
ព្រះរាជាណាចក្រកម្ពុជា
ជាតិ សាសនា ព្រះមហាក្សត្រ


# Software Engineering

## Project: Research Conference Registration and Session Management System

Week1


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

Lecturer: ROEUN Pacharoth (TP)


**ACADEMIC YEAR 2025-2026**

# 1. Project Overview

The Research Conference Registration and Session Management System is designed to support the management of academic conferences by providing functionality for user registration, session selection, administrative scheduling, and session chair responsibilities. The system aims to simulate a real-world conference management platform using role-based access control and structured data management.

This project was selected from the provided list as it involves multiple system components, including security, database design, backend logic, and frontend presentation, making it suitable for applying software engineering principles.

# 2. System Requirements Analysis

The system defines three primary user roles:

- **Administrator**: Responsible for managing users, creating and scheduling conference sessions, and overseeing system data.
- **Session Chair**: Assigned to specific sessions and responsible for managing session-related activities.
- **Participant**: Registers for the conference and selects sessions to attend.

The main functional requirements of the system include:

- User authentication and authorization using role-based access control
- Creation and management of conference sessions
- Participant registration and session selection
- Controlled access to system features based on user roles
- Web-based user interface using server-side rendering

# 3. Task Allocation and Development Plan

Based on the project specification, the system development was divided into the following major components:

1. Security and Authentication
2. Main Entity CRUD Operations

3. Secondary Module CRUD Operations
4. Frontend Development with Thymeleaf
5. Database Design and Management

Each component was further divided into smaller subtasks to allow incremental development and parallel progress. This task division helps reduce dependency issues and improves development efficiency.

# 4. Work Completed (Current Progress)

## Project Setup (SOPHAT Odom)

The Spring Boot project was successfully configured with required dependencies, including Spring Web, Spring Data JPA, Thymeleaf, MySQL, and Flyway. The application runs successfully and connects to the database.

## Security and Authentication (THY Pharoth)

Backend structures for authentication and authorization were prepared by creating the User entity, Role enum (ADMIN, CHAIR, PARTICIPANT), and User repository. These components provide the foundation for implementing login, role-based access control, and secure password management in the next development phase.

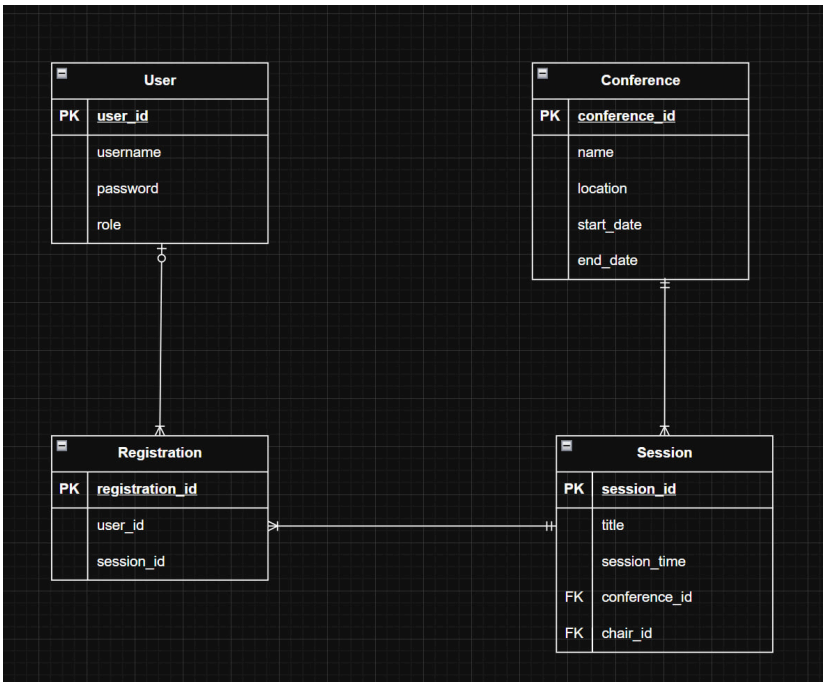
```java
package main.java.com.project5.rcrsms;

public enum Role {
    ADMIN,
    CHAIR,
    PARTICIPANT
}
```

