


Coding Problem – College Management System I

Problem Statement

 Classroom

Spring Data JPA Queries
[Coding Problem - College Management System](#)

Problem Submissions Solutions Doubts

✓ College Management System - Spring Data JPA Queries

Easy • Score 80/80 • Advanced Java • Spring Hibernate

Problem statement [Send feedback](#)

Baruch College has developed a student management system to record and manage student information, course enrollments, and grades. The system has been developed using Java and Spring Boot and uses JPA for database access. The database schema, JPA entities, and REST APIs have already been implemented. The college administration has identified a need to add a new feature to the system that fetches all the students enrolled in a course. Based on the given template, explore the student management system current working and add new functionality as per the mentioned task. (Sample Screenshots of entities are given as reference below)

Tasks:

1. Write a **Derived query** to retrieve the list of all students enrolled in a given course. The function should only take one parameter: course name.
2. In your **MainController** class create a method **getAllStudentsByCourse()** with the following properties:
 - API Path: GET /students/{course}
 - It takes a string of type course as input and returns the list of students.
3. Also, add a method **getAllStudentsByCourse(String course)** in your **MainService** class which does the following:
 - It takes a string of type course as input.
 - It should return a list of students from the student repository.
4. Lastly, write the **Derived Query** in the **Student Repository**.

Special Instruction for submitting the solution:

1. Remove the target folder from the root directory of your project.
2. Remove the "test" folder from your "src" folder.

Note:-

1. Don't change the versions of spring-boot (3.0.0) and Java (17). If needed then install the same.
2. This project uses the "H2 Database".
3. Don't change the application.yml file as it contains "h2 database" configurations.

Screenshot for reference:-

Screenshots for references

```
@Entity
public class Student {

    private Integer id;
    private String name;

    @ManyToMany
    //Using HashSet to avoid duplicate entries.
    private Set<Courses> courses = new HashSet<>();
    @OneToMany(cascade = CascadeType.ALL, mappedBy="Student")
    private Set<Grade> grades = new HashSet<>();
}
```

```

@Entity
public class Course {

    private Integer id;
    private String name;

    @ManyToMany(mappedBy = "courses", cascade = CascadeType.ALL)
    private Set<Student> students = new HashSet<>();
    @OneToMany(cascade = CascadeType.ALL, mappedBy="course")
    private List<Grade> grades;
}

```

```

@Entity
public class Grade {

    private Integer id;
    private double marks;

    @ManyToOne
    @JoinColumn(name="student_id", nullable=false)
    private Student student;

    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name="course_id", nullable=false)
    private Course course;
}

```