package com.tuitionfee.demo.model;

import jakarta.persistence.Entity;

@Entity

public class Product {

   private Long id;

   private String name;

   private String description;

   private Double price;

   // Getters and Setters

   public Long getId() {

       return id;

   }

   public void setId(Long id) {

       this.id = id;

   }

   public String getName() {

       return name;

   }

   public void setName(String name) {

       this.name = name;

   }

   public String getDescription() {

       return description;

   }

   public void setDescription(String description) {

       this.description = description;

   }

   public Double getPrice() {

       return price;

   }

   public void setPrice(Double price) {

       this.price = price;

   }

}

public interface ProductRepository extends JpaRepository<Product, Long> {

}

package com.tuitionfee.demo.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.tuitionfee.demo.model.Product;

import com.tuitionfee.demo.repository.ProductRepository;

@Service

public class ProductService {

    @Autowired

    private ProductRepository productRepository;

    public List<Product> getAllProducts() {

        return productRepository.findAll();

    }

    public Product addProduct(Product product) {

        return productRepository.save(product);

    }

}

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

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

import { Observable } from 'rxjs';

import { Product } from '../model/model';

import { environment } from 'src/environments/environment.development';

@Injectable({

  providedIn: 'root'

})

export class ProductService {

  private apiUrl = environment.apiUrl;

  //private apiUrl = 'http://localhost:8080/api/products';

  constructor(private http: HttpClient) {}

  getProducts(): Observable<Product[]> {

    return this.http.get<Product[]>(this.apiUrl);

  }

}

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

import { Product } from '../model/model';

import { ProductService } from '../service/product.service';

@Component({

  selector: 'app-product-list',

  templateUrl: './product-list.component.html',

  styleUrls: ['./product-list.component.css']

})

export class ProductListComponent implements OnInit {

  products: Product[] = [];

  constructor(private productService: ProductService) {}

  ngOnInit(): void {

    this.productService.getProducts().subscribe(data => {

      this.products = data;

    });

  }

}

package com.tuitionfee.demo.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Book {

Long bookId;

String title;

String author;

String genre;

double price;

public Book(Long bookId, String title, String author, String genre, double price) {

    this.bookId = bookId;

    this.title = title;

    this.author = author;

    this.genre = genre;

    this.price = price;

}

public Long getBookId() {

    return bookId;

}

public void setBookId(Long bookId) {

    this.bookId = bookId;

}

public String getTitle() {

    return title;

}

public void setTitle(String title) {

    this.title = title;

}

public String getAuthor() {

    return author;

}

public void setAuthor(String author) {

    this.author = author;

}

public String getGenre() {

    return genre;

}

public void setGenre(String genre) {

    this.genre = genre;

}

public double getPrice() {

    return price;

}

public void setPrice(double price) {

    this.price = price;

}

}

public interface BookRepository extends JpaRepository<Book,Long> {

}

package com.tuitionfee.demo.services;

import java.util.List;

import java.util.Optional;

import org.springframework.stereotype.Service;

import com.tuitionfee.demo.model.Book;

import com.tuitionfee.demo.repository.BookRepository;

@Service

public class BookService {

   private final BookRepository bookRepository;

   public BookService(BookRepository bookRepository) {

       this.bookRepository = bookRepository;

   }

   public List<Book> getAllBooks() {

       return bookRepository.findAll();

   }

   public Book addBook(Book book) {

       return bookRepository.save(book);

   }

   public Book getBookById(Long id) {

       Optional<Book> optionalBook = bookRepository.findById(id);

       return optionalBook.orElse(null);

   }

}

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

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

import { Observable } from 'rxjs';

import { environment } from 'src/environments/environment.development';

import { Book } from '../model/model';

@Injectable({

  providedIn: 'root',

})

export class BookService {

private apiUrl = `${environment.apiUrl}/api/books`;

constructor(private http: HttpClient) {}

getBooks(): Observable<Book[]> {

  return this.http.get<Book[]>(this.apiUrl);

}

addBook(book: Book): Observable<Book> {

  return this.http.post<Book>(this.apiUrl, book);

