**Assignment 2c**

1. AuthService (auth.service.ts)

(Manages user state and sign-in card visibility)

TypeScript

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

import { BehaviorSubject, Observable } from 'rxjs';

// Define a simple user interface (adjust as needed)

export interface User {

  id: string;

  name: string;

  email: string;

  // Add other relevant user properties

}

@Injectable({

  providedIn: 'root' // Singleton service available throughout the app

})

export class AuthService {

  // BehaviorSubject holds the current value and emits it to new subscribers

  private currentUserSubject = new BehaviorSubject<User | undefined>(undefined);

  // Expose the current user state as an Observable

  public currentUser$: Observable<User | undefined> = this.currentUserSubject.asObservable();

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

  public showSignInCard$: Observable<boolean> = this.showSignInCardSubject.asObservable();

  constructor() {

    // Optional: Initialize user from local storage or API on app start

    // this.loadInitialUser();

  }

  // Method to update the current user state

  setCurrentUser(user: User | undefined): void {

    this.currentUserSubject.next(user);

  }

  // Method to get the current user value synchronously (use with caution)

  getCurrentUserValue(): User | undefined {

    return this.currentUserSubject.value;

  }

  // Method to control the sign-in card visibility

  setShowSignInCard(show: boolean): void {

    this.showSignInCardSubject.next(show);

  }

  // You might add methods here to check authentication status, etc.

  // isLoggedIn(): boolean {

  //   return !!this.currentUserSubject.value;

  // }

}

2. NotificationService (notification.service.ts)

(Handles success/error notifications - using console.log for simplicity here, replace with a real toast library like ngx-toastr)

TypeScript

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

@Injectable({

  providedIn: 'root'

})

export class NotificationService {

  constructor() { }

  success(message: string): void {

    // Replace with actual toastr implementation

    console.log('SUCCESS:', message);

    // Example: this.toastr.success(message);

  }

  error(message: string): void {

    // Replace with actual toastr implementation

    console.error('ERROR:', message);

     // Example: this.toastr.error(message);

  }

}

**3. Loader Component (**loader.component.ts **&** loader.component.html**)**

* loader.component.ts:
* TypeScript

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

@Component({

  selector: 'app-loader',

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

  // Add styles if needed: styleUrls: ['./loader.component.css']

  standalone: true, // Use standalone component if preferred

})

export class LoaderComponent { }

* loader.component.html: (Basic example)
* HTML

<div class="loader-overlay">

  <div class="loader-spinner"></div>

  Loading...

</div>

* *(You'll need CSS for .loader-overlay and .loader-spinner)*

**4. Navbar Component (**navbar.component.ts**)**

TypeScript

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

import { Router, NavigationEnd, Event as RouterEvent } from '@angular/router';

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

import { Subscription, Observable } from 'rxjs';

import { filter, finalize, catchError, tap, take } from 'rxjs/operators'; // Import take

import { AuthService, User } from './auth.service'; // Adjust path

import { NotificationService } from './notification.service'; // Adjust path

@Component({

  selector: 'app-navbar',

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

  styleUrls: ['./navbar.component.css'] // Add component-specific styles if needed

})

export class NavbarComponent implements OnInit, OnDestroy {

  currentUser$: Observable<User | undefined>; // Observable for the template

  currentUser: User | undefined; // Hold the current user value for logic

  isLoading = false;

  currentPath: string = '';

  private userSubscription: Subscription | undefined;

  private routerSubscription: Subscription | undefined;

  constructor(

    private router: Router,

    private http: HttpClient,

    private authService: AuthService,

    private notificationService: NotificationService

  ) {

    this.currentUser$ = this.authService.currentUser$;

  }

  ngOnInit(): void {

    // Subscribe to user changes

    this.userSubscription = this.authService.currentUser$.subscribe(user => {

      this.currentUser = user;

    });

    // Subscribe to router events to get the current path

    this.currentPath = this.router.url; // Initial path

    this.routerSubscription = this.router.events.pipe(

      filter((event: RouterEvent): event is NavigationEnd => event instanceof NavigationEnd)

