INSTITUTION OF TECHNOLOGY OF CAMBODIA

DEPARTMENT: GIC I4-B

Report of Software Engineering

Project 3: Course Enrollment and Classroom Scheduling System

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

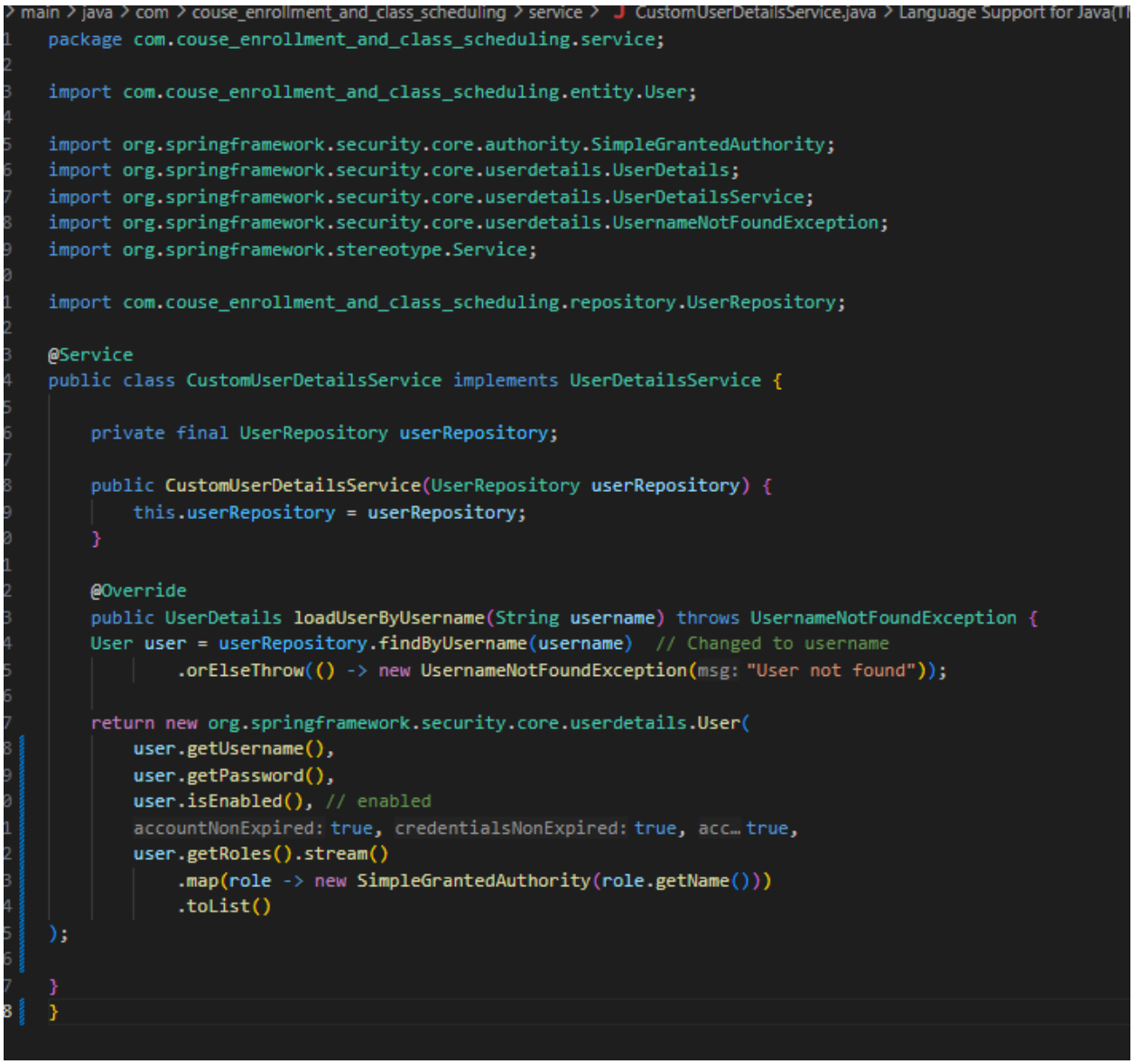
Soeury Sreyno

Vicheth Sokhsedtha

- **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

```java
@Configuration
@EnableMethodSecurity
public class SecurityConfig {

    private final CustomUserDetailsService userDetailsService;
    private final CustomLoginSuccessHandler loginSuccessHandler;

    public SecurityConfig(CustomUserDetailsService userDetailsService,
                          CustomLoginSuccessHandler loginSuccessHandler) {
        this.userDetailsService = userDetailsService;
        this.loginSuccessHandler = loginSuccessHandler;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .requestMatchers(...patterns: "/login", "/register", "/css/**").permitAll()
                .requestMatchers(...patterns: "/admin/**").hasRole(role: "ADMIN")
                .requestMatchers(...patterns: "/lecturer/**").hasRole(role: "LECTURER")
                .requestMatchers(...patterns: "/student/**").hasRole(role: "STUDENT")
                .anyRequest().authenticated()
            )
            .formLogin(login -> login
                .loginPage(loginPage: "/login")
                .successHandler(loginSuccessHandler)
                .permitAll()
            )
            .exceptionHandling(ex -> ex
                .accessDeniedPage(accessDeniedUrl: "/accessDenied")
            )
            .logout(logout -> logout
                .logoutSuccessUrl(logoutSuccessUrl: "/login?logout")
            );

        return http.build();
    }

    @Bean
    public AuthenticationManager authenticationManager(
            AuthenticationConfiguration config) throws Exception {
        return config.getAuthenticationManager();
    }
}
```

- **User Management:** Create the `UserDetailsService` implementation to load users from the database

