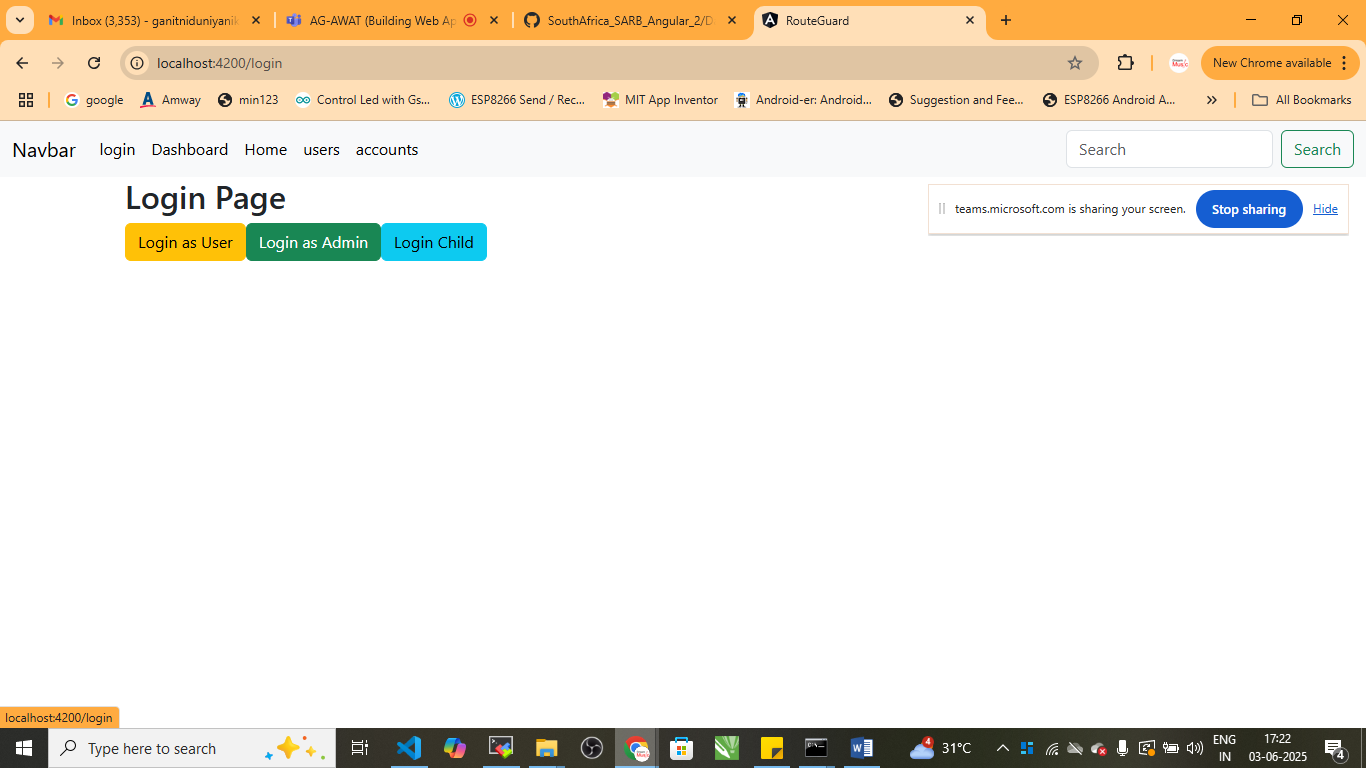
    <button class="btn btn-warning" (click)="loginAsUser()">Login as User</button>

