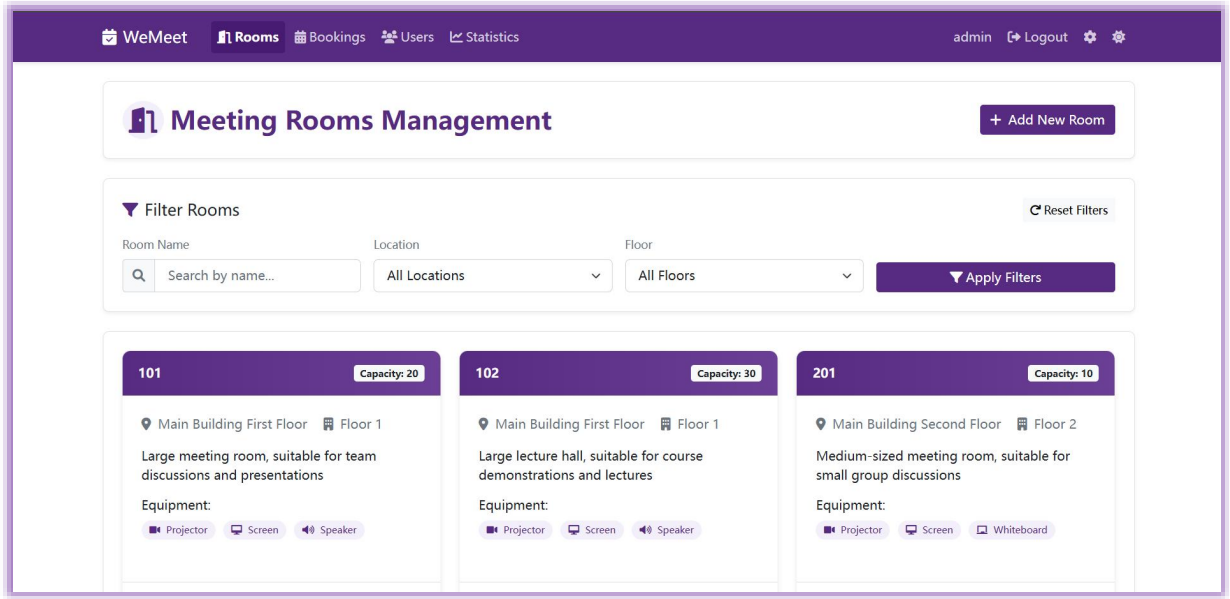
WeMeet

-- An Online Meeting Booking System



Group 50

URL: *http://118.178.57.227:8080*

Xi'an Jiaotong-Liverpool University
西交利物浦大学

2025/4/30

OVERVIEW

I. Introduction

II. Architectural Design

III. Software Design

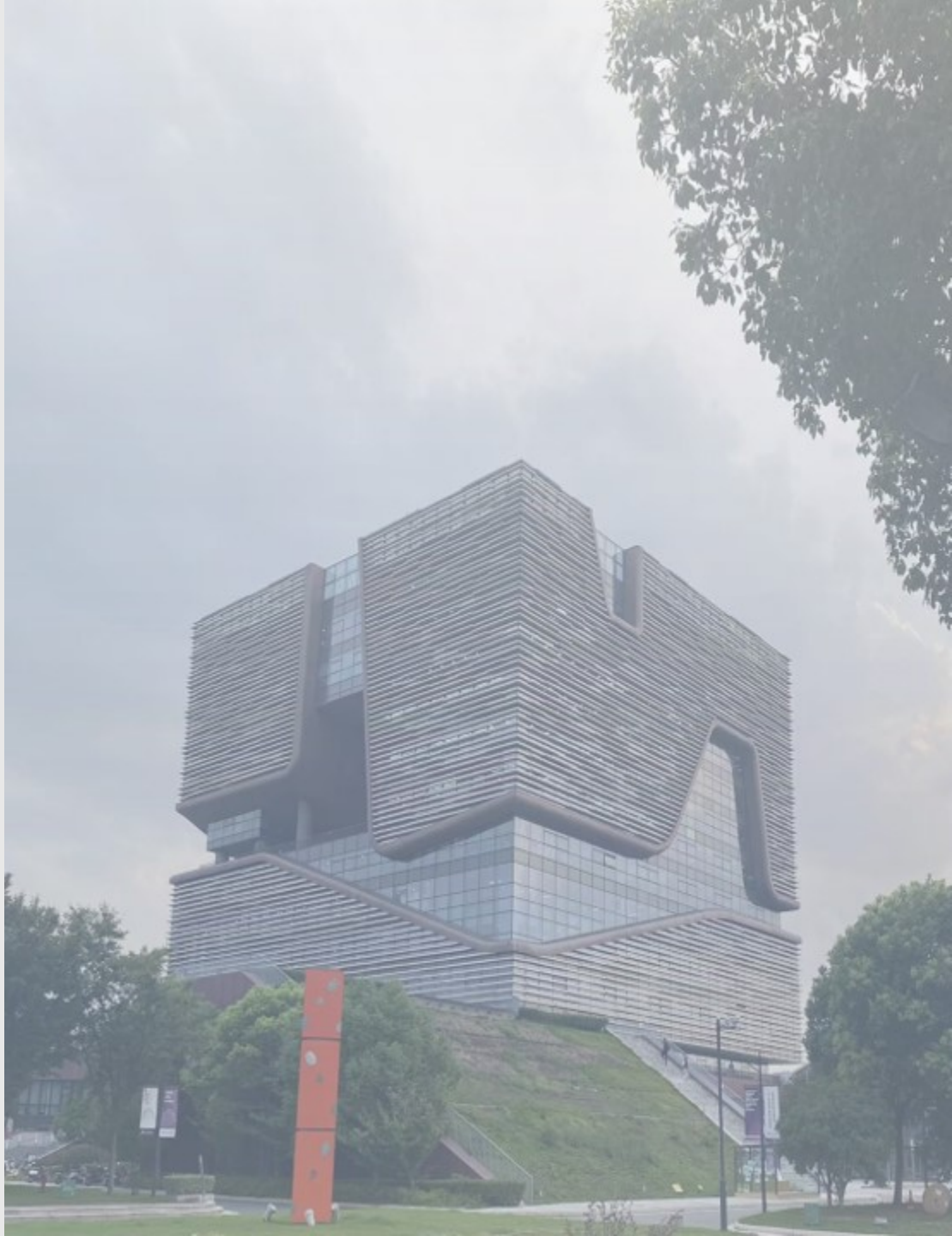
IV. Software Testing

V. Conclusion

Xi'an Jiaotong-Liverpool University
西交利物浦大学

PART I. Introduction

1. Problems & Improvements
2. User Characteristics
3. Assumptions & Dependencies
4. Project Scope
5. Project Risks & Mitigation



1.1 Problems and Improvements

Problem Statement

- Only Basic Login & Booking
- No Room Filtering or Search
- No Detailed Room Information
- Manual Administrative Process
- No Real-time Updates
- No Real-time Conflict Checking

Improvement

- Email Verification & User Profiles
- Filter by Capacity, Location, Equipment
- Photos, Amenities, Schedule Views
- Admin Dashboard with Analytics
- Live Availability & Booking Validation

Goal: Build a Flexible, Transparent, User-Centered System.

1.2 User Characteristics

User

- Room Search & Filters
- Availability Overview
- Booking & History Access
- Simple Interface & Secure Login

Admin

- Room Information Management
- Booking Logs & Dashboards
- User Control & Approvals
- Real-time Alerts & Monitoring

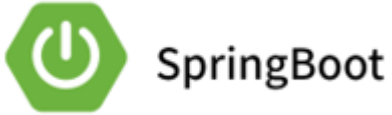
1.3 Assumptions and Dependencies

Assumptions

- Stable Internet & Desktop Access
- University Email Required
- Admin Room Management
- User Booking Compliance

Improvement

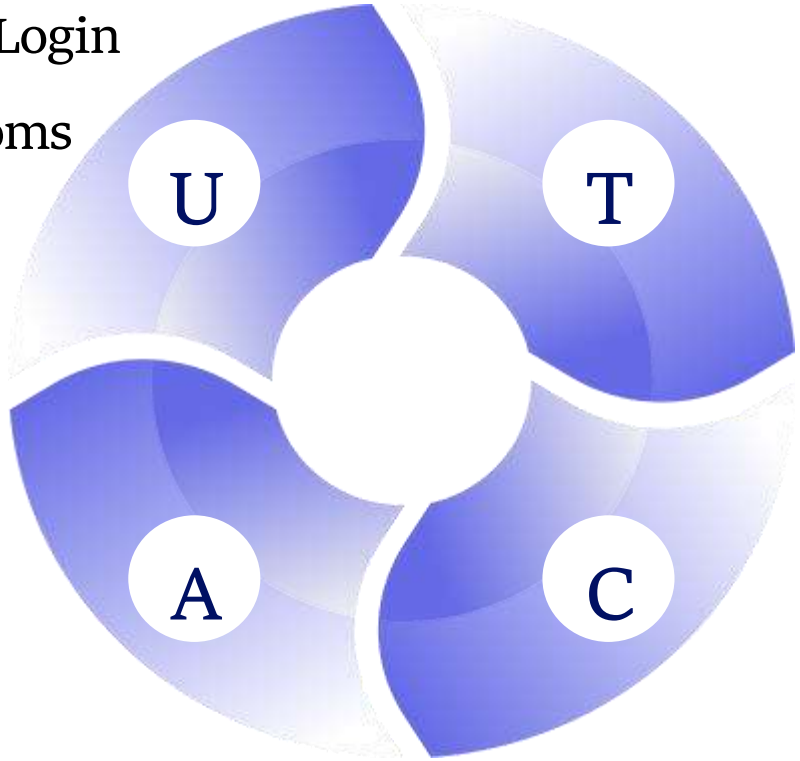
- Reliable Email Service
- *Docker* Containerization
- *Alibaba Cloud H*osting
- *Spring Boot*, *Thymeleaf* & *MySQL*
- *GitHub* for Version Control
- *Postman* for Testing



1.4 Project Scope

User Functions

- Register, Verify Email, Login
- Search/Filter/Book Rooms



Tech Stack

- *Spring Boot* + *Thymeleaf* + *MySQL*
- Deployed via *Docker* on *Alibaba Cloud*