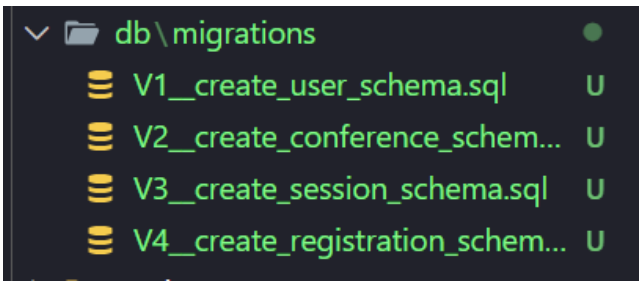
```java
package main.java.com.project5.rcrsms;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.Optional;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
}
```

```java
package com.project5.demo.security;
import jakarta.persistence.*;

@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String username;

    @Column(nullable = false)
    private String password;

    @Enumerated(EnumType.STRING)
    private Role role;

    public User() {}

    public User(String username, String password, Role role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    public Long getId() { return id; }

    public String getUsername() { return username; }

    public String getPassword() { return password; }

    public Role getRole() { return role; }
}
```

## Database Design (SOPHAT Odom)

Core database entities were identified, including User, Session, and Registration. Relationships between these entities were defined to support session assignment and participant registration. Flyway was introduced to manage database schema versioning and ensure controlled database evolution.






```sql
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('ADMIN', 'CHAIR', 'PARTICIPANT') NOT NULL
);
```

```sql
CREATE TABLE conferences (
    conference_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    location VARCHAR(255) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL
);
```