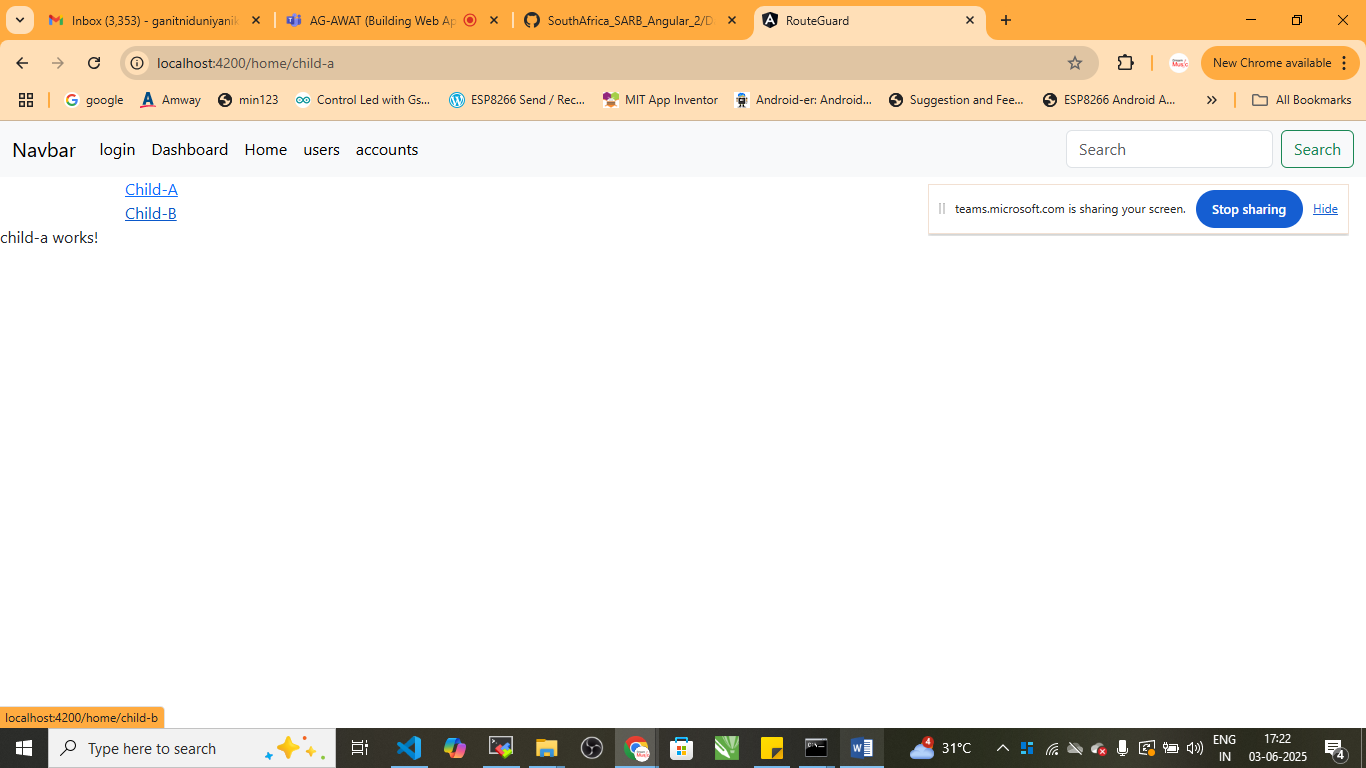
    <button class="btn btn-success" (click)="loginAsAdmin()">Login as Admin</button>

<button class="btn btn-info" (click)="loginChild()">Login Child</button>

</div>

Output:





Now try to access: localhost:4200/home/child-a

In the browser it will not work

Topic: canDeactivate

The **canDeactivate** guard is used to **prevent users from accidentally navigating away from a component**- typically when there is unsaved data in a form

Consider an example of edit-form

App/

-can-deactivate.gaurd.ts

-can-deactivate.interface.ts

-edit-profile

-edit-profile.component.html

-edit-profilr.component.ts

-app.component.html

-app.module.ts

**Step:1 Generate guard can-deactivate**

* **ng generate guard can-deactivate**

choose ( ) canDeactivate

canDeactivate.guard.ts

import { CanDeactivateFn } from '@angular/router';

import { CanComponentDeactivate } from './can-deactivate.interface';

export const canDeactivateGuard:CanDeactivateFn<CanComponentDeactivate> =(component,currentRoute,currentState,nextState)=>{

  return component.canDeactivate? component.canDeactivate():true;

