ព្រះរាជាណាចក្រកម្ពុជា

ជាតិ សាសនា ព្រះមហាក្សត្រ

REPORT OF Progress Week 3

Software Engineering

GROUP: I4-GIC-A (5)

Name of Students	ID of Students	Score
1.SOPHAT ODOM	e20221559	…….
2.SOPHEAP SOTHIPHAK	e20221038	…….
3.THY PHAROTH	e20220886	…….
4.RA SOCHEATEY	e20221446	…….
5. PHE RITHIKA	e20220245	…….

Lecturer: TAL Tongsreng (Course)

ROEUN Pacharoth (TP)

Academic year 2025-2026

Merge into Development Branch:
https://github.com/KingChocoLate/SE-Registration-and-Session-Management-System/tree/Development

Member	Goal	Work done
Sophat Odom	Build the Room system that the other members need for their logic.	• Create room entity • Link rooms to sessions • Room database access • Make the connection between member code • Merge codes
Sopheap Sothiphak	Connect the website buttons to the new logic and show Room information.	• Security Implementation • Create Admin and Conference Controller • Admin Dashboard UI • Implement Registration Logic • Controller Logic: Updated SessionController to retrieve available rooms • Update Session UI design
Thy Pharoth	Connect the real database login and organize the data files.	• Integrated AuthenticationManager to authenticate against the real database • Move Login req/res to dto folder • Added @PostMapping("/register") endpoint with full registration logic • Integrated passwordEncoder.encode() before saving passwords to database • Update the SecurityConfig.java to ensure /api/login or /register is public, but paths like /registrations/add or /admin/** require a logged-in user.
		RegistrationController.java - Only handles HTTP

Phe Rithika	Write the "brain" of the app to handle registration rules and room limits.	requests/responses ● Delegates all business logic to the service ● Catches exceptions and returns appropriate HTTP status codes RegistrationService.java - Business Logic Layer (The "Brain") ● Contains all registration rules and validation ● Room capacity checking ● Duplicate registration prevention ● Transaction management ● Database operations
Ra Socheatey	Stop the app from showing messy code errors and verify security.	● Implemented a Global Exception Handler to catch system errors and show friendly user messages. ● Added data validation to Session forms to prevent empty or invalid submissions. ●

Screenshots of the code we have done:

1. Sophat Odom
    - Create rooms entity

```java
package com.project5.rcrsms.Entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "rooms")
public class Room {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "room_id")
    private Long roomId;

    @Column(nullable = false)
    private String name;

    @Column(nullable = false)
    private Integer capacity;

    public Room() {
    }

    // Getters and Setters
    public Long getRoomId() {
        return roomId;
    }

    public void setRoomId(Long roomId) {
        this.roomId = roomId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getCapacity() {
        return capacity;
    }

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

- Link rooms to sessions

```java
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "room_id", nullable = false)
private Room room;
```

- Room database access

```java
package com.project5.rcrsms.Repository;

import java.time.LocalDateTime;
import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.project5.rcrsms.Entity.Room;

@Repository
public interface RoomRepository extends JpaRepository<Room, Long> {
    Optional<Room> findByName(String name);
    long countBySession_SessionId(Long sessionId);
    boolean existsByAttendee_UserIdAndSession_SessionId(Long userId, Long sessionId);
    boolean existsByRoom_RoomIdAndSessionTime(Long roomId, LocalDateTime sessionTime);
}
```

2. Sopheap Sothiphak
    - Security Implementation

```java
→ BCryptPasswordEncoder DefaultSecurityFilterChain | You, 6 minutes ago | 2 authors (You and one other)
@Configuration
@EnableMethodSecurity
public class SecurityConfig {

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

    → 1 bean | ← 1 bean
    @Bean       Thy Pharoth, 7 days ago • add userdetailservice securityconfig loginreque…
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .csrf(csrf -> csrf.disable()) // Keep disabled for now
            .authorizeHttpRequests(auth -> auth
                // Allow Home, Login, Register, and Static Resources (CSS/JS)
                .requestMatchers(...patterns:"/", "/index.html", "/login", "/register", "/css/**", "/js/**", "/images/**").permitAl
                // Everything else requires login
                .anyRequest().authenticated()
            )
            // Enable standard Form Login (like in your Test config)
            .formLogin(form -> form
                .loginPage(loginPage:"/login") // If you have a custom login page
                .defaultSuccessUrl(defaultSuccessUrl:"/", alwaysUse:true) // Go to home after login
                .permitAll()
            )
            .logout(logout -> logout
                .logoutSuccessUrl(logoutSuccessUrl:"/login?logout")
                .permitAll()
            );

        return http.build();
    }
}
```

- Create Admin and Conference Controller

```java
@Controller
@RequestMapping("/admin")
public class AdminController {

    @Autowired
    private SessionService sessionService;

