**1. Set up Angular 12 Project**

Make sure you have Node.js and Angular CLI installed. If not, you can install them using:

bash

npm install -g @angular/cli

**Create a new Angular project:**

bash

ng new angular-auth-app

cd angular-auth-app

**Install necessary dependencies:**

For this example, you will need angular/forms for forms and @angular/router for routing.

bash

ng add @angular/forms

npm install --save @angular/router

**2. Create Components**

We will create the following components:

* **Registration**: for registering a new user.
* **Login**: for user login.
* **Admin**: for the admin panel.
* **Home**: for the homepage.

Use the Angular CLI to generate components:

bash

ng generate component registration

ng generate component login

ng generate component admin

ng generate component home

**3. Create Services**

Create a service to handle authentication (auth.service.ts) and potentially an admin service (admin.service.ts) to interact with an API or mock data.

bash

ng generate service auth

**4. Set Up Routing**

In app-routing.module.ts, configure routes for the different components:

typescript

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

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

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

import { RegistrationComponent } from './registration/registration.component';

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

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

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

const routes: Routes = [

{ path: '', component: HomeComponent },

{ path: 'register', component: RegistrationComponent },

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

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

{ path: '\*\*', redirectTo: '' }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

Here, AuthGuard is used to protect the admin route.

**5. Implement Authentication Logic**

Create an auth.service.ts to handle registration, login, and authentication status. For simplicity, we’ll use a mock backend (could later be replaced with a real API).

**auth.service.ts**

typescript

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

import { BehaviorSubject } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class AuthService {

private currentUserSubject: BehaviorSubject<any>;

public currentUser: any;

constructor() {

this.currentUserSubject = new BehaviorSubject<any>(JSON.parse(localStorage.getItem('currentUser') || '{}'));

this.currentUser = this.currentUserSubject.asObservable();

}

// Registration method

register(username: string, password: string): void {

const user = { username, password, role: 'user' }; // default role is user

localStorage.setItem('currentUser', JSON.stringify(user));

this.currentUserSubject.next(user);

}

// Login method

login(username: string, password: string): boolean {

const user = { username, password, role: 'user' };

localStorage.setItem('currentUser', JSON.stringify(user));

this.currentUserSubject.next(user);

return true;

}

// Logout method

logout() {

localStorage.removeItem('currentUser');

this.currentUserSubject.next(null);

}

// Get current user

getCurrentUser() {

return this.currentUser;

}

// Check if the user is logged in

isLoggedIn() {

return !!this.currentUserSubject.value;

}

}

**6. Registration Form**

In registration.component.ts, use ReactiveFormsModule for the form.

**registration.component.ts**

typescript

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

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

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

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

@Component({

selector: 'app-registration',

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

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

})

export class RegistrationComponent implements OnInit {

registrationForm: FormGroup;

constructor(

private fb: FormBuilder,

private authService: AuthService,

private router: Router

) {}

ngOnInit(): void {

this.registrationForm = this.fb.group({

username: ['', [Validators.required, Validators.minLength(3)]],

password: ['', [Validators.required, Validators.minLength(6)]]

});

}

onSubmit() {

if (this.registrationForm.valid) {

const { username, password } = this.registrationForm.value;

this.authService.register(username, password);

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

}

}

}

**registration.component.html**

html

<form [formGroup]="registrationForm" (ngSubmit)="onSubmit()">

<div>

<label for="username">Username:</label>

<input id="username" formControlName="username" />

<div \*ngIf="registrationForm.controls['username'].invalid && registrationForm.controls['username'].touched">

Username is required.

</div>

</div>

<div>

<label for="password">Password:</label>

<input type="password" id="password" formControlName="password" />

<div \*ngIf="registrationForm.controls['password'].invalid && registrationForm.controls['password'].touched">

Password is required.

</div>

</div>

<button type="submit" [disabled]="registrationForm.invalid">Register</button>

</form>

**7. Login Form**

Create a similar login.component.ts for the login functionality.

**login.component.ts**

typescript

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

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

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 implements OnInit {

loginForm: FormGroup;

constructor(

private fb: FormBuilder,

private authService: AuthService,

private router: Router

) {}

ngOnInit(): void {

this.loginForm = this.fb.group({

username: ['', [Validators.required]],

password: ['', [Validators.required]]

});

}

onSubmit() {

if (this.loginForm.valid) {

const { username, password } = this.loginForm.value;

if (this.authService.login(username, password)) {

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

} else {

alert('Login failed');

}

}

}

}

**8. Admin Panel**

**admin.component.ts**

The admin panel can simply check for an authenticated user, and display content accordingly.

typescript

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

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

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

@Component({

selector: 'app-admin',

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

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

})

export class AdminComponent implements OnInit {

user: any;

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

ngOnInit(): void {

this.user = this.authService.getCurrentUser();

if (!this.user || this.user.role !== 'admin') {

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

}

}

logout() {

this.authService.logout();

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

}

}

**9. Auth Guard**

Create an AuthGuard to protect the admin route.

bash

ng generate guard auth

In auth.guard.ts, define the guard to check if the user is logged in:

typescript

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

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

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

@Injectable({

providedIn: 'root'

})

export class AuthGuard implements CanActivate {

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

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {

if (this.authService.isLoggedIn()) {

return true;

}

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

return false;

}

}

**10. Run the Application**

Finally, run the Angular application:

bash

ng serve