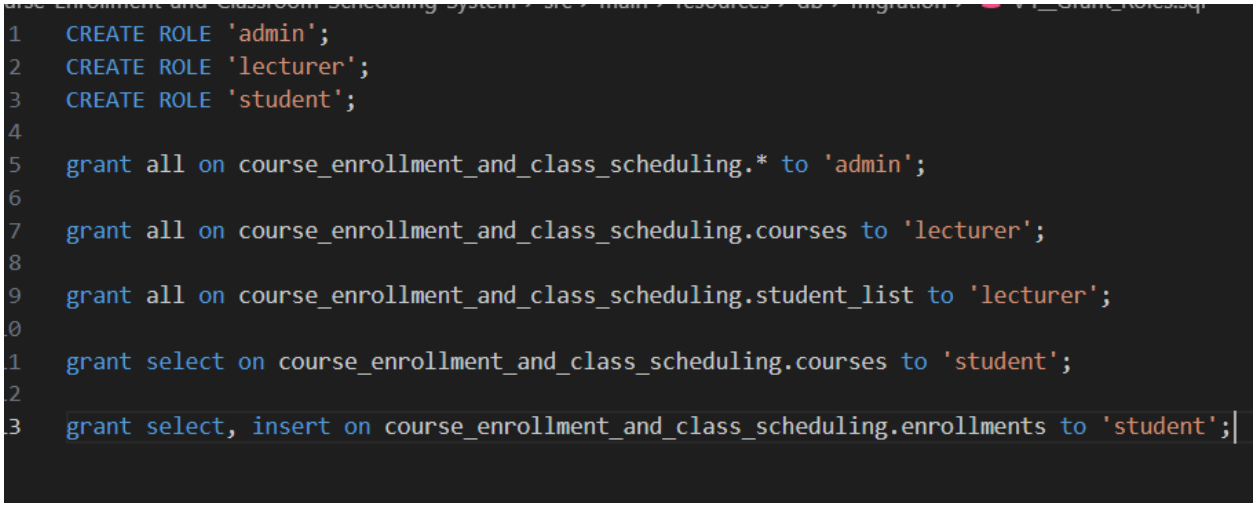


- **AuthController.java**

```java
@Controller
public class AuthController {

    private final UserRepository userRepository;
    private final RoleRepository roleRepository;
    private final PasswordEncoder passwordEncoder;

    public AuthController(UserRepository userRepository,
                          RoleRepository roleRepository,
                          PasswordEncoder passwordEncoder) {
        this.userRepository = userRepository;
        this.roleRepository = roleRepository;
        this.passwordEncoder = passwordEncoder;
    }

    @GetMapping("/login")
    public String login() {
        return "login";
    }

    @GetMapping("/register")
    public String registerForm(Model model) {
        model.addAttribute(attributeName: "user", new User());
        return "register";
    }

    @PostMapping("/register")
    public String register(@Valid @ModelAttribute("user") User user,
                           BindingResult bindingResult,
                           RedirectAttributes redirectAttributes) {

        // Validate input
        if (bindingResult.hasErrors()) {
            return "register";
        }

        // Check if username already exists
        if (userRepository.existsByUsername(user.getUsername())) {
            bindingResult.rejectValue(field: "username", errorCode: "error.user",
                    defaultMessage: "Username already exists");
            return "register";
        }

        // Check if email already exists
        if (userRepository.existsByEmail(user.getEmail())) {
            bindingResult.rejectValue(field: "email", errorCode: "error.user",
                    defaultMessage: "Email already registered");
            return "register";
        }
```

```
        // Encode password
        user.setPassword(passwordEncoder.encode(user.getPassword()));

        // Initialize roles collection
        if (user.getRoles() == null) {
            user.setRoles(new HashSet<>());
        }

        // Assign ROLE_STUDENT by default
        Optional<Role> studentRole = roleRepository.findByName(name: "ROLE_STUDENT");
        if (studentRole.isEmpty()) {
            redirectAttributes.addFlashAttribute(attributeName: "error",
                    attributeValue: "System error: Default role not found");
            return "redirect:/register";
        }

        user.getRoles().add(studentRole.get());

        // Set user as active
        user.setEnabled(enabled: true);

        // Save user
        try {
            userRepository.save(user);
            redirectAttributes.addFlashAttribute(attributeName: "success",
                    attributeValue: "Registration successful! Please login.");
            return "redirect:/login?registered";
        } catch (Exception e) {
            redirectAttributes.addFlashAttribute(attributeName: "error",
                    "Registration failed: " + e.getMessage());
            return "redirect:/register";
        }
    }

    @GetMapping("/accessDenied")
    public String accessDenied(Model model, Authentication auth) {
        if (auth != null) {
            String roles = auth.getAuthorities().stream()
                    .map(a -> a.getAuthority())
                    .reduce((a, b) -> a + ", " + b)
                    .orElse(other: "No roles");

            model.addAttribute(attributeName: "userRoles", roles);
            model.addAttribute(attributeName: "requiredRole", attributeValue: "ADMIN / LECTURER / STUDENT");
        }
        return "accessDenied";
    }

}
```

- **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



```sql
CREATE ROLE 'admin';
CREATE ROLE 'lecturer';
CREATE ROLE 'student';

grant all on course_enrollment_and_class_scheduling.* to 'admin';

grant all on course_enrollment_and_class_scheduling.courses to 'lecturer';

grant all on course_enrollment_and_class_scheduling.student_list to 'lecturer';

grant select on course_enrollment_and_class_scheduling.courses to 'student';

grant select, insert on course_enrollment_and_class_scheduling.enrollments to 'student';
```

Nget Darapich

- **Create ViewCourseScheduleController.java**
- **Create ClassScheduleService.java**
- **Insert datas for testing**
- **Create class-schedule.html and my-course.html in templates**

main > java > com > couse_enrollment_and_class_scheduling > controller > ViewCourseScheduleController.java > Language Support for Java(TM) by

```java
package com.couse_enrollment_and_class_scheduling.controller;

import com.couse_enrollment_and_class_scheduling.service.CourseService;
import com.couse_enrollment_and_class_scheduling.service.EnrollmentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@Controller
public class ViewCourseScheduleController {

    private final CourseService courseService;
    private final EnrollmentService enrollmentService;

    @Autowired
    public ViewCourseScheduleController(CourseService courseService, EnrollmentService enrollmentService) {
        this.courseService = courseService;
        this.enrollmentService = enrollmentService;
    }

    @GetMapping("/class-schedule")

    public String showClassSchedule(Model model) {
        model.addAttribute(attributeName: "courses", courseService.getAllCourses());
        return "class-schedule"; // Looks for templates/class-schedule.html
    }

    @GetMapping("/my-courses/{studentId}")
    public String showMyCourses(@PathVariable Long studentId, Model model) {
        model.addAttribute(attributeName: "myEnrollments", enrollmentService.getStudentSchedule(studentId));
        return "my-course"; // FIXED: Changed from "my-courses" to "my-course"
    }
}
```

main > java > com > couse_enrollment_and_class_scheduling > service > ClassScheduleService.java > ...

```java
public class ClassScheduleService {

    public ClassSchedule addSchedule(ClassSchedule schedule) {

        // Validation 1: Classroom double booking
        List<ClassSchedule> classroomConflicts = classScheduleRepository.findConflicts(
                schedule.getClassroom().getId(),
                schedule.getDayOfWeek(),
                schedule.getStartTime(),
                schedule.getEndTime());

        if (!classroomConflicts.isEmpty()) {
            throw new IllegalArgumentException(
                    s: "Classroom is already booked during this time");
        }

        // Validation 2: Course double scheduling
        List<ClassSchedule> courseConflicts = classScheduleRepository.findCourseConflicts(
                schedule.getCourse().getId(),
                schedule.getDayOfWeek(),
                schedule.getStartTime(),
                schedule.getEndTime());

        if (!courseConflicts.isEmpty()) {
            throw new IllegalArgumentException(
                    s: "Course already has a schedule during this time");
        }

        return classScheduleRepository.save(schedule);
    }

    // --------------------
    // Update existing schedule
    // --------------------
    @Transactional
```