    // Inject repositories directly for simple stats (or use Services if you have them)
    @Autowired
    private ConferenceRepository conferenceRepository;

    @Autowired
    private UserRepository userRepository;

    @GetMapping("/dashboard")
    public String dashboard(Model model) {
        // 1. Fetch Stats for the Cards
        model.addAttribute(attributeName:"totalSessions", sessionService.countSessions());
        model.addAttribute(attributeName:"totalConferences", conferenceRepository.count());
        model.addAttribute(attributeName:"totalUsers", userRepository.count());

        // 2. Fetch Recent Sessions (Just getting all for now, you can limit this later)
        model.addAttribute(attributeName:"recentSessions", sessionService.getAllSessions());

        return "admin/dashboard";
    }

    @GetMapping("/schedule")
    public String schedule(Model model) {
        // Fetch all sessions to display in the table
        model.addAttribute(attributeName:"sessions", sessionService.getAllSessions());
        return "admin/schedule";
    }
}
```

```java
@Controller
@RequestMapping("/conferences")
public class ConferenceController {

    @Autowired
    private ConferenceRepository conferenceRepo;

    // 1. List All Conferences
    @GetMapping("") // Matches /conferences
    public String listConferences(Model model) {
        model.addAttribute(attributeName:"conferences", conferenceRepo.findAll());
        return "conference/list"; // You'll need to create this file later if you want a public list
    }

    // 2. Show Create Form
    @GetMapping("/create")
    public String showCreateForm(Model model) {
        model.addAttribute(attributeName:"conference", new Conference());
        return "conference/create";
    }

    // 3. Save Conference
    @PostMapping("/save")
    public String saveConference(@ModelAttribute("conference") Conference conference) {
        conferenceRepo.save(conference);
        return "redirect:/admin/dashboard"; // Redirect to dashboard after saving
    }
}
```

- Admin Dashboard UI

```
src > main > resources > templates > admin > dashboard.html > html > head
You, 16 minutes ago | 1 author (You)
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layout/main_layout}">
<head>
    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Admin Dashboard</title>
</head>     You, 16 minutes ago • Uncommitted changes
<body>

<div layout:fragment="content">
    <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">
        <h1 class="h2">Admin Dashboard</h1>
        <div class="btn-toolbar mb-2 mb-md-0">
            <a th:href="@{/sessions/create}" class="btn btn-sm btn-outline-primary">
                <i class="bi ＋"></i> New Session
            </a>
        </div>
    </div>

    <div class="row g-4 mb-4">
        <div class="col-xl-3 col-md-6">
            <div class="card bg-primary text-white h-100 shadow-sm">
                <div class="card-body">
                    <div class="d-flex justify-content-between align-items-center">
                        <div>
                            <h6 class="text-uppercase mb-1">Total Sessions</h6>
                            <h2 class="mb-0 fw-bold" th:text="${totalSessions ?: 0}">0</h2>
                        </div>
                        <i class="bi ☑ fs-1 opacity-50"></i>
                    </div>
                </div>
            </div>
        </div>

        <div class="col-xl-3 col-md-6">
            <div class="card bg-success text-white h-100 shadow-sm">
                <div class="card-body">
```

- Session Controller Logic

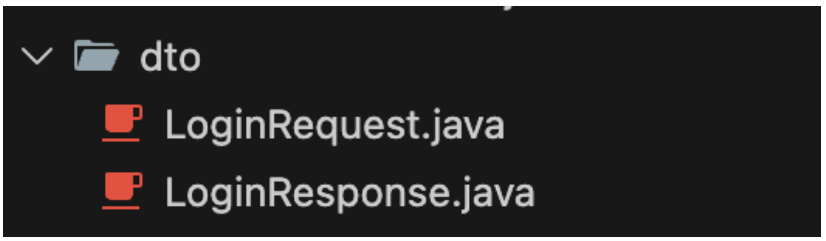
```
    // 2. NEW: Inject the RoomRepository
    @Autowired
    private RoomRepository roomRepo;
```