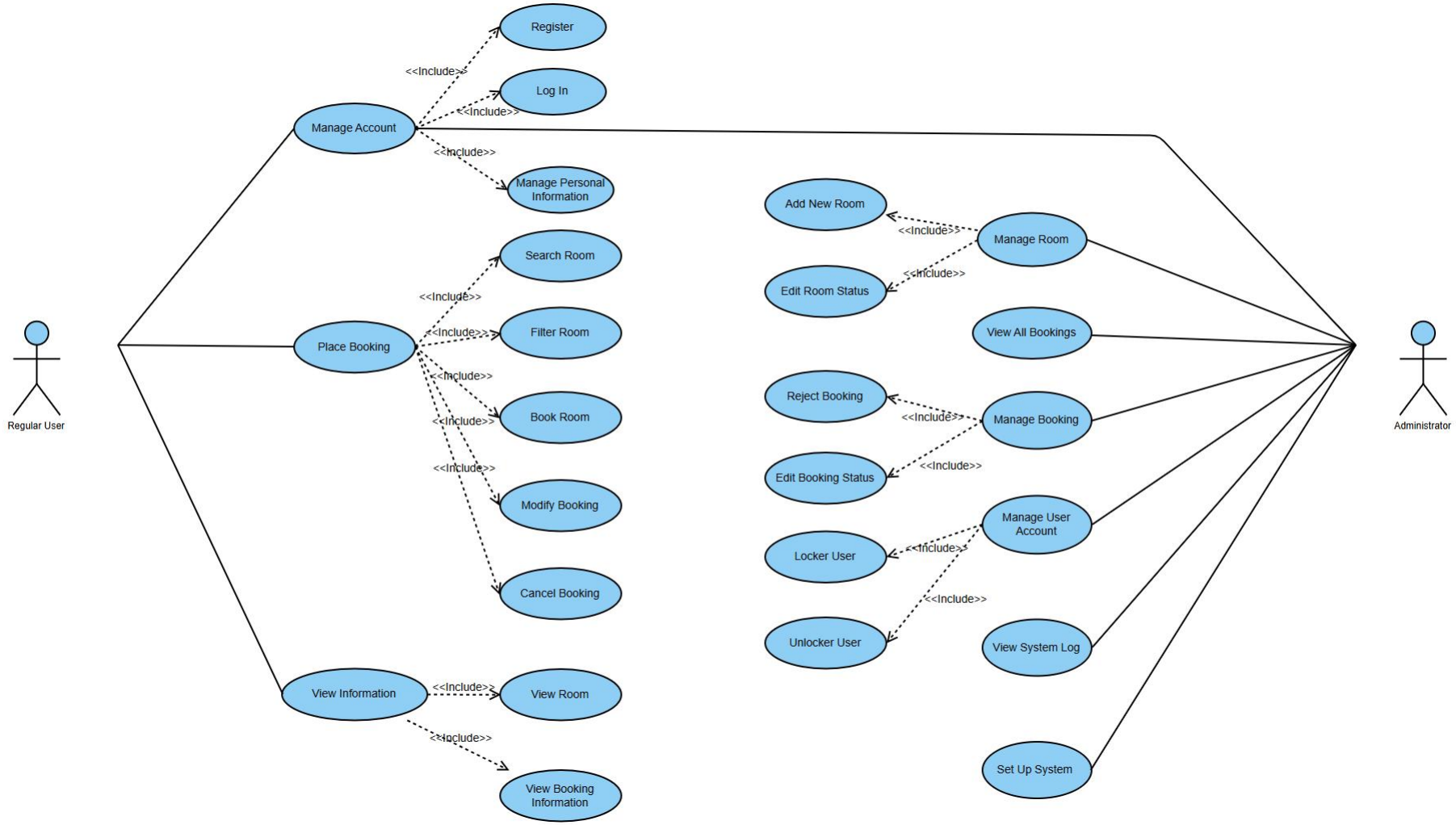
Admin Functions

- Manage Rooms
- Manage User Accounts
- Manage User Accounts

Current Limitations

- Only Desktop Browsers
- Use Demo Data Only

1.4 Project Scope



1.5 Project Risks and Mitigation

Risks

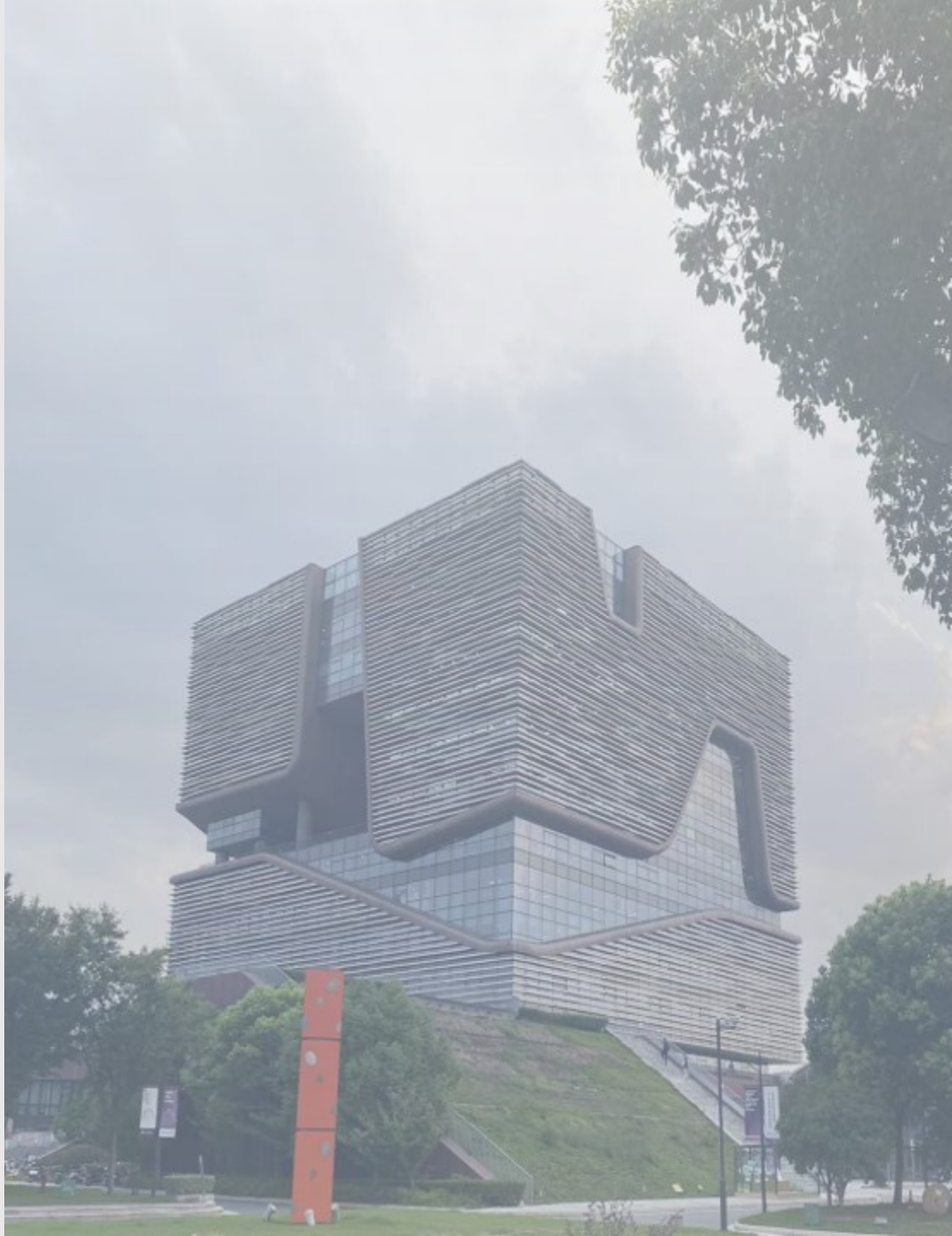
- Module Integration Issues
- Deployment Configuration Failures
- Team Coordination Delays
- Merge Conflicts

Mitigation

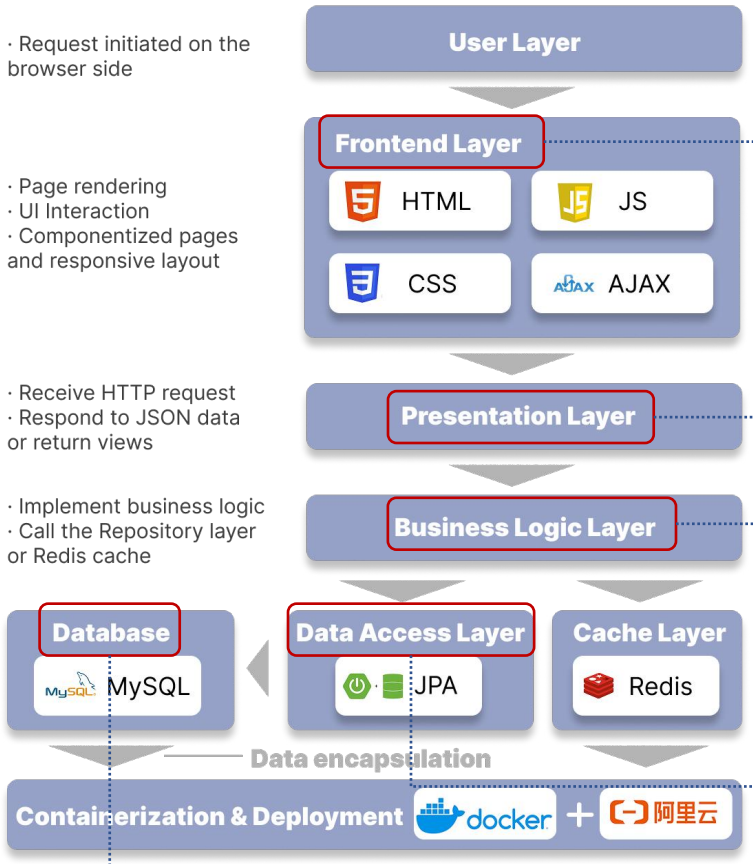
- Regular Meetings
- Clear Task Assignments
- Sprint Development
- *GitHub* Collaboration & Code Reviews

PART II. Architectural Design

1. System Architecture
2. Frontend System
3. Backend System
4. High-level Database Design



2.1 System Architecture



(1) Layered Architecture

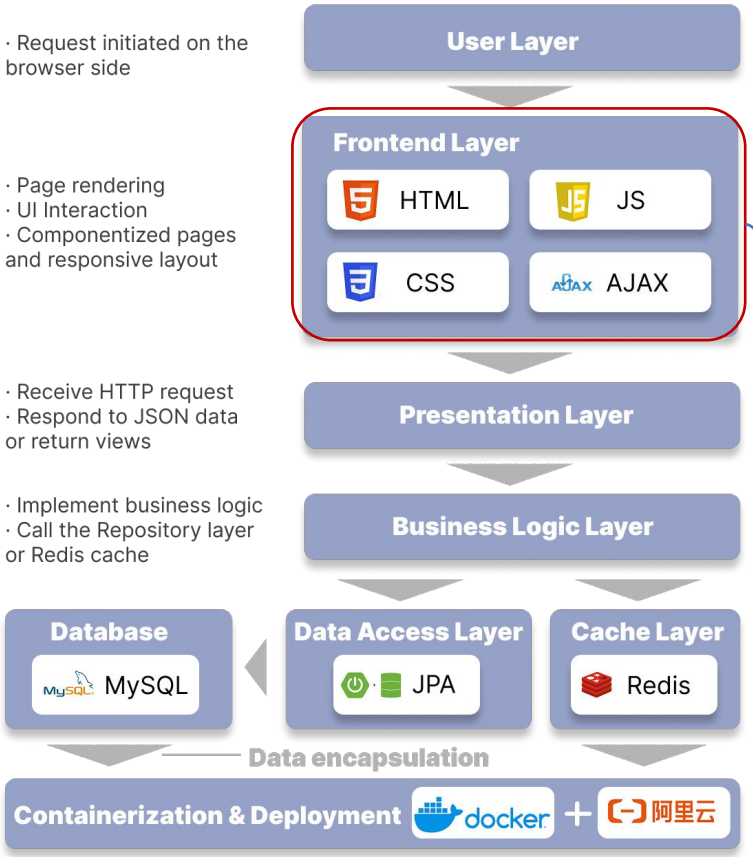
- Clear Separation of Concerns
- Easier Testing, Maintenance & Scalability

Architecture
Layered Architecture
Feature-Based Modularization

The code is organized using Feature-Based Modularization for better scalability & cooperation.

WeMeet system adopts a classic *Layered Architecture* combined with *Feature-Based Modularization* to achieve clear separation of concerns and high extensibility.