main > resources > db > migration > V3__Insert_Courses_Classrooms_Schedules.sql

```sql
-- 1. Insert Lecturers first
INSERT INTO lecturers (id, full_name, department, office_hours)
VALUES
(1, 'Dr. Alice Smith', 'Computer Science', 'Mon 10:00-12:00'),
(2, 'Dr. Bob Johnson', 'Information Technology', 'Tue 14:00-16:00');

-- 2. Insert Classrooms
INSERT INTO classrooms (id, room_number, building, max_capacity)
VALUES
(1, 'Room 304', 'Building A', 40),
(2, 'Room 305', 'Building A', 30);

-- 3. Insert Courses (Linked to Lecturers and Classrooms)
-- Note: We include lecturer_id and classroom_id here so course.lecturer.fullName works!
INSERT INTO courses (id, course_code, course_name, credits, capacity, lecturer_id, classroom_id)
VALUES
(1, 'CS301', 'Database Systems', 3, 30, 1, 1),
(2, 'CS302', 'Operating Systems', 3, 2, 2, 2);

-- 4. Insert Schedules (For the calendar/time view)
INSERT INTO class_schedules (course_id, classroom_id, day_of_week, start_time, end_time)
VALUES
(1, 1, 'MONDAY', '09:00:00', '11:00:00'),
(2, 2, 'MONDAY', '11:00:00', '13:00:00');

-- 5. Insert a Test Student (so you can test enrollment)
INSERT INTO users (id, username, password, email, full_name, enabled)
VALUES
(1, 'student1', '$2a$10$dXJ3SW6G7P50lGekeFaboe3QwvGz7z0yENRKSLYSlZhsFhQH.vA7u', 'john@test.com', 'John Doe', 1),
(2, 'student2', '$2a$10$dXJ3SW6G7P50lGekeFaboe3QwvGz7z0yENRKSLYSlZhsFhQH.vA7u', 'jane@test.com', 'Jane Smith', 1);
```

main > resources > templates > class-schedule.html > html > body.container.mt-5

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Available Class Schedule</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body class="container mt-5">

    <div class="d-flex justify-content-between align-items-center mb-3">
        <h2>Available Class Schedule</h2>
        <a href="/my-courses/1" class="btn btn-outline-primary">My Enrolled Courses</a>
    </div>

    <div th:if="${courses == null || #lists.isEmpty(courses)}" class="alert alert-warning">
        No courses available at the moment.
    </div>

    <table th:unless="${courses == null || #lists.isEmpty(courses)}" class="table table-striped mt-3">
        <thead class="table-dark">
            <tr>
                <th>Course Name</th>
                <th>Lecturer</th>
                <th>Capacity</th>
                <th>Status</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="course : ${courses}">
                <td th:text="${course.courseName}">Course Name</td>
                <td th:text="${course.lecturer != null ? course.lecturer.fullName : 'TBA'}">Lecturer</td>
                <td th:text="${course.currentStudents != null ? course.currentStudents : 0} + '/' + ${cours
                <td>
                    <span th:if="${course.currentStudents != null && course.currentStudents >= course.capa
                    <span th:unless="${course.currentStudents != null && course.currentStudents >= course.
                </td>
                <td>
```

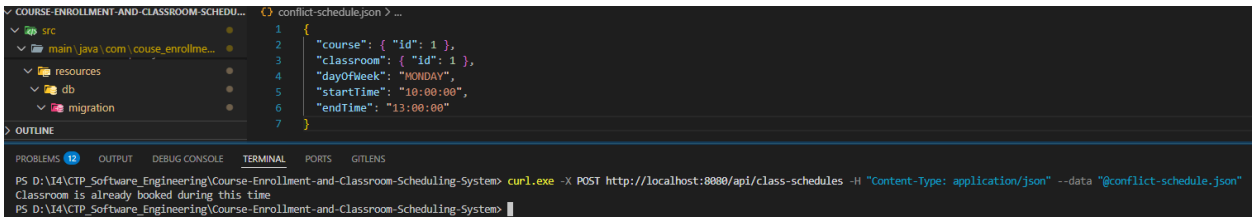
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Schedule</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body class="container mt-5">

    <div class="d-flex justify-content-between align-items-center mb-3">
        <h2>My Enrolled Courses</h2>
        <a href="/class-schedule" class="btn btn-outline-secondary">Back to Schedule</a>
    </div>

    <div th:if="${myEnrollments == null || #lists.isEmpty(myEnrollments)}" class="alert alert-info mt-4">
        You are not enrolled in any courses yet.
        <a href="/class-schedule" class="alert-link">Browse available courses</a>
    </div>

    <table th:unless="${myEnrollments == null || #lists.isEmpty(myEnrollments)}" class="table table-hover mt-3">
        <thead class="table-info">
            <tr>
                <th>Course Name</th>
                <th>Enrollment Date</th>
                <th>Lecturer</th>
                <th>Building/Room</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="enrollment : ${myEnrollments}">
                <td th:text="${enrollment.course.courseName}" class="fw-bold">Course</td>
                <td th:text="${enrollment.enrollmentDate != null ? #temporals.format(enrollment.enrollmentDate, 'dd-MM-yyyy HH:mm') : 'N/A'}">Date</td>
                <td th:text="${enrollment.course.lecturer != null ? enrollment.course.lecturer.fullName : 'TBA'}">Lecturer</td>
                <td>
                    <span th:if="${enrollment.course.classroom != null}"
                          th:text="${enrollment.course.classroom.building} + ' - ' + ${enrollment.course.classroom.roomNumber}">Building 1 - Room 101</span>
                    <span th:unless="${enrollment.course.classroom != null}">TBA</span>
                </td>
            </tr>
        </tbody>
    </table>

</body>
</html>
```

- **Validation: double booking**.

