CS9053 Java Final Project Proposal

Project Title: Campus Events and Entertainment Platform

Student Name: Cheng Yang(cy2932), Yixin Leng(yl13851), Zuqing Gao(zg2921)

Date: December 15, 2025

1. Project Overview

This project proposes to develop a Campus Events and Entertainment Platform, a centralized web-based system that helps university communities organize, discover, and participate in campus events and activities. The system serves as a digital hub to enhance student engagement, simplify event management, and strengthen social connections among students, faculty, and staff.

Target Users:
- Event Participants: Students or staff who join and attend campus events.
- Event Holders: Clubs, associations, or faculty members who organize events.
- Administrators: System operators who review and manage content.

2. Overall Goals

The main goal is to create a centralized, user-friendly, and intelligent campus event system that:
- Enables effortless event creation, publication, and management.
- Allows users to explore and register for events based on personal interests.
- Encourages campus socialization through comments, reviews, and chat features.
- Enhances community spirit and promotes a vibrant, participatory campus culture.

3. Key Functionalities

1. Events: Users can create events with either capacity-based or form-based enrollment. The system supports event enrollment, rating, searching, and Markdown-based event descriptions. Event authors can edit their own events and export enrollment lists in Excel format.

2. Posts: Users can publish posts, like, collect, and comment on content. Posts can be filtered based on related eventId to support event-centered discussions.

3. User Profile: Users can edit personal information and avatars, and view personalized pages including “My Events,” “My Posts,” “Favorite Events,” and “Favorite Posts.”

4. Messaging: The platform supports private direct messaging by clicking user avatars. Real-time chat functionality is implemented using WebSocket.

5. Administration: Administrators can manage user permissions and create, edit, or delete events and posts to maintain system integrity..

4. Technical Design

Frontend: Thymeleaf is used to generate dynamic server-side views and integrate with Spring MVC controllers.

Backend: The backend is implemented using Spring Boot, applying the Spring Framework's dependency injection and MVC architecture.

Database: H2 is used for persistent data storage, with JPA providing Java-based database access and object–relational mapping as covered in Java database programming.

Communication: WebSocket is used to support real-time, client–server communication for features such as live chat.

Security: The system applies Java-based authentication and authorization mechanisms, including JWT token generation and verification and password hashing, to ensure secure access control.

Advanced Java Concepts Applied

This project integrates four major advanced Java concepts with the Final Project requirements:

1. Spring Framework & Dependency Injection – Used to manage application components and implement a layered MVC architecture.

2. Java Database Programming (JDBC / JPA) – Used for persistent data storage, SQL-based operations, and secure database access.

3. Networking & Client–Server Communication – Implemented through WebSocket-based real-time chat following the client–server model.

4. Security in Java Applications – Includes JWT-based authentication, password hashing, and secure access control.

5. JUnit- The project uses JUnit, the standard Java testing framework introduced in class, to validate the correctness of core backend logic. Test classes are written to verify key application workflows through automated assertions, including homepage accessibility, user authentication, JWT generation and validation, and user registration and login behavior. Tests are executed in a mock web environment using an H2 in-memory database to isolate test data from the production system.

6. Multithreading & Concurrency – The backend handles multiple concurrent users and real-time interactions, requiring thread-safe design.

7. Maven-Apache Maven is used to manage project dependencies and the build lifecycle. All third-party Java libraries (Spring Boot, JDBC driver, WebSocket, JWT, and testing frameworks) are centrally managed through pom.xml, ensuring modularity and reproducibility.

By combining Spring-based REST APIs, concurrency control, JPA persistence, WebSocket networking, and Maven integration, the system demonstrates multiple advanced Java features.

5. Expected Outcomes

The platform follows a typical client–server architecture. Users interact with the system through a web interface rendered by Thymeleaf. Requests are handled by Spring MVC controllers, which coordinate with service-layer components and JPA repositories to process business logic and persist data.

After authentication via JWT, users can browse events, register for activities, and participate in real-time discussions through WebSocket-based chat. Administrators have elevated privileges to manage users and events.