INSTITUTION OF TECHNOLOGY OF CAMBODIA

DEPARTMENT: GIC I4-B

Report of Software Engineering

Project 3: Course Enrollment and Classroom Scheduling System

Topic : Progress 2

Name of Students	Students ID
1. PANG Lythong	e20220161
2. SOEURY Sreyno	e20220908
3. VICHETH Sokhsedtha	e202201549
4. NGET Darapich	e20221646
5. KEO Chanponlork	e20220660

Lecturer: ROEUN Pacharoth

Academic year 2025-2026

1. Security and Authentication Lead (Vicheth Sokhsedtha) Done all Tasks

Goal: You are the gatekeeper. You ensure only the right people get in and see the right things.

- ✓ ~~**Key Entities:** `User`, `Role`, `Privilege`.~~
- ✓ ~~**Specific Tasks:**~~
    - ✓ ~~**Spring Security Config:** Create the `SecurityConfig.java` class. Configure the `SecurityFilterChain` to define which URLs are public (e.g., `/login`, `/css/**`) and which are private.~~
    - ✓ ~~**User Management:** Create the `UserDetailsService` implementation to load users from the database.~~
    - ✓ ~~**Roles:** Implement 3 distinct roles:~~
        - `ROLE_ADMIN`: Can create courses, assign lecturers, and manage classrooms.
        - `ROLE_LECTURER`: Can view their assigned courses and see the student list.
        - `ROLE_STUDENT`: Can browse courses and enroll.
    - ✓ ~~**Registration:** Build the logic to register new accounts (encrypting passwords using `BCryptPasswordEncoder`).~~
- ✓ ~~**Deliverables:** Login page, Registration page, and "Access Denied" error handling.~~

2. Main Entity CRUD Lead (The Core Data) (Keo Chanponlok)

Goal: You manage the "nouns" of the system—the physical things that exist regardless of the schedule.

- ✓ ~~**Key Entities:** `Course`, `Classroom`, `Department` (optional).~~
- ✓ ~~**Specific Tasks:**~~
    - **Course Management:**

- ☑ ~~Create `CourseController`, `CourseService`, and `CourseRepository`.~~
    - ☑ ~~Fields needed: `courseName`, `courseCode` (e.g., CS101), `credits`, `description`, `capacity`.~~
    - ☑ ~~**Validation:** Ensure `courseCode` is unique and `capacity` is a positive number.~~
  - **Classroom Management:**
    - ☑ ~~Create `ClassroomController` etc.~~
    - ☑ ~~Fields needed: `roomNumber`, `building`, `maxCapacity`.~~
  - **Lecturer Profile:** (If not handled by Security) Create a `Lecturer` entity to store specific details like "Department" or "Office Hours."
- ☑ ~~**Deliverables:** Admin pages to "Add New Course," "Edit Classroom," and "List All Courses."~~

I have completed the basic tasks for the Main Entity CRUD module, including creating and managing core entities such as Course, Classroom, and Lecturer. The required controllers, services, repositories, and validations have been implemented, and the system is ready for integration with other modules. I will continue to improve this part by adding more complex logic, validations, and admin interfaces in the next phase of development.

Nget Darapich

Entities Created

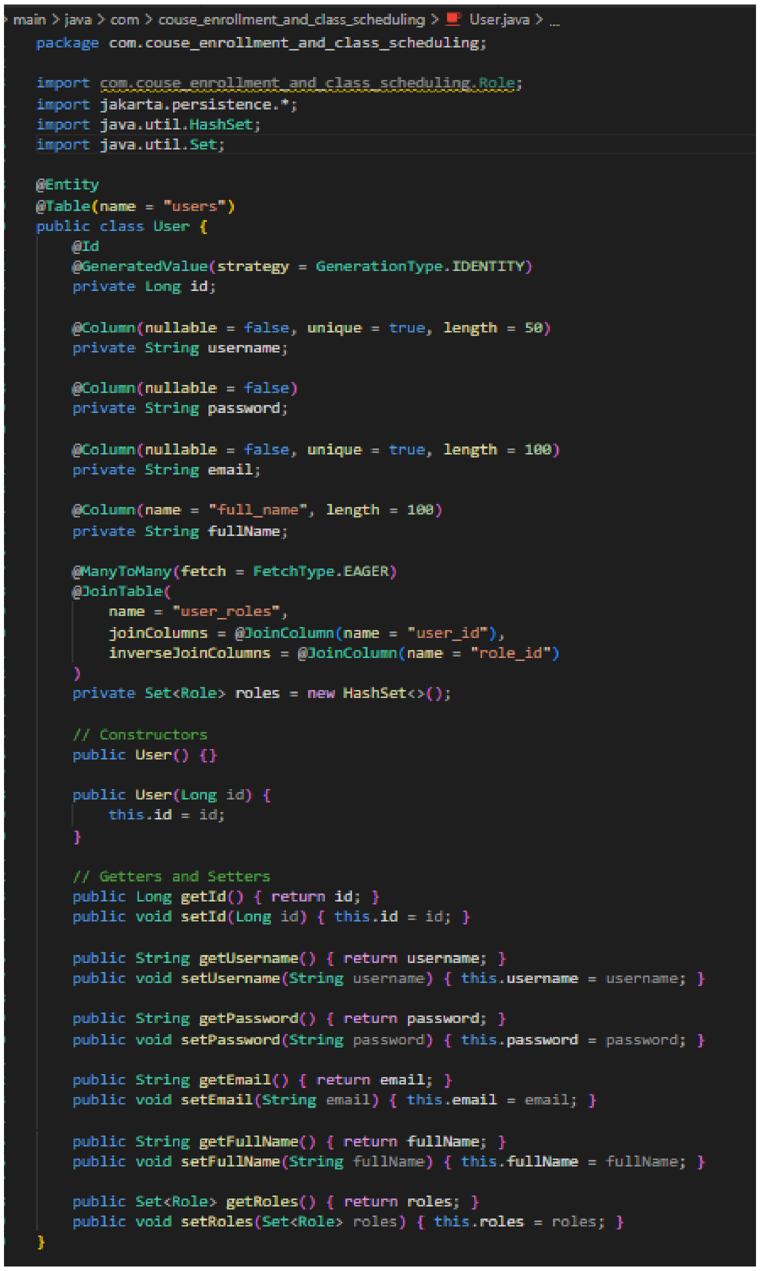
You have defined multiple **JPA entities** representing the system domain:

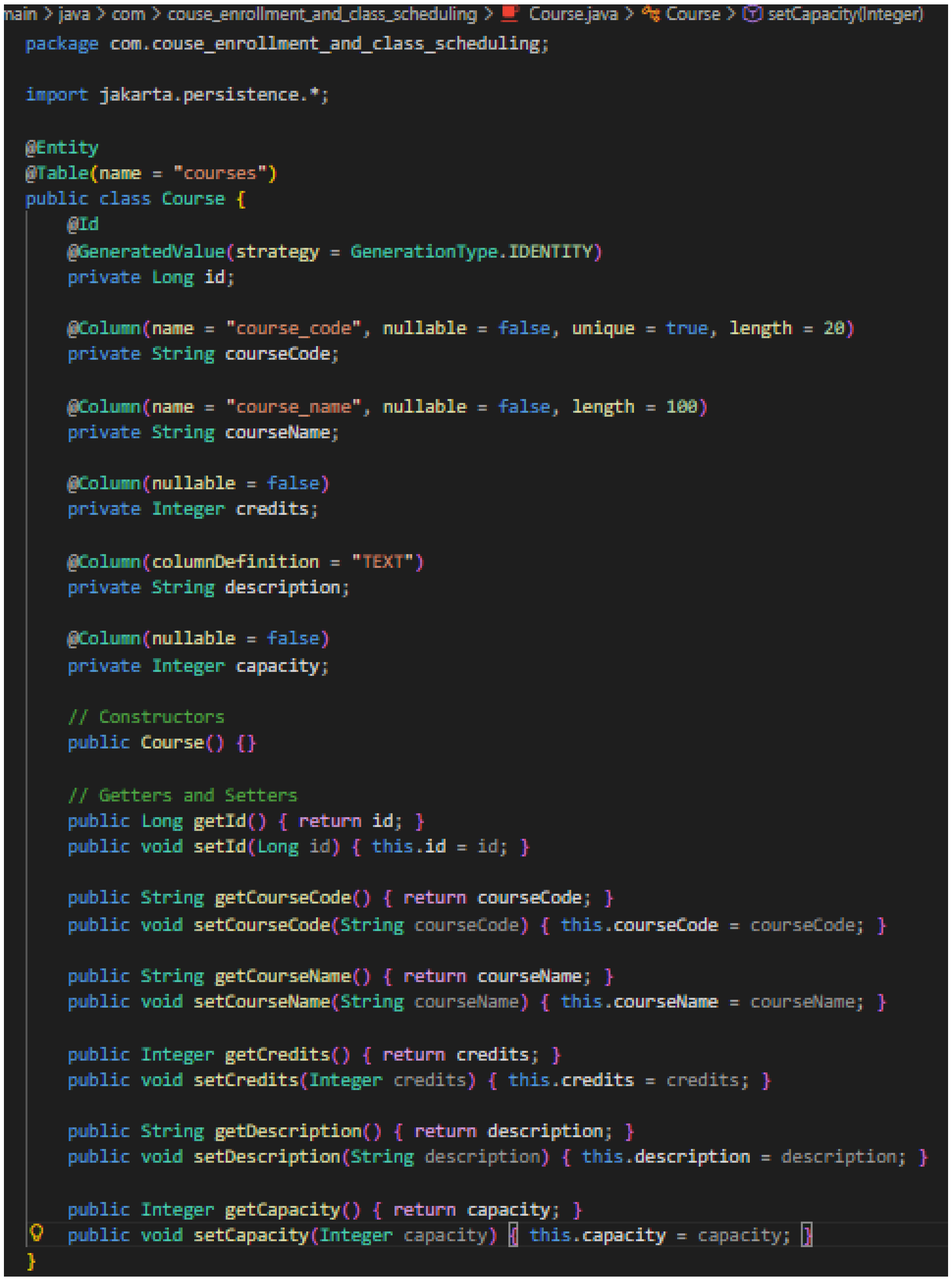
- **User**
  - Fields: `id`, `username`, `password`, `email`, `fullName`
- **Role**
  - Represents user roles (e.g., ADMIN, STUDENT, INSTRUCTOR)