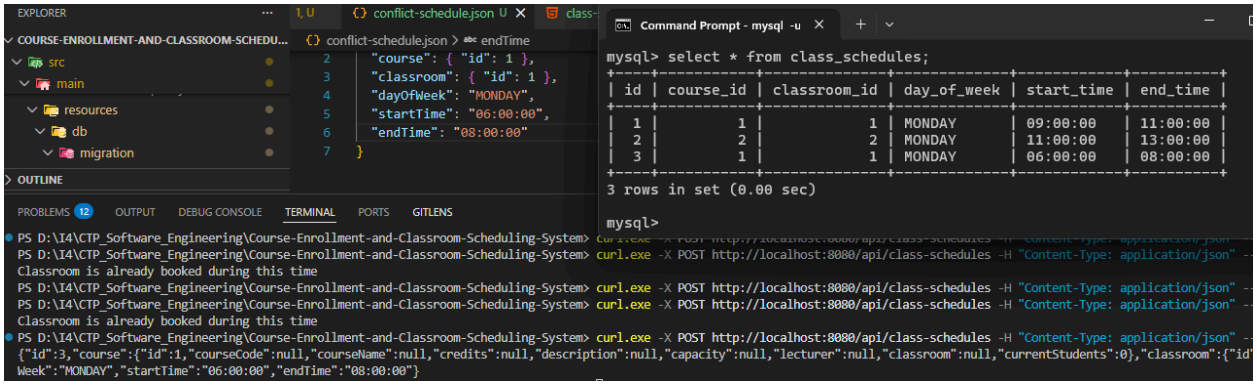
First :

```
mysql> select * from class_schedules;
+----+-----------+--------------+-------------+------------+----------+
| id | course_id | classroom_id | day_of_week | start_time | end_time |
+----+-----------+--------------+-------------+------------+----------+
|  1 |         1 |            1 | MONDAY      | 09:00:00   | 11:00:00 |
|  2 |         2 |            2 | MONDAY      | 11:00:00   | 13:00:00 |
+----+-----------+--------------+-------------+------------+----------+
2 rows in set (0.00 sec)

mysql>
```

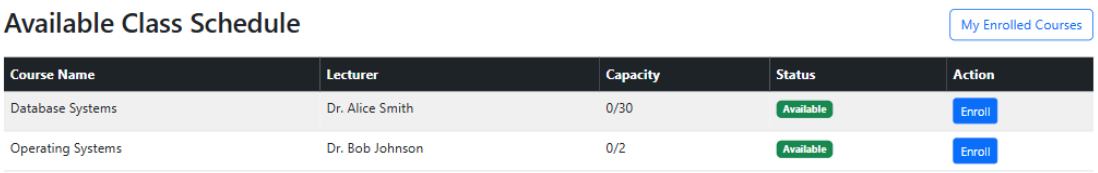
Try insert overlap schedule :



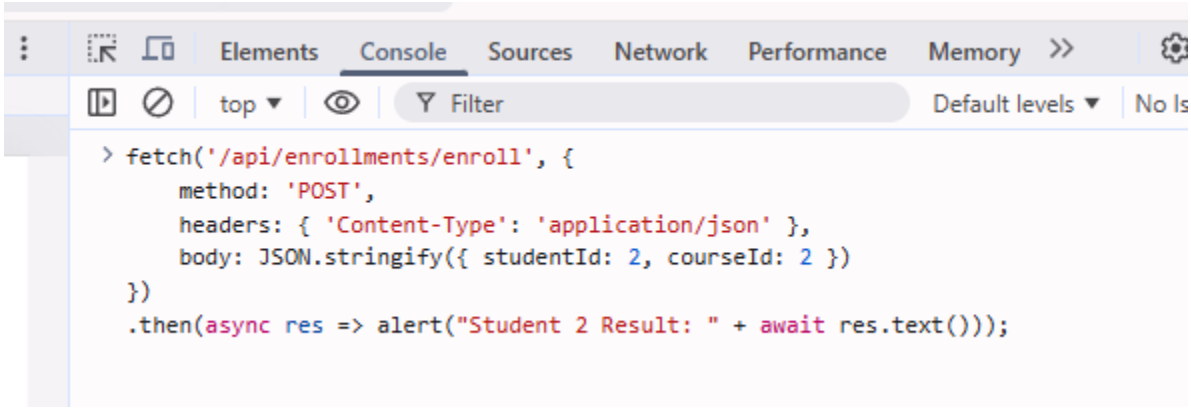
Try insert normally :



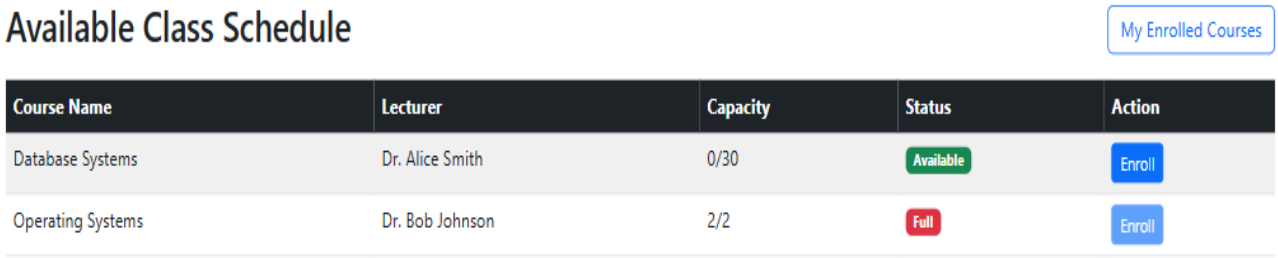
- /class-schedule



After click enroll (as student 1) and past this as student 2

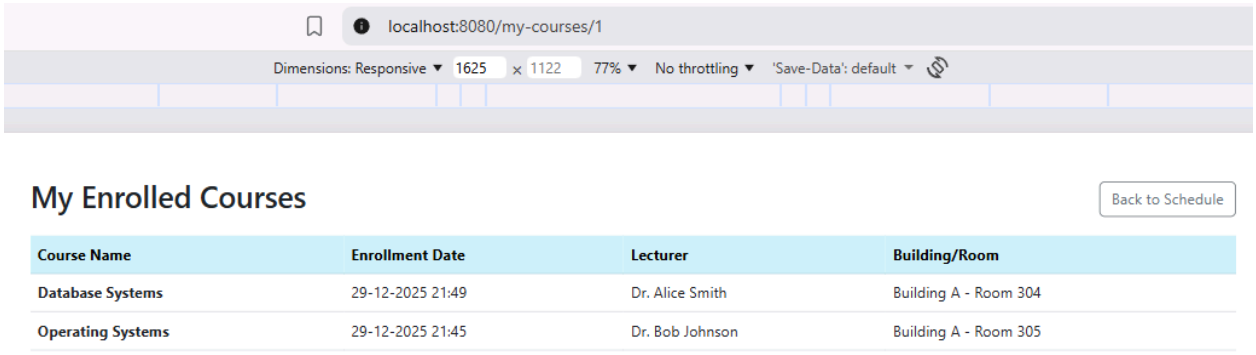


The result shown **Validation:** Check if the course is full (`currentStudents < capacity`). Check if the student is already enrolled.



