



# **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\_LLECTURER:** 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.

# 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 > cose_enrollment_and_class_scheduling > Userjava > ...
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; }
}

```

main > java > com > cose\_enrollment\_and\_class\_scheduling > Course.java > Course > setCapacity(Integer)

```
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
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; }
}
```

```

c > main > java > com > couse_enrollment_and_class_scheduling > ClassSchedule.java > ClassSchedule >
1  package com.couse_enrollment_and_class_scheduling;
2  import jakarta.persistence.*;
3  import java.time.DayOfWeek;
4  import java.time.LocalDateTime;
5
6  @Entity
7  @Table(name = "class_schedules")
8  public class ClassSchedule {
9      @Id
10     @GeneratedValue(strategy = GenerationType.IDENTITY)
11     private Long id;
12
13     @ManyToOne(fetch = FetchType.LAZY)
14     @JoinColumn(name = "course_id", nullable = false)
15     private Course course;
16
17     @ManyToOne(fetch = FetchType.LAZY)
18     @JoinColumn(name = "classroom_id", nullable = false)
19     private Classroom classroom;
20
21     @Enumerated(EnumType.STRING)
22     @Column(name = "day_of_week", nullable = false, length = 10)
23     private DayOfWeek dayOfWeek;
24
25     @Column(name = "start_time", nullable = false)
26     private LocalDateTime startTime;
27
28     @Column(name = "end_time", nullable = false)
29     private LocalDateTime endTime;
30
31     // Constructors
32     public ClassSchedule() {}
33
34     // Getters and Setters
35     public Long getId() { return id; }
36     public void setId(Long id) { this.id = id; }
37
38     public Course getCourse() { return course; }
39     public void setCourse(Course course) { this.course = course; }
40
41     public Classroom getClassroom() { return classroom; }
42     public void setClassroom(Classroom classroom) { this.classroom = classroom; }
43
44     public DayOfWeek getDayOfWeek() { return dayOfWeek; }
45     public void setDayOfWeek(DayOfWeek dayOfWeek) { this.dayOfWeek = dayOfWeek; }
46
47     public LocalDateTime getStartTime() { return startTime; }
48     public void setStartTime(LocalDateTime startTime) { this.startTime = startTime; }
49
50     public LocalDateTime getEndTime() { return endTime; }
51     public void setEndTime(LocalDateTime endTime) { this.endTime = endTime; }
52 }
53

```

```
> main > java > com > couse_enrollment_and_class_scheduling > Enrollment.java > Enrollment
10 public class Enrollment {
16     public void setCourse(Course course) {
17         this.course = course;
18     }
19
20     public LocalDateTime getEnrollmentDate() {
21         return enrollmentDate;
22     }
23
24     public void setEnrollmentDate(LocalDateTime enrollmentDate) {
25         this.enrollmentDate = enrollmentDate;
26     }
27 }
```

---

## Relationships Implemented

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

```
@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 > ...
1  package com.couse_enrollment_and_class_scheduling;
2
3  import com.couse_enrollment_and_class_scheduling.User;
4  import org.springframework.data.jpa.repository.JpaRepository;
5  import org.springframework.stereotype.Repository;
6  import java.util.Optional;
7
8  @Repository
9  public interface UserRepository extends JpaRepository<User, Long> {
10     Optional<User> findByUsername(String username);
11     boolean existsByUsername(String username);
12     boolean existsByEmail(String email);
13 }
14
```

```
src > main > java > com > couse_enrollment_and_class_scheduling > RoleRepository.java > ...
1  package com.couse_enrollment_and_class_scheduling;
2  import com.couse_enrollment_and_class_scheduling.Role;
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import org.springframework.stereotype.Repository;
5  import java.util.Optional;
6
7  @Repository
8  public interface RoleRepository extends JpaRepository<Role, Long> {
9      Optional<Role> findByName(String name);
10 }
11
```

```
src > main > java > com > couse_enrollment_and_class_scheduling > CourseRepository.java > ...
1  package com.couse_enrollment_and_class_scheduling;
2
3  import com.couse_enrollment_and_class_scheduling.Course;
4  import org.springframework.data.jpa.repository.JpaRepository;
5  import org.springframework.stereotype.Repository;
6
7  @Repository
8  public interface CourseRepository extends JpaRepository<Course, Long> {
9      boolean existsByCourseCode(String courseCode);
10 }
11
```

src > main > java > com > couse\_enrollment\_and\_class\_scheduling > ClassScheduleRepository.java > Language Support

