Final Project: Software Tools Lab - Fall 2025-2026

Smart Meeting Room & Management System Backend
Teams of two/soloS

Overview

In this project, you will be tasked with creating the backend system for a Smart Meeting Room and Management System. This system will consist of four services that manage various aspects of meeting room bookings, resources, room availability, and review management. These services should communicate with each other through API calls. You will containerize all services using Docker and test them using Postman (not developing the frontend). The goal is to build a scalable backend with well-defined services, ensuring performance, security, and seamless integration.

Project Requirements

Part I Tasks (80%)

1. Service Development:

You are supposed to develop the following APIs:

Service 1: Users

- ✓ Manage user accounts, including their personal details, roles (admin, regular user), and room booking history.
- ✓ Implement APIs for:
- ✓ User registration (with name, username, password, email, role)
- ✓ User login and authentication
- ✓ Updating user details (profile information)
- ✓ Deleting a user account
- ✓ Getting all users or a specific user by username
- ✓ Viewing a user's booking history

Service 2: Rooms

Manage meeting room availability and room details.
Implement APIs for:

- ✓ Adding a new meeting room (room name, capacity, equipment, location)
- ✓ Updating room details (e.g., capacity, equipment)
- ✓ Deleting a room
- ✓ Retrieving available rooms (based on capacity, location, and equipment)
- ✓ Room status (available or booked)

Service 3: Bookings

Manage meeting room bookings, ensuring that users can reserve available rooms.
Implement APIs for:

- ✓ Viewing all bookings (with details like user, room, time slot)
- ✓ Making a booking (room, time, user)
- ✓ Updating or canceling bookings (time, room)
- ✓ Checking room availability (based on time and date)
- ✓ Storing booking history for each user

Service 4: Reviews
Handle room and service reviews from users, providing feedback on meeting room experiences.
Implement APIs for:
- ✓ Submitting a review for a meeting room (rating, comment)
- ✓ Updating a review (rating, comment)
- ✓ Deleting a review
- ✓ Retrieving reviews for a specific room
- ✓ Moderating reviews (flagging inappropriate reviews)

All in all, below is a practical recommended RBAC plan for users roles, doing the corresponding functionalities for each service and scaling with Part II:

A.Admin (a.k.a. system administrator):

- ✓ Users service: create/read/update/delete users; assign roles; reset passwords; view any user's booking history.
- ✓ Rooms service: add/update/delete rooms; set capacity/equipment/location; toggle availability.
- ✓ Bookings service: view all bookings; override/force-cancel; resolve conflicts; unblock stuck states.
- ✓ Reviews service: full moderation: remove/restore reviews; mark/clear flags; see all reviews.
- ✓ Ops: view audit logs & dashboards if you implement logging/monitoring in Part II.

B.Regular User:

- ✓ Users: manage own profile; view own booking history.
- ✓ Rooms: search & view room details/availability.
- ✓ Bookings: create/update/cancel own bookings.
- ✓ Reviews: create/update/delete own reviews.

C.Facility Manager (power user for inventory/space)
- ✓ Everything a Regular User can do plus: manage rooms (create/update/delete), set equipment lists, mark rooms out-of-service; read all bookings (for planning), but no user/role admin.

D.Moderator (lightweight review admin):

- ✓ Moderate reviews only (hide/remove/flag/unflag, see reports).

E.Auditor/Read-only
- ✓ Read-only access to Users/Rooms/Bookings/Reviews and audit logs; no writes. Useful for the "Auditing and Logging" enhancement.

F.Service Account (API)

Non-human account for inter-service calls (e.g., asynchronous notifications, metrics exporters). It has the least-privilege.

Part I Tasks (80%):

1.Name your project smartmeetingroom_FullNameMember1_FullNameMember2 in case you are in a team or only smartmeetingroom_FullName if you are solo.
2. Develop 4 services that runs on Docker using different ports.
3. You can use the database of your choice. You can create it from scratch or you can use
Dockerhub to download DB image and run it on a dedicated container.
4. Create your project on VScode and your virtual environment.
5. Start your design by writing your API calls for each service using Postman. Create a collection for the project and for each API call write an example, description and comments for the fields. Test all APIs using Postman. Create a Postman collection with all required requests, descriptions, and comments for each API endpoint.
6. Create a Github repo for the project and start by the service of your choice. I need to see
that you were using Github effectively by pulling/pushing your code regularly.
For Version Control: Use GitHub for version control. Commit your work regularly and use branching as necessary.
7. Write unit tests using Pytest for all the services to validate your API calls. These tests should insure that your API calls are well tested and ensure code quality.
8. Do Documentation using Docstings and Sphinx. Provide clear documentation of all APIs using sphinx. I need to see your html file for documentation.
9. Performance Profiling: Perform performance, memory, and code coverage profiling and include snapshots in your final report.
10.Containerization:Package your application and your dependencies into containers for consistent and efficient deployment across various environments. All services should be containerized using Docker and run on separate ports.
Database containerization:Use an appropriate database to store data. You set up your own Docker image for the database.
11Validation and Sanitization: Ensure that user inputs for reviews are validated and sanitized to prevent SQL injection and other security issues.
12. User Authentication: Ensure that only authenticated users can submit, update, or delete reviews.
Describe how user authentication is handled and how authorization is managed.
13. Moderation: Implement moderation features to handle inappropriate reviews, which could be flagged by users or administrators.