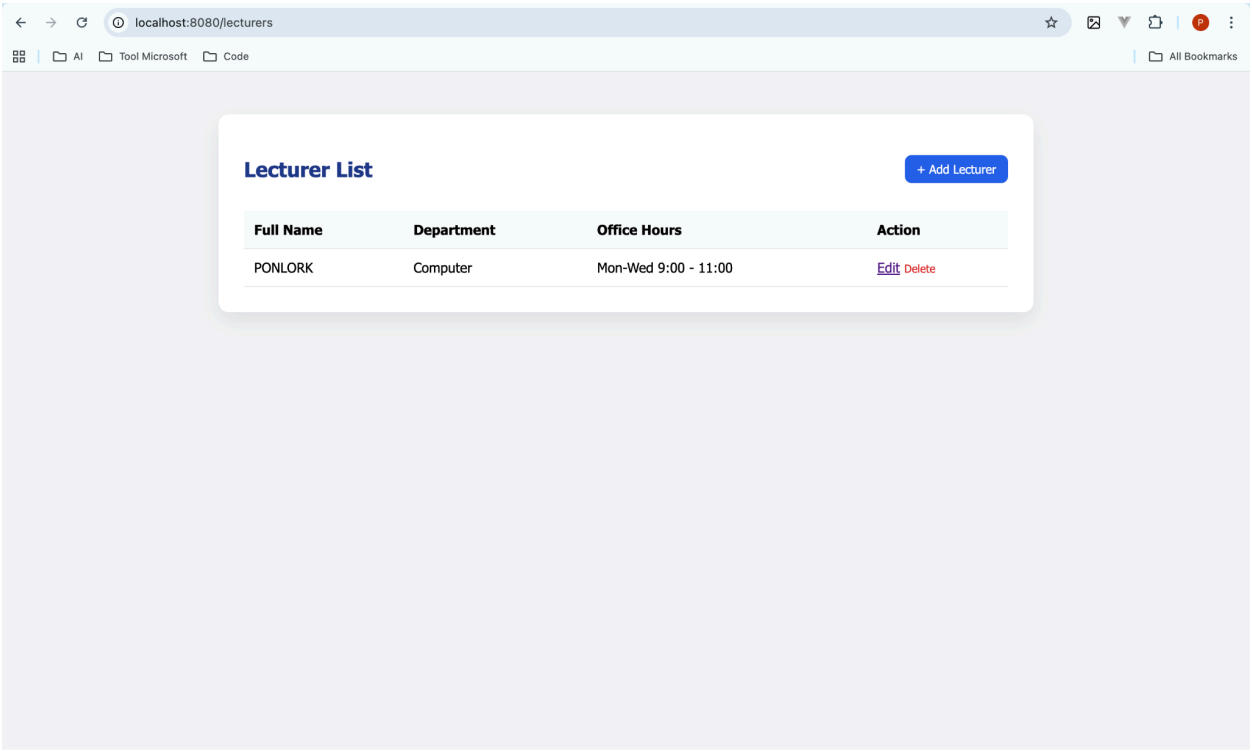
- **Views:** Logic for "My Schedule" (showing a student only *their* classes).

After I click enroll both course (as student 1) :