```
    // 2. Show Create Form/
    @GetMapping("/create")
    public String showCreateForm(Model model) {
        model.addAttribute(attributeName:"session", new Session());
        model.addAttribute(attributeName:"conferences", conferenceRepo.findAll());
        model.addAttribute(attributeName:"chairs", userRepo.findAll());

        // 3. NEW: Add Rooms to the model so the dropdown works
        model.addAttribute(attributeName:"rooms", roomRepo.findAll());

        return "session/create";
    }
```

3. Thy Pharoth

```java
@Controller
public class AuthenticationController {

    @Autowired
    private AuthenticationManager authenticationManager;

    @PostMapping("/api/login")
    public ResponseEntity<LoginResponse> loginAPI(
            @RequestBody LoginRequest loginRequest) {

        try {
            // Authenticate against the real database
            authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(
                    loginRequest.getUsername(),
                    loginRequest.getPassword()
                )
            );

            // Authentication successful
            LoginResponse response = new LoginResponse(
                    loginRequest.getUsername(),
                    password: "Authentication successful"
            );
            return ResponseEntity.ok(response);

        } catch (AuthenticationException e) {
            // Authentication failed
            return ResponseEntity.status(status: 401).body(
                new LoginResponse(username: null, password: "Invalid credentials")
            );
        }
    }
}
```

- Integrated AuthenticationManager to authenticate against the real database
- Updated loginAPI() to call authenticationManager.authenticate() with credentials from LoginRequest
- Added proper error handling for invalid credentials (401 status)



- Created new package: com.project5.rcrsms.dto
- Moved LoginRequest.java and LoginResponse.java to proper DTO package
- Updated import statements in AuthenticationController

```java
    @PostMapping("/register")
    public String registerUser(
            @RequestParam String username,
            @RequestParam String password,
            @RequestParam(defaultValue = "USER") String role,
            RedirectAttributes redirectAttributes) {

        try {
            // Check if username already exists
            if (userRepository.findByUsername(username).isPresent()) {
                redirectAttributes.addFlashAttribute(attributeName: "error", attributeValue: "Username already exists");
                return "redirect:/register";
            }

            // Create new user with encoded password
            UserEntity newUser = new UserEntity();
            newUser.setUsername(username);
            newUser.setPassword(passwordEncoder.encode(password));
            newUser.setRole(Role.valueOf(role.toUpperCase()));

            // Save to database
            userRepository.save(newUser);

            redirectAttributes.addFlashAttribute(attributeName: "success", attributeValue: "Registration successful! Please login.");
            return "redirect:/login";

        } catch (Exception e) {
            redirectAttributes.addFlashAttribute(attributeName: "error", "Registration failed: " + e.getMessage());
            return "redirect:/register";
        }
    }
```

- Added @PostMapping("/register") endpoint with full registration logic
- Integrated passwordEncoder.encode() before saving passwords to database
- Added username uniqueness validation
- Implemented proper error handling and redirect messages

```java
@Configuration
@EnableMethodSecurity
public class SecurityConfig {
    @Bean
    public PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }

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

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
        httpSecurity
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .requestMatchers(...patterns: "/css/**", "/js/**", "/api/login", "/register", "/login", "/er
                .requestMatchers(...patterns: "/admin/**").authenticated()
                .requestMatchers(...patterns: "/registrations/add").authenticated()
                .anyRequest().authenticated()
            )
            .formLogin(form -> form
                .loginPage(loginPage: "/login")
                .defaultSuccessUrl(defaultSuccessUrl: "/", alwaysUse: true)
                .permitAll()
            )
            .logout(logout -> logout
                .logoutSuccessUrl(logoutSuccessUrl: "/login?logout")
                .permitAll()
            );
        return httpSecurity.build();
    }
}
```

- Public routes: /css/**, /js/**, /api/login, /register, /login, /error
- Protected routes: /admin/** and /registrations/add require authentication
- Added form login configuration with login page and logout functionality Configured
- AuthenticationManager bean for dependency injection

4. Phe Rithika

Conference UI:

+ Form

```
<html xmlns:th="http://www.thymeleaf.org">
<body>
    <div class="container form-container">
        <div class="main-card">

            <!-- Form Body -->
            <div class="card-body p-4 p-md-5">
                <form th:action="@{${conference.conferenceId != null ? '/conferences/' + conference.conferenceI
                      method="post" th:object="${conference}">

