**Online Course Registration System**

```
package com.example.course.controller;

import com.example.course.entity.Course;
import com.example.course.entity.Student;
import com.example.course.service.CourseService;
import com.example.course.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api")
public class CourseController {
    @Autowired
    private CourseService courseService;

    @Autowired
    private StudentService studentService;

    // Course endpoints
    @PostMapping("/courses")
    public Course addCourse(@RequestBody Course course) {
        return courseService.addCourse(course);
    }

    @GetMapping("/courses")
    public List<Course> getAllCourses() {
        return courseService.getAllCourses();
    }

    // Student endpoints
    @PostMapping("/students")
    public Student addStudent(@RequestBody Student student) {
        return studentService.addStudent(student);
    }

    @GetMapping("/students")
    public List<Student> getAllStudents() {
        return studentService.getAllStudents();
    }

    @PostMapping("/students/{studentId}/courses/{courseId}/enroll")
    public ResponseEntity<Void> enrollStudent(@PathVariable Long studentId, @PathVariable Long
courseId) {
        courseService.enrollStudent(studentId, courseId);
        return ResponseEntity.ok().build();
    }

    @PostMapping("/students/{studentId}/courses/{courseId}/unenroll")
    public ResponseEntity<Void> unenrollStudent(@PathVariable Long studentId, @PathVariable Long
courseId) {
        courseService.unenrollStudent(studentId, courseId);
        return ResponseEntity.ok().build();
    }
}

package com.example.course.entity;

import java.util.HashSet;
import java.util.Set;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;

@Entity
public class Course {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   private String title;
   private String description;
   private int credits;

   @ManyToMany(mappedBy = "courses", cascade = CascadeType.ALL)
   private Set<Student> students = new HashSet<>();

   // Getters and Setters
   public Long getId() { return id; }
   public void setId(Long id) { this.id = id; }
   public String getTitle() { return title; }
   public void setTitle(String title) { this.title = title; }
   public String getDescription() { return description; }
   public void setDescription(String description) { this.description = description; }
   public int getCredits() { return credits; }
   public void setCredits(int credits) { this.credits = credits; }
   public Set<Student> getStudents() { return students; }
```

```
    public void setStudents(Set<Student> students) { this.students = students; }
}

package com.example.course.entity;

import java.util.HashSet;
import java.util.Set;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.JoinTable;
import jakarta.persistence.ManyToMany;

@Entity
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String email;

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(
        name = "student_course",
        joinColumns = @JoinColumn(name = "student_id"),
        inverseJoinColumns = @JoinColumn(name = "course_id")
    )
    private Set<Course> courses = new HashSet<>();

    // 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 getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public Set<Course> getCourses() { return courses; }
    public void setCourses(Set<Course> courses) { this.courses = courses; }
}

package com.example.course.repository;

import com.example.course.entity.Student;
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Long> {
    Student findByEmail(String email);
}

package com.example.course.repository;

import com.example.course.entity.Course;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CourseRepository extends JpaRepository<Course, Long> {
}
```

```
package com.example.course.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.example.course.entity.Course;
import com.example.course.entity.Student;
import com.example.course.repository.CourseRepository;
import com.example.course.repository.StudentRepository;

@Service
@Transactional
public class CourseService {
  @Autowired
  private CourseRepository courseRepository;

  @Autowired
  private StudentRepository studentRepository;

  public Course addCourse(Course course) {
    return courseRepository.save(course);
  }

  public List<Course> getAllCourses() {
    return courseRepository.findAll();
  }

  public void enrollStudent(Long studentId, Long courseId) {
    Course course = courseRepository.findById(courseId).orElseThrow(() -> new
RuntimeException("Course not found"));

    Student student = studentRepository.findById(studentId).orElseThrow(() -> new
RuntimeException("Student not found"));

    if (student.getCourses().contains(course)) {

        throw new RuntimeException("Student is already enrolled in this course");

    }

    student.getCourses().add(course);

    course.getStudents().add(student);

    courseRepository.save(course);

    studentRepository.save(student);

  }


  public void unenrollStudent(Long studentId, Long courseId) {

    Course course = courseRepository.findById(courseId).orElseThrow(() -> new
RuntimeException("Course not found"));

    Student student = studentRepository.findById(studentId).orElseThrow(() -> new
RuntimeException("Student not found"));

    student.getCourses().remove(course);

    course.getStudents().remove(student);

    courseRepository.save(course);

    studentRepository.save(student);

  }

}


package com.example.course.service;


import com.example.course.entity.Student;

import com.example.course.repository.StudentRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;
```

```java
import java.util.List;


@Service
@Transactional
public class StudentService {
   @Autowired
   private StudentRepository studentRepository;


   public Student addStudent(Student student) {
      return studentRepository.save(student);
   }


   public List<Student> getAllStudents() {
      return studentRepository.findAll();
   }


   public Student getStudentById(Long id) {
      return studentRepository.findById(id)
            .orElseThrow(() -> new RuntimeException("Student not found"));
   }
}
```

```properties
spring.application.name=online-course-registration-system
#spring.datasource.url=jdbc:h2:mem:testdb
#spring.datasource.driverClassName=org.h2.Driver
#spring.datasource.username=sa
#spring.datasource.password=password
#spring.h2.console.enabled=true
#spring.jpa.hibernate.ddl-auto=update
# H2 Database settings
```

```
spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console
```

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

	<modelVersion>4.0.0</modelVersion>

	<parent>

		<groupId>org.springframework.boot</groupId>

		<artifactId>spring-boot-starter-parent</artifactId>

		<version>3.3.4</version>

		<relativePath/> <!-- lookup parent from repository -->

	</parent>

	<groupId>com.example</groupId>

	<artifactId>course</artifactId>

	<version>0.0.1-SNAPSHOT</version>

	<name>online-course-registration-system</name>

	<description>Demo project for Spring Boot</description>

	<url/>

	<licenses>

		<license/>

	</licenses>

	<developers>

		<developer/>

	</developers>
```

```
<scm>
	<connection/>
	<developerConnection/>
	<tag/>
	<url/>
</scm>
<properties>
	<java.version>17</java.version>
</properties>
<dependencies>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jpa</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>

	<dependency>
		<groupId>com.h2database</groupId>
		<artifactId>h2</artifactId>
		<scope>runtime</scope>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-test</artifactId>
		<scope>test</scope>
	</dependency>
	<dependency><groupId>com.example</groupId>
		<artifactId>demo</artifactId>
```

```
                        <version>0.0.1-SNAPSHOT</version>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>
```