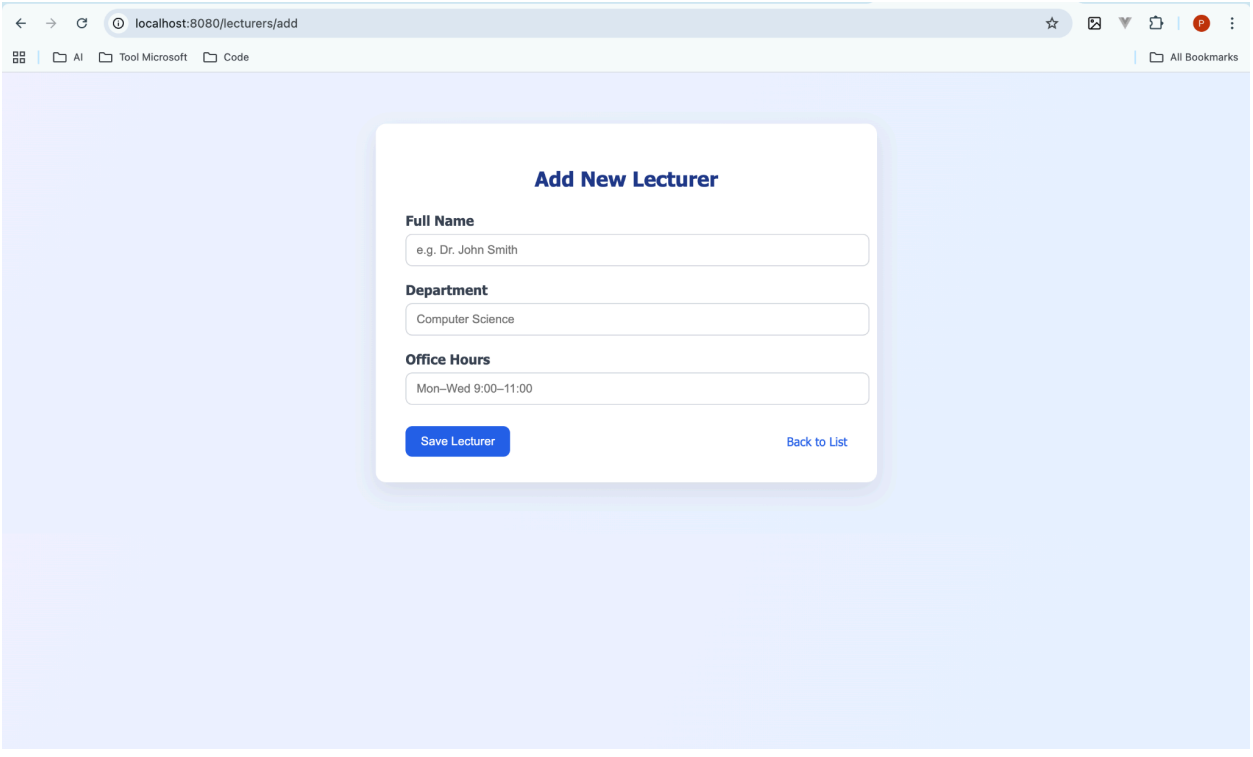


KEO CHANPONLORK

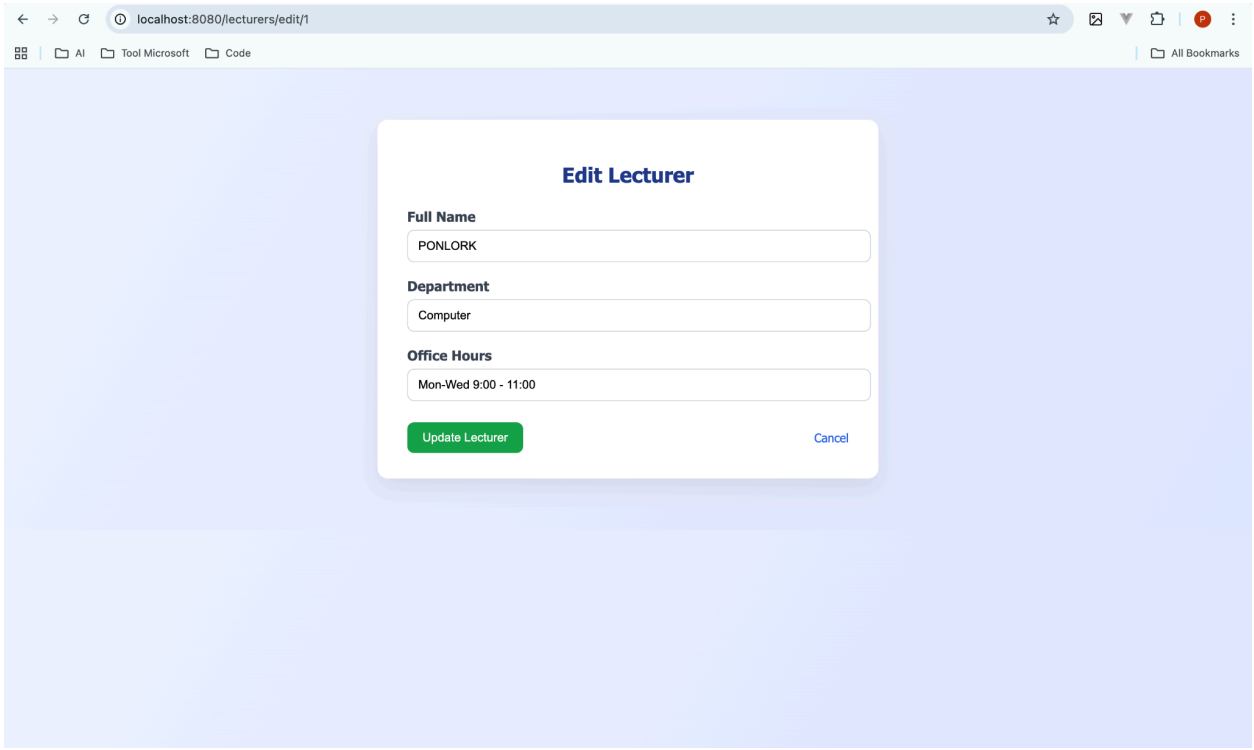
- **Lecturer**
  - Lecturer list



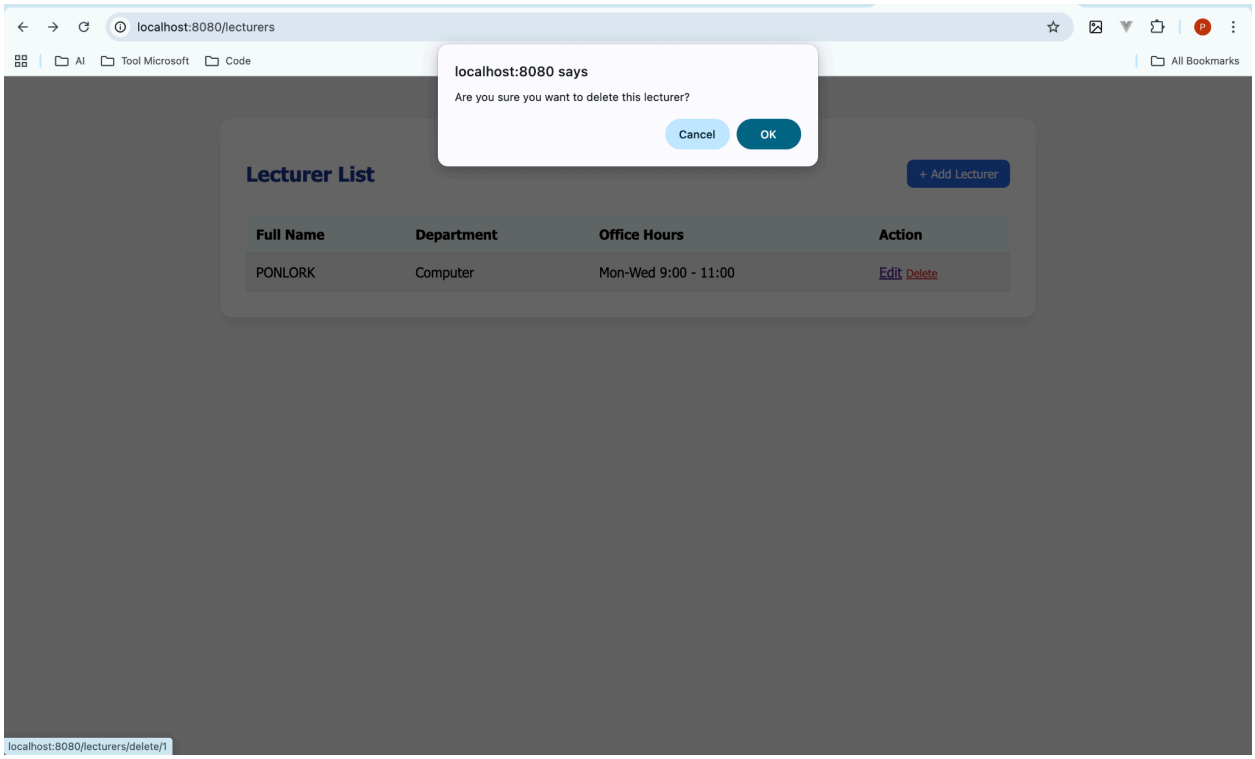
  - Add lecturer



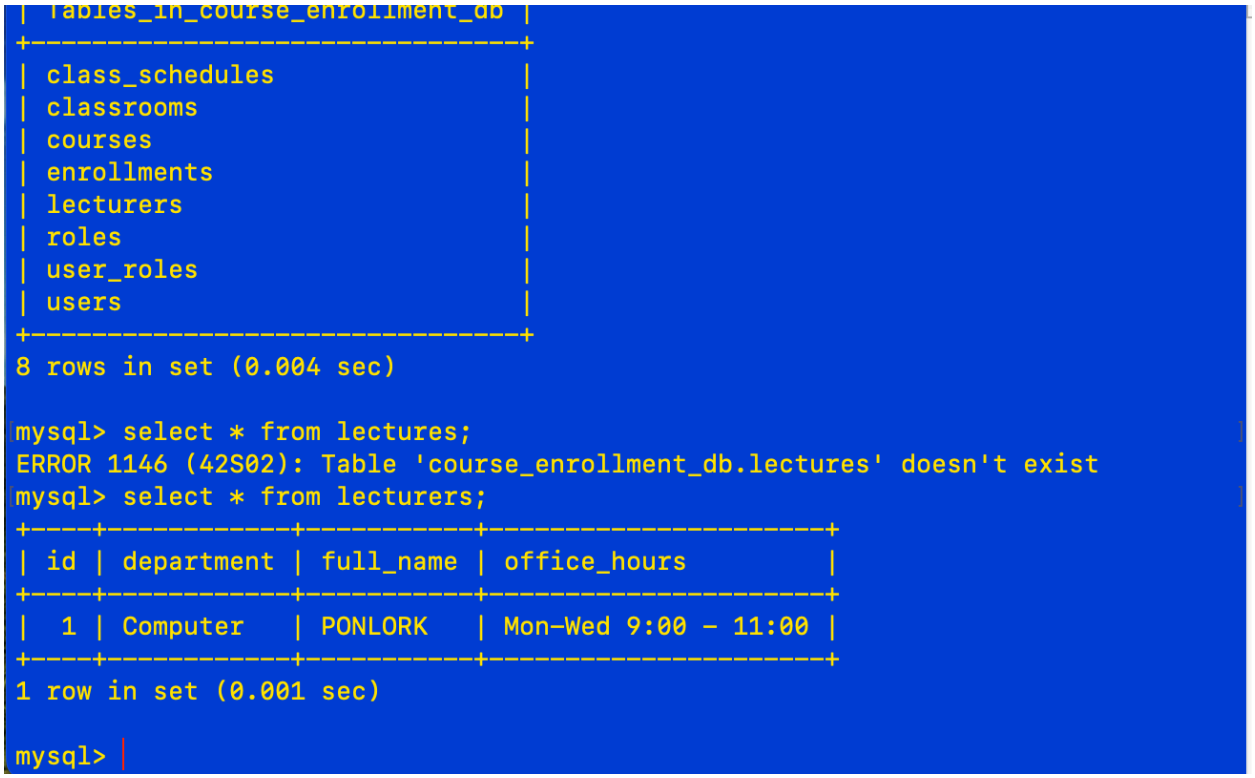
- Edit lecturer



- Detele lecturer



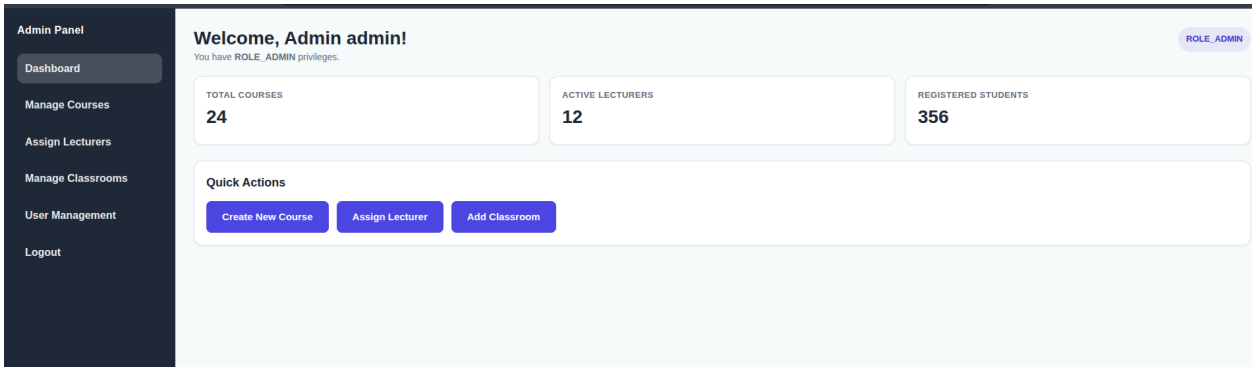
- Lecturer database

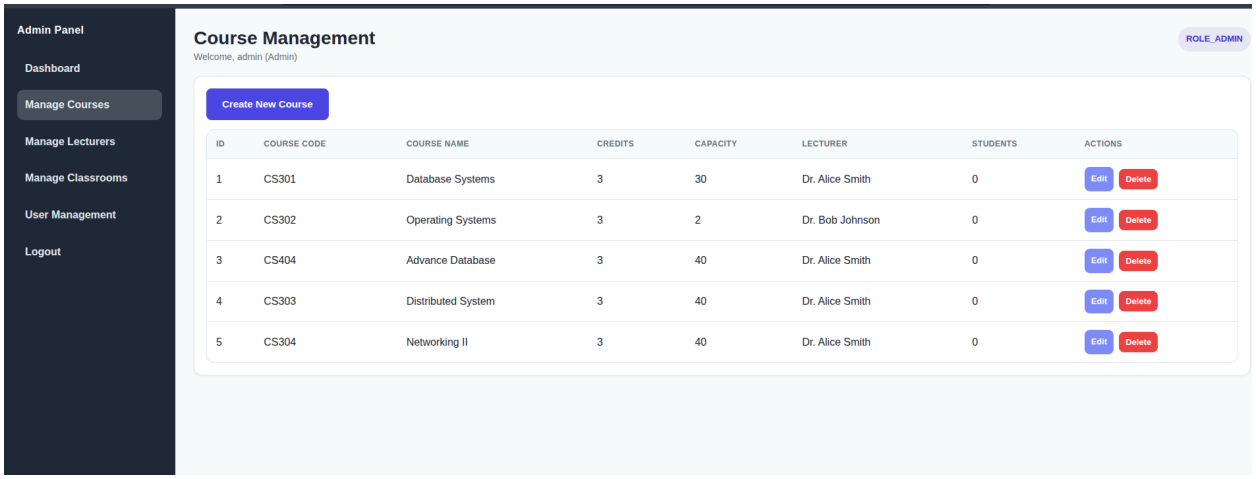


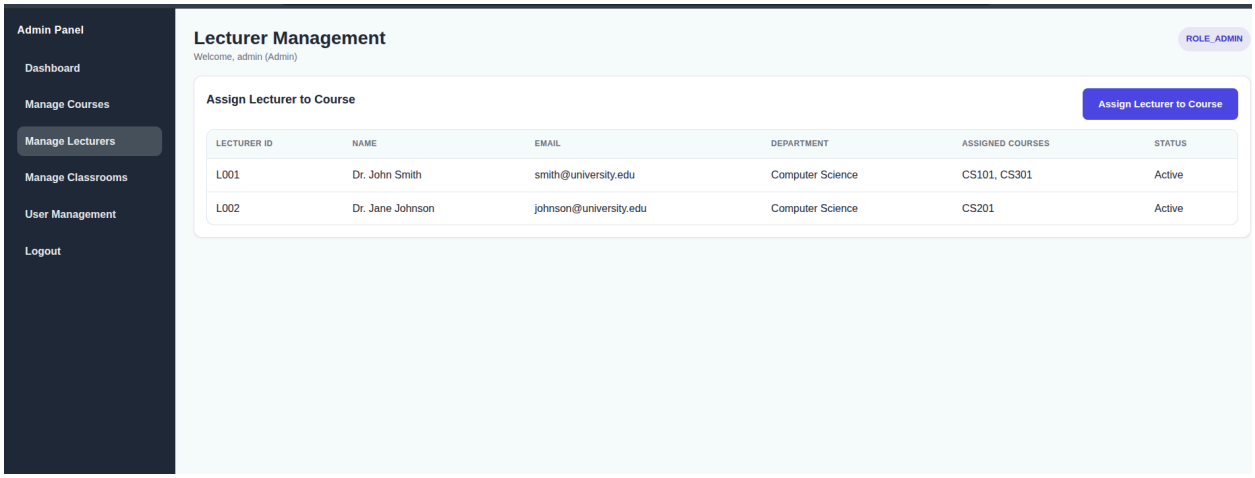
```
| Tables_in_course_enrollment_db |
+--------------------------------+
| class_schedules                |
| classrooms                     |
| courses                        |
| enrollments                    |
| lecturers                      |
| roles                          |
| user_roles                     |
| users                          |
+--------------------------------+
8 rows in set (0.004 sec)

mysql> select * from lectures;
ERROR 1146 (42S02): Table 'course_enrollment_db.lectures' doesn't exist
mysql> select * from lecturers;
+----+------------+-----------+----------------------+
| id | department | full_name | office_hours         |
+----+------------+-----------+----------------------+
|  1 | Computer   | PONLORK   | Mon-Wed 9:00 - 11:00 |
+----+------------+-----------+----------------------+
1 row in set (0.001 sec)

mysql>
```