}

**Step:2 Create interface** can-deactivate.interface.ts

export interface CanComponentDeactivate{

  canDeactivate: ()=> boolean | Promise<boolean>;

}

Step:3 edit-profile.component.ts

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

import { CanComponentDeactivate } from '../can-deactivate.interface';

@Component({

  selector: 'app-edit-profile',

  templateUrl: './edit-profile.component.html',

  styleUrls: ['./edit-profile.component.css']

})

export class EditProfileComponent  implements CanComponentDeactivate  {

  name='';

  saved=false;

  save(){

    this.saved=true;

    alert('Profile Saved');

  }

  canDeactivate():boolean{

    if(!this.saved && this.name){

      return confirm('You have Unsaved changes. Leave anyway?');

    }

    return true;

  }

}

edit-profile.component.html

<div class="container">

    <h2> Edit Profile</h2>

    <input [(ngModel)]="name" placeholder="Enter your Name"/>

    <button class="btn btn-danger" (click)="save()"> Save</button>

    <br>

    <br>

    <a routerLink="/"> Go Home</a>

</div>

Step:4 Create Navigation in app.component.html

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="edit-profile" href="#">Edit-Profile</a>

        </li>

Step:5 app.routing.module.ts

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

import { RouterModule, Routes } from '@angular/router';

import { DashboardComponent } from './dashboard/dashboard.component';

import { HomeComponent } from './home/home.component';

import { UsersComponent } from './users/users.component';

import { AccountsComponent } from './accounts/accounts.component';

import { LoginComponent } from './login/login.component';

import { authChildGuard, authGuard } from './auth.guard';

import { AdminComponent } from './admin/admin.component';

import { roleGuard } from './role.guard';

import { NotAuthorizedComponent } from './not-authorized/not-authorized.component';

import { ChildAComponent } from './child-a/child-a.component';

import { ChildBComponent } from './child-b/child-b.component';

import { EditProfileComponent } from './edit-profile/edit-profile.component';

import { canDeactivateGuard } from './can-deactivate.guard';

const routes: Routes = [

{ path: '', redirectTo: 'login', pathMatch: 'full' },

  {path:"login",component:LoginComponent},

  {path:"dashboard",

    component:DashboardComponent,

    canActivate:[authGuard]

  },

  {

    path:"home",

    component:HomeComponent,

    canActivate:[authGuard],

    canActivateChild:[authChildGuard],

    children:[

      {path:"child-a",component:ChildAComponent},

      {path:"child-b",component:ChildBComponent}

    ]

  },

  {path:"admin/users",component:UsersComponent,canActivate: [roleGuard],

    data: { expectedRole: 'admin' }},

  {path:"admin/accounts",component:AccountsComponent,canActivate: [roleGuard],

    data: { expectedRole: 'admin' }},

  {

    path: 'admin',

    component: AdminComponent,

    canActivate: [roleGuard],

    data: { expectedRole: 'admin' }

  },

  {path:"edit-profile",

    component:EditProfileComponent,

    canDeactivate:[canDeactivateGuard]

  },

  { path: 'not-authorized', component: NotAuthorizedComponent },

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

Step:5 Finally Import Forms Module in app.module.ts file

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

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { HomeComponent } from './home/home.component';

import { UsersComponent } from './users/users.component';

import { AccountsComponent } from './accounts/accounts.component';

import { DashboardComponent } from './dashboard/dashboard.component';

import { LoginComponent } from './login/login.component';

import { AdminComponent } from './admin/admin.component';

import { NotAuthorizedComponent } from './not-authorized/not-authorized.component';

import { ChildAComponent } from './child-a/child-a.component';

import { ChildBComponent } from './child-b/child-b.component';

import { ProfileComponent } from './profile/profile.component';

import { EditProfileComponent } from './edit-profile/edit-profile.component';

import { FormsModule } from '@angular/forms';