}

getBookById(bookId: number): Observable<Book> {

  return this.http.get<Book>(`${this.apiUrl}/${bookId}`);

}

}

package com.tuitionfee.demo.model;

public class Course {

    private Long courseId;

    private String courseName;

    private int credits;

    public Course() {}

    public Course(Long courseId, String courseName, int credits) {

        this.courseId = courseId;

        this.courseName = courseName;

        this.credits = credits;

    }

    public Course(String courseName, int credits) {

        this.courseName = courseName;

        this.credits = credits;

    }

    public Long getCourseId() {

        return courseId;

    }

    public void setCourseId(Long courseId) {

        this.courseId = courseId;

    }

    public String getCourseName() {

        return courseName;

    }

    public void setCourseName(String courseName) {

        this.courseName = courseName;

    }

    public int getCredits() {

        return credits;

    }

    public void setCredits(int credits) {

        this.credits = credits;

    }

}

package com.tuitionfee.demo.model;

public class Student {

    private Long studentId;

    private String studentName;

    private String course;

    public Student(String studentName, String course) {

      this.studentName = studentName;

      this.course = course;

    }

    public Student() {}

    public Student(Long studentId, String studentName, String course) {

        this.studentId = studentId;

        this.studentName = studentName;

        this.course = course;

    }

    public Long getStudentId() {

        return studentId;

    }

    public void setStudentId(Long studentId) {

        this.studentId = studentId;

    }

    public String getStudentName() {

        return studentName;

    }

    public void setStudentName(String studentName) {

        this.studentName = studentName;

    }

    public String getCourse() {

        return course;

    }

    public void setCourse(String course) {

        this.course = course;

    }

}

package com.tuitionfee.demo.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.tuitionfee.demo.model.Course;

import com.tuitionfee.demo.repository.CourseRepository;

@Service

public class CourseService {

    @Autowired

    private CourseRepository courseRepository;

    public List<Course> getAllCourses() {

        return courseRepository.findAll();

    }

    public Course addCourse(Course course) {

        return courseRepository.save(course);

    }

}

package com.tuitionfee.demo.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.tuitionfee.demo.model.Student;

import com.tuitionfee.demo.repository.StudentRepository;

@Service

public class StudentService {

    @Autowired

    private StudentRepository studentRepository;

    public List<Student> getAllStudents() {

        return studentRepository.findAll();

    }

    public Student addStudent(Student student) {

        return studentRepository.save(student);

    }

}

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

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

import { Observable } from 'rxjs';

import { Course } from 'src/app/model/model';

import { environment } from 'src/environments/environment.development';

@Injectable({

  providedIn: 'root'

})

export class CourseService {

  private apiUrl = `${environment.apiUrl}/api/courses`;

  constructor(private http: HttpClient) {}

  getCourses(): Observable<Course[]> {

    return this.http.get<Course[]>(this.apiUrl);

  }

  addCourse(course: Course): Observable<Course> {

    return this.http.post<Course>(this.apiUrl, course);

  }

}

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

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

import { Observable } from 'rxjs';

import { Student } from '../model/model'; // Adjust path if needed

import { environment } from 'src/environments/environment.development';

@Injectable({

  providedIn: 'root'

})

export class StudentService {

  private apiUrl = `${environment.apiUrl}/api/students`;

  constructor(private http: HttpClient) {}

  getStudents(): Observable<Student[]> {

    return this.http.get<Student[]>(this.apiUrl);

  }

  addStudent(student: Student): Observable<Student> {

    return this.http.post<Student>(this.apiUrl, student);

  }

}

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()) {

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

      return false;

    }

    const requiredRole = route.data['role'];

    if (requiredRole && this.authService.getUserRole() !== requiredRole) {

      this.router.navigate(['/']);

      return false;

    }

    return true;

  }

}

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

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

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

@Component({

  selector: 'app-login',

  templateUrl: './login.component.html'

})

export class LoginComponent {

  username = '';

  password = '';

  errorMessage = '';