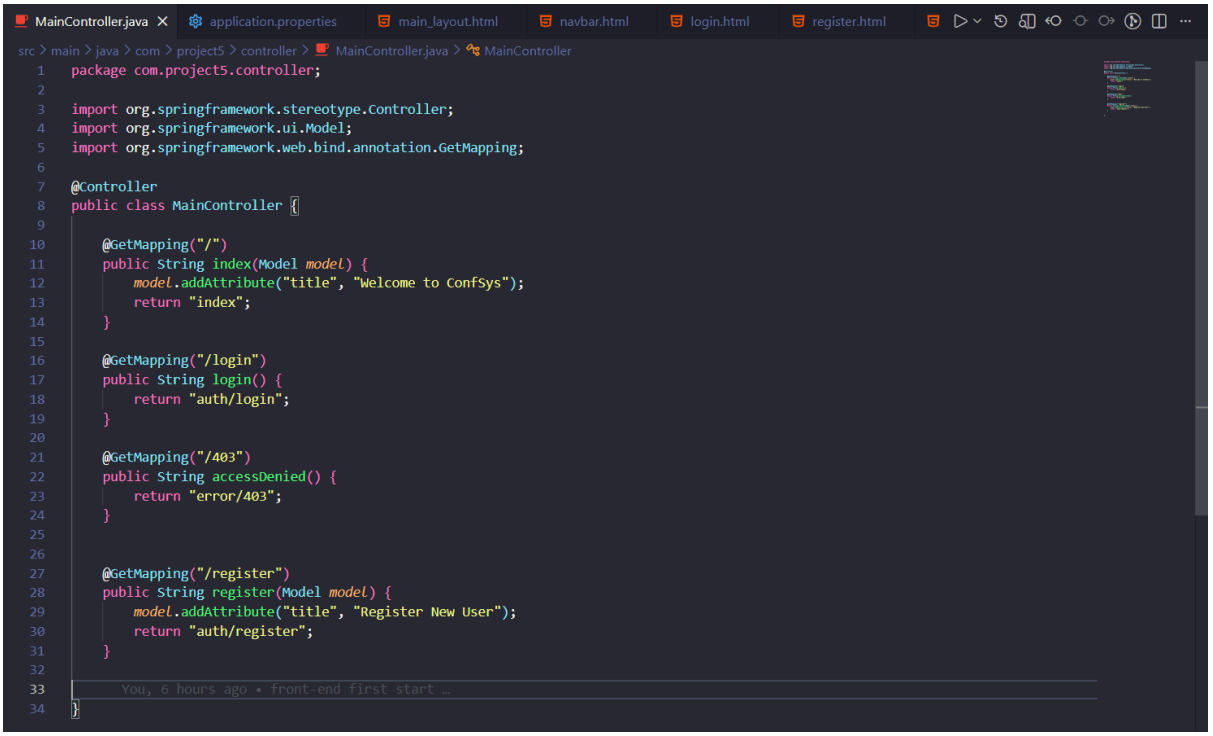
```sql
CREATE TABLE sessions (
    session_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    session_time DATETIME NOT NULL,
    conference_id INT NOT NULL,
    chair_id INT NOT NULL,
    FOREIGN KEY (conference_id) REFERENCES conferences(conference_id),
    FOREIGN key (chair_id) REFERENCES users(user_id)
);
```

```sql
CREATE TABLE registrations (
    registration_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    session_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (session_id) REFERENCES sessions(session_id),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

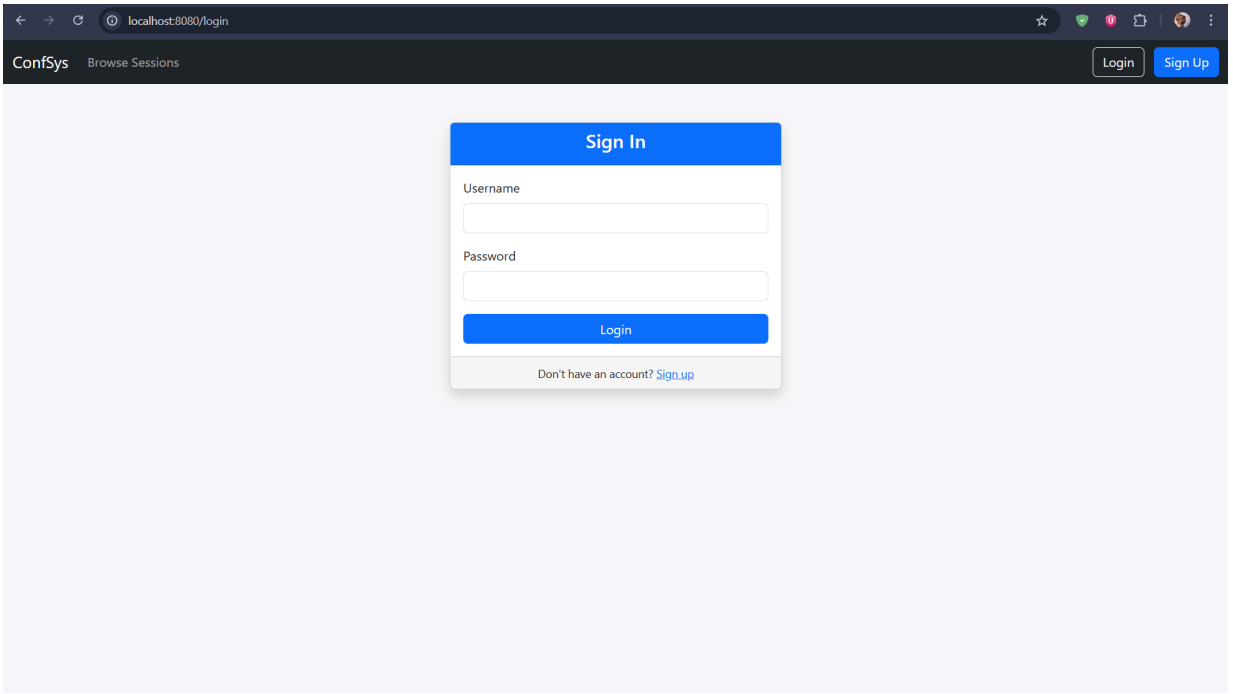
## Frontend/Thymeleaf (SOPHEAP Sothiphak)

Initial Thymeleaf templates were prepared for login and role-based pages. Page structure and navigation flow were designed to align with backend security rules, providing the foundation for user interaction and