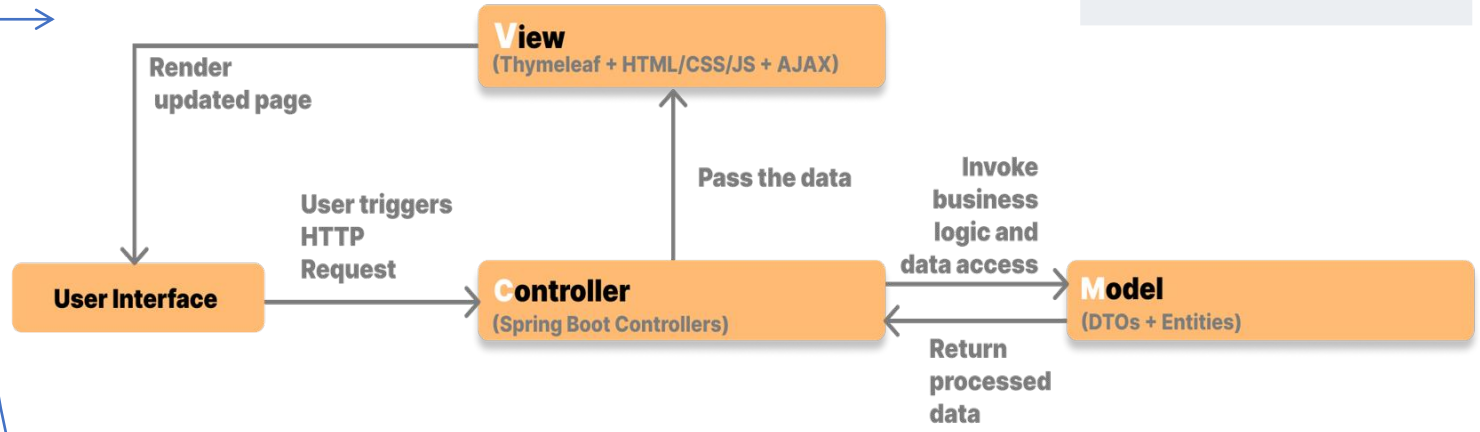
2.1 System Architecture



(2) MVC Architecture

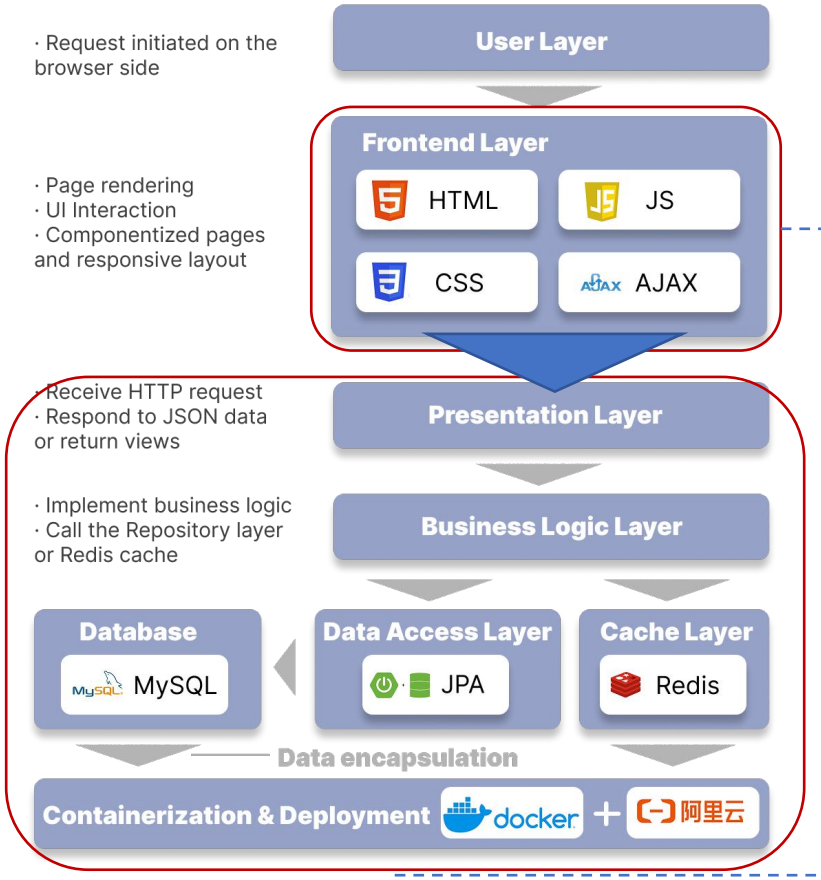
- Separation of UI, Logic & Data
- Simplified Code Management
- Flexible System Extension

Architecture
Layered Architecture
Feature-Based Modularization
MVC Architecture



In frontend layer, we used MVC Architecture.

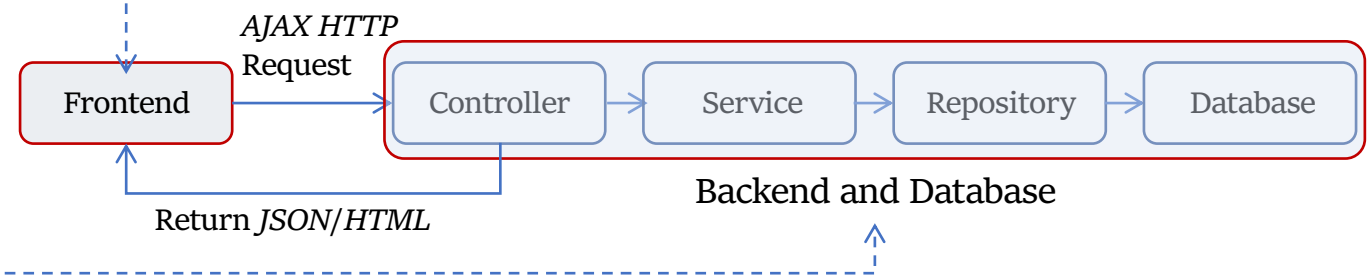
2.1 System Architecture



(3) B/S Architecture

- Browser-Based Access, No Client Installation
- Asynchronous Data Communication
- Seamless User Experience

Architecture
Layered Architecture
Feature-Based Modularization
MVC Architecture
B/S Architecture



We use B/S Architecture with *AJAX* to communicate frontend and backend.

2.2 Frontend System

Technology	Justification
HTML5 + CSS3	- Define Page Structure & Style - Ensure Cross-Browser Compatibility
JavaScript + jQuery	- Handle User Events & DOM Manipulation - Facilitate *AJAX* Calls
Bootstrap 5	- Implement Responsive Layout - Provide Ready-To-Use UI Components
Thymeleaf	- Server-Side Dynamic Page Rendering - Bind Backend Data into *HTML* Templates
AJAX (fetch API)	- Enable Asynchronous Data Requests - Update Parts of the Page without Full Reload

2.3 Backend System

Technology	Justification
Spring Boot	Simplified Development, Embedded Servers, Auto-configuration
Spring MVC	RESTful APIs & Web Endpoint Support
Spring Security + JWT	Stateless Authentication & Role-Based Authorization
Spring Data JPA + Hibernate	ORM & Database Abstraction
MySQL (Alibaba RDS)	Reliable Relational Database
Redis	Multi-Level Caching for Performance Improvement
Thymeleaf	Server-Side *HTML* Rendering
Maven	Project Lifecycle & Dependency Management

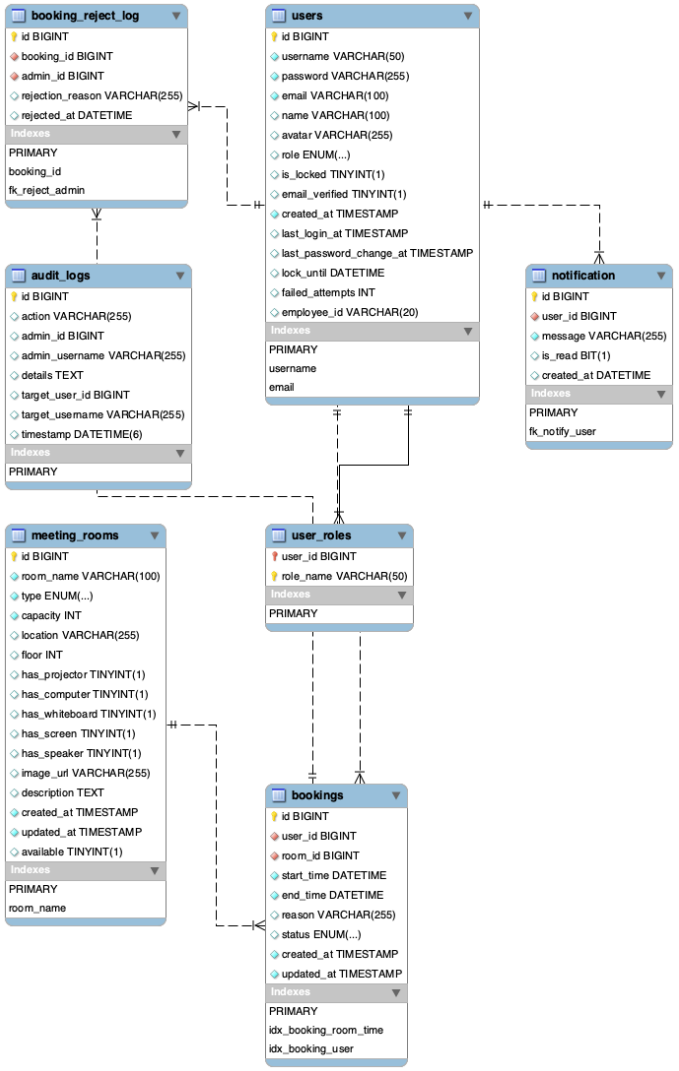
2.4 High-level Database Design

Core Entities

- User
- Meeting Room
- User Role
- Notification
- Booking Reject Log
- Audit Log