- **Course**

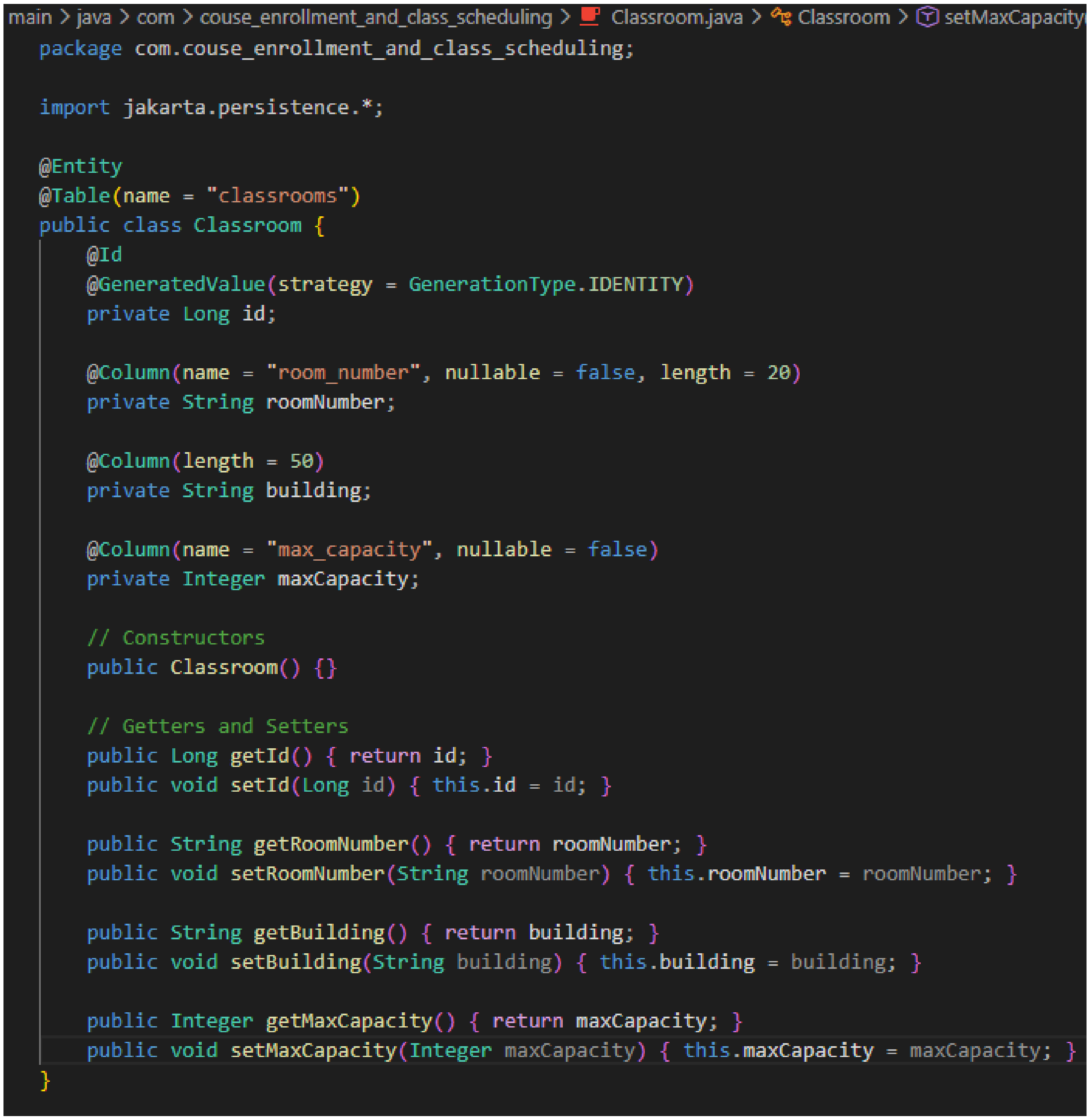
- **Classroom**
- **ClassSchedule**
- **Enrollment**

These entities are annotated with `@Entity`, `@Table`, and proper JPA mappings.

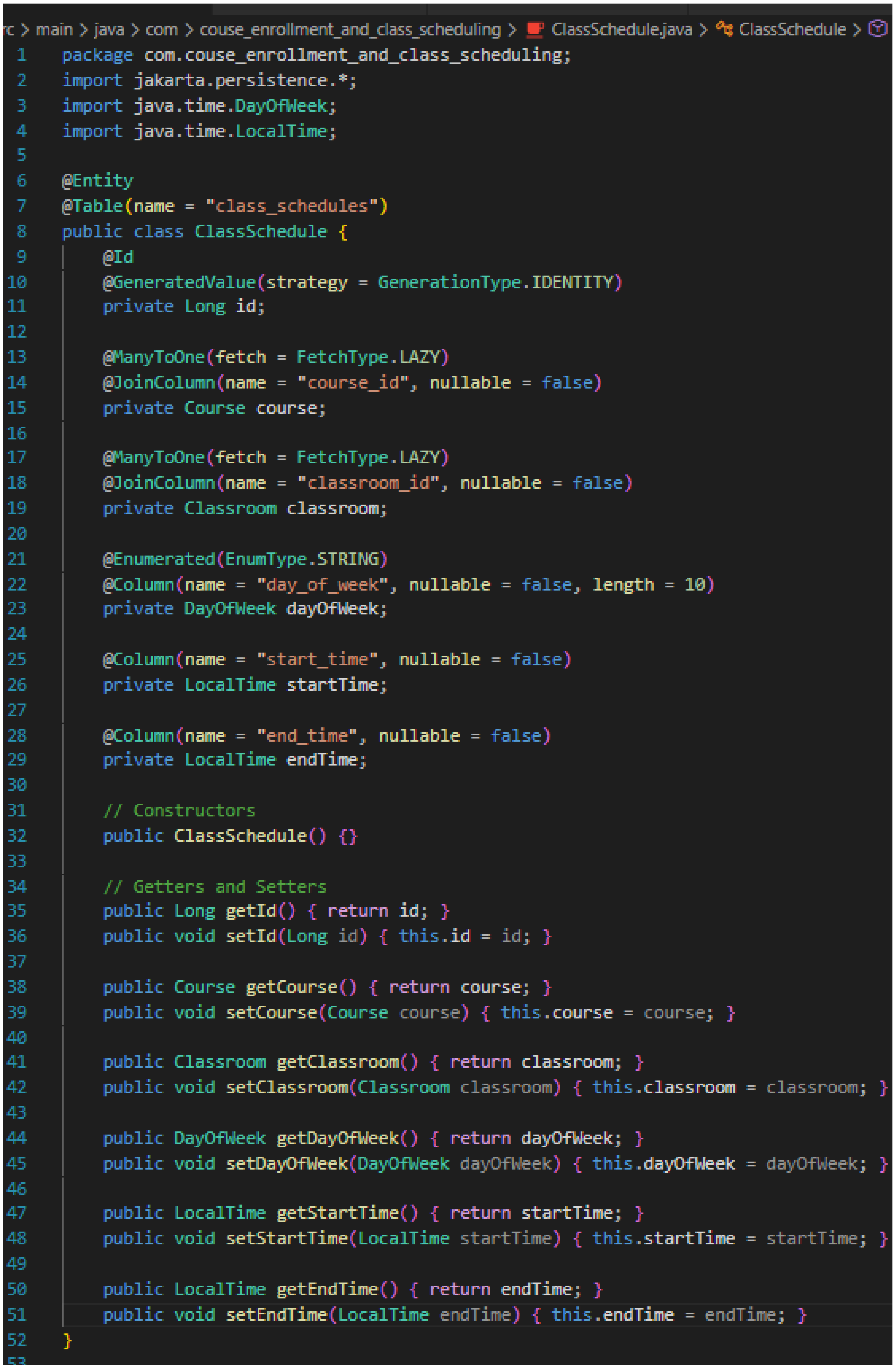

```
› main › java › com › couse_enrollment_and_class_scheduling › User.java › ...
package com.couse_enrollment_and_class_scheduling;

import com.couse_enrollment_and_class_scheduling.Role;
import jakarta.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true, length = 50)
    private String username;

    @Column(nullable = false)
    private String password;

    @Column(nullable = false, unique = true, length = 100)
    private String email;

    @Column(name = "full_name", length = 100)
    private String fullName;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(
        name = "user_roles",
        joinColumns = @JoinColumn(name = "user_id"),
        inverseJoinColumns = @JoinColumn(name = "role_id")
    )
    private Set<Role> roles = new HashSet<>();

    // Constructors
    public User() {}

    public User(Long id) {
        this.id = id;
    }

    // Getters and Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public String getFullName() { return fullName; }
    public void setFullName(String fullName) { this.fullName = fullName; }

    public Set<Role> getRoles() { return roles; }
    public void setRoles(Set<Role> roles) { this.roles = roles; }
}
```


