COMP3095 - Group 34 – Status Report

Group Members:

- **Member 1:** Trang Nguyen (100749684)
- **Member 2:** Nhu Ly (101429112)
- **Member 3:** Adam Simcoe (101442161)
- **Member 4:** Nhan Tran (100898539)

Architecture Overview

1. RoomService: Handled By Trang Nguyen

- Use PostgreSQL / JPA for data storage.
- Manage room resources with attributes like roomName, capacity, features (e.g., projector, whiteboard), and availability.
- Provide endpoints to check room availability for bookings.

2. BookingService: Handled By Adam Simcoe

- Use MongoDB for data storage.
- Handle room booking requests. Each booking should include userId, roomId, startTime, endTime, and purpose.
- Prevent double-booking of rooms using appropriate validation logic.
- Communicate with RoomService to verify room availability before confirming a booking.

3. UserService: Handled By Trang Nguyen

- Use PostgreSQL / JPA to store user information.
- Manage user profiles for students, staff, and faculty with attributes like name, email, role, and userType (student, staff, faculty)
- Implement role-based access

4. EventService: Handled By Nhu Ly

- Use MongoDB to manage events.
- Create events linked to room bookings with attributes such as eventName, organizerId, eventType, and expectedAttendees.

- communicate with the UserService to verify the role of the event organizer before an event is created

5. ApprovalService: Handled By Nhan Tran

- Use MongoDB or PostgreSQL to store approval.
- Allow staff to review and approve/reject event requests. Link event approvals to the user role.
- Communicate with both EventService and UserService to fetch event details and verify if a staff member has the correct privileges to approve events.

Challenges Faced

- Service Coordination: Ensuring that services communicate effectively and handle failure scenarios
- Role-based Access: Implementing role-based access across multiple services. We implemented inter-service communication to verify the user’s role before granting access to certain operations.
- Double Booking Prevention: BookService had to prevent double bookings by ensuring room availability before confirming a booking.
- Docker Configuration: It was time-consuming to contain each microservice and ensure they could be easily deployed with a single docker-compose file. Proper configuration of networking and dependency management between containers was critical for smooth operation.

Lessons Learned

Developing the microservices-based platform for the room booking and event management system presented both challenges and rewards. Throughout the process, we gained valuable insights into microservices architecture, inter-service communication, containerization, and database management. Despite the obstacles, we successfully completed the project, ensuring each service was independently deployable and scalable while seamlessly interacting with other services.