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

  login(): void {

    const success = this.authService.login(this.username, this.password);

    if (success) {

      const role = this.authService.getUserRole();

      if (role === 'student') {

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

      } else if (role === 'instructor') {

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

      }

    } else {

      this.errorMessage = 'Invalid username or password';

    }

  }

}

  <div>

    <h2>Login</h2>

    <form (ngSubmit)="login()">

      <label>

        Username:

        <input [(ngModel)]="username" name="username" required />

      </label>

      <br>

      <label>

        Password:

        <input [(ngModel)]="password" name="password" type="password" required />

      </label>

      <br>

      <button type="submit">Login</button>

    </form>

    <p \*ngIf="errorMessage" style="color: red;">{{ errorMessage }}</p>

  </div>

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

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

import { CourseService } from '../../services/course.service';

import { Course } from '../../models/course';

@Component({

  selector: 'app-add-course',

  templateUrl: './add-course.component.html'

})

export class AddCourseComponent implements OnInit {

  course: Course = {

    title: '',

    description: '',

    duration: '',

    instructor: '',

    category: '',

    price: 0

  };

  isEditMode = false;

  courseId: number | null = null;

  constructor(

    private courseService: CourseService,

    private router: Router,

    private route: ActivatedRoute

  ) {}

  ngOnInit(): void {

    this.route.paramMap.subscribe(params => {

      const id = params.get('id');

      if (id) {

        this.isEditMode = true;

        this.courseId = +id;

        this.courseService.getCourseById(this.courseId).subscribe(course => {

          this.course = course;

        });

      }

    });

  }

  onSubmit(form: any): void {

    if (form.invalid) return;

    if (this.isEditMode && this.courseId !== null) {

      this.courseService.updateCourse(this.courseId, this.course).subscribe(() => {

        this.router.navigate(['/course-list']);

      });

    } else {

      this.courseService.createCourse(this.course).subscribe(() => {

        this.router.navigate(['/course-list']);

      });

    }

  }

}

<form #courseForm="ngForm" (ngSubmit)="onSubmit(courseForm)">

    <input name="title" [(ngModel)]="course.title" required />

    <input name="description" [(ngModel)]="course.description" required />

    <input name="duration" [(ngModel)]="course.duration" required />

    <input name="instructor" [(ngModel)]="course.instructor" required />

    <input name="category" [(ngModel)]="course.category" required />

    <input name="price" type="number" [(ngModel)]="course.price" required />

    <button type="submit" [disabled]="courseForm.invalid">Save</button>

  </form>

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

import { CourseService } from '../../services/course.service';

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

import { Course } from '../../models/course';

@Component({

  selector: 'app-course-list',

  templateUrl: './course-list.component.html'

})

export class CourseListComponent implements OnInit {

  courses: Course[] = [];

  constructor(

    private courseService: CourseService,

    private router: Router

  ) {}

  ngOnInit(): void {

    this.courseService.getAllCourses().subscribe({

      next: (data) => this.courses = data,

      error: (err) => console.error('Error fetching courses', err)

    });

  }

  editCourse(id: number): void {

    this.router.navigate(['/edit-course', id]);

  }

  deleteCourse(id: number): void {

    this.router.navigate(['/delete-course', id]);

  }

  goToAddCourse(): void {

    this.router.navigate(['/add-course']);

  }

}

<div>

    <button (click)="goToAddCourse()">Add Course</button>

  </div>

  <table \*ngIf="courses.length > 0">

    <thead>

      <tr>

        <th>Title</th>

        <th>Description</th>

        <th>Duration</th>

        <th>Instructor</th>

        <th>Category</th>

        <th>Price</th>

        <th>Actions</th>

      </tr>

    </thead>

    <tbody>

      <tr \*ngFor="let course of courses">

        <td>{{ course.title }}</td>

        <td>{{ course.description }}</td>

        <td>{{ course.duration }}</td>

        <td>{{ course.instructor }}</td>

        <td>{{ course.category }}</td>

        <td>{{ course.price }}</td>

        <td>

          <button (click)="editCourse(course.id!)">Edit</button>

          <button (click)="deleteCourse(course.id!)">Delete</button>

        </td>

      </tr>

    </tbody>

  </table>

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

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

import { CourseService } from '../../services/course.service';

import { Course } from '../../models/course';