nain > java > com > couse_enrollment_and_class_scheduling > Course.java > Course > setCapacity(Integer)

```java
package com.couse_enrollment_and_class_scheduling;

import jakarta.persistence.*;

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

    @Column(name = "course_code", nullable = false, unique = true, length = 20)
    private String courseCode;

    @Column(name = "course_name", nullable = false, length = 100)
    private String courseName;

    @Column(nullable = false)
    private Integer credits;

    @Column(columnDefinition = "TEXT")
    private String description;

    @Column(nullable = false)
    private Integer capacity;

    // Constructors
    public Course() {}

    // Getters and Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getCourseCode() { return courseCode; }
    public void setCourseCode(String courseCode) { this.courseCode = courseCode; }

    public String getCourseName() { return courseName; }
    public void setCourseName(String courseName) { this.courseName = courseName; }

    public Integer getCredits() { return credits; }
    public void setCredits(Integer credits) { this.credits = credits; }

    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }

    public Integer getCapacity() { return capacity; }
    public void setCapacity(Integer capacity) { this.capacity = capacity; }
}
```

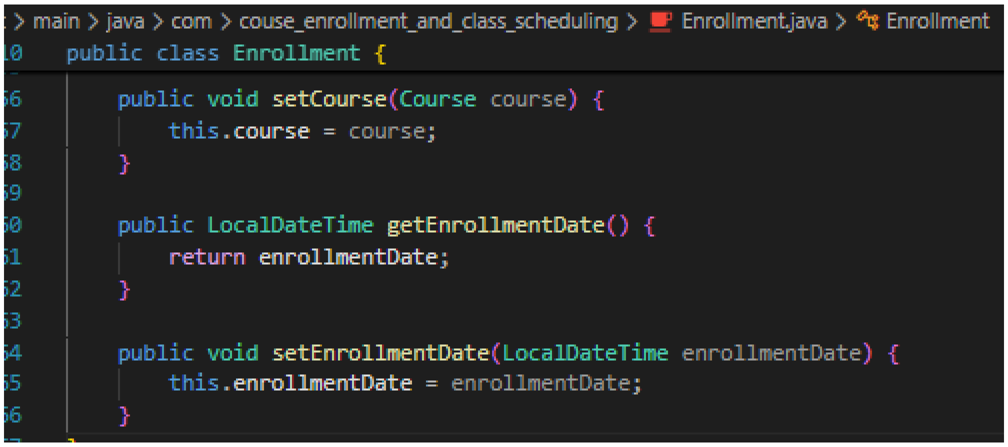

main > java > com > couse_enrollment_and_class_scheduling > Classroom.java > Classroom > setMaxCapacity

```java
package com.couse_enrollment_and_class_scheduling;

import jakarta.persistence.*;

@Entity
@Table(name = "classrooms")
public class Classroom {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "room_number", nullable = false, length = 20)
    private String roomNumber;

    @Column(length = 50)
    private String building;

    @Column(name = "max_capacity", nullable = false)
    private Integer maxCapacity;

    // Constructors
    public Classroom() {}

    // Getters and Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getRoomNumber() { return roomNumber; }
    public void setRoomNumber(String roomNumber) { this.roomNumber = roomNumber; }

    public String getBuilding() { return building; }
    public void setBuilding(String building) { this.building = building; }

    public Integer getMaxCapacity() { return maxCapacity; }
    public void setMaxCapacity(Integer maxCapacity) { this.maxCapacity = maxCapacity; }
}
```



rc > main > java > com > couse_enrollment_and_class_scheduling > ClassSchedule.java > ClassSchedule >

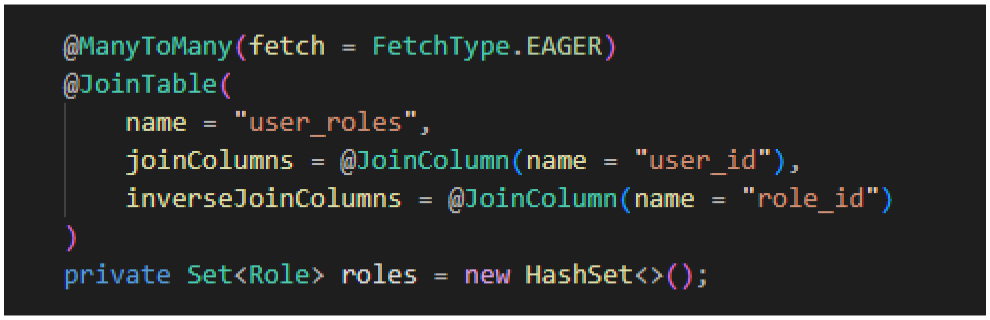
```java
package com.couse_enrollment_and_class_scheduling;
import jakarta.persistence.*;
import java.time.DayOfWeek;
import java.time.LocalTime;

@Entity
@Table(name = "class_schedules")
public class ClassSchedule {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "course_id", nullable = false)
    private Course course;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "classroom_id", nullable = false)
    private Classroom classroom;

    @Enumerated(EnumType.STRING)
    @Column(name = "day_of_week", nullable = false, length = 10)
    private DayOfWeek dayOfWeek;

    @Column(name = "start_time", nullable = false)
    private LocalTime startTime;

    @Column(name = "end_time", nullable = false)
    private LocalTime endTime;

    // Constructors
    public ClassSchedule() {}

    // Getters and Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public Course getCourse() { return course; }
    public void setCourse(Course course) { this.course = course; }

    public Classroom getClassroom() { return classroom; }
    public void setClassroom(Classroom classroom) { this.classroom = classroom; }

    public DayOfWeek getDayOfWeek() { return dayOfWeek; }
    public void setDayOfWeek(DayOfWeek dayOfWeek) { this.dayOfWeek = dayOfWeek; }

    public LocalTime getStartTime() { return startTime; }
    public void setStartTime(LocalTime startTime) { this.startTime = startTime; }

    public LocalTime getEndTime() { return endTime; }
    public void setEndTime(LocalTime endTime) { this.endTime = endTime; }
}
```


> main > java > com > couse_enrollment_and_class_scheduling > Enrollment.java > Enrollment

```java
public class Enrollment {
    public void setCourse(Course course) {
        this.course = course;
    }

    public LocalDateTime getEnrollmentDate() {
        return enrollmentDate;
    }

    public void setEnrollmentDate(LocalDateTime enrollmentDate) {
        this.enrollmentDate = enrollmentDate;
    }
```