@NgModule({

  declarations: [

    AppComponent,

    HomeComponent,

    UsersComponent,

    AccountsComponent,

    DashboardComponent,

    LoginComponent,

    AdminComponent,

    NotAuthorizedComponent,

    ChildAComponent,

    ChildBComponent,

    ProfileComponent,

    EditProfileComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule

  ],

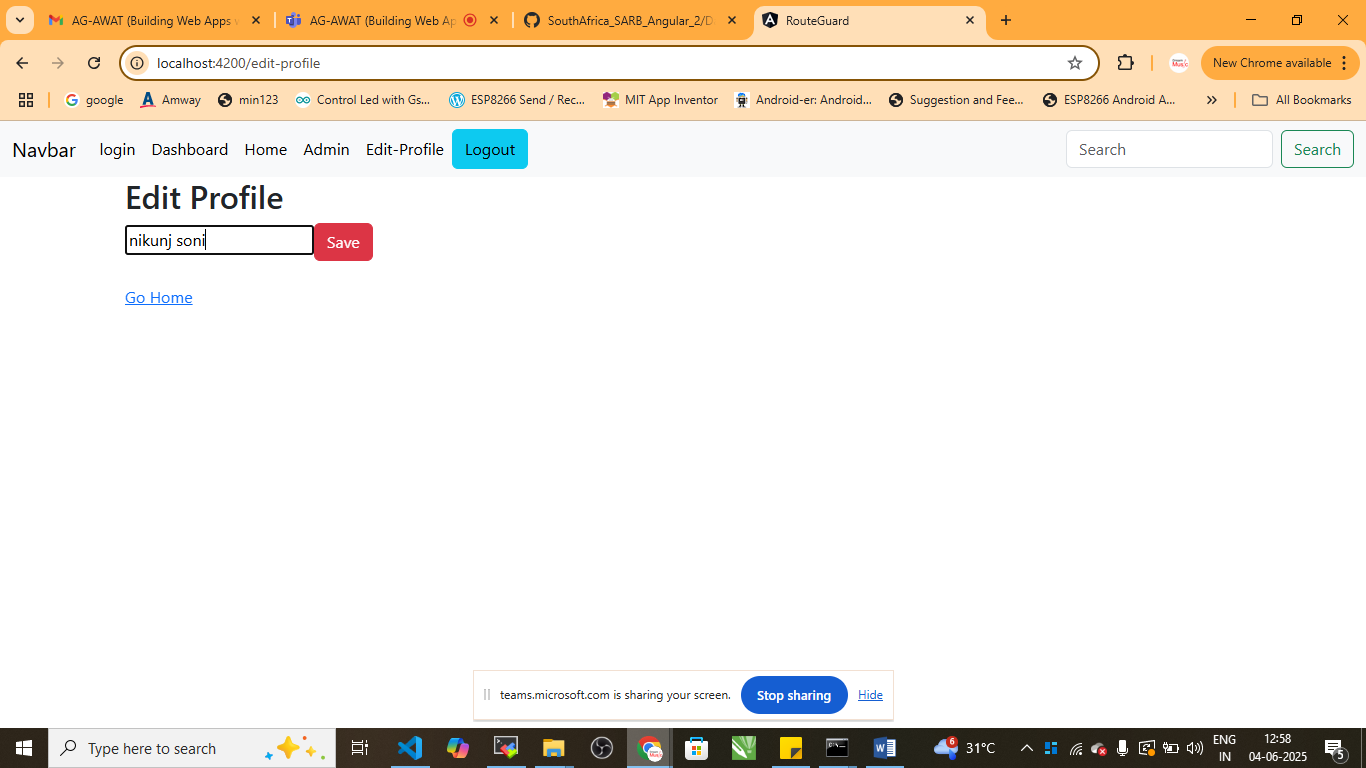
  providers: [],

  bootstrap: [AppComponent]

