ព្រះរាជាណាចក្រកម្ពុជា

ជាតិ សាសនា ព្រះមហាក្សត្រ

REPORT OF Progress Week 4

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

Name	Progress
Sophat Odom	• Fix route for Browse sessions in nav bar • fix wrong sql file name for flyway • fix name conflicts of query method (jpa repository) • fix requestParams have no parameter name for registration (when clicking on register button) • fix enum conflict of roles when choosing in register compare to database or in defined code • configure to connect to a remote database (in railway) • Add hasAnyAuthority to the all admin pages in securityConfig to ensure that other roles cannot access to admin pages.
Sopheap Sothiphak	• Redesign html structures • Create new Chair Controllers • Modify HTML Contents • Add sessionStatus to Session entity • Add session save to adminController
Thy Pharoth	• configure the MainController to ensure each role lands on the appropriate page. • update the SessionController to use SessionService instead of directly accessing the repositories
Phe Rithika	• Update the Login Request : - Displays object information and hiding password as `[PROTECTED]` - Compares LoginRequest objects by username only - implementing the **`hashCode()`** method on username • Debug SessionController
Ra Socheatey	• Update login response • Update the conferenceController from using `conferenceService`

	to uses conferenceService

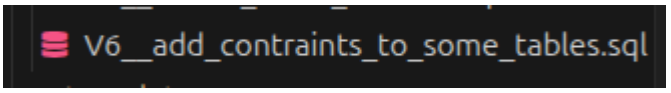
Screenshots of the progress:

1. Sophat Odom

- **Getmapping for sessions**

```
<li class="nav-item" sec:authorize="!isAuthenticated()">
    <a class="nav-link" href="/session/list">Browse Sessions</a>
</li>
```