Relationships Implemented

- **User ↔ Role** → `@ManyToMany`
- **Enrollment** connects users to courses/schedules
- Scheduling entities link **Course**, **Classroom**, and time information



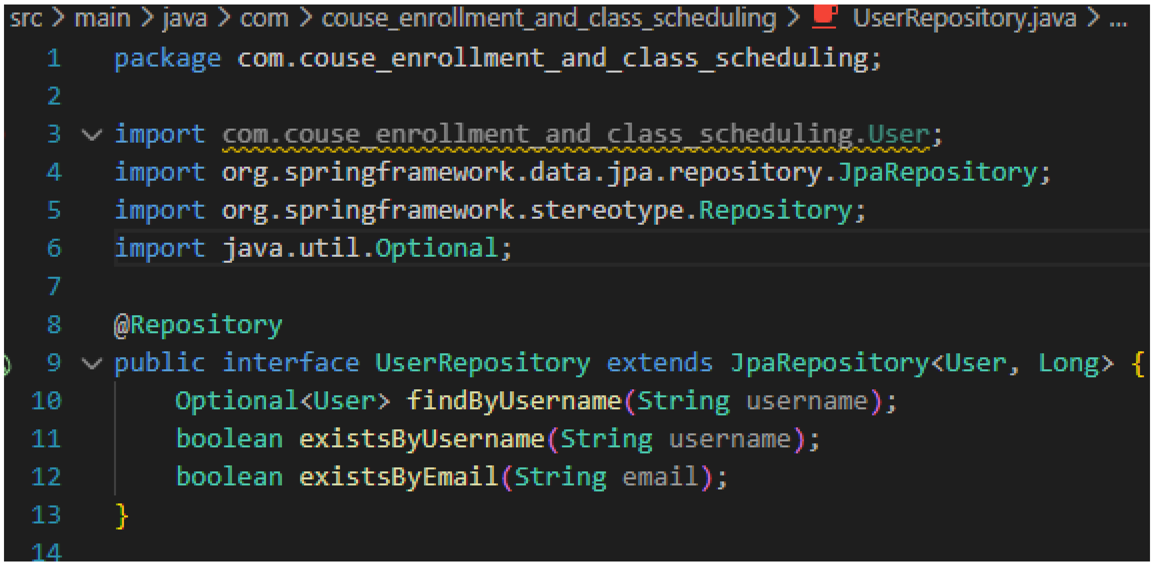
```java
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(
    name = "user_roles",
    joinColumns = @JoinColumn(name = "user_id"),
    inverseJoinColumns = @JoinColumn(name = "role_id")
)
private Set<Role> roles = new HashSet<>();
```

Repositories

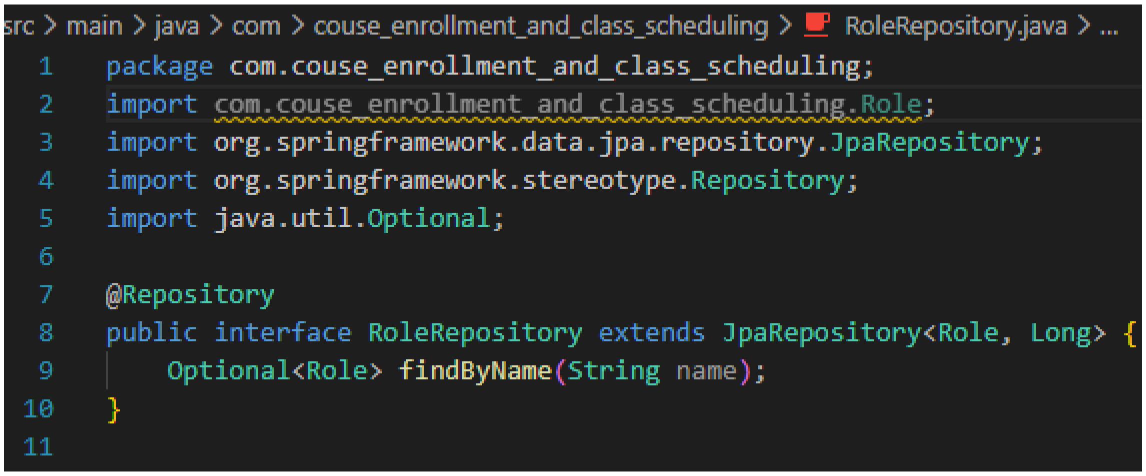
You created **Spring Data JPA repositories** for each main entity:

- `UserRepository`
- `RoleRepository`
- `CourseRepository`
- `ClassScheduleRepository`
- `EnrollmentRepository`

These handle database operations automatically.

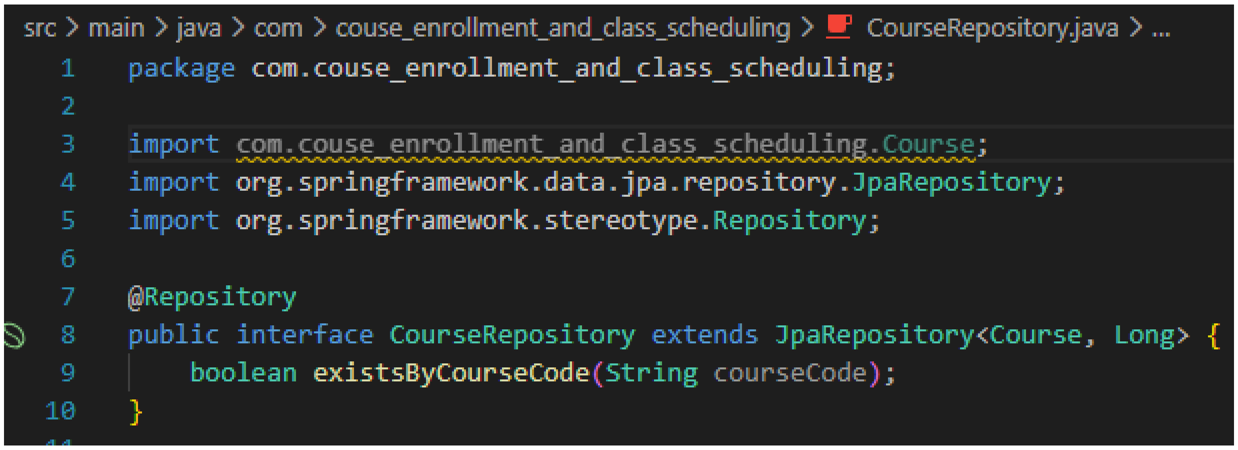

src > main > java > com > couse_enrollment_and_class_scheduling > UserRepository.java > ...

```java
package com.couse_enrollment_and_class_scheduling;

import com.couse_enrollment_and_class_scheduling.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.Optional;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
    boolean existsByUsername(String username);
    boolean existsByEmail(String email);
}
```



