Wee Kim Wee School of Communication and Information

IN6225 Enterprise Application Development

2024-2025 (Semester 2)

Individual Assignment Report

Student Counselling Appointment System

Submitted By:

[redacted]

Introduction

This report presents the complete development lifecycle and architectural details of the Student Counselling Appointment System, developed as part of the IN6225 Enterprise Application Development module. The project was designed to allow students to conveniently schedule appointments with university counsellors. It features secure login, appointment management, calendar integration, and real-time availability checks. The project follows a layered architecture using modern frontend and backend frameworks, with a microservices-ready structure.

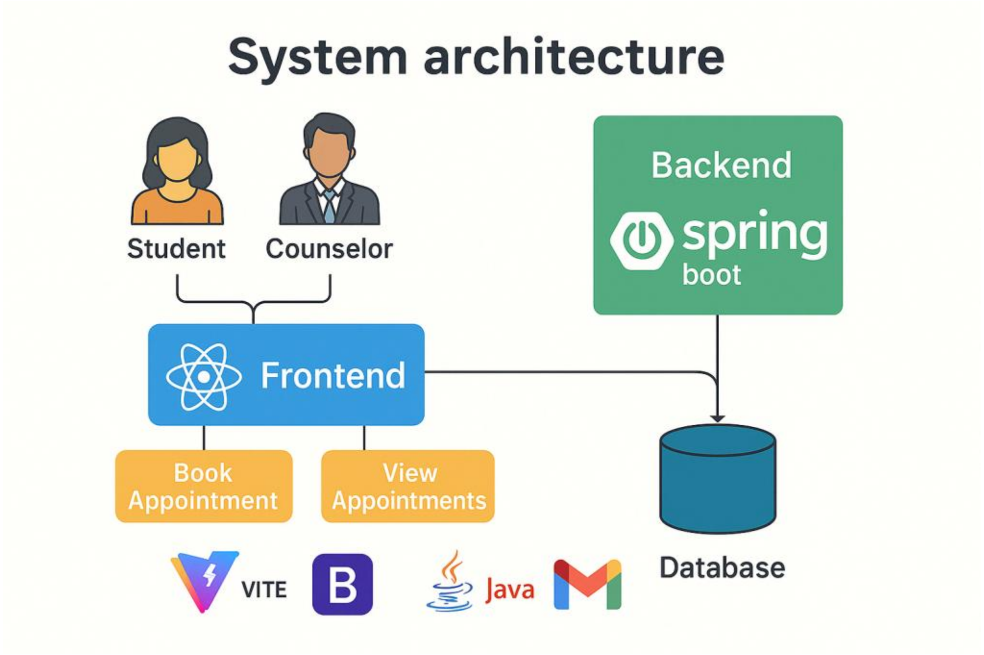
Project Objectives and Scope

The primary objective of the Student Counselling Appointment System is to create an efficient, user-friendly digital platform that simplifies the process of booking, managing, and attending counselling appointments for university students. It eliminates manual booking, reduces administrative workload, and ensures secure, role-based access to critical functionality for both students and counselors.

The scope of the system includes:

- Secure user authentication using JWT for students and counselors.
- Slot-based appointment booking with a cap of 3 students per time block.
- Dashboard functionalities personalized for different roles.
- ICS file generation for calendar integration.
- Automated email confirmations on successful bookings.
- Logical and aesthetic user interfaces that enhance usability.
- REST API-driven backend with full CRUD support for appointment records.

System Features and Architecture



The application is developed based on **Layered (N-Tier) Architecture**, ensuring the separation of responsibilities and ease of maintenance. This consists of:

- **Controller Layer**: Responsible for mapping incoming HTTP requests.
- **Service Layer**: Hosts core business logic and inter-service communication.
- **Repository Layer**: Manages persistence using Spring Data JPA.
- **Entity/DTO Layer**: Models domain objects and transport objects for external communication.
- **Security Layer**: Uses JWT filters and Spring Security configurations.

Design Patterns Applied:

- **DTO Pattern**: Used extensively to encapsulate the structure of requests and responses.
- **Builder Pattern**: Employed via Lombok's @Builder annotation for object construction.
- **Singleton Pattern**: Spring-managed beans default to singleton lifecycle.
- **Factory Method Pattern**: Implemented through Spring’s @Bean configurations.
- **Strategy Pattern**: Underlying mechanism used by Spring Security for authentication flow.
- **Proxy Pattern**: Leverages Spring AOP for aspects like transaction management.
- **Template Method Pattern**: Enabled through extending JpaRepository.