- **Wrong sql file name (forgot __)**

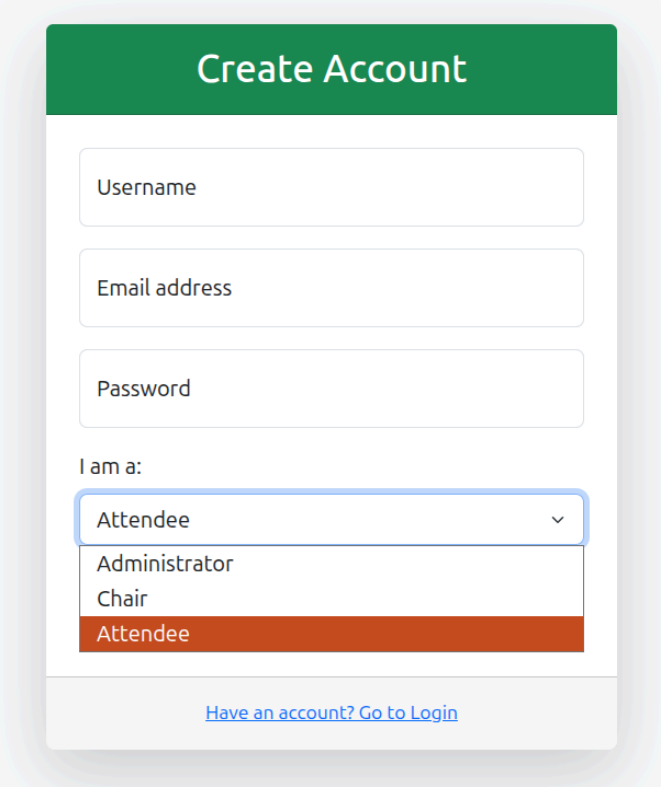


- **RequestParam and change default value**

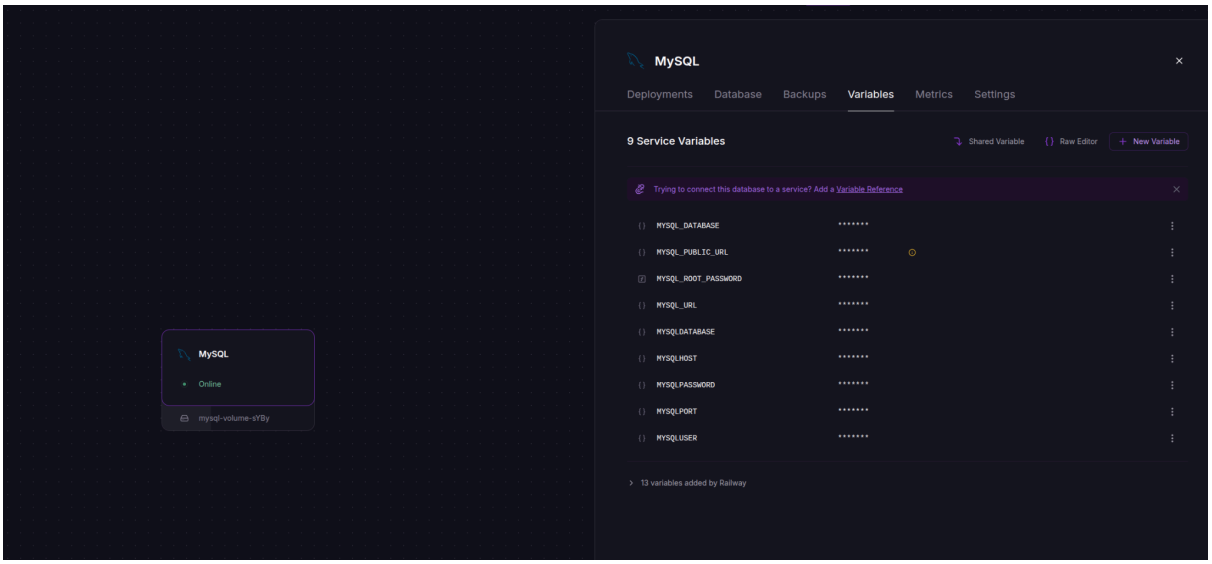
```
@PostMapping("/register")
public String registerUser(
        @RequestParam("username") String username,
        @RequestParam("password") String password,
        @RequestParam(name = "role", defaultValue = "PARTICIPANT") String role,
        RedirectAttributes redirectAttributes) {
```

- **Conflicting enum for roles between interface and the database and code**

```
<div class="mb-3">
    <label class="form-label" for="role">I am a:</label>
    <select class="form-select" id="role" name="role">
        <option value="ADMIN">Administrator</option>
        <option value="CHAIR">Chair</option>
        <option value="PARTICIPANT" selected>Attendee</option>
        </select>
</div>
```



- **Configure remote database (using railway)**



- **Add hasAnyAuthority to Admin in securityConfig**

```
.csrf(csrf -> csrf.disable())
.authorizeHttpRequests(auth -> auth
    .requestMatchers("/css/**", "/js/**", "/api/login", "/register", "/login", "/error", "/sessions").permitAll()
    .requestMatchers("/admin/**").hasAnyAuthority("ADMIN")
    .requestMatchers("/registrations/add").authenticated()
    .anyRequest().authenticated()
)
```