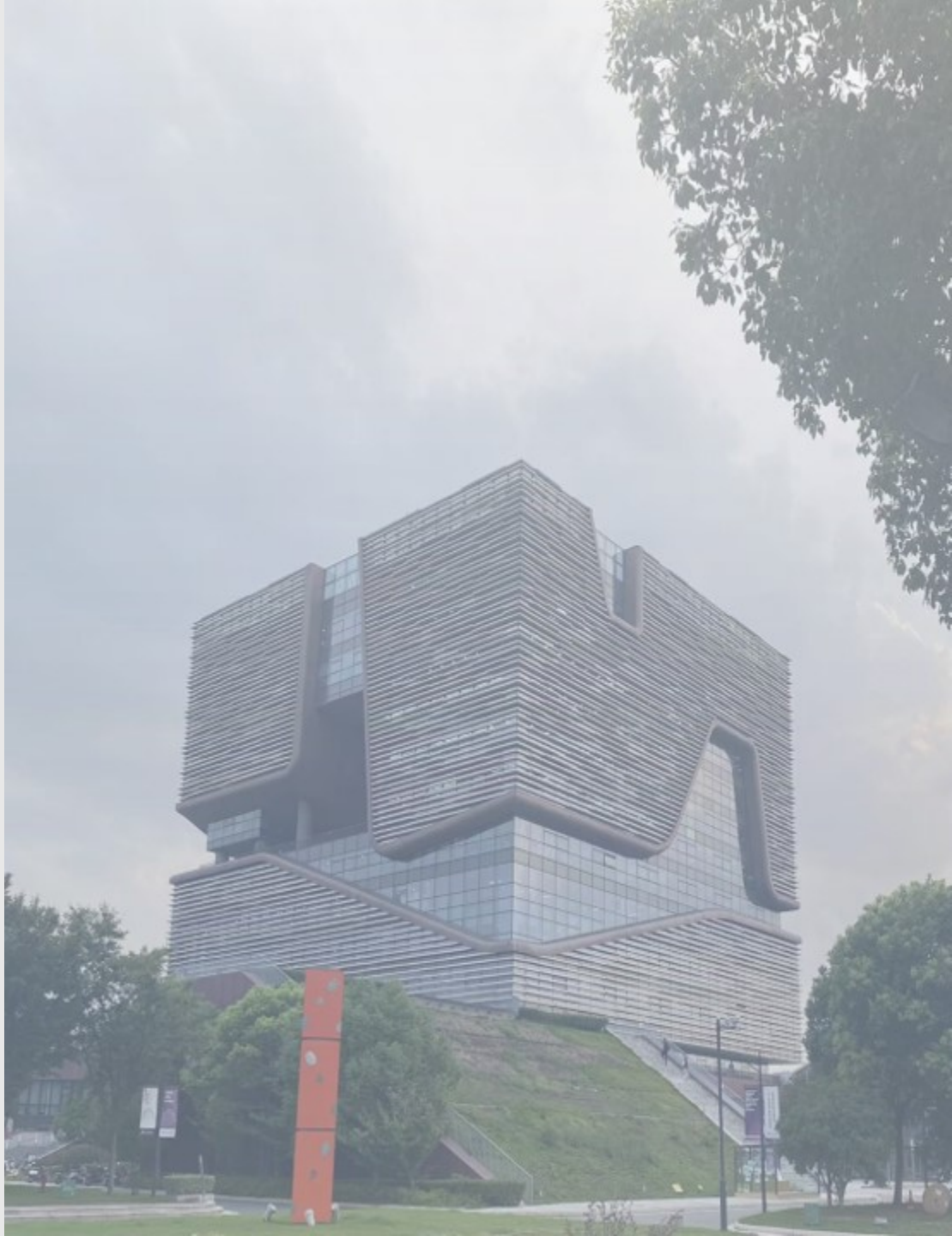
Design Highlights

- Clear Relationships, Scalable Structure
- Flexible Role-Based Access Control
- Optimized for Complex Queries & High Concurrency



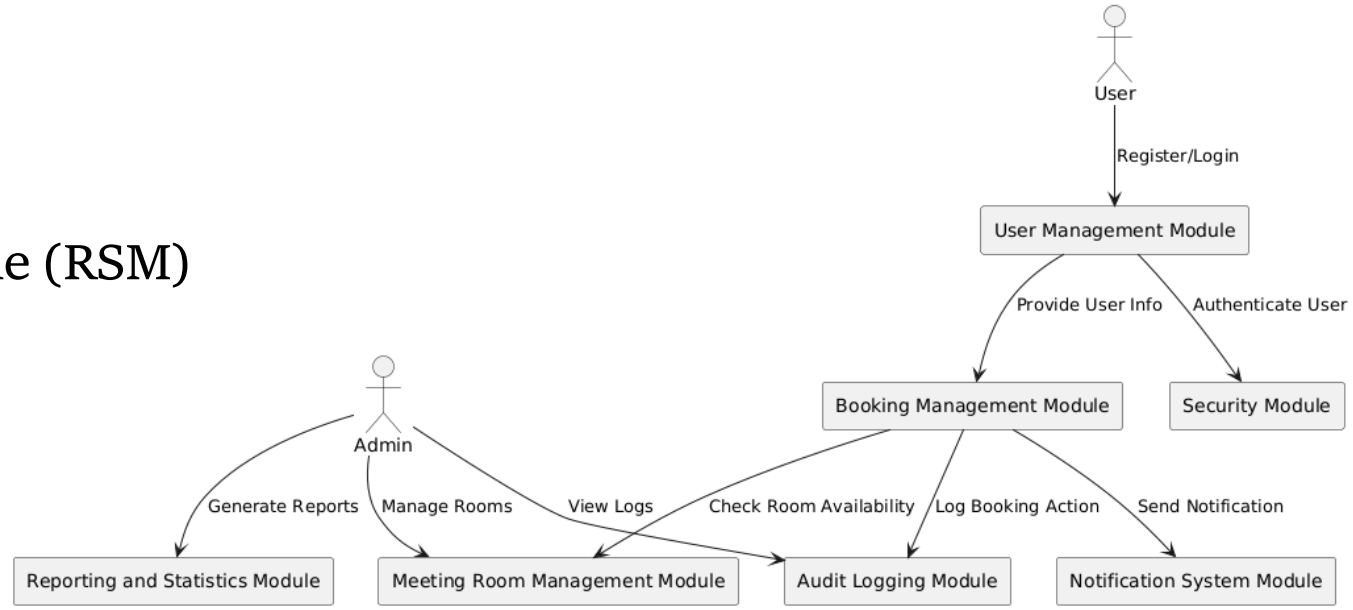
PART III. Software Design

1. Software Modules
2. High-level Process Flow
3. Software Support Services
4. Coding Structure & Convention
5. Software Configuration & Production Environment



3.1 Software Modules

1. User Management Module (UMM)
2. Meeting Room Management Module (MRMM)
3. Booking Management Module (BMM)
4. Notification System Module (NSM)
5. Audit Logging Module (ALM)
6. Security Module (SM)
7. Reporting and Statistics Module (RSM)



3.2 High-level Process Flow

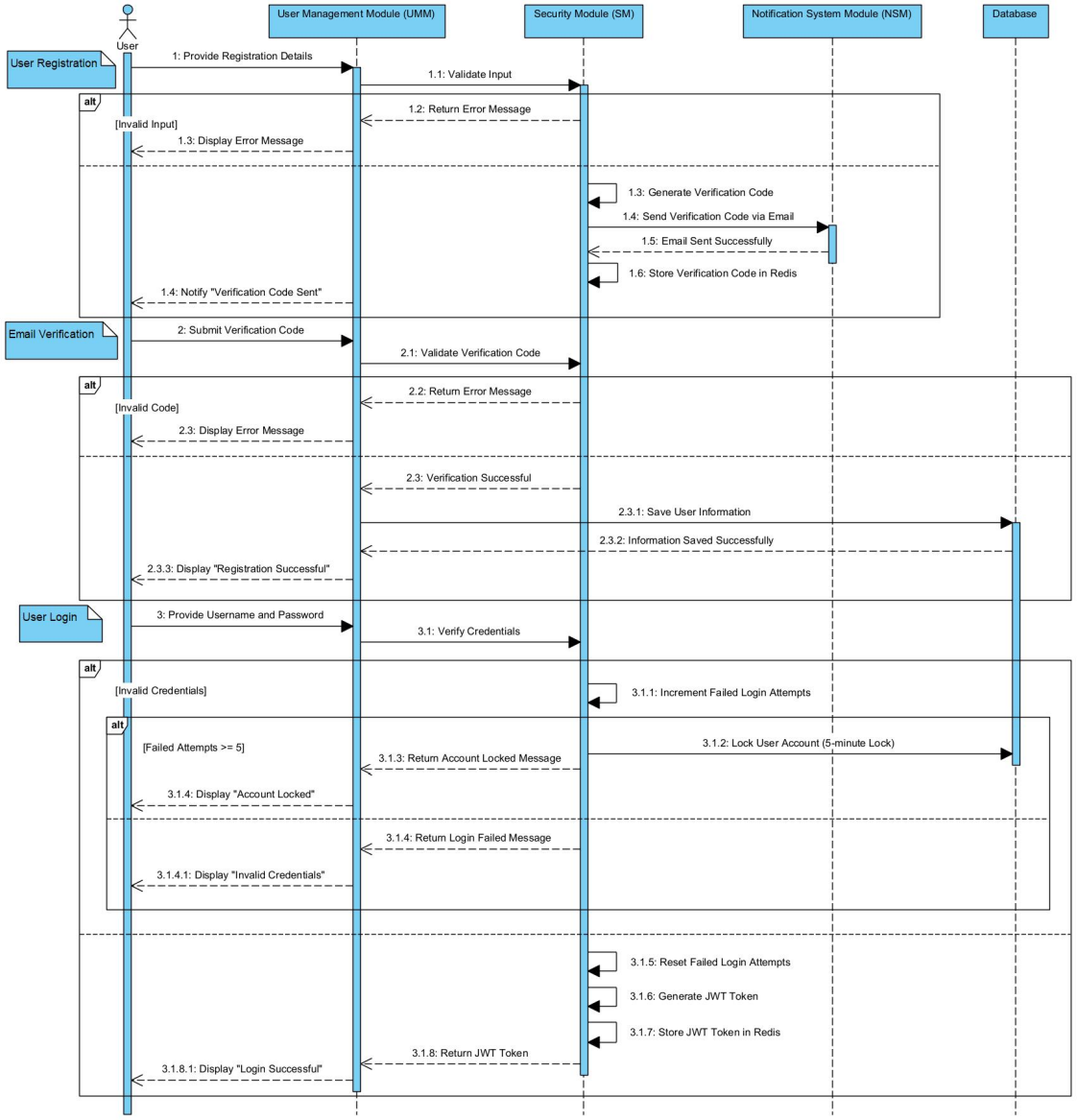
(1) User Functionality

User Registration and Login

- UMM
- SM
- NSM

Key Mechanisms

- BCrypt Password Encryption
- Verification Code Validation
- Login Failure Limits
- JWT-Based Session Management