                    <!-- Basic Information Section -->
                    <div class="form-section">
                        <h5 class="section-title">
                            <span class="section-icon">
                                <i class="bi bi-info-circle"></i>
                            </span>
                            Basic Information
                        </h5>

                        <div class="row">
                            <div class="col-12 mb-4">
                                <label for="name" class="form-label">
                                    Conference Name<span class="required-star">*</span>
                                </label>
                                <input type="text"
                                       class="form-control"
                                       id="name"
                                       name="name"
                                       th:field="*{name}"
                                       placeholder="e.g., International Tech Summit 2025"
                                       required
                                       maxlength="200">
                                <div class="form-text">Give your conference a clear, descriptive name</div>
                            </div>

                            <div class="col-12 mb-4">
                                <label for="description" class="form-label">
                                    Description<span class="required-star">*</span>
                                    <span class="char-counter" id="descCounter">0 / 1000</span>
                                </label>
                                <textarea class="form-control"
                                          id="description"
                                          name="description"
                                          th:field="*{description}"
                                          placeholder="Describe the conference theme, objectives, and target au
                                          required
                                          maxlength="1000"
                                          oninput="updateCharCount('description', 'descCounter', 1000)"></texta
```

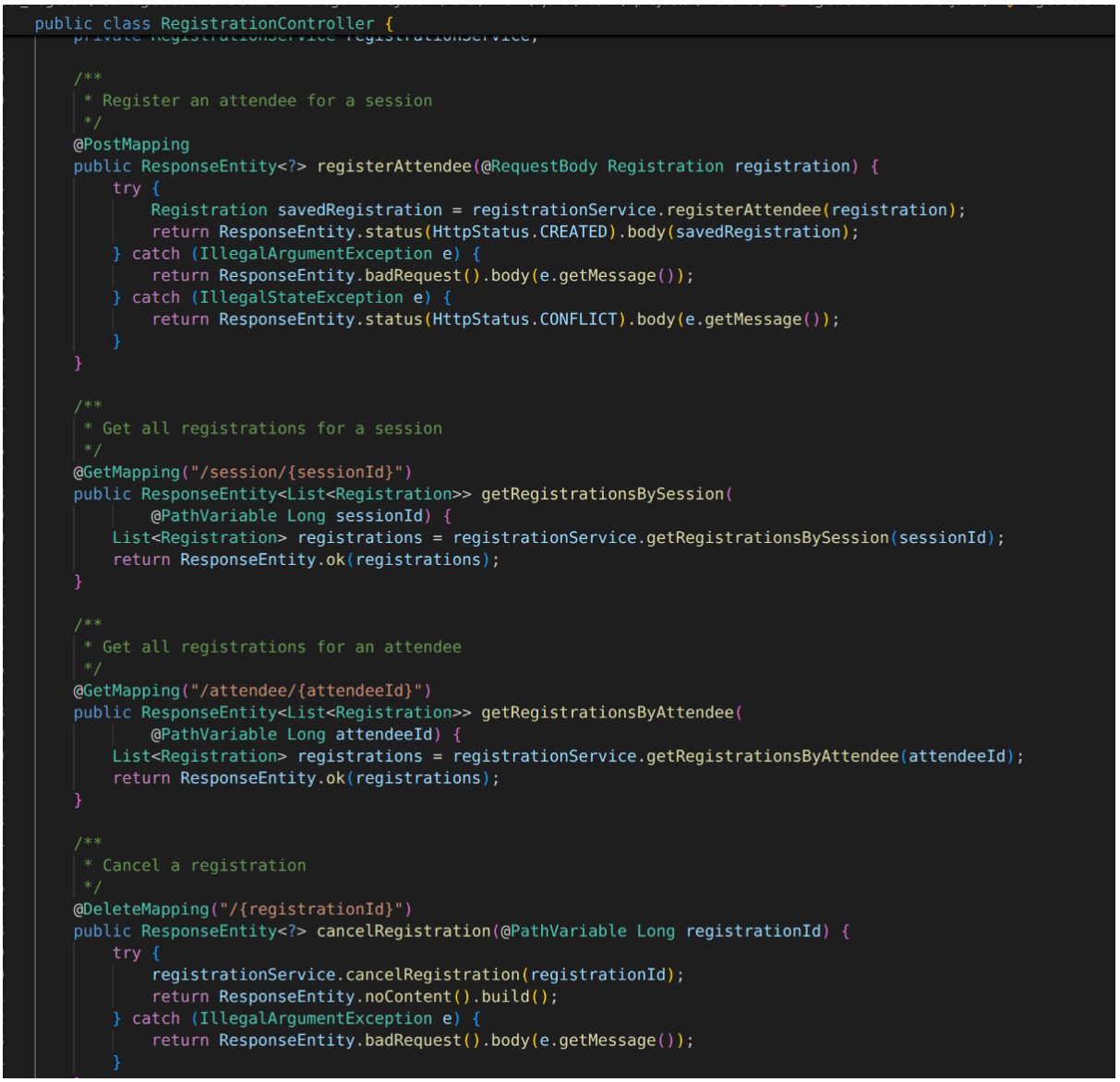
+ List

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <style>
        }
        .status-upcoming { background: #dbeafe; color: #1e40af; }
        .status-ongoing { background: #dcfce7; color: #166534; }
        .status-ended { background: #f3f4f6; color: #6b7280; }
    </style>
</head>
<body>
    <div class="container">
        <div class="main-card">
            <div class="gradient-header">
                <div class="row align-items-center">
                    <div class="col-md-6">
                        <h1 class="fw-bold mb-2">
                            <i class="bi bi-calendar-event me-2"></i>
                            Conferences
                        </h1>
                        <p class="mb-0 opacity-90">Explore upcoming and ongoing academic conferences</p>
                    </div>
                    <div class="col-md-6">
                        <div class="d-flex gap-2">
                            <input type="text" class="form-control search-box" placeholder="Search conferences.
                            <button class="btn btn-light px-4">
                                <i class="bi bi-search"></i>
                            </button>
                        </div>
                    </div>
                </div>
            </div>

            <div class="card-body p-4">
                <div class="row g-4">
                    <!-- Conference Card -->
                    <div class="col-md-6 col-lg-4" th:each="conference : ${conferences}">
                        <div class="conference-card">
                            <div class="conference-header">
                                <h5 class="fw-bold mb-2" th:text="${conference.name}">TechConf 2025</h5>
                                <div>
                                    <span class="badge status-upcoming" th:classappend="${conference.status}">
                                        <i class="bi bi-clock-history me-1"></i>
                                        Upcoming
                                    </span>
                                </div>
                            </div>
                            <div class="conference-body">
                                <div class="mb-3">
                                    <div class="info-badge">
```