@Component({

  selector: 'app-delete-course',

  templateUrl: './delete-course.component.html'

})

export class DeleteCourseComponent implements OnInit {

  courseId!: number;

  course!: Course;

  isLoading = true;

  constructor(

    private route: ActivatedRoute,

    private courseService: CourseService,

    private router: Router

  ) {}

  ngOnInit(): void {

    this.courseId = +this.route.snapshot.paramMap.get('id')!;

    this.courseService.getCourseById(this.courseId).subscribe({

      next: (data) => {

        this.course = data;

        this.isLoading = false;

      },

      error: (err) => {

        console.error('Error loading course', err);

        this.isLoading = false;

      }

    });

  }

  confirmDelete(): void {

    this.courseService.deleteCourse(this.courseId).subscribe({

      next: () => {

        // alert('Course deleted successfully!');

        this.router.navigate(['/course-list']);

      },

      error: (err) => {

        console.error('Error deleting course', err);

      }

    });

  }

}

<form #registerForm="ngForm" (ngSubmit)="register()">

    <input

      name="Username"

      [(ngModel)]="user.Username"

      required

      placeholder="Username"

    />

    <input

      name="email"

      [(ngModel)]="user.email"

      required

      placeholder="Email"

      type="email"

    />

    <input

      name="password"

      [(ngModel)]="user.password"

      required

      placeholder="Password"

      type="password"

    />

    <input

      name="confirmPassword"

      [(ngModel)]="confirmPassword"

      required

      placeholder="Confirm Password"

      type="password"

    />

    <button type="submit" [disabled]="!isFormValid()">Register</button>

    <div \*ngIf="errorMessage" class="error">{{ errorMessage }}</div>

  </form>

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

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

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

@Component({

  selector: 'app-register',

  templateUrl: './register.component.html'

})

export class RegisterComponent {

  user = {

    Username: '',

    email: '',

    password: ''

  };

  confirmPassword = '';

  errorMessage = '';

  constructor(

    private authService: AuthService,

    private router: Router

  ) {}

  isFormValid(): boolean {

    return (

      this.user.Username.trim() !== '' &&

      this.user.email.trim() !== '' &&

      this.user.password.trim() !== '' &&

      this.user.password === this.confirmPassword

    );

  }

  register(): void {

    if (!this.isFormValid()) return;

    this.authService.register(this.user).subscribe({

      next: () => {

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

      },

      error: (err) => {

        this.errorMessage = 'Registration failed';

        console.error(err);

      }

    });

  }

}

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

import {

  HttpEvent,

  HttpHandler,

  HttpInterceptor,

  HttpRequest,

  HttpResponse,

  HttpErrorResponse

} from '@angular/common/http';

import { Observable, tap } from 'rxjs';

import { finalize } from 'rxjs/operators';

@Injectable()

export class ApiLoggerInterceptor implements HttpInterceptor {

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

    const started = performance.now();

    console.log('%c[Request]', 'color: blue; font-weight: bold;');

    console.log('URL:', req.url);

    console.log('Method:', req.method);

    console.log('Body:', req.body);

    return next.handle(req).pipe(

      tap({

        next: (event) => {

          if (event instanceof HttpResponse) {

            const elapsed = performance.now() - started;

            console.log('%c[Response]', 'color: green; font-weight: bold;');

            console.log('Status:', event.status);

            console.log('Body:', event.body);

            console.log(`Response time: ${elapsed.toFixed(2)} ms`);

          }

        },

        error: (error: HttpErrorResponse) => {

          const elapsed = performance.now() - started;

          console.error('%c[Error Response]', 'color: red; font-weight: bold;');

          console.error('Status:', error.status);

          console.error('Error:', error.message);

          console.error(`Response time: ${elapsed.toFixed(2)} ms`);

        }

      })

    );