3.2 High-level Process Flow

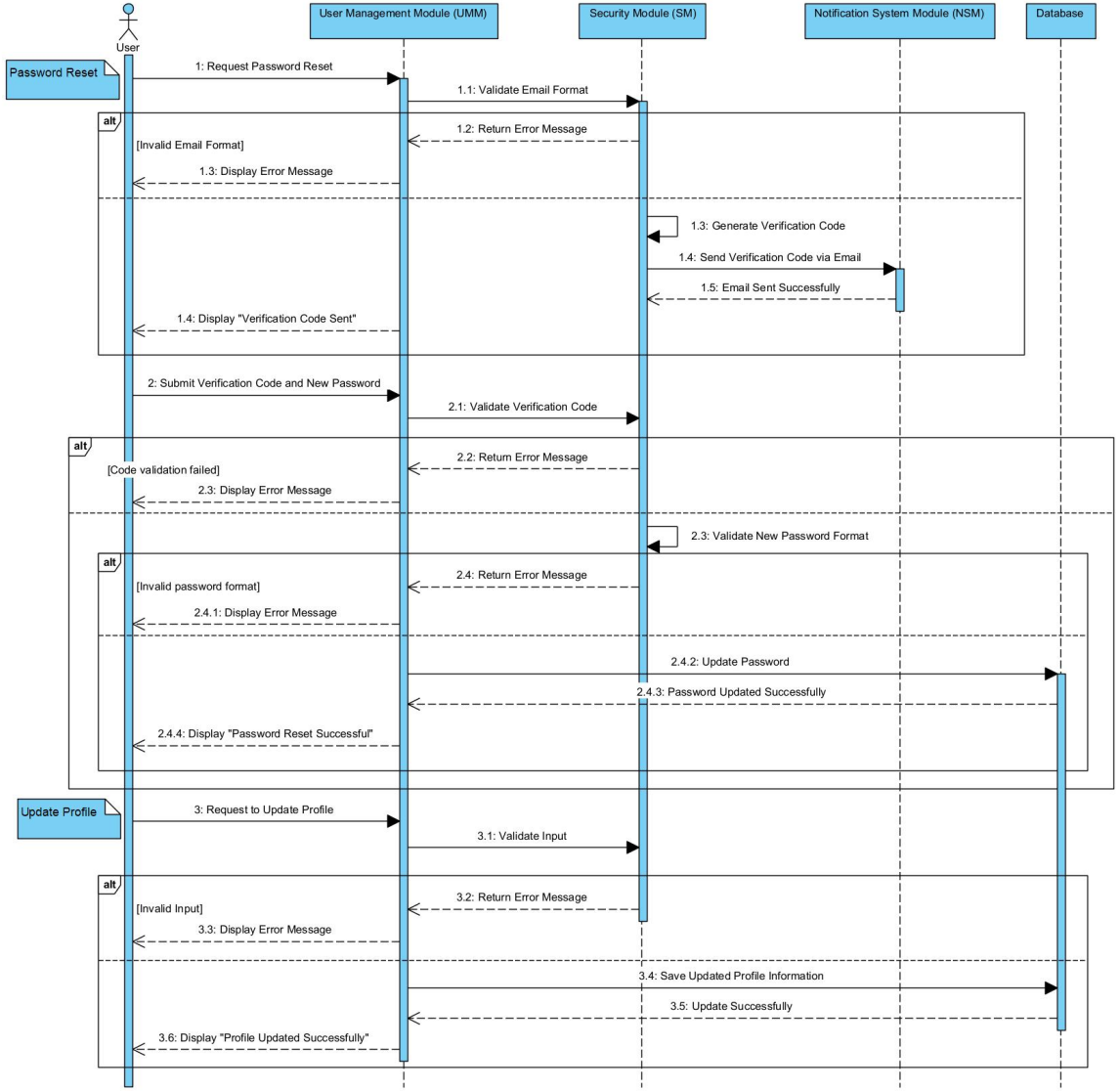
(1) User Functionality

Personal Profile Management

- UMM
- SM
- NSM

Key Mechanisms

- BCrypt Password Encryption
- Verification Code Validation
- Input Validation



3.2 High-level Process Flow

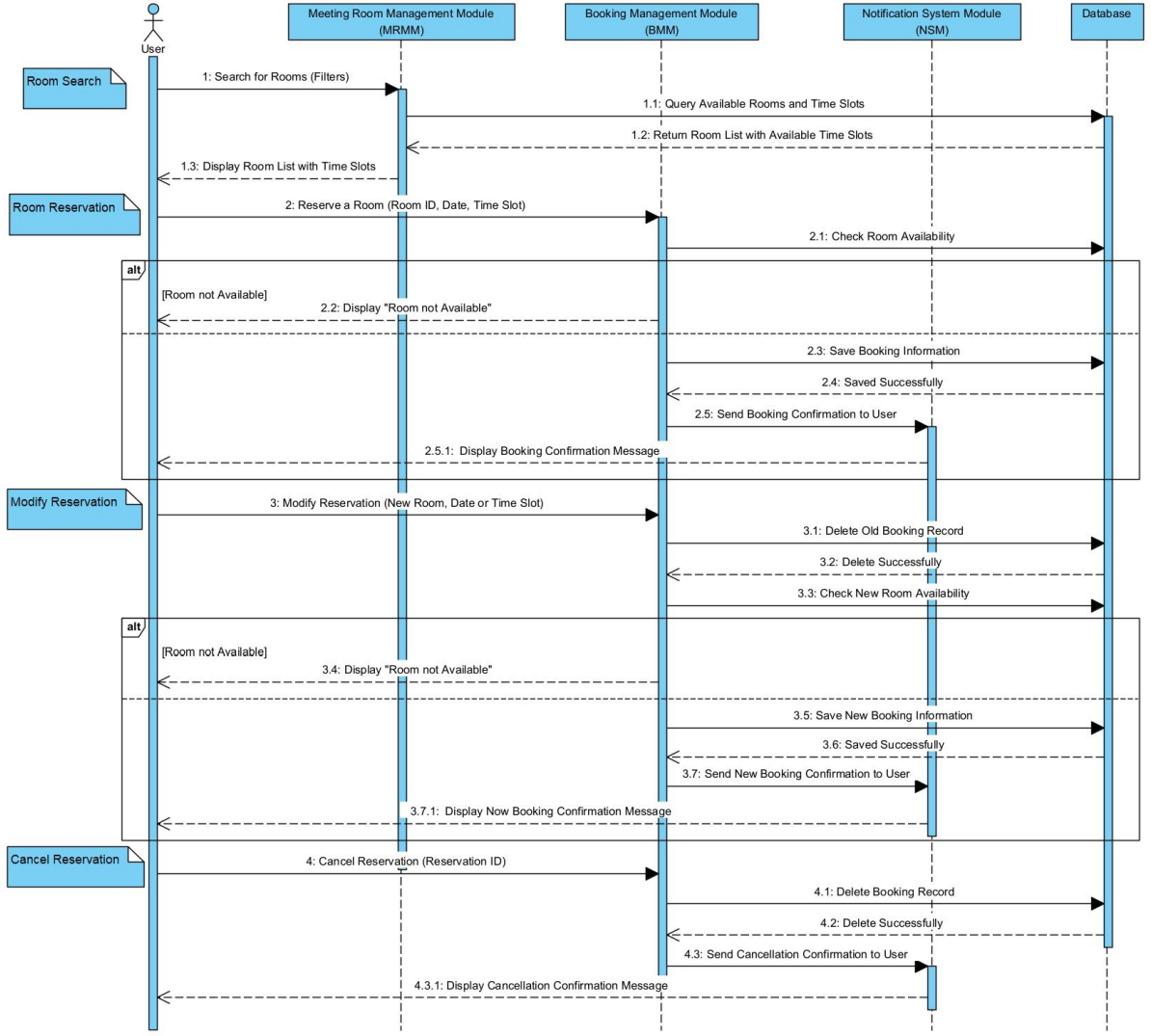
(1) User Functionality

Room Search and Reservation

- MRMM
- BMM
- NSM

Key Mechanisms

- Real-Time Availability Checks
- Conflict Resolution
- User Notifications



3.2 High-level Process Flow

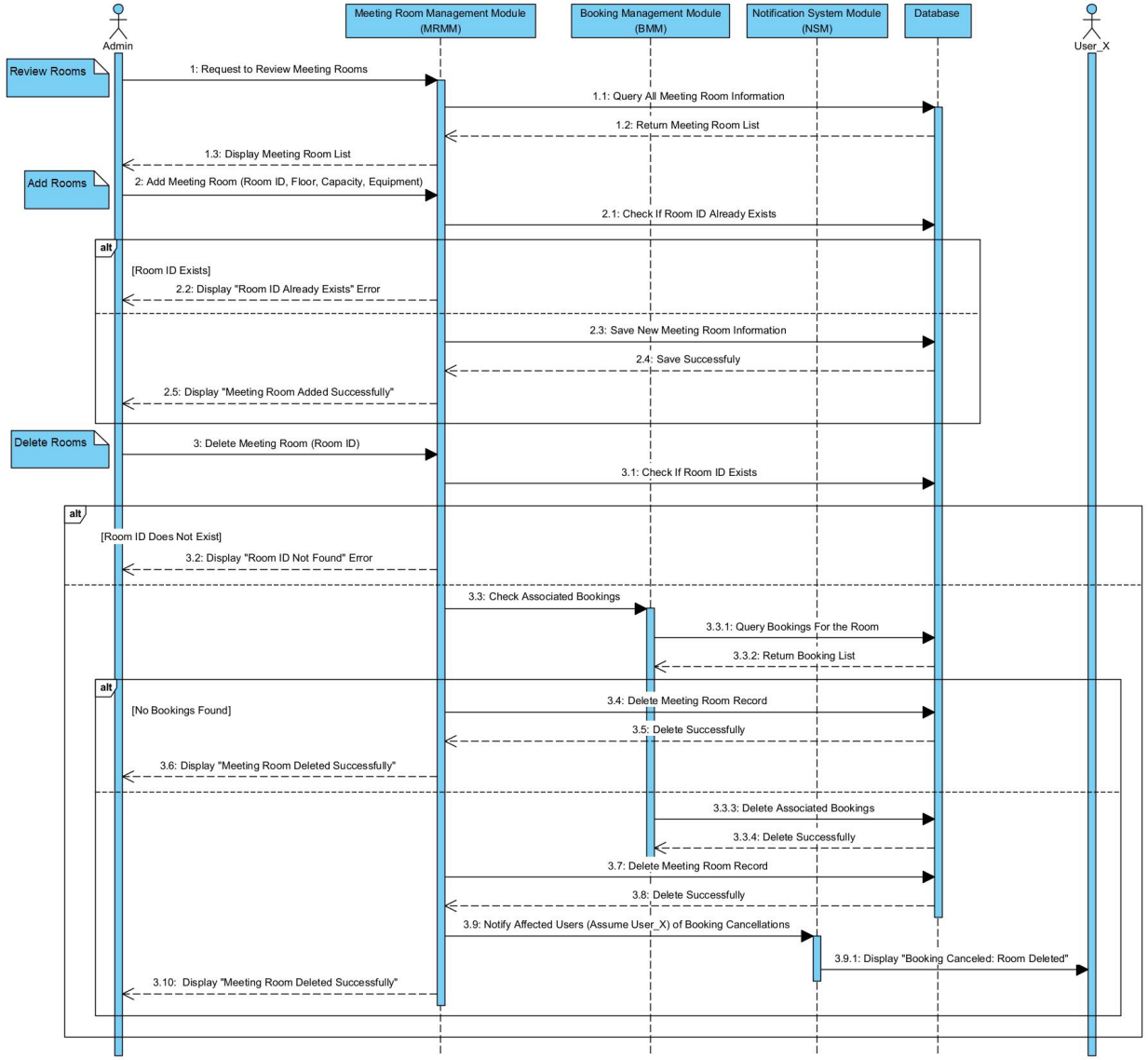
(2) Admin Functionality

Meeting Room Management

- MRMM
- BMM
- NSM

Key Mechanisms

- Input Validation
- Booking Checks
- User Notifications



3.2 High-level Process Flow

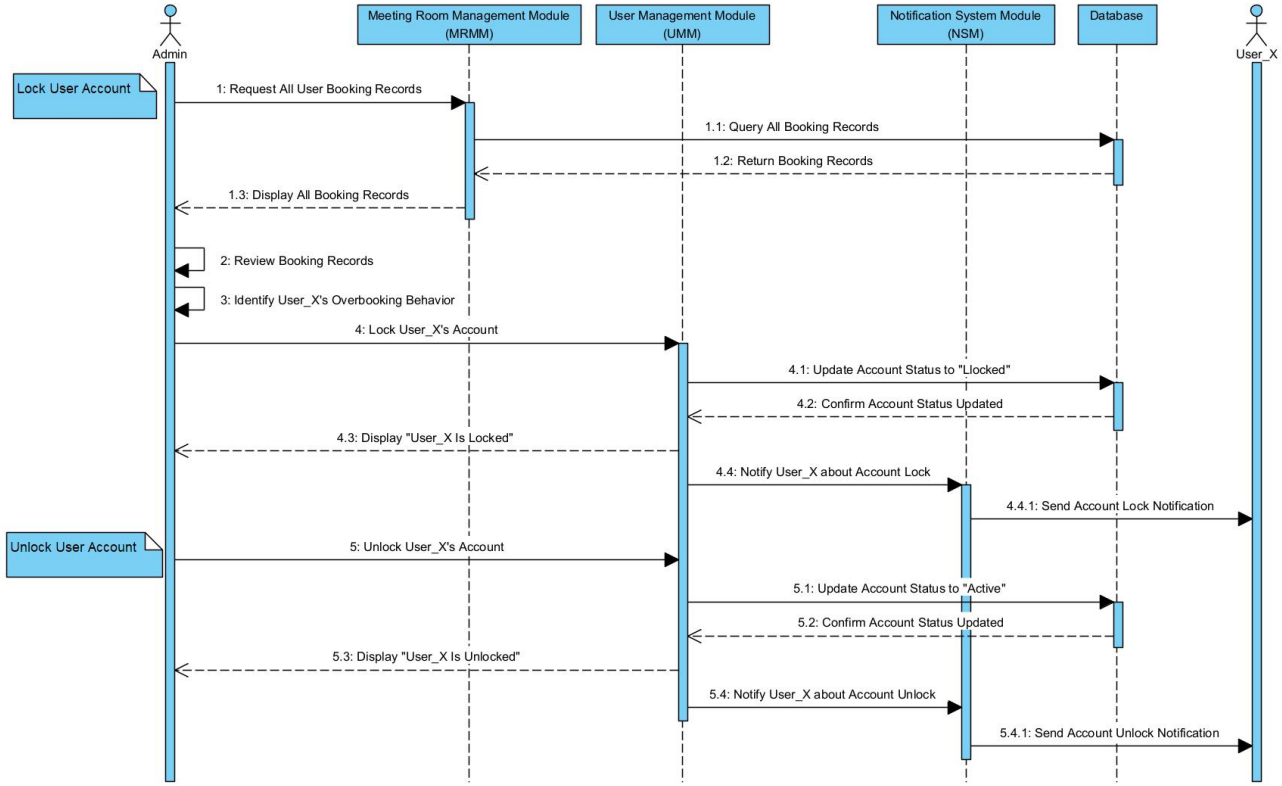
(2) Admin Functionality

User Account Management

- MRMM
- UMM
- NSM

Key Mechanisms

- Booking Data Analysis
- Account Status Management
- User Notifications



3.2 High-level Process Flow

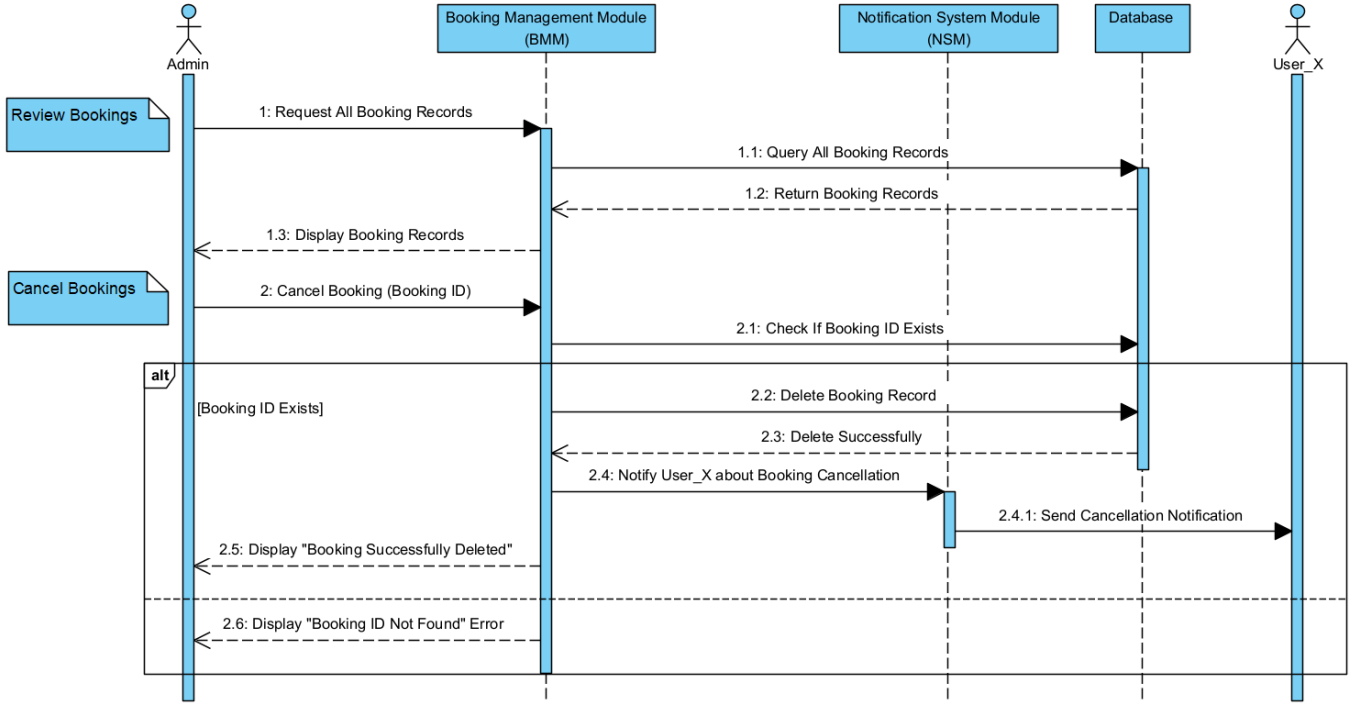
(2) Admin Functionality

Booking Management

- MRMM
- NSM

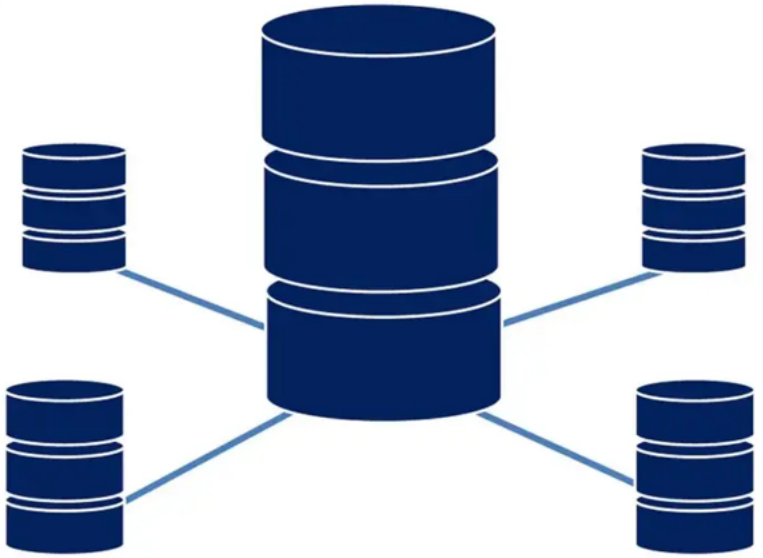
Key Mechanisms

- Booking Record Retrieval
- Booking Validation
- User Notifications



3.3 Software Support Services

(1) Database Services



Relational Database Integration

- *MySQL 8.0* for Storing Users, Rooms & Bookings
- *UTF-8 Encoding* to Support Multilingual Data
- *Spring Data JPA* for ORM Layer & Easy CRUD