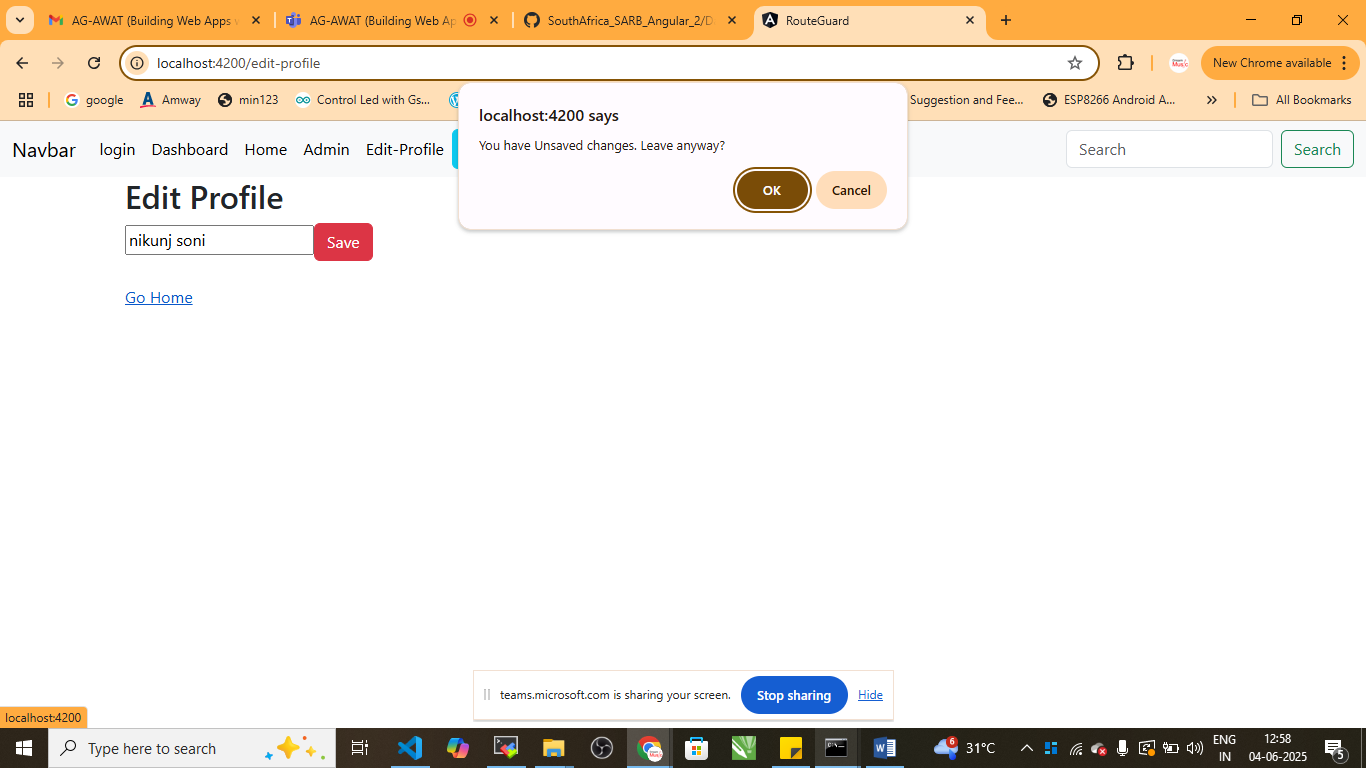
})

export class AppModule { }

Output:



Now click on gohome>



**Topic : CanMatch**

- You want to block or allow route Matching before lazy loading.

