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

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

import { Observable, throwError } from 'rxjs';

import { catchError, map,tap  } from 'rxjs/operators';

import { CookieService } from 'ngx-cookie-service';

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

@Injectable({

  providedIn: 'root'

})

export class AuthentificationService {

  private apiUrl = 'http://localhost:3001/api/users';

  constructor(

    private http: HttpClient,

    private cookieService: CookieService

  ) { }

  registerUser(userData: any): Observable<any> {

    return this.http.post(`${this.apiUrl}/register`, userData, {

      headers: { 'Content-Type': 'application/json' }

    }).pipe(

      catchError(this.handleError)

    );

  }

  loginUser(email: string, password: string): Observable<any> {

    const url = `${this.apiUrl}/login-user`;

    const body = { email, password };

    return this.http.post<any>(url, body).pipe(

      catchError((error) => {

        console.error('Error during login:', error);

        return throwError('Login failed. Please try again.');

      }),

      tap(response => {

        console.log('API Response:', response); // Log pour vérifier la réponse ici

      })

    );

  }

  getUsers(): Observable<any> {

    return this.http.get<any>(`${this.apiUrl}/all`, { headers: this.getAuthHeaders() })

      .pipe(catchError(this.handleError));

  }

  getUserById(id: string): Observable<any> {

    return this.http.get<any>(`${this.apiUrl}/${id}`, { headers: this.getAuthHeaders() })

      .pipe(catchError(this.handleError));

  }

  updateUser(id: string, userData: any): Observable<any> {

    return this.http.put<any>(`${this.apiUrl}/${id}`, userData, { headers: this.getAuthHeaders() })

      .pipe(catchError(this.handleError));

  }

  deleteUser(id: string): Observable<any> {

    return this.http.delete<any>(`${this.apiUrl}/${id}`, { headers: this.getAuthHeaders() })

      .pipe(catchError(this.handleError));

  }

  // Helper method to retrieve headers with token for secure requests

  private getAuthHeaders(): HttpHeaders {

    const token = this.cookieService.get('token');

    return new HttpHeaders({

      'Authorization': `Bearer ${token}`

    });

  }

  // Method to validate if token is present and valid

  isTokenValid(token: string): boolean {

    // Placeholder validation logic (e.g., expiration check)

    return !!token;

  }

  // Centralized error handling

  private handleError(error: any): Observable<never> {

    console.error('API error occurred:', error);

    return throwError('An error occurred; please try again.');

  }

  getUserRole(): string {

    return this.cookieService.get('role'); // Lire le rôle depuis un cookie

  }

  isAdmin(): boolean {

    return this.getUserRole() === 'admin';

  }

}