Connection Pool Management

- *HikariCP* for Optimized High-Concurrency Access

Caching Infrastructure

- *Redis* for Captcha & Temporary Data Caching
- *ConcurrentHashMap* Fallback Mechanism

"Efficient data management is the foundation of system stability."

3.3 Software Support Services

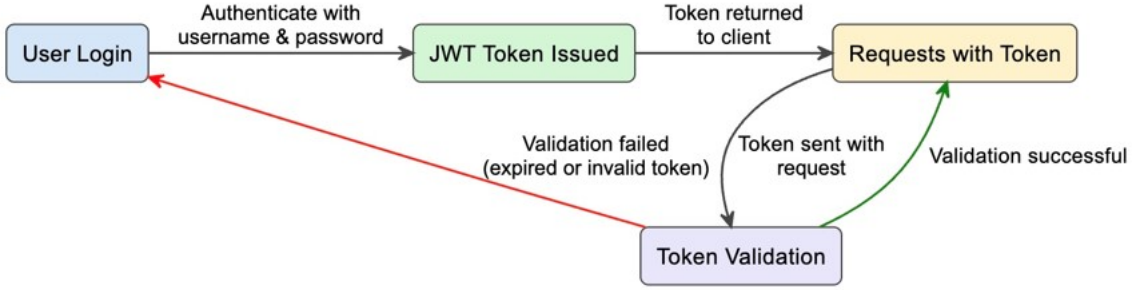
(2) Security Services

Authentication & Authorization

- *Spring Security* Role-Based Access Control
- *BCrypt* Password Hashing & Account Lockout

JWT-Based Stateless Authentication

- Tokens Stored in Headers or Cookies
- No Server-Side Session Storage Required



Security Auditing & Protection

- Audit Logs, XSS Prevention & File Validatio

"Security is not an option; it is the system's lifeline."

3.3 Software Support Services

(3) File Storage Services

User Avatar Management

- *Alibaba OSS* Stores Avatars with UUID Filenames
- Old Avatars are Deleted on Updates

Validation

- File Type (jpg/jpeg/png/bmp) & Size Restrictions (10MB Maximun)

"Efficient and orderly file management ensures a seamless user experience."

3.3 Software Support Services

(4) Notification and Messaging Services

Email Notification Service

- *Spring Mail* Sends *HTML* Verification Emails
- *Redis* Integration Accelerates Captcha Verification

Internal Notification System

- Users Notified of Booking Status Changes
- Visual Distinction Between Read & Unread Messages

"Every timely notification builds user trust."