- If you want to control access based on user roles, authentication, feature flags

- Unlike CanActivate, CanMatch stops Angular from even considering the route

Use Case: There is one WebApplication of SARB Bank and you want to Restrict the access to the path you can implement the same here

What you need:

1. guards
   1. resolve.guard.ts
2. admin
   1. admin.module.ts
   2. admin.component.ts
3. app-routing.module.ts
4. app.component.ts
5. app.module.ts

Step 1: Generate Resolve Auth

* + ng generate guard resolve

Step 2: Create **resolve.guard.ts** – canMatchGuard

import { CanActivateFn } from '@angular/router';

const isAdmin= ()=>{

  return true;

}

export const resolveGuard: CanActivateFn = (route, state) => {

  const allowed =isAdmin();

  console.log('Admin route access:',allowed);

  return allowed;

};

Step 3: admin.component.html

<div class="container">

    <div class="col">

        <a routerLink="accounts">Accounts</a>

        <a routerLink="users"> users</a>

    </div>

</div>

<router-outlet></router-outlet>

<div class="container bg-warning">

    <h1>This is Admin Dashboard</h1>

    <p>Admin................</p>

</div>

<div class="container bg-info">

    <h1>This is Admin Dashboard</h1>

    <p>Admin................</p>

</div>

<div class="container bg-danger ">

    <h1>This is Admin Dashboard</h1>

    <p>Admin................</p>

</div>

Step 3: admin.module.ts

import { NgModule } from "@angular/core";

import { RouterModule } from "@angular/router";

import { AdminComponent } from "./admin.component";

@NgModule({

  imports:[

    RouterModule.forChild([

      {path:'',component:AdminComponent}

    ])

  ]

})

export class AdminModule{}

Step:4 app-routing.module.ts

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

import { RouterModule, Routes } from '@angular/router';

import { DashboardComponent } from './dashboard/dashboard.component';

import { HomeComponent } from './home/home.component';

import { UsersComponent } from './users/users.component';

import { AccountsComponent } from './accounts/accounts.component';

import { LoginComponent } from './login/login.component';

import { authChildGuard, authGuard } from './auth.guard';

import { AdminComponent } from './admin/admin.component';

import { roleGuard } from './role.guard';

import { NotAuthorizedComponent } from './not-authorized/not-authorized.component';

import { ChildAComponent } from './child-a/child-a.component';

import { ChildBComponent } from './child-b/child-b.component';

import { EditProfileComponent } from './edit-profile/edit-profile.component';

import { canDeactivateGuard } from './can-deactivate.guard';

import { resolveGuard } from './resolve.guard';

const routes: Routes = [

{ path: '', redirectTo: 'login', pathMatch: 'full' },

  {path:"login",component:LoginComponent},

  {path:"dashboard",

    component:DashboardComponent,

    canActivate:[authGuard]

  },

  {

    path:"home",

    component:HomeComponent,

    canActivate:[authGuard],

    canActivateChild:[authChildGuard],

    children:[

      {path:"child-a",component:ChildAComponent},

      {path:"child-b",component:ChildBComponent}

    ]

  },

  {path:"admin/users",component:UsersComponent,canActivate: [roleGuard],

    data: { expectedRole: 'admin' }},

  {path:"admin/accounts",component:AccountsComponent,canActivate: [roleGuard],

    data: { expectedRole: 'admin' }},

  {

    path: 'admin',

    component: AdminComponent,

    canMatch:[resolveGuard],

    loadChildren:()=> import('./admin/admin.module').then(m=>m.AdminModule)

   //canActivate: [roleGuard],

    //data: { expectedRole: 'admin' }

  },

  {path:"edit-profile",

    component:EditProfileComponent,

    canDeactivate:[canDeactivateGuard]

  },

  { path: 'not-authorized', component: NotAuthorizedComponent },

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

Step:5 app.component.html

<nav class="navbar navbar-expand-lg bg-body-tertiary">

  <div class="container-fluid">

    <a class="navbar-brand" href="#">Navbar</a>

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">

      <ul class="navbar-nav me-auto mb-2 mb-lg-0">

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="login" href="#">login</a>

        </li>

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="dashboard" href="#">Dashboard</a>

        </li>

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="home" href="#">Home</a>

        </li>

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="admin" href="#">Admin</a>

        </li>

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" routerLink="edit-profile" href="#">Edit-Profile</a>

