



# 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

```
@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

```
> main > java > com > couse_enrollment_and_class_scheduling > service > CustomUserDetailsService.java > Language Support for Java(11)
1 package com.couse_enrollment_and_class_scheduling.service;
2
3 import com.couse_enrollment_and_class_scheduling.entity.User;
4
5 import org.springframework.security.core.authority.SimpleGrantedAuthority;
6 import org.springframework.security.core.userdetails.UserDetails;
7 import org.springframework.security.core.userdetails.UserDetailsService;
8 import org.springframework.security.core.userdetails.UsernameNotFoundException;
9 import org.springframework.stereotype.Service;
0
1 import com.couse_enrollment_and_class_scheduling.repository.UserRepository;
2
3 @Service
4 public class CustomUserDetailsService implements UserDetailsService {
5
6     private final UserRepository userRepository;
7
8     public CustomUserDetailsService(UserRepository userRepository) {
9         this.userRepository = userRepository;
0     }
1
2     @Override
3     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
4         User user = userRepository.findByUsername(username) // Changed to username
5             .orElseThrow(() -> new UsernameNotFoundException(msg: "User not found"));
6
7         return new org.springframework.security.core.userdetails.User(
8             user.getUsername(),
9             user.getPassword(),
0             user.isEnabled(), // enabled
1             accountNonExpired: true, credentialsNonExpired: true, acc... true,
2             user.getRoles().stream()
3                 .map(role -> new SimpleGrantedAuthority(role.getName()))
4                 .toList()
5         );
6
7     }
8 }
```

- **AuthController.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("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

```
Course Enrollment and Classroom Scheduling System > src > main > resources > db > migration > 1__Grant_Roles.sql
1 CREATE ROLE 'admin';
2 CREATE ROLE 'lecturer';
3 CREATE ROLE 'student';
4
5 grant all on course_enrollment_and_class_scheduling.* to 'admin';
6
7 grant all on course_enrollment_and_class_scheduling.courses to 'lecturer';
8
9 grant all on course_enrollment_and_class_scheduling.student_list to 'lecturer';
10
11 grant select on course_enrollment_and_class_scheduling.courses to 'student';
12
13 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
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 > ...
4  public class ClassScheduleService {
5      /**
6       * Adds a new class schedule to the database.
7       *
8       * @param schedule The class schedule to add.
9       */
10    public ClassSchedule addSchedule(ClassSchedule schedule) {
11
12        // Validation 1: Classroom double booking
13        List<ClassSchedule> classroomConflicts = classScheduleRepository.findConflicts(
14            schedule.getClassroom().getId(),
15            schedule.getDayOfWeek(),
16            schedule.getStartTime(),
17            schedule.getEndTime());
18
19        if (!classroomConflicts.isEmpty()) {
20            throw new IllegalArgumentException(
21                "Classroom is already booked during this time");
22        }
23
24        // Validation 2: Course double scheduling
25        List<ClassSchedule> courseConflicts = classScheduleRepository.findCourseConflicts(
26            schedule.getCourse().getId(),
27            schedule.getDayOfWeek(),
28            schedule.getStartTime(),
29            schedule.getEndTime());
30
31        if (!courseConflicts.isEmpty()) {
32            throw new IllegalArgumentException(
33                "Course already has a schedule during this time");
34        }
35
36        return classScheduleRepository.save(schedule);
37    }
38
39    // -----
40    // Update existing schedule
41    // -----
42    @Transactional

```

```

> main > resources > db > migration > V3_Insert_Courses_Classrooms_Schedules.sql
1  -- 1. Insert Lecturers first
2  INSERT INTO lecturers (id, full_name, department, office_hours)
3  VALUES
4  (1, 'Dr. Alice Smith', 'Computer Science', 'Mon 10:00-12:00'),
5  (2, 'Dr. Bob Johnson', 'Information Technology', 'Tue 14:00-16:00');
6
7  -- 2. Insert Classrooms
8  INSERT INTO classrooms (id, room_number, building, max_capacity)
9  VALUES
0  (1, 'Room 304', 'Building A', 40),
1  (2, 'Room 305', 'Building A', 30);
2
3  -- 3. Insert Courses (Linked to Lecturers and Classrooms)
4  -- Note: We include lecturer_id and classroom_id here so course.lecturer.fullName works!
5  INSERT INTO courses (id, course_code, course_name, credits, capacity, lecturer_id, classroom_id)
6  VALUES
7  (1, 'CS301', 'Database Systems', 3, 30, 1, 1),
8  (2, 'CS302', 'Operating Systems', 3, 2, 2, 2);
9
0  -- 4. Insert Schedules (For the calendar/time view)
1  INSERT INTO class_schedules (course_id, classroom_id, day_of_week, start_time, end_time)
2  VALUES
3  (1, 1, 'MONDAY', '09:00:00', '11:00:00'),
4  (2, 2, 'MONDAY', '11:00:00', '13:00:00');
5
6  -- 5. Insert a Test Student (so you can test enrollment)
7  INSERT INTO users (id, username, password, email, full_name, enabled)
8  VALUES
9  (1, 'student1', '$2a$10$dXJ3SW6G7P50lGekeFaboe3QwvGz7z0yENRKSLS1ZhsFhQH.vA7u', 'john@test.com', 'John Doe', 1),
0  (2, 'student2', '$2a$10$dXJ3SW6G7P50lGekeFaboe3QwvGz7z0yENRKSLS1ZhsFhQH.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/" 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} + '/' + ${course.capacity}">
                    <span th:if="${course.currentStudents != null && course.currentStudents >= course.capacity}">Full</span>
                    <span th:unless="${course.currentStudents != null && course.currentStudents >= course.capacity}">Available</span>
                </td>
                <td>
                    <a href="#">View Details</a>
                    <a href="#">Edit</a>
                    <a href="#">Delete</a>
                </td>
            </tr>
        </tbody>
    </table>
</body>
</html>

```