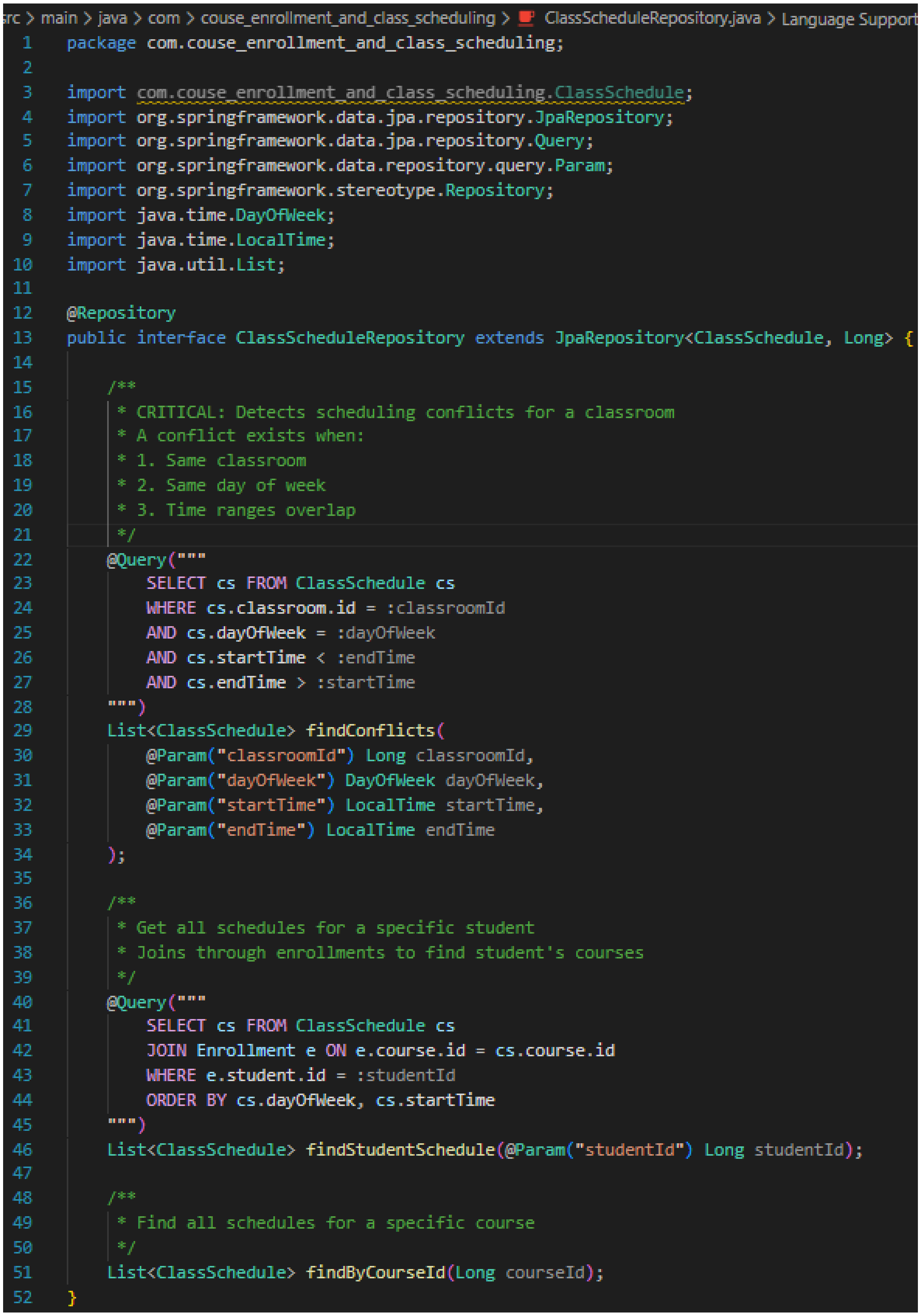
src > main > java > com > couse_enrollment_and_class_scheduling > RoleRepository.java > ...

```java
package com.couse_enrollment_and_class_scheduling;
import com.couse_enrollment_and_class_scheduling.Role;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.Optional;

@Repository
public interface RoleRepository extends JpaRepository<Role, Long> {
    Optional<Role> findByName(String name);
}

```



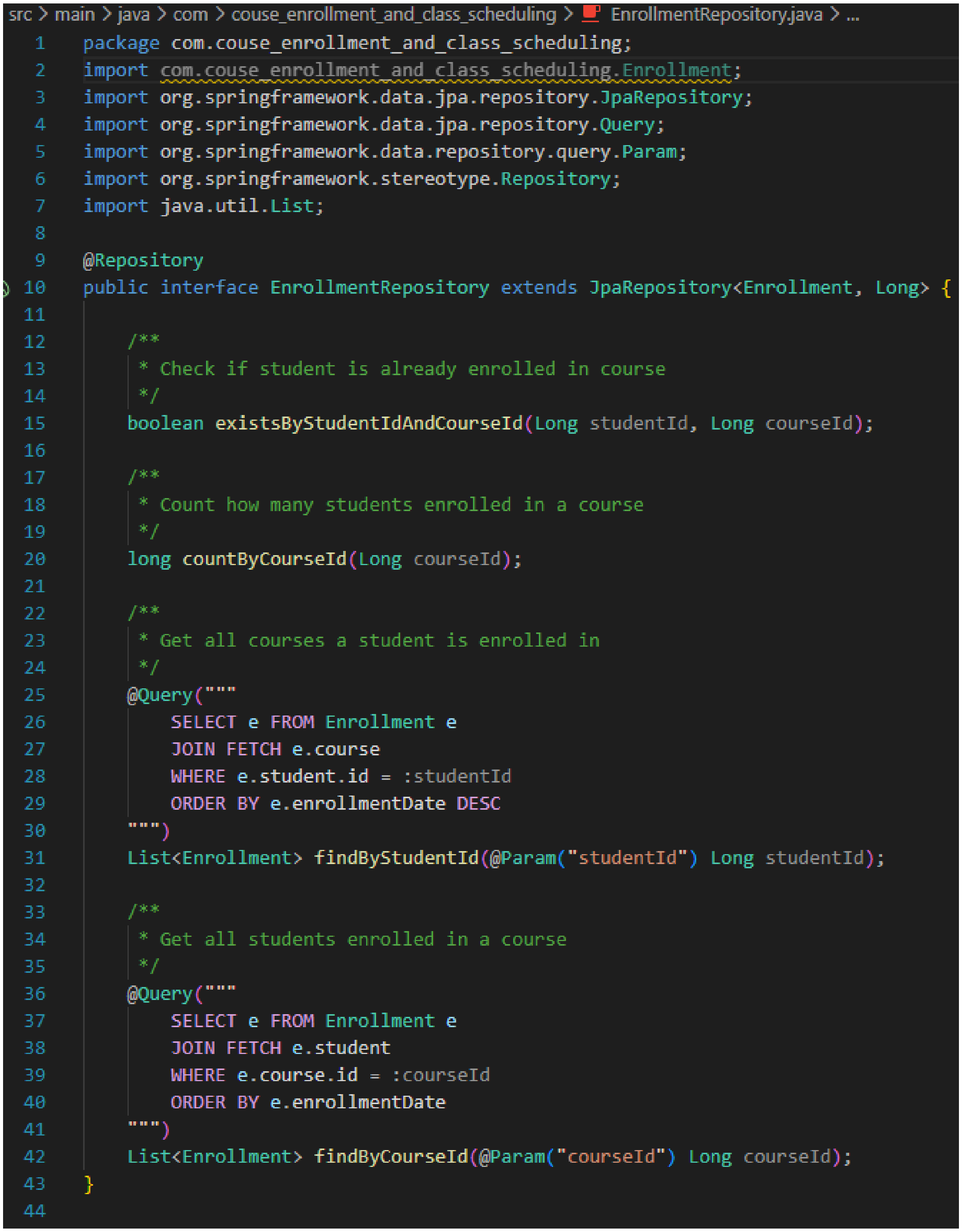
src > main > java > com > couse_enrollment_and_class_scheduling > CourseRepository.java > ...

```java
package com.couse_enrollment_and_class_scheduling;

import com.couse_enrollment_and_class_scheduling.Course;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface CourseRepository extends JpaRepository<Course, Long> {
    boolean existsByCourseCode(String courseCode);
}
```



rc > main > java > com > couse_enrollment_and_class_scheduling > ClassScheduleRepository.java > Language Support

```java
package com.couse_enrollment_and_class_scheduling;

import com.couse_enrollment_and_class_scheduling.ClassSchedule;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import java.time.DayOfWeek;
import java.time.LocalTime;
import java.util.List;

@Repository
public interface ClassScheduleRepository extends JpaRepository<ClassSchedule, Long> {

    /**
     * CRITICAL: Detects scheduling conflicts for a classroom
     * A conflict exists when:
     * 1. Same classroom
     * 2. Same day of week
     * 3. Time ranges overlap
     */
    @Query("""
        SELECT cs FROM ClassSchedule cs
        WHERE cs.classroom.id = :classroomId
        AND cs.dayOfWeek = :dayOfWeek
        AND cs.startTime < :endTime
        AND cs.endTime > :startTime
    """)
    List<ClassSchedule> findConflicts(
        @Param("classroomId") Long classroomId,
        @Param("dayOfWeek") DayOfWeek dayOfWeek,
        @Param("startTime") LocalTime startTime,
        @Param("endTime") LocalTime endTime
    );

    /**
     * Get all schedules for a specific student
     * Joins through enrollments to find student's courses
     */
    @Query("""
        SELECT cs FROM ClassSchedule cs
        JOIN Enrollment e ON e.course.id = cs.course.id
        WHERE e.student.id = :studentId
        ORDER BY cs.dayOfWeek, cs.startTime
    """)
    List<ClassSchedule> findStudentSchedule(@Param("studentId") Long studentId);

    /**
     * Find all schedules for a specific course
     */
    List<ClassSchedule> findByCourseId(Long courseId);
}
```