Part II Tasks (20%)

Each student should choose **two (or more)** tasks from the following list to enhance the system. Each task must be explained and accompanied by appropriate screenshots in your final report.

1. Enhanced Inter-Service Communication

Circuit Breaker Pattern: Implement fault tolerance using a circuit breaker pattern.
Rate Limiting and Throttling: Apply rate-limiting strategies for API endpoints to prevent misuse.

2. Advanced Security Measures

Auditing and Logging: Implement detailed logging of API requests and responses for auditing purposes.
Encryption: Encrypt sensitive data transfers (e.g., user information, booking details).
Secure Configuration Management: Store sensitive credentials (API keys, DB passwords) securely using a service like **AWS Secrets Manager**.

3. Performance Optimization

Caching Mechanism: Introduce caching for frequently accessed data (e.g., room availability, user data).
Load Balancing: Implement load balancing across services using **HAProxy** or **Nginx**.
Database Indexing: Optimize database queries by indexing fields like room names and booking dates.

4. Scalability and Reliability

Asynchronous Messaging: Implement **RabbitMQ** or **Kafka** for asynchronous communication between services (e.g., notifications after booking).
Health Checks: Add health check APIs to monitor service availability.
Horizontal Scaling: Use **Docker Compose** or **Kubernetes** to scale services based on demand.

5. Analytics and Insights

Real-Time Dashboards: Create dashboards using **Grafana** or **Kibana** to monitor service performance (e.g., booking frequency, user activity).
Data Analytics: Aggregate and analyze data, such as total room bookings, average room ratings, etc.

6. User and Access Management

Multi-Factor Authentication (MFA): Implement MFA for sensitive operations (e.g., deleting bookings or users).

7. Advanced Development Practices

Custom Exception Handling: Implement a global error handling framework for standardized error messages across services.
API Versioning: Implement API versioning to ensure backward compatibility.

8. Continuous Monitoring and Alerts

Prometheus: Set up **Prometheus** for monitoring and alerting anomalies (e.g., high CPU usage or downtime).
Error Tracking: Use **Sentry** or similar tools to track runtime errors in services.

9. Extending the Project Scope

Recommendation System: Implement a recommendation engine that suggests available rooms based on user preferences (e.g., capacity, equipment).

Wishlist Feature: Allow users to save rooms they are interested in for future bookings.

Abandoned Booking Notifications: Notify users about abandoned booking attempts (if the session expires or if they leave without confirming).

10. Integration with Third-Party Services

Notification System: Integrate a service like **Twilio** or **SendGrid** to notify users about bookings and cancellations.

Final Project Policy(Make sure to read and abide by the below "Zero-Tolerance Checklist ")

- **Your project name is supposed to be FirstNameLastName_FirstNameLastName_meetingroom on both your project code and your word document report or else it's a zero.**
- **We are not responsible for the accuracy of what you upload on Moodle. You are responsible for your own work. Uploading a wrong folder or a folder which fails to open is your mistake, not ours and will be given no chance to recorrect it after dead line.**
- **Not inserting an outline via PageLayout->table of contents gives you a grade of zero, i.e. typing your own table of contents without inserting it based on "PageLayout->table of contents" gives you a grade of zero.**
- **For each of the titles in your outline of your project you are supposed to mention the name of the team member who did it. You are not allowed to mention two student names working on the same sub task.You are supposed to distribute the tasks /sub tasks in Part I and Part II evenly between you. One member works for example on functionalities,etc.for user role A and its sub tasks, while the other team member on user role B and its sub tasks, etc. And so on. You are supposed to divide user roles and corresponding subtasks evenly between you. A team member cannot do Part I tasks only or Part II tasks only; he/she has to do both.**
- **Both team members can agree on the database part together but still dividing its creation, etc. Evenly between you.**
- **Please note that there will be no grade for the entire project but each student will be graded separately based on what he/she did, so divide the services/tasks and sub-services/tasks evenly between you.**
- **You are supposed to divide Part I tasks wevely between both team members and each team member do two tasks from Part II (or more as optional) and work on it correspondingly from A to Z, i.e. you are asked personally to push your work to github, etc. and not only one student doing the task(s) on behalf of both.**

- **For each of the sections and sub sections in your project, include the necessary snapshots with corresponding descriptions to prove your work.**
- **Snapshots with corresponding descriptions should reside in your report itself and not in a snapshots folder , and not in any appendix section, or else it's a zero. They should have a caption and title too or else it's a zero on it.**
- **Submitting a snapshot off the code and not a snapshot of your actual test/required task gives you a grade of zero on that service. Anytime I say test snapshot, this means it will be a trial of the service. Do not post your pytest code done for it.Trial means showing input, process, output and not only the end result.**
- **Not having snapshots in your report in any of the sections gives you a grade of zero on that section.**
- **Not submitting your project on the announced due date will give you a zero grade automatically.**