    ).subscribe((event: NavigationEnd) => {

      this.currentPath = event.urlAfterRedirects; // Use urlAfterRedirects for accuracy

    });

  }

  ngOnDestroy(): void {

    // Unsubscribe to prevent memory leaks

    this.userSubscription?.unsubscribe();

    this.routerSubscription?.unsubscribe();

  }

  get postRedirectPath(): string {

    // Getter similar to useMemo, calculated on demand

    return this.currentUser ? '/post' : '';

  }

  handlePostClick(): void {

    if (!this.currentUser) {

      this.authService.setShowSignInCard(true);

      // Optionally prevent navigation if path is empty, though routerLink handles this

      if (!this.postRedirectPath) return;

    }

    // Navigation will be handled by routerLink if currentUser exists

  }

  logout(): void {

    this.isLoading = true;

    this.http.post<{ message: string }>('/user/logout', {}).pipe( // Specify expected response type

      tap(response => { // Use tap for side-effects on success

        this.notificationService.success(response.message);

        this.authService.setCurrentUser(undefined);

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

      }),

      catchError((error: HttpErrorResponse) => {

        console.error('Logout failed:', error);

        const errorMessage = error.error?.message || 'Logout failed. Please try again.';

        this.notificationService.error(errorMessage);

        // Rethrow or return an observable indicating failure if needed elsewhere

        // return throwError(() => new Error(errorMessage));

        return []; // Return empty observable to complete the stream gracefully here

      }),

      finalize(() => { // finalize runs whether success or error

        this.isLoading = false;

      }),

      take(1) // Ensure the subscription completes after one emission

    ).subscribe(); // Need to subscribe to trigger the request

  }

  navigateToSignIn(): void {

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

  }

}

**5. Navbar Component Template (**navbar.component.html**)**

HTML

<app-loader \*ngIf="isLoading"></app-loader> <ng-container \*ngIf="currentUser$ | async as user">

  </ng-container>

<div class="grid grid-cols-2 sm:grid-cols-3 gap-4 w-screen py-6 px-2 sm:px-4 border-black items-center fixed top-0 filter backdrop-filter backdrop-blur-lg rounded-b-md shadow-lg">

  <a routerLink="/" class="font-poppins font-bold text-4xl">

    Narrate

  </a>

  <a

    [routerLink]="postRedirectPath"

    (click)="handlePostClick()"

    class="font-poppins text-2xl border-2 border-black rounded-full py-2 sm:px-6 px-4 font-semibold m-auto shadow-stone-300 shadow-lg hover:bg-black hover:text-white invisible w-0 md:w-28 md:visible absolute sm:static">

    post

  </a>

  <div class="ml-auto flex items-center">

    <button \*ngIf="currentPath === '/profile'" (click)="logout()" class="max-w-32 bg-red-500 my-0 py-2 px-4 rounded font-semibold text-white">

      Logout

    </button>

    <a \*ngIf="currentUser && currentPath !== '/profile'"

       routerLink="/profile"

       class="flex gap-2 py-2 px-4 font-poppins border border-gray-400 rounded-3xl min-w-1 items-center">

      <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor" class="size-8 invisible w-0 sm:w-8 sm:visible">

        <path stroke-linecap="round" stroke-linejoin="round" d="M15.75 6a3.75 3.75 0 1 1-7.5 0 3.75 3.75 0 0 1 7.5 0ZM4.501 20.118a7.5 7.5 0 0 1 14.998 0A17.933 17.933 0 0 1 12 21.75c-2.676 0-5.216-.584-7.499-1.632Z" />

      </svg>

      <div class="text-xl truncate">{{ currentUser?.name }}</div>

    </a>

    <button \*ngIf="!currentUser && currentPath !== '/profile'"

            (click)="navigateToSignIn()"

            class="max-w-32 bg-black text-white py-2 px-4 rounded font-semibold">

      Sign in

    </button>

  </div>

</div>