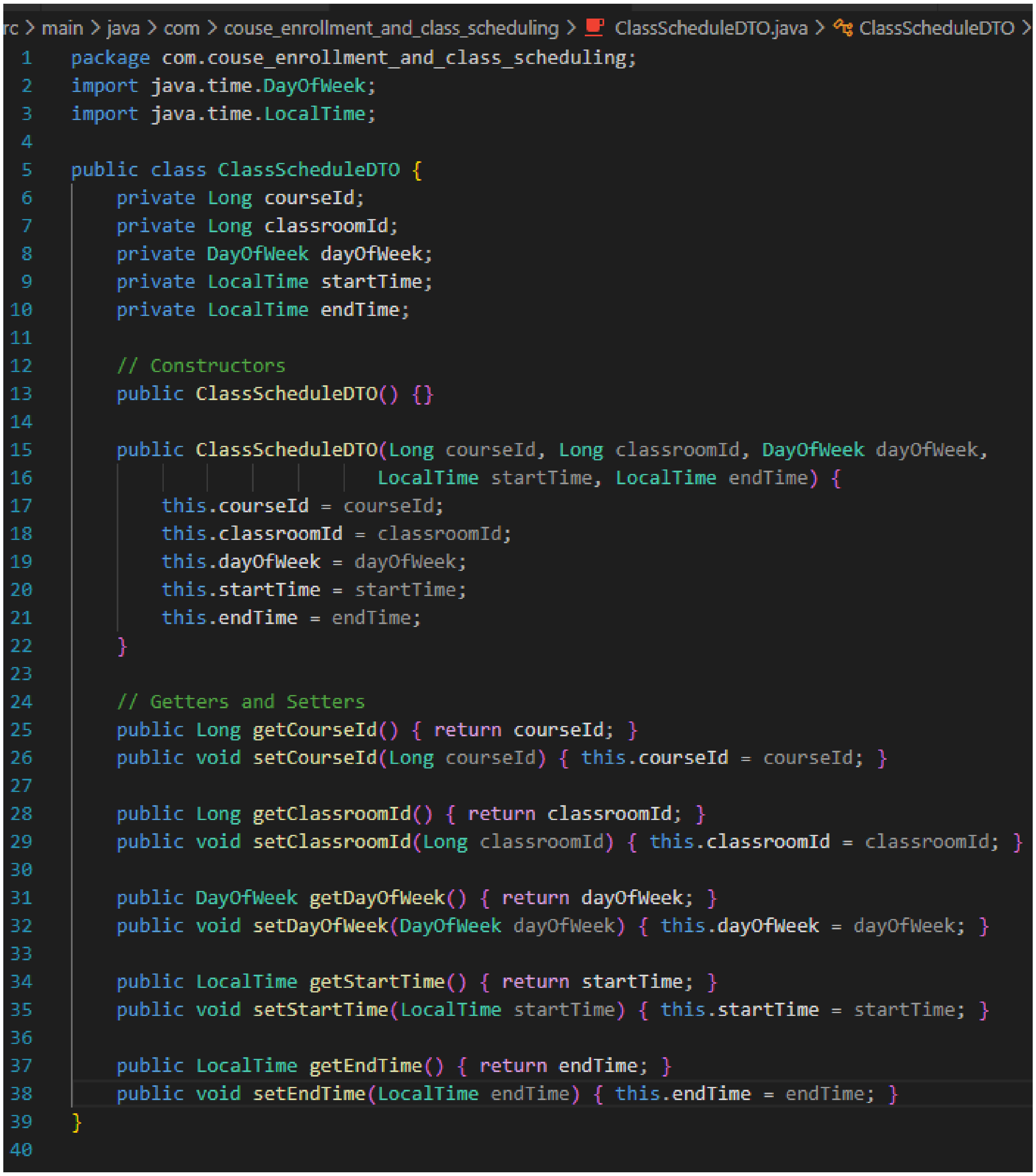


src > main > java > com > couse_enrollment_and_class_scheduling > EnrollmentRepository.java > ...

```java
package com.couse_enrollment_and_class_scheduling;
import com.couse_enrollment_and_class_scheduling.Enrollment;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public interface EnrollmentRepository extends JpaRepository<Enrollment, Long> {

    /**
     * Check if student is already enrolled in course
     */
    boolean existsByStudentIdAndCourseId(Long studentId, Long courseId);

    /**
     * Count how many students enrolled in a course
     */
    long countByCourseId(Long courseId);

    /**
     * Get all courses a student is enrolled in
     */
    @Query("""
        SELECT e FROM Enrollment e
        JOIN FETCH e.course
        WHERE e.student.id = :studentId
        ORDER BY e.enrollmentDate DESC
    """)
    List<Enrollment> findByStudentId(@Param("studentId") Long studentId);

    /**
     * Get all students enrolled in a course
     */
    @Query("""
        SELECT e FROM Enrollment e
        JOIN FETCH e.student
        WHERE e.course.id = :courseId
        ORDER BY e.enrollmentDate
    """)
    List<Enrollment> findByCourseId(@Param("courseId") Long courseId);
}

```

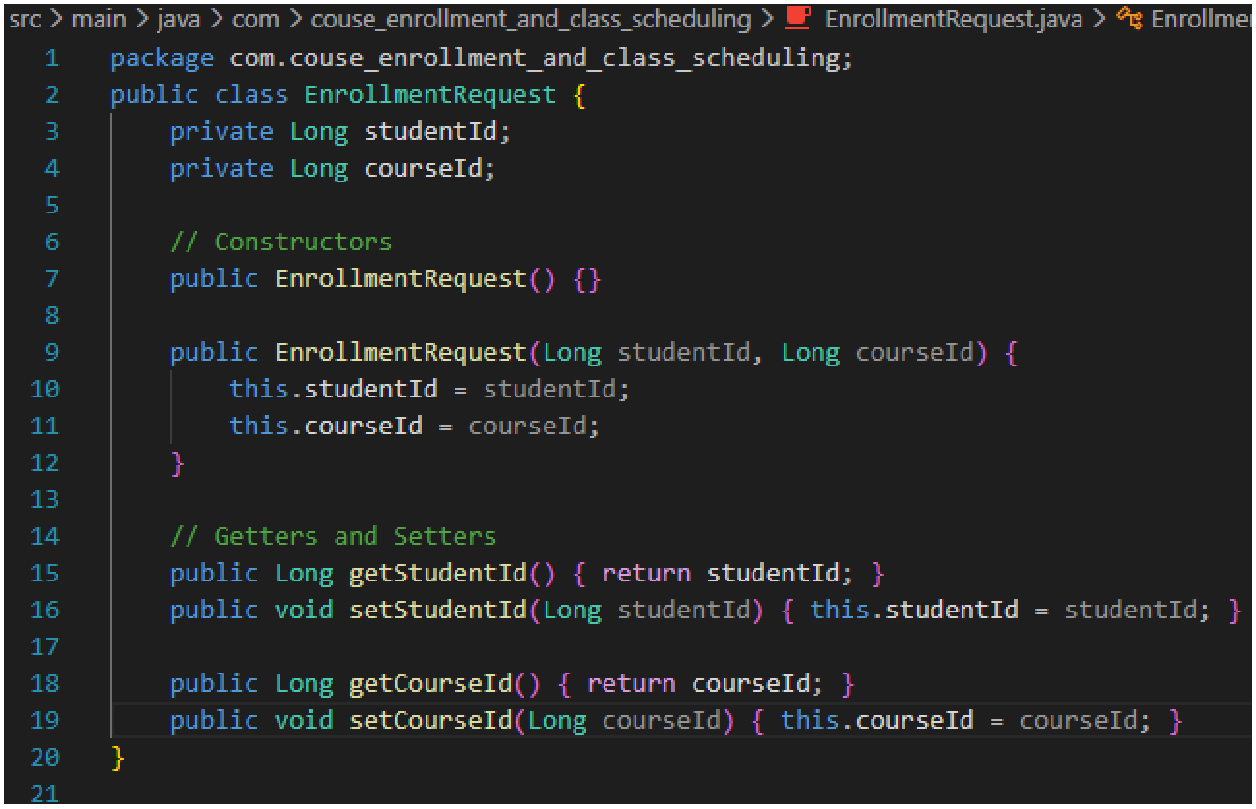
DTOs & Requests

- `ClassScheduleDTO`
- `EnrollmentRequest`



rc > main > java > com > couse_enrollment_and_class_scheduling > ClassScheduleDTO.java > ClassScheduleDTO >