```
1  package com.couse_enrollment_and_class_scheduling;
2
3  import com.couse_enrollment_and_class_scheduling.ClassSchedule;
4  import org.springframework.data.jpa.repository.JpaRepository;
5  import org.springframework.data.jpa.repository.Query;
6  import org.springframework.data.repository.query.Param;
7  import org.springframework.stereotype.Repository;
8  import java.time.DayOfWeek;
9  import java.time.LocalDateTime;
10 import java.util.List;
11
12 @Repository
13 public interface ClassScheduleRepository extends JpaRepository<ClassSchedule, Long> {
14
15     /**
16      * CRITICAL: Detects scheduling conflicts for a classroom
17      * A conflict exists when:
18      * 1. Same classroom
19      * 2. Same day of week
20      * 3. Time ranges overlap
21      */
22     @Query("""
23         SELECT cs FROM ClassSchedule cs
24         WHERE cs.classroom.id = :classroomId
25         AND cs.dayOfWeek = :dayOfWeek
26         AND cs.startTime < :endTime
27         AND cs.endTime > :startTime
28     """)
29     List<ClassSchedule> findConflicts(
30         @Param("classroomId") Long classroomId,
31         @Param("dayOfWeek") DayOfWeek dayOfWeek,
32         @Param("startTime") LocalDateTime startTime,
33         @Param("endTime") LocalDateTime endTime
34     );
35
36     /**
37      * Get all schedules for a specific student
38      * Joins through enrollments to find student's courses
39      */
40     @Query("""
41         SELECT cs FROM ClassSchedule cs
42         JOIN Enrollment e ON e.course.id = cs.course.id
43         WHERE e.student.id = :studentId
44         ORDER BY cs.dayOfWeek, cs.startTime
45     """)
46     List<ClassSchedule> findStudentSchedule(@Param("studentId") Long studentId);
47
48     /**
49      * Find all schedules for a specific course
50      */
51     List<ClassSchedule> findByCourseId(Long courseId);
52 }
```

```
src > main > java > com > cose_enrollment_and_class_scheduling > EnrollmentRepository.java > ...
1  package com.couse_enrollment_and_class_scheduling;
2  import com.couse_enrollment_and_class_scheduling.Enrollment;
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import org.springframework.data.jpa.repository.Query;
5  import org.springframework.data.repository.query.Param;
6  import org.springframework.stereotype.Repository;
7  import java.util.List;
8
9  @Repository
10 public interface EnrollmentRepository extends JpaRepository<Enrollment, Long> {
11
12     /**
13      * Check if student is already enrolled in course
14      */
15     boolean existsByStudentIdAndCourseId(Long studentId, Long courseId);
16
17     /**
18      * Count how many students enrolled in a course
19      */
20     long countByCourseId(Long courseId);
21
22     /**
23      * Get all courses a student is enrolled in
24      */
25     @Query("""
26         SELECT e FROM Enrollment e
27         JOIN FETCH e.course
28         WHERE e.student.id = :studentId
29         ORDER BY e.enrollmentDate DESC
30     """)
31     List<Enrollment> findByStudentId(@Param("studentId") Long studentId);
32
33     /**
34      * Get all students enrolled in a course
35      */
36     @Query("""
37         SELECT e FROM Enrollment e
38         JOIN FETCH e.student
39         WHERE e.course.id = :courseId
40         ORDER BY e.enrollmentDate
41     """)
42     List<Enrollment> findByCourseId(@Param("courseId") Long courseId);
43 }
44
```

## DTOs & Requests

- ClassScheduleDTO
- EnrollmentRequest

```
rc > main > java > com > couse_enrollment_and_class_scheduling > ClassScheduleDTO.java > ClassScheduleDTO >
1  package com.couse_enrollment_and_class_scheduling;
2  import java.time.DayOfWeek;
3  import java.time.LocalTime;
4
5  public class ClassScheduleDTO {
6      private Long courseId;
7      private Long classroomId;
8      private DayOfWeek dayOfWeek;
9      private LocalTime startTime;
10     private LocalTime endTime;
11
12     // Constructors
13     public ClassScheduleDTO() {}
14
15     public ClassScheduleDTO(Long courseId, Long classroomId, DayOfWeek dayOfWeek,
16                             LocalTime startTime, LocalTime endTime) {
17         this.courseId = courseId;
18         this.classroomId = classroomId;
19         this.dayOfWeek = dayOfWeek;
20         this.startTime = startTime;
21         this.endTime = endTime;
22     }
23
24     // Getters and Setters
25     public Long getCourseId() { return courseId; }
26     public void setCourseId(Long courseId) { this.courseId = courseId; }
27
28     public Long getClassroomId() { return classroomId; }
29     public void setClassroomId(Long classroomId) { this.classroomId = classroomId; }
30
31     public DayOfWeek getDayOfWeek() { return dayOfWeek; }
32     public void setDayOfWeek(DayOfWeek dayOfWeek) { this.dayOfWeek = dayOfWeek; }
33
34     public LocalTime getStartTime() { return startTime; }
35     public void setStartTime(LocalTime startTime) { this.startTime = startTime; }
36
37     public LocalTime getEndTime() { return endTime; }
38     public void setEndTime(LocalTime endTime) { this.endTime = endTime; }
39 }
40
```

```
src > main > java > com > couse_enrollment_and_class_scheduling > EnrollmentRequest.java > EnrollmentRequest.java
1  package com.couse_enrollment_and_class_scheduling;
2  public class EnrollmentRequest {
3      private Long studentId;
4      private Long courseId;
5
6      // Constructors
7      public EnrollmentRequest() {}
8
9      public EnrollmentRequest(Long studentId, Long courseId) {
10         this.studentId = studentId;
11         this.courseId = courseId;
12     }
13
14     // Getters and Setters
15     public Long getStudentId() { return studentId; }
16     public void setStudentId(Long studentId) { this.studentId = studentId; }
17
18     public Long getCourseId() { return courseId; }
19     public void setCourseId(Long courseId) { this.courseId = courseId; }
20 }
21
```

---

## 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.

- ☒ **Key Technologies:** ~~Thymeleaf, HTML5, CSS (Bootstrap or Tailwind), JavaScript.~~
- ☐ **Specific Tasks:**
  - ☒ **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.
- ☒ **Deliverables:** ~~All `.html` templates in the `src/main/resources/templates` folder.~~