+ View:

```java
public class MainController {
    public String listSessions(Model model) {
        mockSessions.add(Map.of(
            k1: "id", v1: 1,
            k2: "title", v2: "The Future of AI in Healthcare",
            k3: "status", v3: "APPROVED",
            k4: "submitter", Map.of(k1: "username", v1: "Dr. Alan Turing")
        ));

        mockSessions.add(Map.of(
            k1: "id", v1: 2,
            k2: "title", v2: "Sustainable Energy Grid Systems",
            k3: "status", v3: "PENDING",
            k4: "submitter", Map.of(k1: "username", v1: "Tesla_Fan_99")
        ));

        model.addAttribute(attributeName: "sessions", mockSessions);
        return "session/list";
    }

    @GetMapping("/sessions/edit/{id}")
    public String editSession() {
        return "index";
    }

    @GetMapping("/sessions/submit")
    public String showSubmitForm(Model model) {
        return "session/create";
    }
    @org.springframework.web.bind.annotation.PostMapping("/sessions/submit")
    public String handleSubmission(@org.springframework.web.bind.annotation.RequestParam String title,
                                   @org.springframework.web.bind.annotation.RequestParam String abstractText) {


        System.out.println(x: "New Session Submitted:");
        System.out.println("Title: " + title);
        System.out.println("Abstract: " + abstractText);


        return "redirect:/sessions";
    }


    @GetMapping("/admin/schedule")
    public String adminSchedule() {
        return "admin/schedule";
    }
}
```