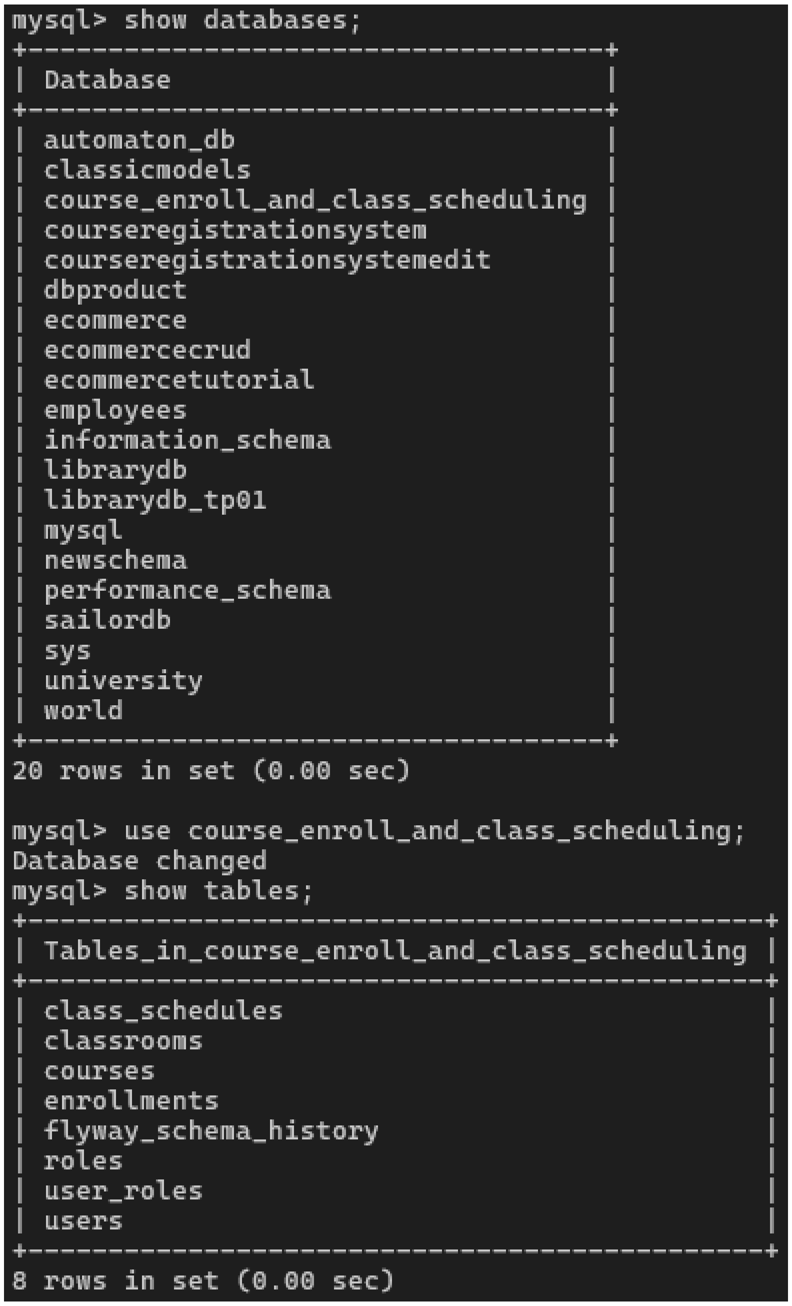
```java
package com.couse_enrollment_and_class_scheduling;
import java.time.DayOfWeek;
import java.time.LocalTime;

public class ClassScheduleDTO {
    private Long courseId;
    private Long classroomId;
    private DayOfWeek dayOfWeek;
    private LocalTime startTime;
    private LocalTime endTime;

    // Constructors
    public ClassScheduleDTO() {}

    public ClassScheduleDTO(Long courseId, Long classroomId, DayOfWeek dayOfWeek,
                          LocalTime startTime, LocalTime endTime) {
        this.courseId = courseId;
        this.classroomId = classroomId;
        this.dayOfWeek = dayOfWeek;
        this.startTime = startTime;
        this.endTime = endTime;
    }

    // Getters and Setters
    public Long getCourseId() { return courseId; }
    public void setCourseId(Long courseId) { this.courseId = courseId; }

    public Long getClassroomId() { return classroomId; }
    public void setClassroomId(Long classroomId) { this.classroomId = classroomId; }

    public DayOfWeek getDayOfWeek() { return dayOfWeek; }
    public void setDayOfWeek(DayOfWeek dayOfWeek) { this.dayOfWeek = dayOfWeek; }

    public LocalTime getStartTime() { return startTime; }
    public void setStartTime(LocalTime startTime) { this.startTime = startTime; }

    public LocalTime getEndTime() { return endTime; }
    public void setEndTime(LocalTime endTime) { this.endTime = endTime; }
}
```

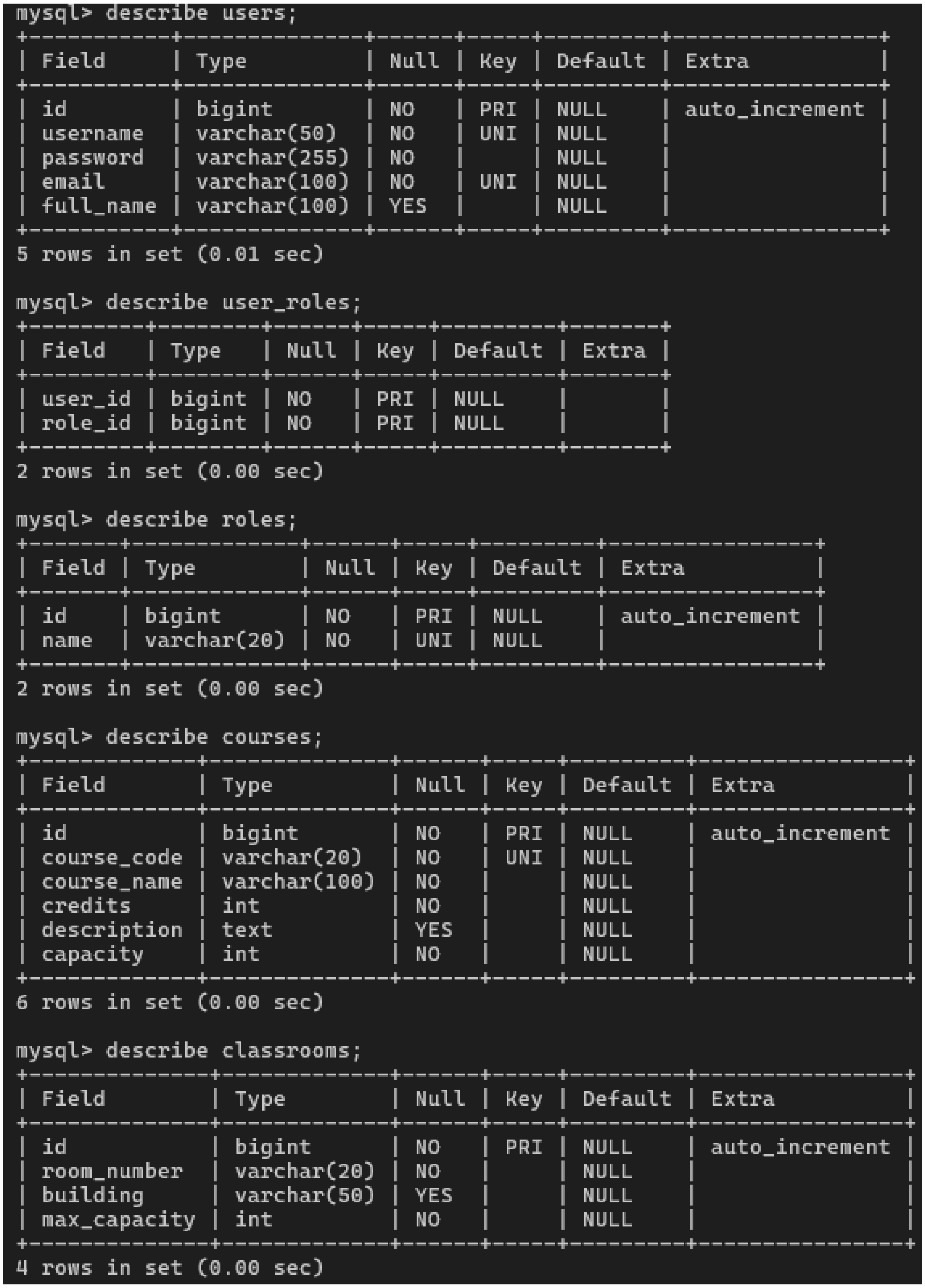

src > main > java > com > couse_enrollment_and_class_scheduling > EnrollmentRequest.java > Enrollme

```java
package com.couse_enrollment_and_class_scheduling;
public class EnrollmentRequest {
    private Long studentId;
    private Long courseId;

    // Constructors
    public EnrollmentRequest() {}

    public EnrollmentRequest(Long studentId, Long courseId) {
        this.studentId = studentId;
        this.courseId = courseId;
    }

    // Getters and Setters
    public Long getStudentId() { return studentId; }
    public void setStudentId(Long studentId) { this.studentId = studentId; }

    public Long getCourseId() { return courseId; }
    public void setCourseId(Long courseId) { this.courseId = courseId; }
}
```