3.3 Software Support Services

(5) Development Tools and Services

Version Control

- Git for Collaboration, Pull Request Workflow

Build & Dependency Management

- Maven Standardizes Builds & Dependency Resolution

Containerization

- Docker & Docker Compose Accelerate Environment Setup

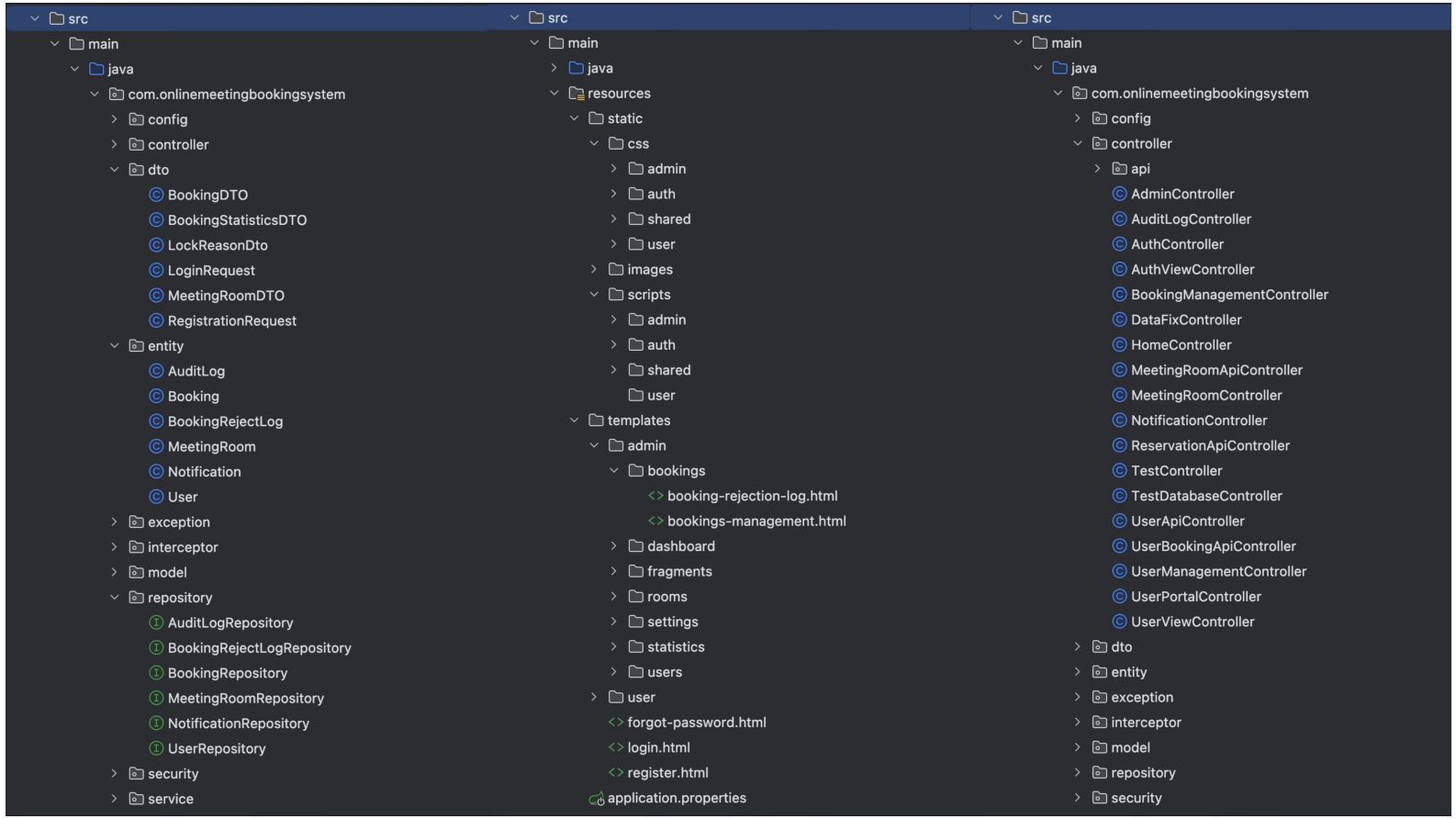
Developer Productivity Tools

- Lombok Reduces Boilerplate
- Spring Boot DevTools Enables Hot Reloads

"Efficient development and standardized management empower continuous product evolution."

3.4 Coding Structure & Convention

(1) Coding Structure: MVC Model



3.4 Coding Structure & Convention

(2) Code Convention

```java
public class AuthViewController {
    /**
     * Forgot password page
     */
    @GetMapping("/forgot-password")
    public String forgotPassword() { return "forgot-password"; }

    /**
     * Provide an automatic login function without password for convenience of testing
     * @param username username
     * @param isAdmin whether is admin or not
     * @return redirect to the dashboard
     */
    @GetMapping("/autoLogin")
    public String autoLogin(@RequestParam(defaultValue = "admin") String username,
                            @RequestParam(defaultValue = "false") boolean isAdmin,
                            RedirectAttributes redirectAttributes) {
        try {
            // try to search for existing users
            Optional<User> userOpt = userService.findByUsername(username);
            User user;

            if (userOpt.isPresent()) {
                user = userOpt.get();
            } else {
                // if user is not exist, create a virtual user
                user = new User();
                user.setUsername(username);
                user.setEmail(username + "@example.com");

                if (isAdmin || "admin".equals(username)) {
                    user.setRole(User.Role.Admin);
```

Overall Code Layout

```java
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import jakarta.persistence.*;
import java.time.LocalDateTime;

@Entity
@Table(name = "notification")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Notification {

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

    @Column(name = "user_id", nullable = false)
    private Long userId;

    @Column(name = "message", nullable = false, length = 255)
    private String message;

    @Column(name = "is_read")
    private boolean isRead = false;

    @Column(name = "created_at")
    private LocalDateTime createdAt = LocalDateTime.now();

    public void setIsRead(boolean isRead) { this.isRead = isRead; }
}
```

OOP Principle

3.5 Software Configuration & Production Environment

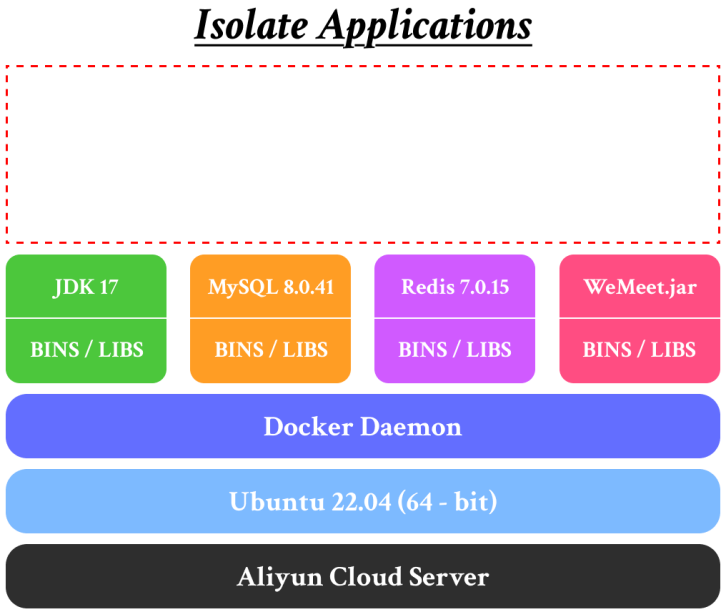
Frontend: *HTML*, *CSS*, and *JavaScript*

Backend: *Java* and *Spring Boot*

Database: *MySQL* and *Redis*

Table 1: Alibaba Cloud Server Configuration

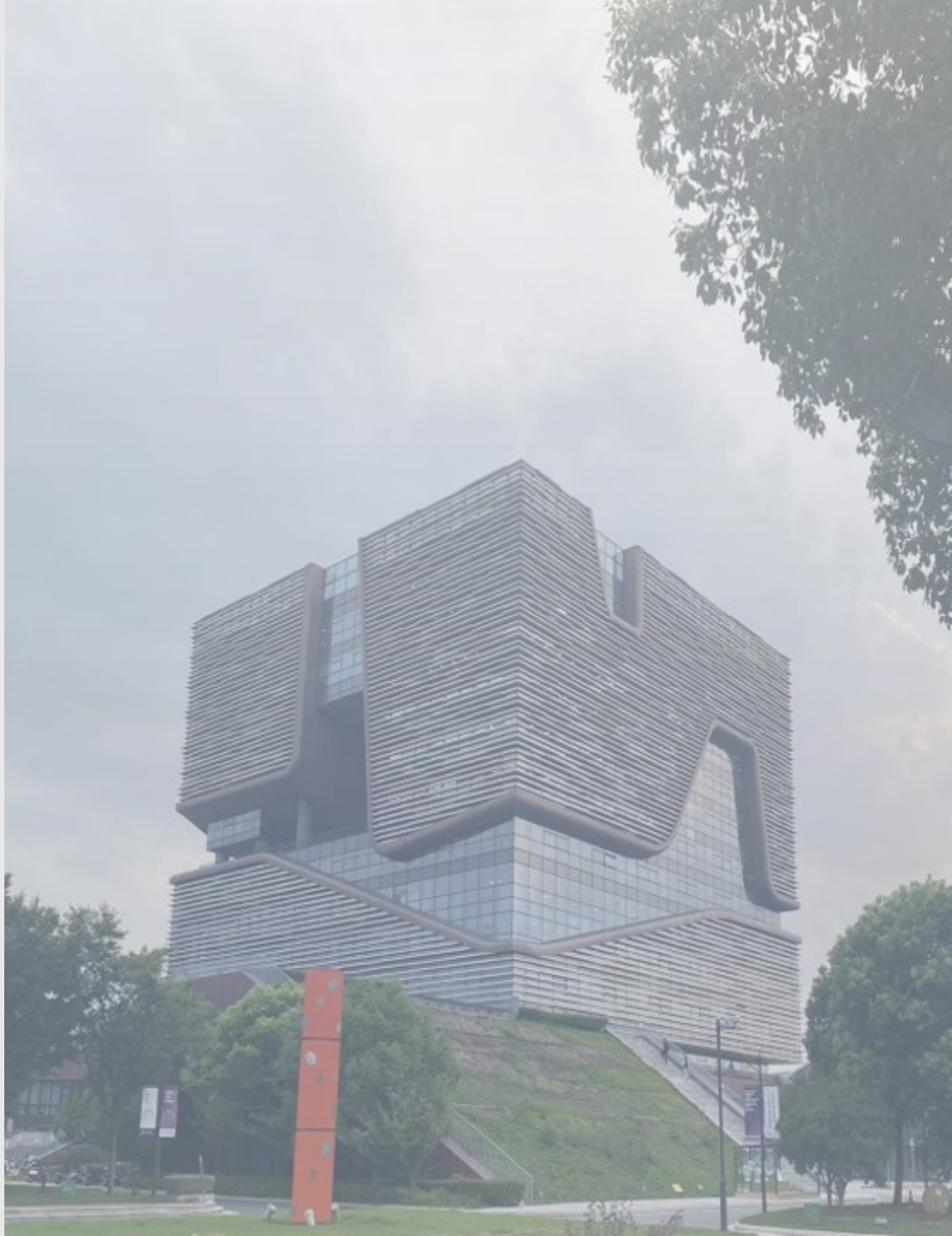
Operating System	Ubuntu 22.04 (64-bit)
CPU & Memory	4 core (vCPU) & 8 GiB
Public Network Bandwidth	5Mbps (Max)



Docker Container

PART IV. Software Testing

1. Unit Testing
2. Integration Testing
3. Acceptance Testing



4.1 Unit Testing

- Unit Testing for Functional Correctness
- Multi-Level Testing Strategy
- 542 Unit Tests with *JUnit 5* & *Mockito*
- Coverage of 8 Major Functional Areas

```java
@Test
void testLockUser_Success() {
    // Arrange
    String username = "testuser";
    User user = new User();
    user.setUsername(username);
    user.setAccountNonLocked(true); // Initially unlocked

    // Mock dependencies using Mockito
    when(userMapper.findByUsername(username)).thenReturn(
     user);
    when(userMapper.updateUserLockStatus(username, false)
     ).thenReturn(1);

    // Act: Call the method under test
    adminService.lockUser(username);

    // Assert: Verify outcomes and interactions
    assertFalse(user.isAccountNonLocked()); // Check user
      object state
    verify(userMapper, times(1)).findByUsername(username)
     ;
    verify(userMapper, times(1)).updateUserLockStatus(
     username, false);
}
```