+ RegistrationController:Provides REST API endpoints and delegates all work to the service.



```java
public class RegistrationController {
    private RegistrationService registrationService;

    /**
     * Register an attendee for a session
     */
    @PostMapping
    public ResponseEntity<?> registerAttendee(@RequestBody Registration registration) {
        try {
            Registration savedRegistration = registrationService.registerAttendee(registration);
            return ResponseEntity.status(HttpStatus.CREATED).body(savedRegistration);
        } catch (IllegalArgumentException e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        } catch (IllegalStateException e) {
            return ResponseEntity.status(HttpStatus.CONFLICT).body(e.getMessage());
        }
    }

    /**
     * Get all registrations for a session
     */
    @GetMapping("/session/{sessionId}")
    public ResponseEntity<List<Registration>> getRegistrationsBySession(
            @PathVariable Long sessionId) {
        List<Registration> registrations = registrationService.getRegistrationsBySession(sessionId);
        return ResponseEntity.ok(registrations);
    }

    /**
     * Get all registrations for an attendee
     */
    @GetMapping("/attendee/{attendeeId}")
    public ResponseEntity<List<Registration>> getRegistrationsByAttendee(
            @PathVariable Long attendeeId) {
        List<Registration> registrations = registrationService.getRegistrationsByAttendee(attendeeId);
        return ResponseEntity.ok(registrations);
    }

    /**
     * Cancel a registration
     */
    @DeleteMapping("/{registrationId}")
    public ResponseEntity<?> cancelRegistration(@PathVariable Long registrationId) {
        try {
            registrationService.cancelRegistration(registrationId);
            return ResponseEntity.noContent().build();
        } catch (IllegalArgumentException e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
```

+ RegistrationService: Handles all business logic and validation rules for registrations.

```java
public class RegistrationService {

    @Autowired
    private SessionRepository sessionRepository;

    /**
     * Register an attendee for a session with capacity validation
     */
    @Transactional
    public Registration registerAttendee(Registration registration) {
        // Validate session exists
        Session session = sessionRepository.findById(registration.getSession().getId())
            .orElseThrow(() -> new IllegalArgumentException(
                "Session not found with ID: " + registration.getSession().getId()));

        // Get the room for this session
        Room room = session.getRoom();
        if (room == null) {
            throw new IllegalStateException(
                s: "Session does not have an assigned room");
        }

        // Count current registrations for this session
        long currentCount = registrationRepository.countBySessionId(session.getId());

        // Check room capacity
        if (currentCount >= room.getCapacity()) {
            throw new IllegalStateException(
                String.format(format: "Session '%s' is at full capacity (%d/%d)",
                    session.getTitle(),
                    currentCount,
                    room.getCapacity()));
        }

        // Check for duplicate registration (same attendee, same session)
        boolean alreadyRegistered = registrationRepository
            .existsByAttendeeIdAndSessionId(
                registration.getAttendee().getId(),
                session.getId());

        if (alreadyRegistered) {
            throw new IllegalStateException(
                s: "Attendee is already registered for this session");
        }

        // Save the registration
        return registrationRepository.save(registration);
    }
```

5. Ra Socheatey

GlobalExceptionHandler.java:Implementation of the Global Exception Handler to catch AccessDeniedException. This redirects unauthorized users to the session list with a security alert instead of showing a 403 error page.

```java
package com.project5.rcrsms.exception;

import org.springframework.security.access.AccessDeniedException;
import org.springframework.ui.Model;
```

```java
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import
org.springframework.web.servlet.mvc.support.RedirectAttributes;

@ControllerAdvice
public class GlobalExceptionHandler {

    // Catch Access Denied (e.g., Participant trying to enter
Admin pages)
    @ExceptionHandler(AccessDeniedException.class)
    public String handleAccessDenied(AccessDeniedException ex,
RedirectAttributes ra) {
        ra.addFlashAttribute("error", "Security Alert: You do not
have permission to access that section.");
        return "redirect:/sessions";
    }

    // Catch Logic Errors (Duplicate Registration, "User not
found", etc.)
    @ExceptionHandler(IllegalStateException.class)
    public String handleLogicErrors(IllegalStateException ex,
RedirectAttributes ra) {
        ra.addFlashAttribute("error", ex.getMessage());
        return "redirect:/sessions";
    }

    @ExceptionHandler(Exception.class)
    public String handleGeneralError(Exception ex, Model model) {
        model.addAttribute("error", "Something went wrong: " +
ex.getMessage());
        return "error";
    }
}
```

Task: Form Validation

- **Entity Constraints**: Implemented `@NotBlank` and `@Size` validation in the `Session` entity to ensure all submissions have a valid title and a description of at least 10 characters.
- **Controller Logic**: Integrated `@Valid` in `SessionController` to catch empty submissions before they reach the database.

Create.html

```
<div class="mb-3">
    <label class="form-label">Session Description</label>

    <textarea th:field="*{description}"
              class="form-control"
              placeholder="Enter session details here..."
              rows="4"></textarea>

    <div class="text-danger"
         th:if="${#fields.hasErrors('description')}"
         th:errors="*{description}">
    </div>
</div>
```