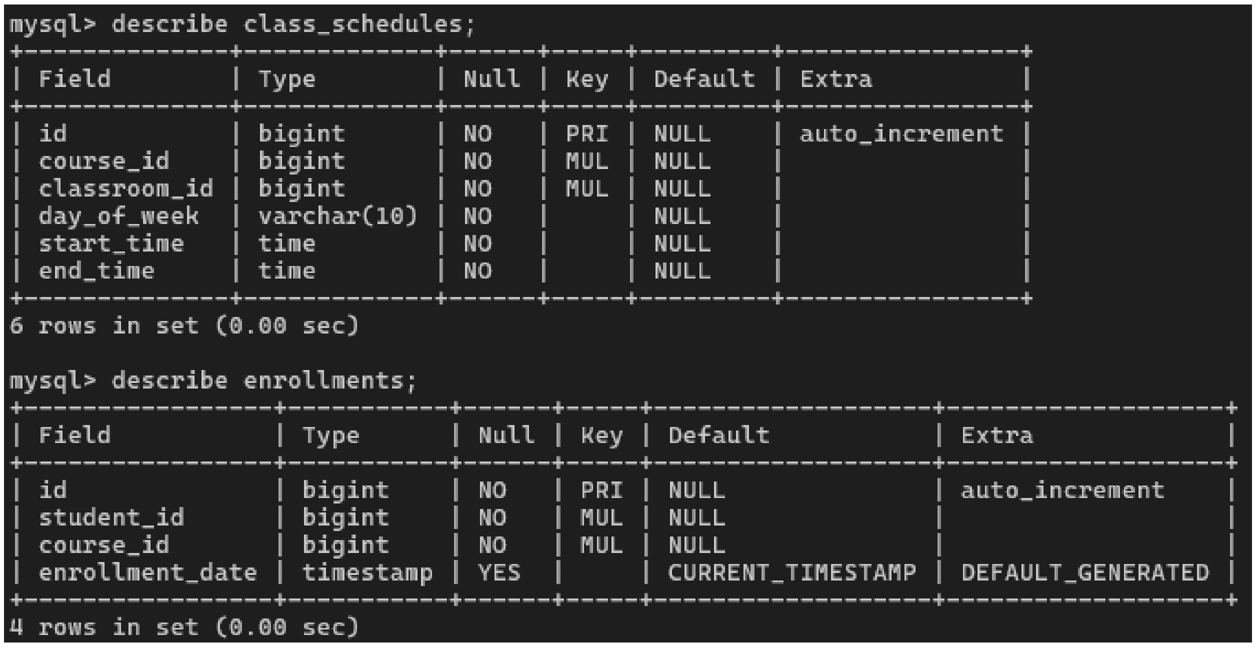
Database & Migration

- Integrated **Flyway**
- Created migration file:
    - `V1__Create_Tables.sql`
- Database successfully initializes and migrates on startup


```
mysql> show databases;
+-------------------------------------+
| Database                            |
+-------------------------------------+
| automaton_db                        |
| classicmodels                       |
| course_enroll_and_class_scheduling |
| courseregistrationsystem            |
| courseregistrationsystemedit        |
| dbproduct                           |
| ecommerce                           |
| ecommercecrud                       |
| ecommercetutorial                   |
| employees                           |
| information_schema                  |
| librarydb                           |
| librarydb_tp01                      |
| mysql                               |
| newschema                           |
| performance_schema                  |
| sailordb                            |
| sys                                 |
| university                          |
| world                               |
+-------------------------------------+
20 rows in set (0.00 sec)

mysql> use course_enroll_and_class_scheduling;
Database changed
mysql> show tables;
+----------------------------------------------+
| Tables_in_course_enroll_and_class_scheduling |
+----------------------------------------------+
| class_schedules                              |
| classrooms                                   |
| courses                                      |
| enrollments                                  |
| flyway_schema_history                        |
| roles                                        |
| user_roles                                   |
| users                                        |
+----------------------------------------------+
8 rows in set (0.00 sec)
```



```
mysql> describe users;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | bigint       | NO   | PRI | NULL    | auto_increment |
| username  | varchar(50)  | NO   | UNI | NULL    |                |
| password  | varchar(255) | NO   |     | NULL    |                |
| email     | varchar(100) | NO   | UNI | NULL    |                |
| full_name | varchar(100) | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)

mysql> describe user_roles;
+---------+--------+------+-----+---------+-------+
| Field   | Type   | Null | Key | Default | Extra |
+---------+--------+------+-----+---------+-------+
| user_id | bigint | NO   | PRI | NULL    |       |
| role_id | bigint | NO   | PRI | NULL    |       |
+---------+--------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> describe roles;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | bigint      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(20) | NO   | UNI | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> describe courses;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | bigint       | NO   | PRI | NULL    | auto_increment |
| course_code | varchar(20)  | NO   | UNI | NULL    |                |
| course_name | varchar(100) | NO   |     | NULL    |                |
| credits     | int          | NO   |     | NULL    |                |
| description | text         | YES  |     | NULL    |                |
| capacity    | int          | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql> describe classrooms;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id           | bigint      | NO   | PRI | NULL    | auto_increment |
| room_number  | varchar(20) | NO   |     | NULL    |                |
| building     | varchar(50) | YES  |     | NULL    |                |
| max_capacity | int         | NO   |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```


```
mysql> describe class_schedules;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id           | bigint      | NO   | PRI | NULL    | auto_increment |
| course_id    | bigint      | NO   | MUL | NULL    |                |
| classroom_id | bigint      | NO   | MUL | NULL    |                |
| day_of_week  | varchar(10) | NO   |     | NULL    |                |
| start_time   | time        | NO   |     | NULL    |                |
| end_time     | time        | NO   |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql> describe enrollments;
+-----------------+-----------+------+-----+-------------------+-------------------+
| Field           | Type      | Null | Key | Default           | Extra             |
+-----------------+-----------+------+-----+-------------------+-------------------+
| id              | bigint    | NO   | PRI | NULL              | auto_increment    |
| student_id      | bigint    | NO   | MUL | NULL              |                   |
| course_id       | bigint    | NO   | MUL | NULL              |                   |
| enrollment_date | timestamp | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------------+-----------+------+-----+-------------------+-------------------+
4 rows in set (0.00 sec)
```

4. Frontend/Thymeleaf Lead (Soeury Sreyno)

Goal: You make the application usable and ensure the UI adapts to the user.

- [x] ~~**Key Technologies:** Thymeleaf, HTML5, CSS (Bootstrap or Tailwind), JavaScript.~~
- [ ] **Specific Tasks:**
    - [x] ~~**Master Layout:** Create a `layout.html` fragment (Header, Footer, Sidebar) so every page looks consistent.~~
    - [ ] **Role-Based UI:** Use `sec:authorize` tags to hide buttons.
        - *Example:* Only show the "Delete Course" red button if the user is an `ADMIN`.
        - *Example:* Show "Enroll" button only to `STUDENT` users.
    - [ ] **Feedback:** Design alert boxes for success messages ("Enrolled successfully!") and error messages ("Course is full!").
    - [ ] **Mockups:** You often need to write the HTML *before* the CRUD leads finish their logic so they have a template to work with.
- [x] ~~**Deliverables:** All `.html` templates in the `src/main/resources/templates` folder.~~

5. Database Lead (Pang Lythong)

- [x] ~~**Goal:** You provide the foundation. If you change the database halfway through, everyone else breaks, so you must plan early.~~
- [x] ~~**Key Technologies:** MySQL/PostgreSQL, Flyway, JPA Relationships.~~
- [x] ~~**Specific Tasks:**~~
    - [x] ~~**ER Diagram:** Draw the map.~~
        - [x] ~~*One to Many:* One `Course` has many `ClassSchedule` entries.~~
        - [x] ~~*Many to Many:* `Students` and `Courses` are linked via the `Enrollment` table.~~
    - [x] ~~**Flyway Migrations:**~~
        - [x] ~~`V1__Create_Tables.sql`: The initial script to create all tables.~~
        - [x] ~~`V2__Insert_Dummy_Data.sql`: Add 5 dummy courses, 2 classrooms, and 3 users so the team has data to test with immediately.~~
    - [x] ~~**Optimization:** Ensure columns like `email` or `course_code` have `UNIQUE` constraints.~~
- [x] ~~**Deliverables:** The `db/migration` folder content and the ER Diagram image for the final report.~~