        </li>

        <button class="btn btn-info" (click)="logout()">Logout</button>

      </ul>

      <form class="d-flex" role="search">

        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search"/>

        <button class="btn btn-outline-success" type="submit">Search</button>

      </form>

    </div>

  </div>

</nav>

<router-outlet></router-outlet>

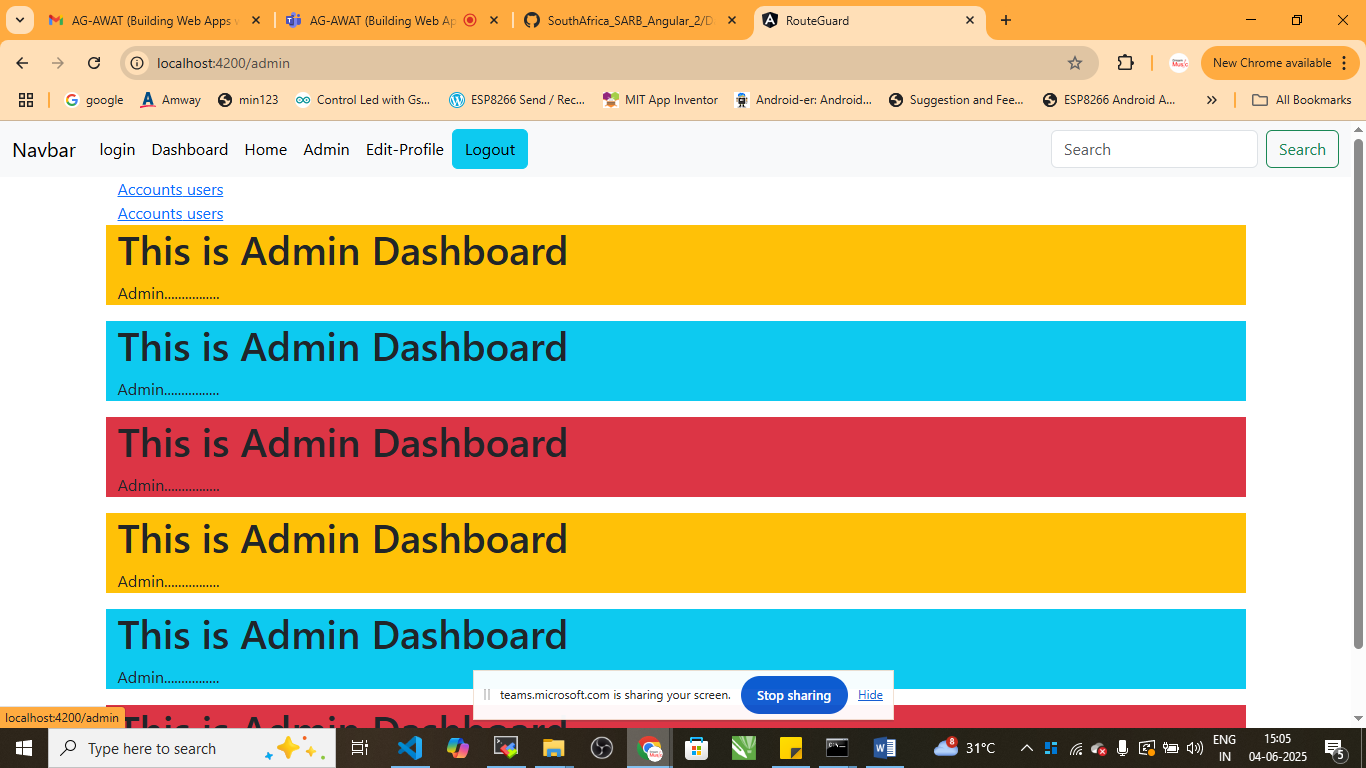
Try changing the value in resolve.guard.ts

const isAdmin= ()=>{

  return true;

  //return false

}



const isAdmin= ()=>{

  return false;

  //return false

}