access-controlled views. We have builded the testing Login Page, Register Page and Landing Page and also a MainController.java with a mock data for testing the frontend.

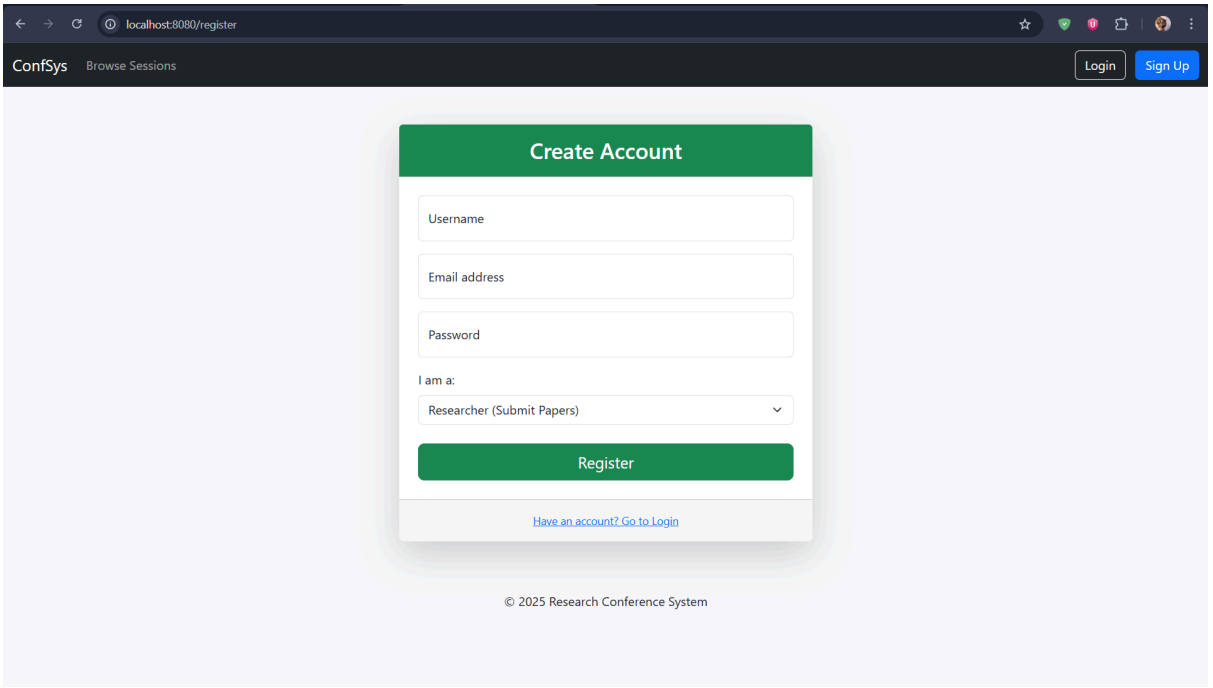
## MainController.java



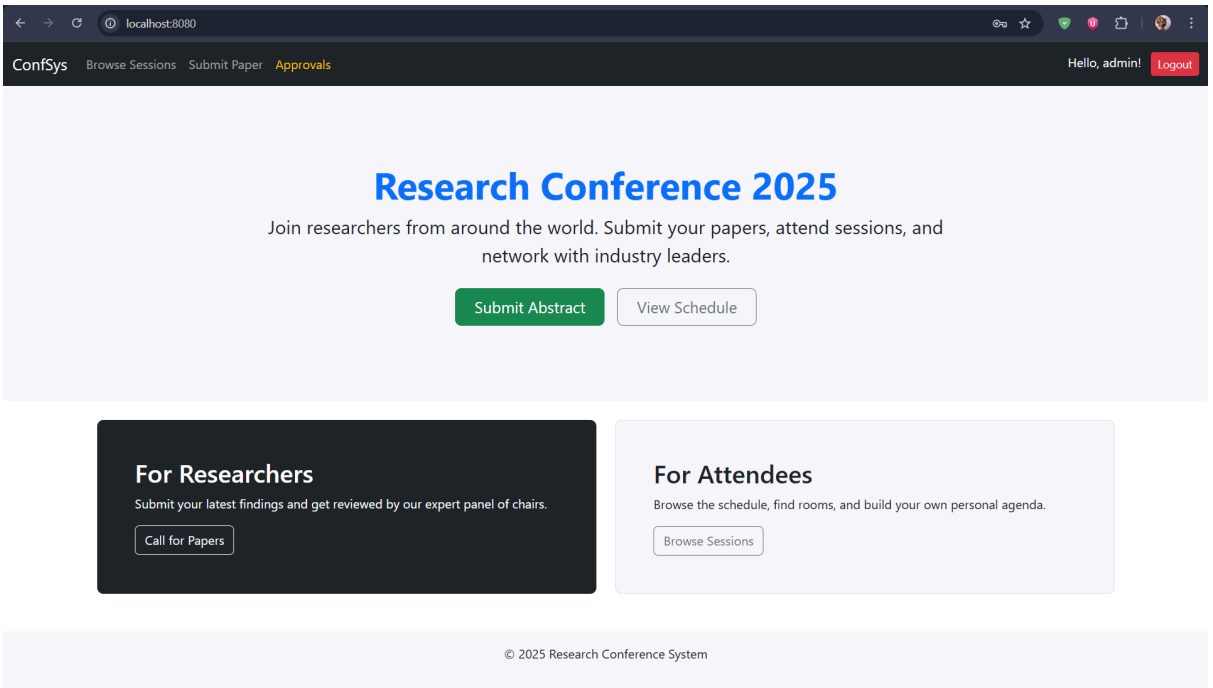

## Login Page



## Register Page




## Landing Page



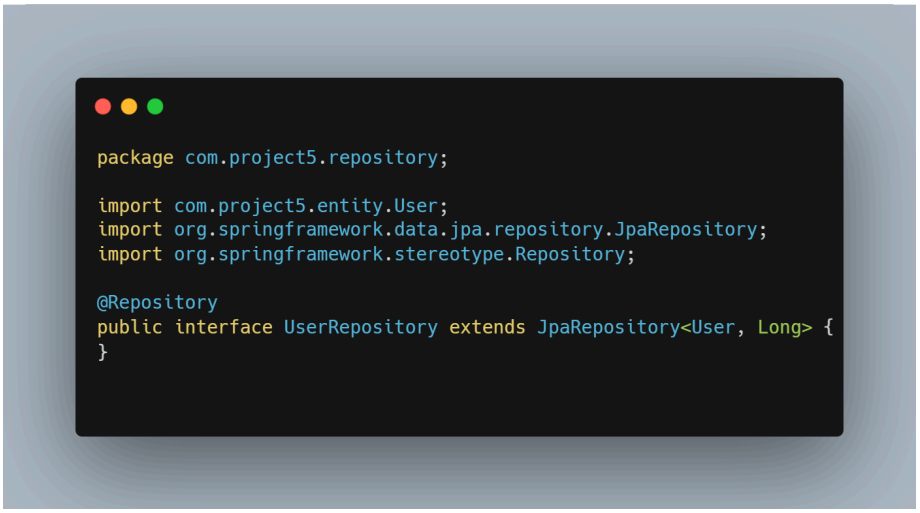
## Backend Entity and CRUD Design (PHE Rithika & RA Socheatey)

Entity classes were created to map application objects to database tables. Repository interfaces were prepared to support basic CRUD operations, forming the foundation for backend business logic such as user management and session selection.

## PHE Rithika work:

- **User Entity & Repository**

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

import jakarta.persistence.*;
import jakarta.validation.constraints.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "user_id")
    private Long userId;

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

    @NotBlank(message = "Password is required")
    @Size(min = 6, message = "Password must be at least 6 characters")
    private String password;

    @NotBlank(message = "Role is required")
    private String role;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Registration> registrations = new ArrayList<>();

    // Constructors
    public User() {}

    // 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 getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

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

    public List<Registration> getRegistrations() { return registrations; }
    public void setRegistrations(List<Registration> registrations) { this.registrations = registrations; }
}
```



```java
package com.project5.repository;

import com.project5.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {
}
```