  }

}

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('jwtToken');

    if (token) {

      const cloned = req.clone({

        setHeaders: {

          Authorization: `Bearer ${token}`

        }

      });

      return next.handle(cloned);

    }

    return next.handle(req);

  }

}

export interface Course {

  id?: number;

  title: string;

  description: string;

  instructor: string;

  category: string;

  duration: string;

  price: number;

}

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

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

import { Observable } from 'rxjs';

import { environment } from '../../environments/environment';

@Injectable({

  providedIn: 'root'

})

export class AuthService {

  private apiUrl = `${environment.apiUrl}/users`;

  constructor(private http: HttpClient) {}

  register(user: { email: string; password: string; Username?: string }): Observable<any> {

    return this.http.post(this.apiUrl, user);

  }

  // You can add login, logout, and token management methods here as needed

}

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

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

import { Observable } from 'rxjs';

import { Course } from '../models/course';

import { environment } from '../../environments/environment';

@Injectable({

  providedIn: 'root'

})

export class CourseService {

  private apiUrl = `${environment.apiUrl}/courses`;

  constructor(private http: HttpClient) {}

  getAllCourses(): Observable<Course[]> {

    return this.http.get<Course[]>(this.apiUrl);

  }

  getCourseById(id: number): Observable<Course> {

    return this.http.get<Course>(`${this.apiUrl}/${id}`);

  }

  createCourse(course: Course): Observable<Course> {

    return this.http.post<Course>(this.apiUrl, course);

  }

  updateCourse(id: number, course: Course): Observable<void> {

    return this.http.put<void>(`${this.apiUrl}/${id}`, course);

  }

  deleteCourse(id: number): Observable<void> {

    return this.http.delete<void>(`${this.apiUrl}/${id}`);

  }

}

S2A3 direct test and submit

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

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

@Component({

  selector: 'app-login-page',

  template: `

    <form (ngSubmit)="onLogin()" #loginForm="ngForm">

      <input name="username" [(ngModel)]="loginUser.username" required />

      <input name="password" [(ngModel)]="loginUser.password" required type="password" />

      <button type="submit">Login</button>

    </form>

    <p>{{ loginMessage }}</p>

  `

})

export class LoginPageComponent {

  loginUser = {

    username: '',

    password: ''

  };

  loginMessage = '';

  constructor(private authService: AuthService) {}

  onLogin(): void {

    this.authService.loginByUsername(this.loginUser.username, this.loginUser.password).subscribe((users: any[]) => {

      if (users.length > 0) {

        this.loginMessage = 'Login successful!';

      } else {

        this.loginMessage = 'Invalid credentials!';

      }

    });

  }

}

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

import { Observable, of } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class AuthService {

  loginByUsername(username: string, password: string): Observable<any[]> {

    // This would normally make an HTTP request

    return of([]);

  }

 register(user: { username: string; email: string; password: string }): Observable<any> {

  // This would normally make an HTTP request

  return of({});

}

}

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

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

@Component({

  selector: 'app-sign-up',

  template: `

    <form (ngSubmit)="onRegister()" #signUpForm="ngForm">

      <input name="username" [(ngModel)]="user.username" required />

      <input name="email" [(ngModel)]="user.email" required type="email" />

      <input name="password" [(ngModel)]="user.password" required type="password" />

      <button type="submit">Register</button>

    </form>

    <p>{{ registerMessage }}</p>

  `

})

export class SignUpComponent {

  user = {

    username: '',

    email: '',

    password: ''

  };

  registerMessage = '';

  constructor(private authService: AuthService) {}

  onRegister(): void {

    if (this.user.password.length < 8) {

      this.registerMessage = 'Password must be at least 8 characters long.';

      return;

    }

    this.authService.register(this.user).subscribe(() => {

      this.registerMessage = 'Registration successful!';

    });

  }

}

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

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

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

import { environment } from '../../environments/environment';

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css'],

})

export class HomeComponent implements OnInit {

 ngOnInit(): void {

   throw new Error('Method not implemented.');