Technology Stack and Tools Used

Layer	**Technologies / Libraries**
Frontend	React.js, Bootstrap 5, Vite, Axios, JavaScript
Backend	Java 17, Spring Boot 3, Spring Security, Spring Data JPA, JavaMail
Database	MySQL, Hibernate ORM
Development	IntelliJ IDEA, VSCode, MySQL Workbench, Postman
Deployment Tools	Git, GitHub (public repository)

Backend Implementation

The backend is a robust Spring Boot application with distinct layers organized for scalability and testability. Key components include:

- **Controllers**: Handle endpoints for appointments, students, counselors, and authentication.

- **Services**: Implement logic like JWT generation, slot validations, and appointment filtering.
- **Repositories**: Provide abstraction over data access using JPA.
- **DTOs**: Models such as AppointmentRequestDTO and AppointmentResponseDTO bridge the frontend and backend.
- **SecurityConfig.java**: Defines access policies, password encoding, and JWT filter chains.

Additional integrations:

- JavaMailSender for confirmation emails.
- Swagger for API testing (configurable).
- Exception handling via global advice classes.
- Input validations with Hibernate Validator.

Frontend Implementation

The React-based frontend leverages componentization, Bootstrap styling, and modern practices:

- **LoginComponent.jsx**: Authenticates user and stores tokens.
- **HeaderComponent/FooterComponent**: Static navigational elements.
- **BookAppointmentComponent.jsx**: Enables date and slot selection, validates booking.
- **ViewAppointmentsComponent.jsx**: Displays sorted appointment data.
- **EmailConfirmationModal.jsx**: Shows popup on successful booking.
- **TokenUtils.js**: Decodes JWT to extract role and user info.

Highlights:

- Axios used for secure API interactions with Authorization headers.
- .ICS generation and download for calendar clients.
- Modal confirmation UI with automated reset logic.
- Color-coded UI for appointment statuses.
- Role-based PrivateRoute protection of routes.

Database Design

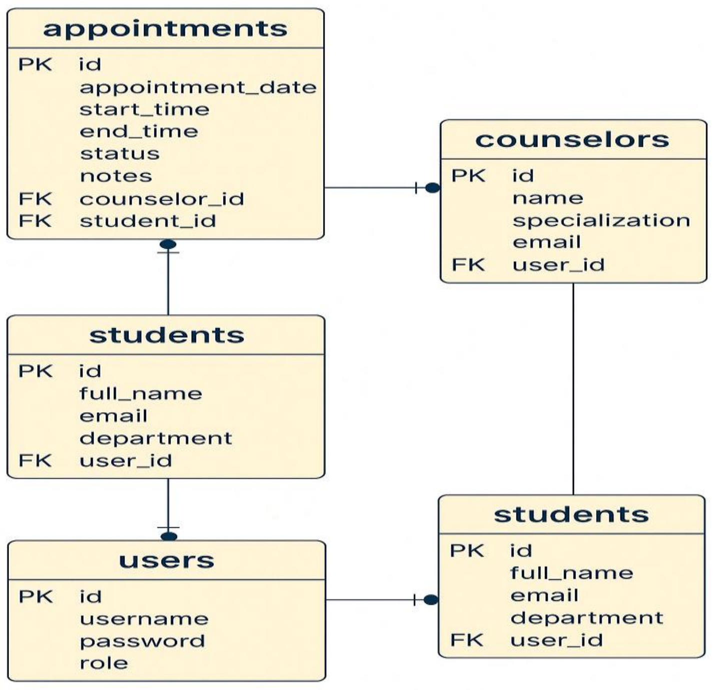
Database: appointments_db

Tables:

- users: Contains login credentials (username, password, role).

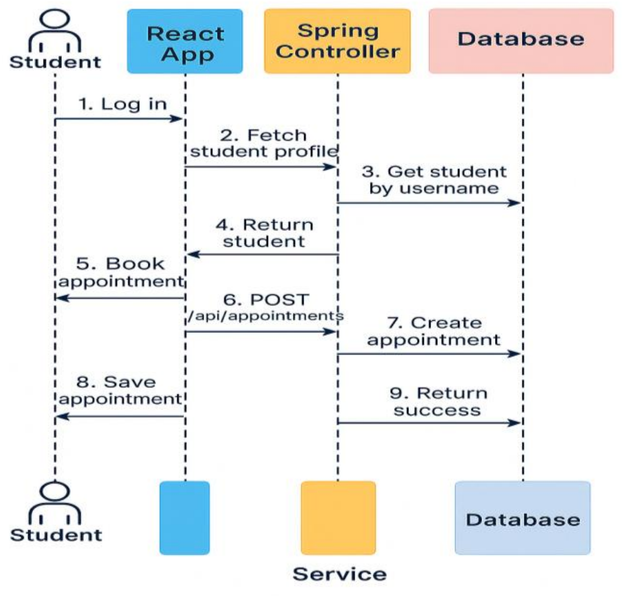
- students: Contains personal info and FK to users.
- counselors: Stores counselor details and FK to users.
- appointments: Core appointment data including FK to students and counselors.