- **Conference Entity & Repository**

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

import com.project5.rcrsms.model.Conference;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ConferenceRepository extends JpaRepository<Conference, Long> {
}
```

```java
    @NotNull(message = "Start date is required")
    @Column(name = "start_date")
    private LocalDate startDate;

    @NotNull(message = "End date is required")
    @Column(name = "end_date")
    private LocalDate endDate;

    @OneToMany(mappedBy = "conference", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Session> sessions = new ArrayList<>();

    public Conference() {}


    public Long getConferenceId() { return conferenceId; }
    public void setConferenceId(Long conferenceId) { this.conferenceId = conferenceId; }

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

    public String getLocation() { return location; }
    public void setLocation(String location) { this.location = location; }

    public LocalDate getStartDate() { return startDate; }
    public void setStartDate(LocalDate startDate) { this.startDate = startDate; }

    public LocalDate getEndDate() { return endDate; }
    public void setEndDate(LocalDate endDate) { this.endDate = endDate; }

    public List<Session> getSessions() { return sessions; }
    public void setSessions(List<Session> sessions) { this.sessions = sessions; }
}
```

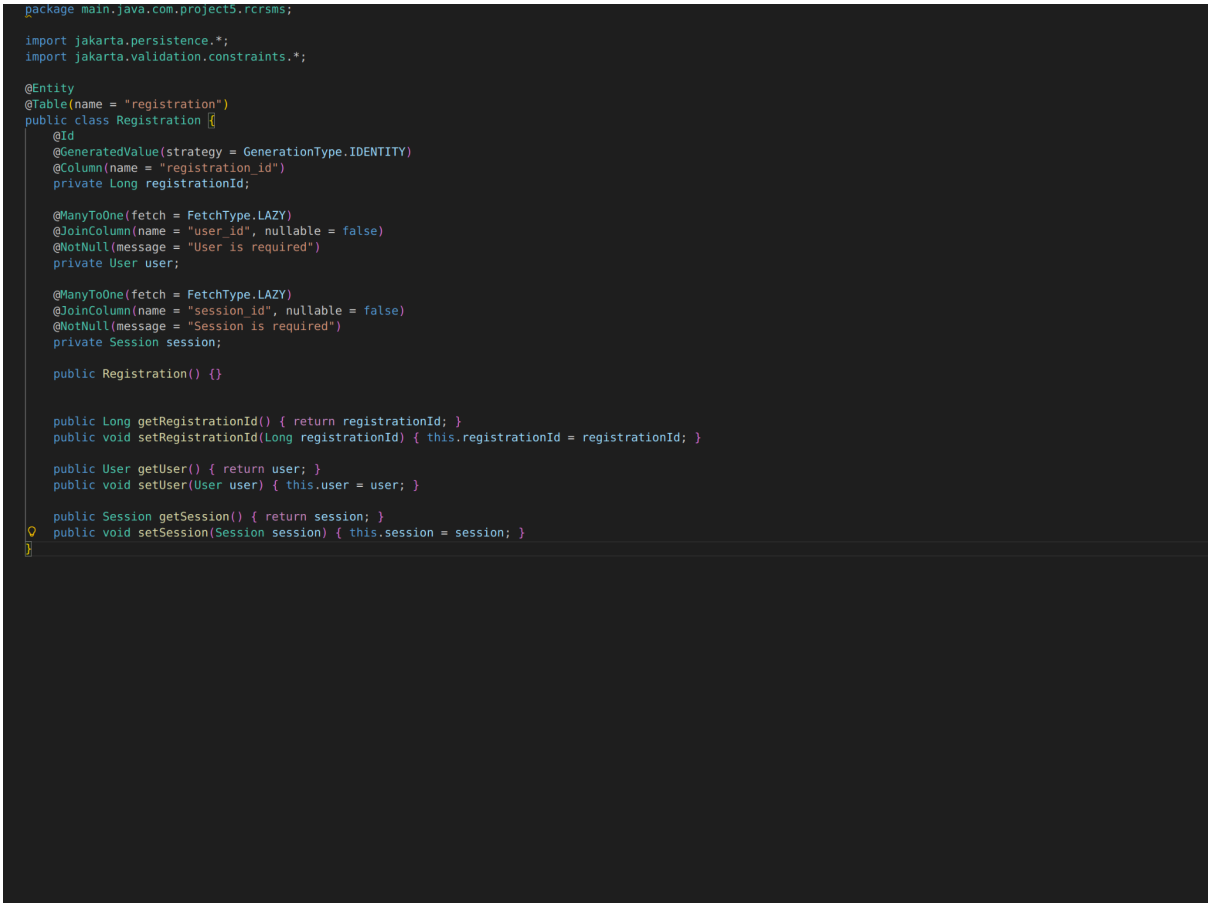
- **Session Entity & Repository**

