**Angular 15 Project scenario** with **RxJS**, **Lifecycle Hooks**, **Route Guards**, **State Management**, and **HTTP Interceptors**. The project simulates a **Task Management System** where:

* Users can log in and view/manage tasks.
* Admins can also create/delete users.
* API calls are intercepted for JWT and loading spinner.(Optional)
* Components use lifecycle hooks and RxJS streams.
* State is managed using a simple service (or optionally NgRx if needed).

**🔧 Tech Stack**

* Angular 15
* RxJS
* Services for state management
* HTTP Interceptors
* Route Guards (Auth + Role-based)
* JSONPlaceholder/mock API

**🔍 Project Scenario**

**Scenario:**

You are building a **Task Manager App** with:

* **Login** page
* **Dashboard** for viewing tasks
* **Admin panel** for managing users
* Use route guards to protect routes based on login and role.
* Use RxJS for reactive task filtering.
* Use state management to maintain the current user and loading state.
* Show a loading spinner with an interceptor.
* Use lifecycle hooks to load initial data in components.

Solution:

# Components

ng generate component components/login

ng generate component components/dashboard

ng generate component components/admin

# Services and Interceptors

ng generate service core/services/state

ng generate interceptor core/interceptors/auth

ng generate interceptor core/interceptors/loader

# Auth Guards

ng generate guard auth/auth

ng generate guard auth/role

## Step 3: Define Models

📄 src/app/models/user.model.ts

ts

CopyEdit

export interface User {

id: number;

name: string;

role: 'admin' | 'user';

}

src/app/models/task.model.ts

ts

CopyEdit

export interface Task {

userId: number;

id: number;

title: string;

completed: boolean;

}

## Step 4: Implement State Management Service

📄 src/app/core/services/state.service.ts

ts

CopyEdit

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

import { BehaviorSubject } from 'rxjs';

import { User } from '../../models/user.model';

@Injectable({ providedIn: 'root' })

export class StateService {

private userSubject = new BehaviorSubject<User | null>(null);

user$ = this.userSubject.asObservable();

private loadingSubject = new BehaviorSubject<boolean>(false);

loading$ = this.loadingSubject.asObservable();

setUser(user: User) {

this.userSubject.next(user);

}

setLoading(status: boolean) {

this.loadingSubject.next(status);

}

}

## Step 5: Create Guards

📄 src/app/auth/auth.guard.ts

ts

CopyEdit

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

export const authGuard: CanActivateFn = () => !!localStorage.getItem('token');

📄 src/app/auth/role.guard.ts

ts

CopyEdit

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

export const roleGuard: CanActivateFn = () => localStorage.getItem('role') === 'admin';

## Step 6: Set Up Interceptors

📄 src/app/core/interceptors/auth.interceptor.ts

ts

CopyEdit

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

import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable()

export class AuthInterceptor implements HttpInterceptor {

intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

const token = localStorage.getItem('token');

if (token) {

req = req.clone({ setHeaders: { Authorization: `Bearer ${token}` } });

}

return next.handle(req);

}

}

📄 src/app/core/interceptors/loader.interceptor.ts

ts

CopyEdit

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

import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';

import { Observable, finalize } from 'rxjs';

import { StateService } from '../services/state.service';

@Injectable()

export class LoaderInterceptor implements HttpInterceptor {

constructor(private state: StateService) {}

intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

this.state.setLoading(true);

return next.handle(req).pipe(finalize(() => this.state.setLoading(false)));

}

}

## Step 7: Create LoginComponent

📄 login.component.ts

ts

CopyEdit

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

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

@Component({ selector: 'app-login', templateUrl: './login.component.html' })

export class LoginComponent {

constructor(private router: Router) {}

loginAsUser() {

localStorage.setItem('token', 'user-token');

localStorage.setItem('role', 'user');

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

}

loginAsAdmin() {

localStorage.setItem('token', 'admin-token');

localStorage.setItem('role', 'admin');

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

}

}

📄 login.component.html

html

CopyEdit

<h2>Login</h2>

<button (click)="loginAsUser()">Login as User</button>

<button (click)="loginAsAdmin()">Login as Admin</button>

## Step 8: Create DashboardComponent with RxJS + Lifecycle

📄 dashboard.component.ts

ts

CopyEdit

import { Component, OnInit, OnDestroy } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Task } from '../../models/task.model';

import { Subject, takeUntil } from 'rxjs';

@Component({ selector: 'app-dashboard', templateUrl: './dashboard.component.html' })

export class DashboardComponent implements OnInit, OnDestroy {

tasks: Task[] = [];

destroy$ = new Subject<void>();

constructor(private http: HttpClient) {}

ngOnInit() {

this.http.get<Task[]>('https://jsonplaceholder.typicode.com/todos')

.pipe(takeUntil(this.destroy$))

.subscribe(data => this.tasks = data.slice(0, 10));

}

ngOnDestroy() {

this.destroy$.next();

this.destroy$.complete();

}

}

📄 dashboard.component.html

html

CopyEdit

<h2>Dashboard</h2>

<ul>

<li \*ngFor="let task of tasks">

{{ task.title }} - {{ task.completed ? 'Done' : 'Pending' }}

</li>

</ul>

## Step 9: Create AdminComponent (Guard Protected)

📄 admin.component.ts

ts

CopyEdit

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

@Component({ selector: 'app-admin', templateUrl: './admin.component.html' })

export class AdminComponent {}

📄 admin.component.html

html

CopyEdit

<h2>Admin Panel</h2>

<p>Only admins can see this!</p>

## Step 10: Configure Routing

📄 app-routing.module.ts

ts

CopyEdit

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

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

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

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

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

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

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

const routes: Routes = [

{ path: '', component: LoginComponent },

{ path: 'dashboard', component: DashboardComponent, canActivate: [authGuard] },

{ path: 'admin', component: AdminComponent, canActivate: [authGuard, roleGuard] }

];

@NgModule({ imports: [RouterModule.forRoot(routes)], exports: [RouterModule] })

export class AppRoutingModule {}

## Step 11: Setup AppModule

📄 app.module.ts

ts

CopyEdit

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

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

import { HttpClientModule, HTTP\_INTERCEPTORS } from '@angular/common/http';

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

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

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

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

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

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

import { AuthInterceptor } from './core/interceptors/auth.interceptor';

import { LoaderInterceptor } from './core/interceptors/loader.interceptor';

@NgModule({

declarations: [AppComponent, LoginComponent, DashboardComponent, AdminComponent],

imports: [BrowserModule, AppRoutingModule, HttpClientModule, ReactiveFormsModule],

providers: [

{ provide: HTTP\_INTERCEPTORS, useClass: AuthInterceptor, multi: true },

{ provide: HTTP\_INTERCEPTORS, useClass: LoaderInterceptor, multi: true },

],

bootstrap: [AppComponent]

})

export class AppModule { }

## Step 12: Run the App

bash

CopyEdit

ng serve

Open in browser: http://localhost:4200