```

<!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="${enrollment.course.classroom != null}">
                        <span th:text="${enrollment.course.classroom.building} - ${enrollment.course.classroom.roomNumber}">Building 1 - Room 101</span>
                        <span th:unless="${enrollment.course.classroom != null}">TBA</span>
                    </span>
                </td>
            </tr>
        </tbody>
    </table>
</body>
</html>

```

- Validation: double booking.

First :

```
Database changed
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 :

The screenshot shows a code editor with a JSON file named `conflict-schedule.json`. The file contains:

```
1 {
  2   "course": { "id": 1 },
  3   "classroom": { "id": 1 },
  4   "dayOfWeek": "MONDAY",
  5   "startTime": "10:00:00",
  6   "endTime": "13:00:00"
}
```

Below the editor is a terminal window with the following output:

```
PS D:\I4\CTP_Software_Engineering\Course-Enrollment-and-Classroom-Scheduling-System> curl.exe -X POST http://localhost:8080/api/class-schedules -H "Content-Type: application/json" --data "@conflict-schedule.json"
Classroom is already booked during this time
PS D:\I4\CTP_Software_Engineering\Course-Enrollment-and-Classroom-Scheduling-System>
```

Try insert normally :

The screenshot shows a code editor with the same `conflict-schedule.json` file. Below it is a terminal window with the following MySQL query and results:

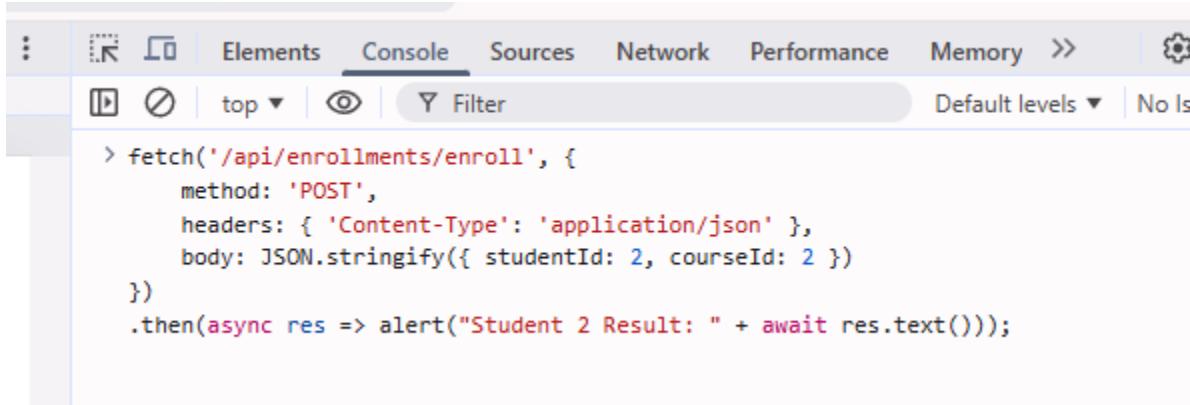
```
mysql> select * from class_schedules;
+----+-----+-----+-----+-----+-----+
| id | course_id | classroom_id | day_of_week | start_time | end_time |
+----+-----+-----+-----+-----+-----+
| 1 | 1 | 1 | MONDAY | 09:00:00 | 11:00:00 |
| 2 | 2 | 2 | MONDAY | 11:00:00 | 13:00:00 |
| 3 | 1 | 1 | MONDAY | 06:00:00 | 08:00:00 |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- /class-schedule

### Available Class Schedule

Course Name	Lecturer	Capacity	Status	Action
Database Systems	Dr. Alice Smith	0/30	Available	Enroll
Operating Systems	Dr. Bob Johnson	0/2	Available	Enroll

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



The screenshot shows the browser's developer tools with the 'Console' tab selected. It displays the following JavaScript code:

```
> fetch('/api/enrollments/enroll', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ studentId: 2, courseId: 2 })
})
.then(async res => alert("Student 2 Result: " + await res.text()));
```

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

## Available Class Schedule

[My Enrolled Courses](#)

Course Name	Lecturer	Capacity	Status	Action
Database Systems	Dr. Alice Smith	0/30	Available	<a href="#">Enroll</a>
Operating Systems	Dr. Bob Johnson	2/2	Full	<a href="#">Enroll</a>

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

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



## My Enrolled Courses

[Back to Schedule](#)

Course Name	Enrollment Date	Lecturer	Building/Room
Database Systems	29-12-2025 21:49	Dr. Alice Smith	Building A - Room 304
Operating Systems	29-12-2025 21:45	Dr. Bob Johnson	Building A - Room 305

# KEO CHANPONLORK

- **Lecturer**
  - Lecturer list

A screenshot of a web browser displaying a list of lecturers. The URL in the address bar is `localhost:8080/lecturers`. The page title is "Lecturer List". There is a button labeled "+ Add Lecturer" in the top right corner. A single row in the table represents the lecturer "PONLORK" from the "Computer" department, with office hours listed as "Mon-Wed 9:00 - 11:00". The "Action" column contains links for "Edit" and "Delete".

Full Name	Department	Office Hours	Action
PONLORK	Computer	Mon-Wed 9:00 - 11:00	<a href="#">Edit</a> <a href="#">Delete</a>

- Add lecturer

A screenshot of a web browser displaying a form to add a new lecturer. The URL in the address bar is `localhost:8080/lecturers/add`. The form has three input fields: "Full Name" (placeholder: "e.g. Dr. John Smith"), "Department" (placeholder: "Computer Science"), and "Office Hours" (placeholder: "Mon-Wed 9:00-11:00"). At the bottom of the form are two buttons: "Save Lecturer" and "Back to List".

**Add New Lecturer**

**Full Name**  
e.g. Dr. John Smith

**Department**  
Computer Science

**Office Hours**  
Mon-Wed 9:00-11:00

**Save Lecturer** **Back to List**

- Edit lecturer

localhost:8080/lecturers/edit/1

All Microsoft Code All Bookmarks

### Edit Lecturer

**Full Name**  
PONLORK

**Department**  
Computer

**Office Hours**  
Mon-Wed 9:00 - 11:00

**Update Lecturer** **Cancel**

- Delete lecturer

localhost:8080 says

Are you sure you want to delete this lecturer?

Cancel OK

Lecturer List

Full Name	Department	Office Hours	Action
PONLORK	Computer	Mon-Wed 9:00 - 11:00	<a href="#">Edit</a> <a href="#">Delete</a>

+ Add Lecturer

localhost:8080/lecturers/delete/1

- 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

**Welcome, Admin admin!**  
You have ROLE\_ADMIN privileges.

**ROLE\_ADMIN**

TOTAL COURSES <b>24</b>	ACTIVE LECTURERS <b>12</b>	REGISTERED STUDENTS <b>356</b>
----------------------------	-------------------------------	-----------------------------------

**Quick Actions**

[Create New Course](#) [Assign Lecturer](#) [Add Classroom](#)

**Course Management**  
Welcome, admin (Admin)

**ROLE\_ADMIN**

[Create New Course](#)

ID	COURSE CODE	COURSE NAME	CREDITS	CAPACITY	LECTURER	STUDENTS	ACTIONS
1	CS301	Database Systems	3	30	Dr. Alice Smith	0	<a href="#">Edit</a> <a href="#">Delete</a>
2	CS302	Operating Systems	3	2	Dr. Bob Johnson	0	<a href="#">Edit</a> <a href="#">Delete</a>
3	CS404	Advance Database	3	40	Dr. Alice Smith	0	<a href="#">Edit</a> <a href="#">Delete</a>
4	CS303	Distributed System	3	40	Dr. Alice Smith	0	<a href="#">Edit</a> <a href="#">Delete</a>
5	CS304	Networking II	3	40	Dr. Alice Smith	0	<a href="#">Edit</a> <a href="#">Delete</a>

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	johson@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	<button>Edit</button> <button>Delete</button>
2	Room 305	Building A	30	<button>Edit</button> <button>Delete</button>
4	705	Building J	35	<button>Edit</button> <button>Delete</button>
5	602	Building J	40	<button>Edit</button> <button>Delete</button>
6	601	Building J	25	<button>Edit</button> <button>Delete</button>

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	<button>Disable</button> <button>Delete</button>
admin1	admin1n@example.com	King Thong	ROLE_STUDENT	Active	<button>Disable</button> <button>Delete</button>
test	rjkkannan@gmail.com	Ly Thong	ROLE_STUDENT	Active	<button>Disable</button> <button>Delete</button>
Luffy123	admin@example.com	Ilok thi	ROLE_STUDENT	Active	<button>Disable</button> <button>Delete</button>