```java
import java.util.List;

@Entity
@Table(name = "session")
public class Session {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "session_id")
    private Long sessionId;

    @NotBlank(message = "Title is required")
    @Size(min = 3, max = 200, message = "Title must be between 3 and 200 characters")
    private String title;

    @NotNull(message = "Session time is required")
    @Column(name = "session_time")
    private LocalDateTime sessionTime;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "conference_id", nullable = false)
    @NotNull(message = "Conference is required")
    private Conference conference;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "chair_id")
    private User chair;

    @OneToMany(mappedBy = "session", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Registration> registrations = new ArrayList<>();

    public Session() {}

    public Long getSessionId() { return sessionId; }
    public void setSessionId(Long sessionId) { this.sessionId = sessionId; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public LocalDateTime getSessionTime() { return sessionTime; }
    public void setSessionTime(LocalDateTime sessionTime) { this.sessionTime = sessionTime; }

    public Conference getConference() { return conference; }
    public void setConference(Conference conference) { this.conference = conference; }

    public User getChair() { return chair; }
    public void setChair(User chair) { this.chair = chair; }

    public List<Registration> getRegistrations() { return registrations; }
    public void setRegistrations(List<Registration> registrations) { this.registrations = registrations; }
}
```

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

import com.project5.rcrsms.model.Session;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface SessionRepository extends JpaRepository<Session, Long> {
}
```

## RA Socheatey work:

- **Registration Entity & Repository**



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

import jakarta.persistence.*;
import jakarta.validation.constraints.*;

@Entity
@Table(name = "registration")
public class Registration {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "registration_id")
    private Long registrationId;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", nullable = false)
    @NotNull(message = "User is required")
    private User user;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "session_id", nullable = false)
    @NotNull(message = "Session is required")
    private Session session;

    public Registration() {}


    public Long getRegistrationId() { return registrationId; }
    public void setRegistrationId(Long registrationId) { this.registrationId = registrationId; }

    public User getUser() { return user; }
    public void setUser(User user) { this.user = user; }

    public Session getSession() { return session; }
    public void setSession(Session session) { this.session = session; }
}
```

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

import com.project5.rcrsms.model.Registration;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

@Repository
public interface RegistrationRepository extends JpaRepository<Registration, Long> {

    Optional<Registration> findByUserIdAndSessionId(Long userId, Long sessionId);
    List<Registration> findBySessionId(Long sessionId);

}
```

# 5. Challenges and Considerations

During development, challenges were encountered related to dependency configuration, particularly database migration tooling and version compatibility. Additional consideration was required to balance system realism with project scope, ensuring the design remains manageable while fulfilling all required features.

# 6. Next Planned Tasks

The next phase of development will include:

- Implementing CRUD operations for conference sessions
- Completing user registration and session selection workflows
- Integrating role-based access control into the frontend views
- Finalizing database migrations and constraints
- Testing system functionality and access restrictions

# 7. Conclusion

At this stage, the project foundation has been successfully established. Core system requirements have been analyzed, the database schema has been designed, and the application structure is in place. The project is progressing according to plan, and future development will focus on completing functionality and improving system integration.