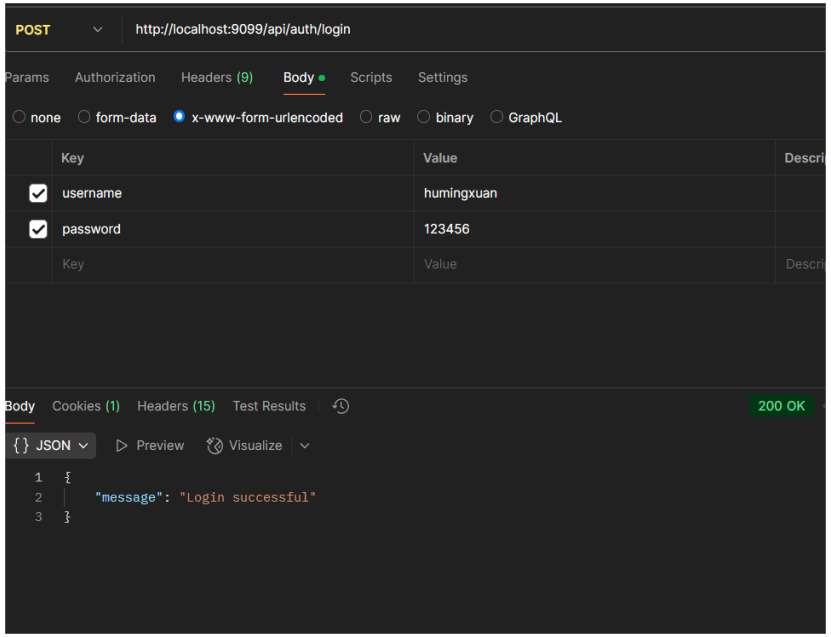
Test 1: Lock a User

```java
@Test
void testUnlockUser_Success() {
    // Arrange
    String username = "testuser";
    User user = new User();
    user.setUsername(username);
    user.setAccountNonLocked(false); // Initially locked

    // Mock dependencies
    when(userMapper.findByUsername(username)).thenReturn(
     user);
    when(userMapper.updateUserLockStatus(username, true))
     .thenReturn(1);
    // Act
    adminService.unlockUser(username);

    // Assert
    assertTrue(user.isAccountNonLocked()); // Check user
     object state
    verify(userMapper, times(1)).findByUsername(username)
     ;
    verify(userMapper, times(1)).updateUserLockStatus(
     username, true);
}
```

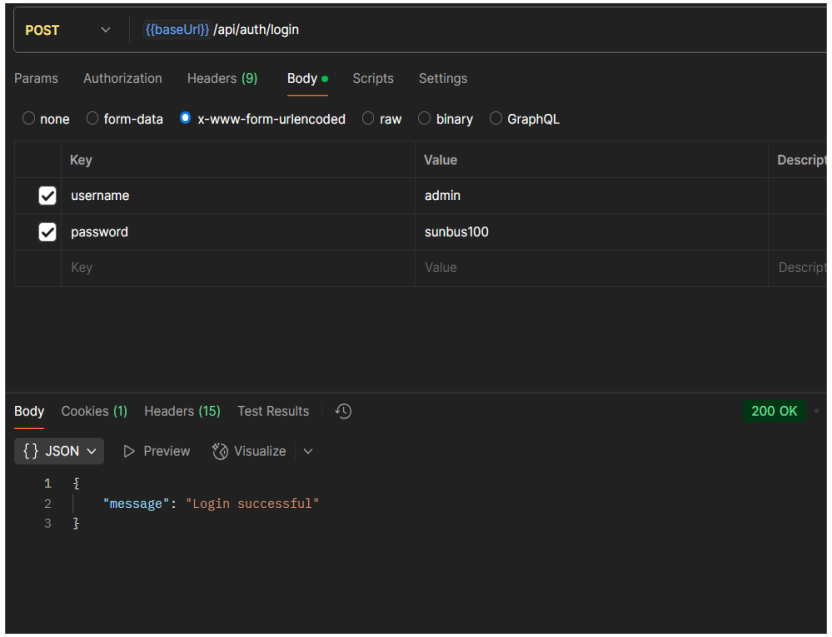
Test 2: Unlock a User

4.2 Integration Testing

- Integration Testing for System Interactions
- Guided by PBIs
- API Endpoint Testing with *Postman*



Test 1: User Login



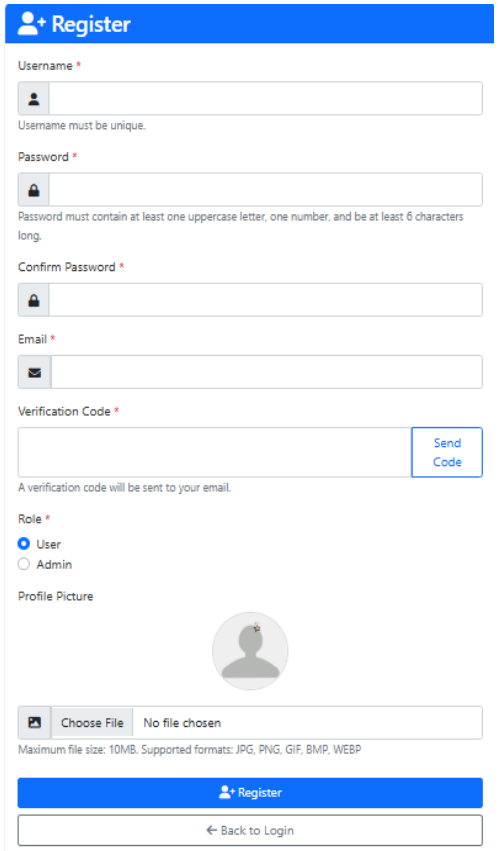
Test 2: Admin Login

4.3 Acceptance Testing

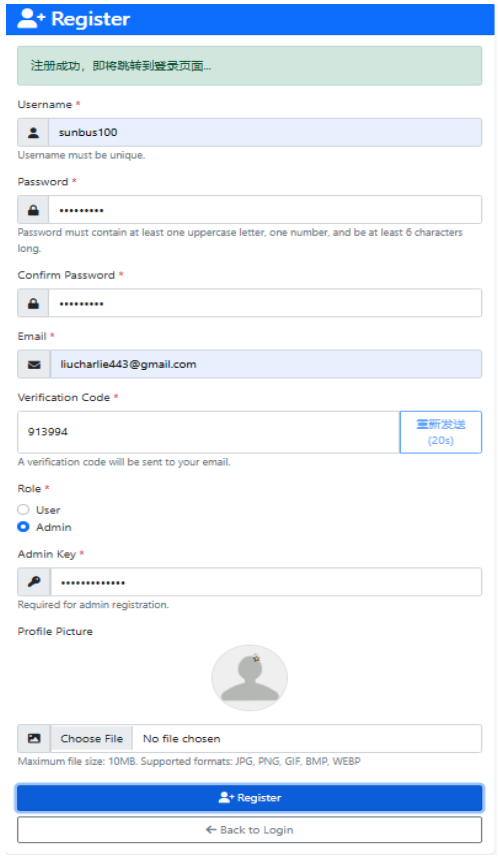
Functional Area	Key Scenario Tested	Status
User Lifecycle	Registration, Verification, Login/Logout & Profile Updates	Pass
Room Discovery	Listing Rooms, Viewing Details, Aearching & Filtering	Pass
User Booking Management	Creating, Viewing, Canceling & Updating Own Bookings	Pass
Administrator Capabilities	Managing Rooms, Managing Users & Managing Bookings	Pass
System Feedback & Usability	Verifying Notifications & Overall Usability	Pass
Security Basics	Verifying Role-Based Access Control	Pass

4.2 Acceptance Testing

- This example demonstrates validating user enrollment requirements in PBI through simulation of interface actions.



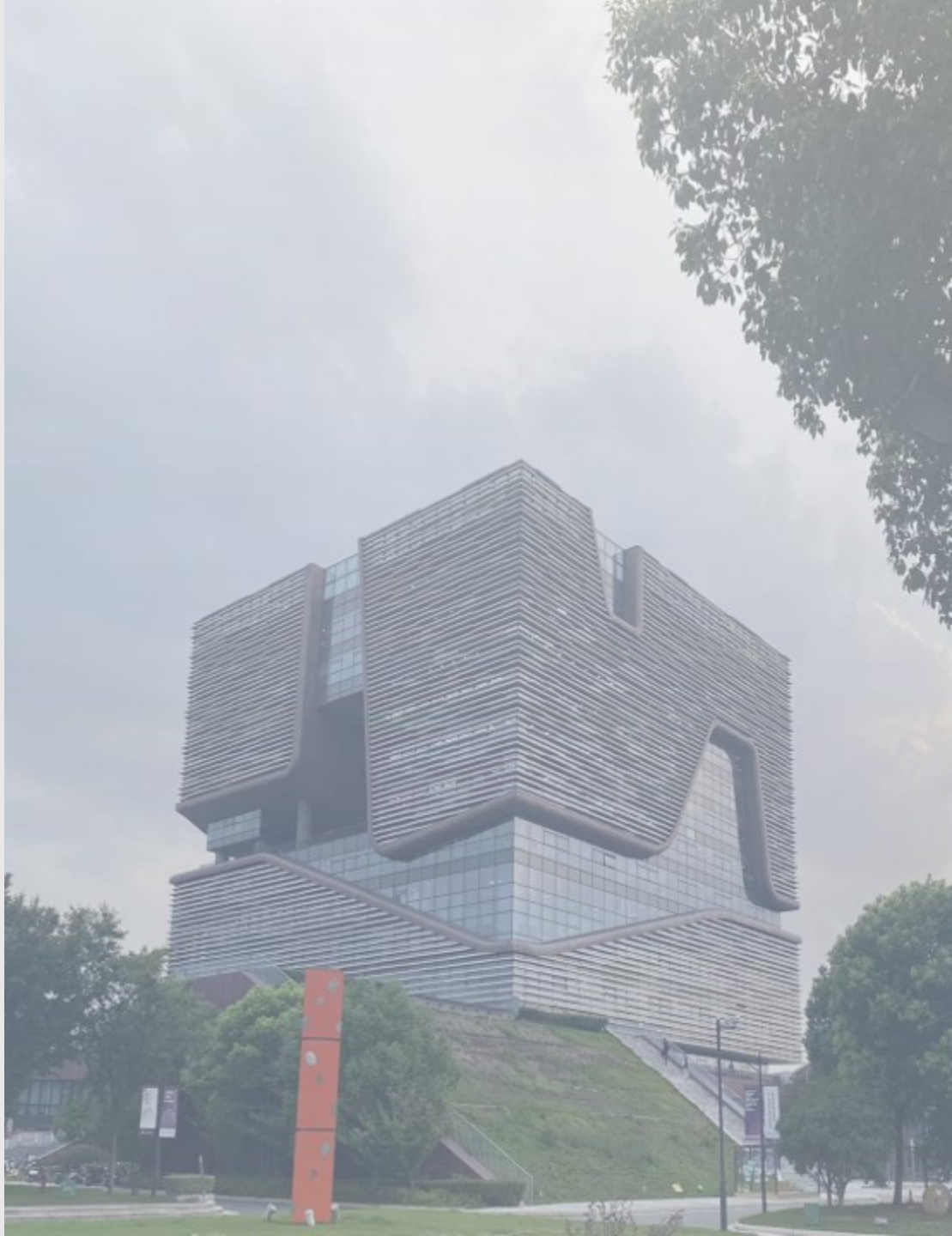
UI 1: Registration



UI 2: Validation

PART V. Conclusion

1. Achievements
2. Limitations
3. Future Work



5.1 Achievements

- Mobile Responsiveness & Accessibility
- Real-Time Database Integration
- Enhanced Room Usage Analytics
- Calendar Sync, Notifications & Conflict Detection
- UI/UX Usability Testing

5.2 Limitations

- Desktop-Only Access, No Mobile Compatibility
- Demo Data Usage, No Live University Data
- Lack of Accessibility, No Screen Reader Support
- Deployment Challenges: Docker Configuration, Email Setup

5.3 Future Work

- Mobile Responsiveness & Accessibility
- Real-Time University Room Database Integration
- Enhanced Analytics for Room Usage Insights
- New Features: Calendar Sync, Notifications, Conflict Detection
- Formal Usability Testing for UI/UX Optimization

SUMMARY

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Please Feel Free To Ask Your Questions

THANK YOU

Xi'an Jiaotong-Liverpool University

西交利物浦大学