2. Sopheap Sothiphak

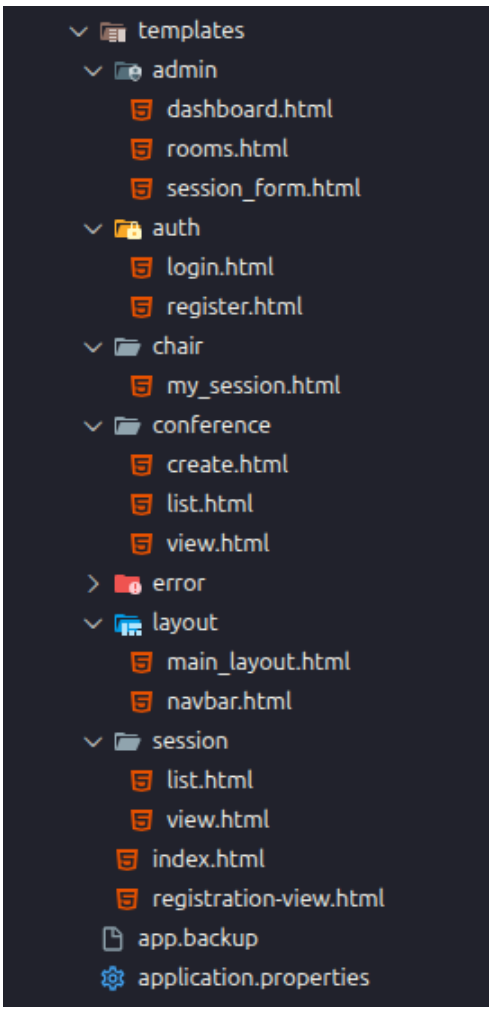
```
    public enum SessionStatus {
        PENDING,
        APPROVED,
        REJECTED,
        SCHEDULED
    }
      You, 8 minutes ago • Uncommitted changes
    // --- 2. ADD THIS FIELD ---
    @Enumerated(EnumType.STRING)
    private SessionStatus status;
```

```
    // --- 2. SAVE SESSION ---
    @PostMapping("/sessions/save")
    public String saveSession(@ModelAttribute Session session) {
        // Admin created it -> Automatically APPROVED
        session.setStatus(SessionStatus.APPROVED);
        sessionRepo.save(session);
        return "redirect:/admin/dashboard?success=Session Created";
    }
}       You, last week • Fix startup errors, config security, and implem…
```

```
@Controller
@RequestMapping("/chair")
@PreAuthorize("hasRole('CHAIR')")
public class ChairController {

    @Autowired private SessionRepository sessionRepo;
    @Autowired private UserRepository userRepo;

    @GetMapping("/my-sessions")
    public String mySessions(Model model, Principal principal) {
        UserEntity chair = userRepo.findByUsername(principal.getName()).orElseThrow();
        // Find sessions assigned to this specific chair
        model.addAttribute(attributeName: "sessions", sessionRepo.findByChair(chair));
        return "chair/my_sessions"; // Points to templates/chair/my_sessions.html
    }
}
```



3. Thy Pharoth

- configure the MainController to ensure each role lands on the appropriate page

```
// Redirect based on user role
switch (user.getRole()) {
    case ADMIN:
        return "redirect:/admin/dashboard";
    case CHAIR:
        return "redirect:/sessions";
    case PARTICIPANT:
        return "redirect:/conferences";
    default:
        model.addAttribute(attributeName: "title", attributeValue: "Welcome to ConfSys");
        return "index";
}
```

- Updated all methods to use service layer

```
@Autowired
private SessionService sessionService;
```

- listSessions() - Now uses sessionService.getAllSessions()

```java
@GetMapping({"/list", "/", ""})
public String listSessions(Model model) {
    model.addAttribute(attributeName: "sessions", sessionService.getAllSessions());
    return "session/list";
}
```

- handleSubmission() - Now uses sessionService.createSession()

```java
public String handleSubmission(@Valid @ModelAttribute("session") Session session,
                               BindingResult result,
                               RedirectAttributes ra) {
    if (result.hasErrors()) {
        return "session/create";
    }
    // Save the session to the database
    sessionService.createSession(session);
    // 3. Add a success message to show on the sessions list page
    ra.addFlashAttribute(attributeName: "success", "Session '" + session.getTitle(
    return "redirect:/sessions";
}
```

- saveSession() - Now uses sessionService.updateSession()

```java
// 3. Handle Create/Update Submission
@PostMapping("/save")
public String saveSession(@ModelAttribute("session") Session session) {
    if (session.getSessionTime() == null) {
        session.setSessionTime(LocalDateTime.now());
    }

    if (session.getSessionId() != null) {
        sessionService.updateSession(session.getSessionId(), session);
    } else {
        sessionService.createSession(session);
    }
    return "redirect:/admin/schedule";
}
```