There is a zero-tolerance policy for cheating & plagiarism. Students who cheat or plagiarize or help other students cheat or plagiarize will receive a grade of zero on the project. Students also may be subject to further disciplinary action as deemed necessary. Using chatGPT for help is tolerated up to

30% maximum for similarity with your code. You can ask AI for for conceptual explanations (e.g., "how to do a required skill in general or what does a required skill mean?").You can also use AI for debugging hints (e.g., "what does this error message usually mean? how to solve this error").It is not allowed to copy-paste full solutions from AI or else, with more than 30% similarity,you will receive grade deductions for this act as AI inserts hidden characters in its answers which you cannot view and which is detected by licensed software like turnitin and free ones like gptzero.

- **Misplacing any of the services in any of the sections of the report, i.e. part I services/tasks in part II services/tasks section or vice versa will give you a zero as a grade of the corresponding services.**
- **Implementing any service and not addressing it in your report (explanation, description, and snapshots) will give you a zero on that service even if you demonstrated it to me in your demo.**

Project Final Report Structure

1. Title Page

- **Project Title**
- **Student Name(s)**
- **Date of Submission**

2. Table of Contents

- List of sections with page numbers and the team member name who worked on each.

3. Introduction

- **Project Overview:** Briefly describe the project, its purpose, and its scope.
- **Objectives:** What were the main goals of your project?

4. System Architecture

- **Service Description:** Describe each of the services (Users, Rooms, Bookings, and Reviews) and their responsibilities.

5. Implementation Details

- **Service-Specific Implementation:Explain each of the functionalities and key APIs requested above for each of the services below: For each user role, detail all his/her/its functionalities related to:**
  - **Service 1 - Users:** Explain the functionality, key APIs,etc..
  - **Service 2 - Rooms:** Describe how rooms are managed, added, and updated,etc..
  - **Service 3 - Bookings:** Outline how bookings are processed, and historical data is handled,etc..
  - **Service 4 - Reviews:** Detail the review submission, updating, moderation, and retrieval processes,etc..
- **Other services if any**
- **You are not supposed to show me your code here. You are supposed to do API Documentation:** Include descriptions of API endpoints, request/response formats, and all calls. Provide screenshots or snippets from Postman collections.

6. Database Design

- **Schema Diagram:** Provide a diagram of the database schema showing tables and corresponding fields.

7. Error Handling and Validation

- **Error Management:** Describe how errors are handled in the system.
- **Validation:** Explain what types of validation are implemented and where.

8. Testing

- **Testing Strategy:** Outline the approach to testing, including unit tests, integration tests, and any other relevant testing.

- **Test Cases:** Provide snapshots of all test cases and corresponding results. Include screenshots or summaries of Pytest results.

9. Deployment and Integration

- **Docker Setup:** Describe how Docker was used to containerize the services. Include Dockerfile and docker-compose.yml configurations.

10. Documentation and Profiling

- **Documentation:** Provide links or references to your project documentation created with Sphinx or other tools.

- **Performance Profiling:** Do performance, memory, and code coverage profiling to your code.Include appropriate snapshots in your report.

11. GitHub and Version Control

- **Repository Links:** Provide links to the GitHub repository and highlight important branches or commits. Provide snapshots of your github repository history,I.e. showing who did what.

12.Docker and images:

Describe how you packaged your application, database, and your dependencies into containers for consistent and efficient deployment across various environments. Provide appropriate screenshots.

13.Validation and sanitization:

Describe how and where you did validation and sanitization to ensure that user inputs for reviews are validated and sanitized to prevent SQL injection and other security issues.Include appropriate screenshots

1. **User Authentication:**
   Describe how you ensure that only authenticated users can submit, update, or delete reviews. Describe how user authentication is handled and how authorization is managed. Include appropriate screenshots.
2. **Moderation:**
   Describe how you Implemented moderation features to handle inappropriate reviews, which could be flagged by users or administrators. Include appropriate screenshots.
3. **Part II tasks:**
   Each student describes how he/she Implemented two (or more) additional features from the available list and what did you handle and how. Include appropriate screenshots.

References

- Include any references or resources used during the project even if it is AI and the purpose behind using it , I.e. where was is used, how, and why.

Submissions to Moodle:
You are supposed to upload to Moodle your full project folder and and database file zipped. **The word document of your report should be uploaded to Moodle on its own and not in the zipped folder. One student should submit the one report word document worked on in the team.**

Good luck!

Grading Rubric

Criteria	Points
Project Management & Organization	10
Service Development (Users, Rooms, Bookings, Reviews)	40
API Documentation & Testing	10
Error Handling & Validation	5
Docker Setup & Integration	5
Performance Profiling & Optimization	5
GitHub & Version Control	5
Part II Tasks (2 unique tasks per each team member)	20