 }

 //Write a code here

 private apiUrl = `${environment.apiUrl}/users`;

 private apiUrl1 = `${environment.apiUrl}/contacts`;

}

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

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

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class AuthGuard implements CanActivate {

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {

    throw new Error('Method not implemented.');

  }

  //Write a code here

}

// auth.service.ts

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

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

import { Observable } from 'rxjs';

import { environment } from '../environments/environment';

@Injectable({

  providedIn: 'root'

})

export class AuthService {

  private apiUrl = environment.apiUrl; // Ensure this is defined in your environment.ts

  constructor(private http: HttpClient) {}

  register(user: { username: string; email: string; password: string }): Observable<any> {

    return this.http.post(`${this.apiUrl}/register`, user);

  }

  loginByUsername(user: { username: string; password: string }): Observable<any> {

    return this.http.post(`${this.apiUrl}/login`, user);

  }

}

// home.component.ts

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

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css']

})

export class HomeComponent {

  username: string = '';

  tasks: any[] = [];

  isEditing: boolean = false;

  newTask: any = null;

  editTask(task: any): void {

    this.isEditing = true;

    this.newTask = task;

  }

}

// sign-up.component.ts

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

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

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

@Component({

  selector: 'app-sign-up',

  templateUrl: './sign-up.component.html',

  styleUrls: ['./sign-up.component.css']

})

export class SignUpComponent {

  user = {

    username: '',

    email: '',

    password: ''

  };

  registerMessage: string = '';

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

  onRegister(): void {

    if (this.user.password.length < 8) {

      this.registerMessage = 'Password must be at least 8 characters long.';

      return;

    }

    this.authService.register(this.user).subscribe({

      next: () => {

        this.registerMessage = 'Registration successful!';

        // Optionally navigate to another page

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

      },

      error: () => {

        this.registerMessage = 'Registration failed.';

      }

    });

  }

}

// login-page.component.ts

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

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

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

@Component({

  selector: 'app-login-page',

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

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

})

export class LoginPageComponent {

  loginUser = {

    username: '',

    password: ''

  };

  loginMessage: string = '';

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

  onLogin(): void {

    this.authService.loginByUsername(this.loginUser).subscribe({

      next: (response: any[]) => {

        if (response.length > 0) {

          this.loginMessage = 'Login successful!';

          // Optionally navigate to another page

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

        } else {

          this.loginMessage = 'Invalid credentials!';

             error: () => {

        this.loginMessage = 'Login failed due to server error.';

      }

  }}})

  }

}

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

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

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

import { environment } from 'src/environments/environment';

@Component({

  selector: 'app-login-page',

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

})

export class LoginPageComponent {

  loginUser = { username: '', password: '' };

  loginMessage = 'Invalid credentials!';

  constructor(private http: HttpClient, private router: Router) {}

  onLogin(): void {

    const { username, password } = this.loginUser;

    this.http.get<any[]>(`${environment.apiUrl}/users?username=${username}&password=${password}`)

      .subscribe({

        next: (users) => {

          if (users.length > 0) {

            this.loginMessage = 'Login successful!';

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

          }

        },

        error: () => {

          this.loginMessage = 'Invalid credentials!';

        }

      });

  }

}

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

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

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

import { environment } from 'src/environments/environment';

@Component({

  selector: 'app-sign-up',

  templateUrl: './sign-up.component.html',

})

export class SignUpComponent {

  user = { username: '', email: '', password: '' };

  registerMessage = '';

  constructor(private http: HttpClient, private router: Router) {}

  onRegister(): void {

    const { username, email, password } = this.user;

    if (!username) {

      this.registerMessage = 'Username is required.';

      return;

    }

    if (password.length < 8) {

      this.registerMessage = 'Password must be at least 8 characters long.';

      return;

    }

    this.http.post(`${environment.apiUrl}/users`, { username, email, password })

      .subscribe({

        next: () => {

          this.registerMessage = 'Registration successful! Redirecting to login page';

          this.user = { username: '', email: '', password: '' };

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

        },

        error: () => {

          this.registerMessage = 'Registration failed!';

        }

      });

  }

}