- showEditForm() - Now uses sessionService.getSessionById()

```java
// 4. Show Edit Form
@GetMapping("/edit/{id}")
public String showEditForm(@PathVariable("id") Long id, Model model) {
    Session session = sessionService.getSessionById(id)
        .orElseThrow(() -> new IllegalArgumentException("Invalid session Id:" + i

    model.addAttribute(attributeName: "session", session);
    model.addAttribute(attributeName: "conferences", conferenceRepo.findAll());
    model.addAttribute(attributeName: "chairs", userRepo.findAll());

    // 3. NEW: Add Rooms here too so editing works
    model.addAttribute(attributeName: "rooms", roomRepo.findAll());

    return "session/create";
}
```

- deleteSession() - Now uses sessionService.deleteSession()

```java
// 5. Delete Session
@GetMapping("/delete/{id}")
public String deleteSession(@PathVariable("id") Long id) {
    sessionService.deleteSession(id);
    return "redirect:/admin/schedule";
}
```

4. Phe Rithika

```java
package com.project5.rcrsms;

import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Size;

public class LoginRequest {

    @NotBlank(message = "Username is required")
    @Size(min = 3, max = 50, message = "Username must be between 3 and 50 characters")
    private String username;

    @NotBlank(message = "Password is required")
    @Size(min = 6, max = 100, message = "Password must be between 6 and 100 characters")
    private String password;

    public LoginRequest() {
    }

    public LoginRequest(String username, String password) {
        this.username = username;
        this.password = password;
    }

    // Getters and Setters
    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;
    }

    // toString (excluding password for security)
    @Override
    public String toString() {
        return "LoginRequest{" +
                "username='" + username + '\'' +
                ", password='[PROTECTED]'" +
                '}';
    }

    // equals and hashCode (optional, but good practice)
    @Override
    public boolean equals(Object o) {
        if (this == o)
            return true;
        if (o == null || getClass() != o.getClass())
            return false;
        LoginRequest that = (LoginRequest) o;
        return username != null ? username.equals(that.username) : that.username == null;
    }

    @Override
    public int hashCode() {
        return username != null ? username.hashCode() : 0;
    }
}
```

```java
// 5. Delete Session
@DeleteMapping("/delete/{id}")
public String deleteSession(@PathVariable("id") Long id) {
    Session session = sessionRepo.findById(id)
        .orElseThrow(() -> new IllegalArgumentException("Invalid session Id:" + id));

    sessionRepo.delete(session);
    return "redirect:/admin/schedule";
}
```

5. Ra Socheatey

Update LoginResponse

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

public class LoginResponse {
    private Long userId;
    private String username;
    private String role;
    private String message;
    private boolean success;

    public LoginResponse() {
    }
    // Constructor for Successful Login
    public LoginResponse(Long userId, String username, String role, String message, boolean success) {
        this.userId = userId;
        this.username = username;
        this.role = role;
        this.message = message;
        this.success = success;
    }

    // Constructor for Failed Login
    public LoginResponse(String message, boolean success) {
        this.message = message;
        this.success = success;
    }
    // Getters and Setters
    public Long getUserId() { return userId; }
    public void setUserId(Long userId) { this.userId = userId; }

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

    public String getRole() { return role; }
    public void setRole(String role) { this.role = role; }

    public String getMessage() { return message; }
    public void setMessage(String message) { this.message = message; }

    public boolean isSuccess() { return success; }
    public void setSuccess(boolean success) { this.success = success; }
}
```

Refactor: Update ConferenceController to use ConferenceService

```
package com.project5.rcrsms.controller;

import com.project5.rcrsms.Entity.Conference;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import com.project5.rcrsms.Service.ConferenceService;

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

    @Autowired
    // private ConferenceRepository conferenceRepo;
    private ConferenceService conferenceService;

    // 1. List All Conferences
    @GetMapping("") // Matches /conferences
    public String listConferences(Model model) {
        // model.addAttribute("conferences", conferenceRepo.findAll());
        model.addAttribute("conferences",
conferenceService.getAllConferences());
        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("conference", new Conference());
        return "conference/create";
    }

    // 3. Save Conference
    @PostMapping("/save")
    public String saveConference(@ModelAttribute("conference")
Conference conference) {
```

```
        //conferenceRepo.save(conference);
        conferenceService.createConference(conference);
        return "redirect:/admin/dashboard"; // Redirect to dashboard
after saving
    }
}
```