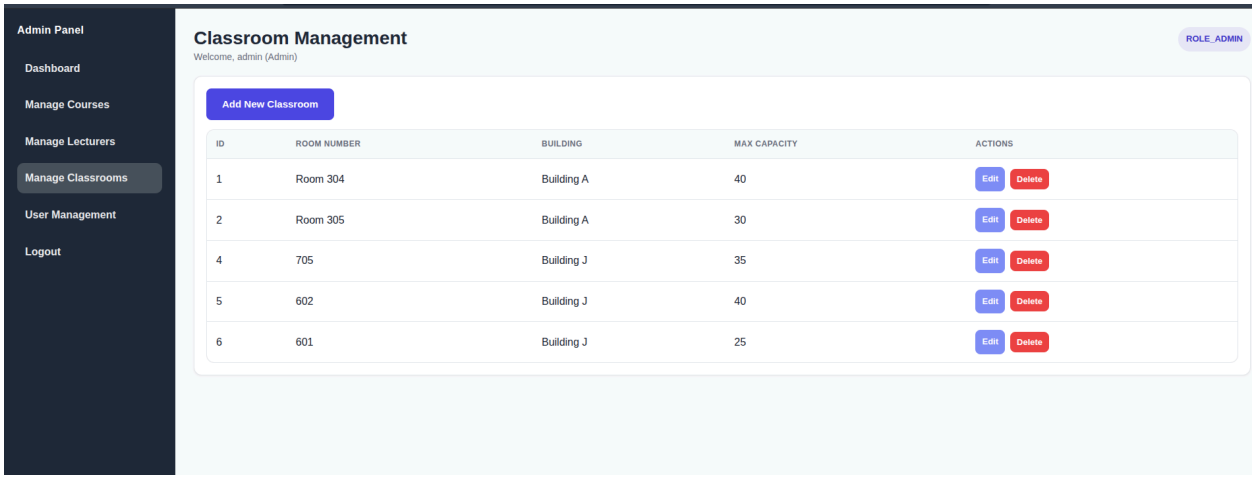
Pang Lythong

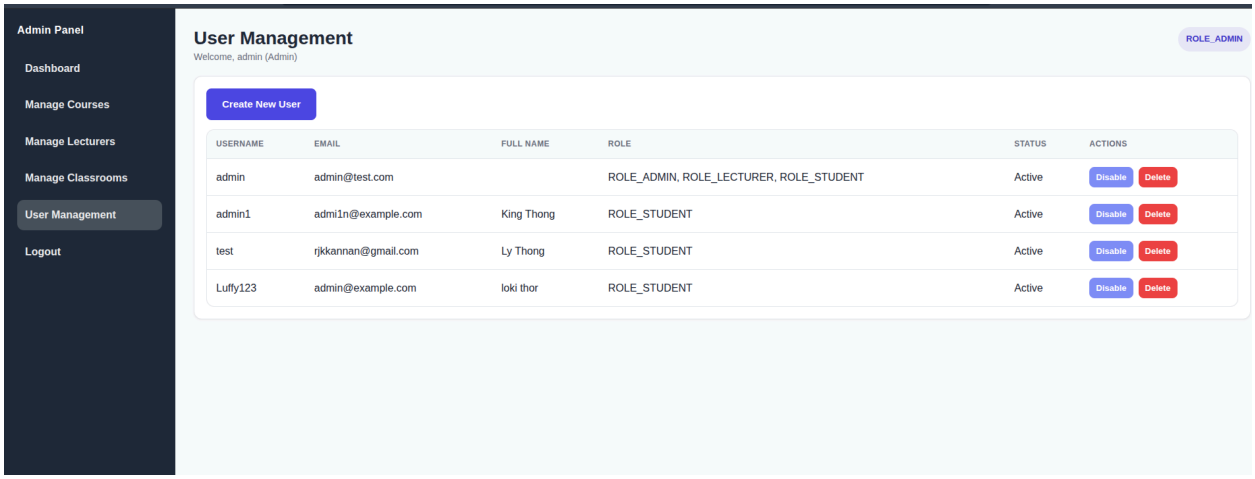
For Admin Page






Admin Panel
Dashboard
Manage Courses
Manage Lecturers
Manage Classrooms
User Management
Logout
Lecturer Management
Welcome, admin (Admin)
ROLE_ADMIN
Assign Lecturer to Course
Assign Lecturer to Course
LECTURER ID
NAME
EMAIL
DEPARTMENT
ASSIGNED COURSES
STATUS
L001
Dr. John Smith
smith@university.edu
Computer Science
CS101, CS301
Active
L002
Dr. Jane Johnson
johnson@university.edu
Computer Science
CS201
Active


Admin Panel
Dashboard
Manage Courses
Manage Lecturers
Manage Classrooms
User Management
Logout
Classroom Management
Welcome, admin (Admin)
ROLE_ADMIN
Add New Classroom
ID
ROOM NUMBER
BUILDING
MAX CAPACITY
ACTIONS
1
Room 304
Building A
40
Edit
Delete
2
Room 305
Building A
30
Edit
Delete
4
705
Building J
35
Edit
Delete
5
602
Building J
40
Edit
Delete
6
601
Building J
25
Edit
Delete


Admin Panel
Dashboard
Manage Courses
Manage Lecturers
Manage Classrooms
User Management
Logout
User Management
Welcome, admin (Admin)
ROLE_ADMIN
Create New User
USERNAME
EMAIL
FULL NAME
ROLE
STATUS
ACTIONS
admin
admin@test.com
ROLE_ADMIN, ROLE_LECTURER, ROLE_STUDENT
Active
Disable
Delete
admin1
admi1n@example.com
King Thong
ROLE_STUDENT
Active
Disable
Delete
test
rjkkannan@gmail.com
Ly Thong
ROLE_STUDENT
Active
Disable
Delete
Luffy123
admin@example.com
loki thor
ROLE_STUDENT
Active
Disable
Delete