ER Diagram

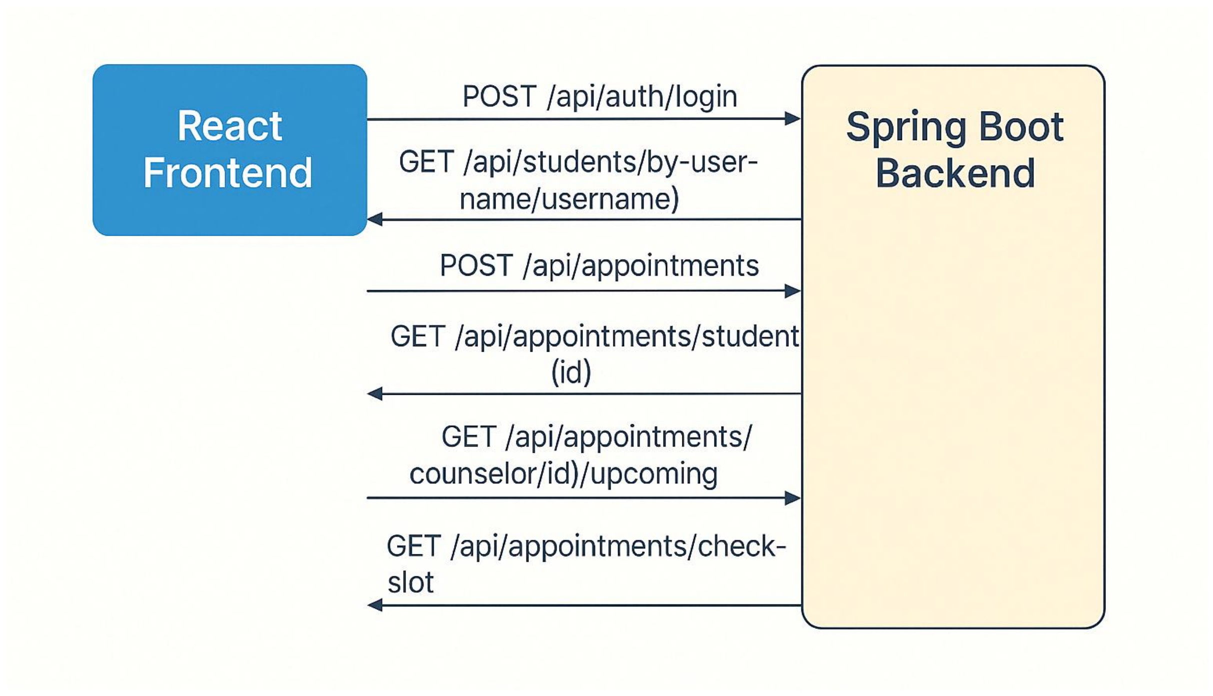


System Flow and Diagrams

Sequence Diagram



REST API Structure


React Frontend
Spring Boot Backend
POST /api/auth/login
GET /api/students/by-user-name/username)
POST /api/appointments
GET /api/appointments/student (id)
GET /api/appointments/counselor/id)/upcoming
GET /api/appointments/check-slot

Challenges Faced and Resolutions

- **Slot Booking Overlap**: Multiple students tried booking same time slot. → Implemented slot availability check.
- **Frontend Role-Based Routing Confusion**: /appointments used by both roles. → Split logic and filtered by JWT role.
- **JWT Malformed Error**: Occurred when token expired or corrupted. → Added robust error handling in TokenUtils.
- **MySQL FK Errors**: FK constraints broke insertions. → Reordered inserts and ensured user_id presence.
- **React Render Failures**: Component crashed on null objects. → Defensive programming using optional chaining.
- **CORS Issues**: Backend rejected frontend requests. → Configured CORS in SecurityConfig.
- **Static File Imports**: ICS and image attachments caused build errors. → Switched to Blob creation and dynamic links.
- **Email Failures**: SMTP error or auth failure. → Enabled App Password in Gmail and reconfigured JavaMailSender.

Scope for Future Enhancements

- **Session History**: Enable viewing past appointments with counselor remarks.
- **Calendar View UI**: Display appointments in a visual calendar component.
- **Multi-role Admin Panel**: Monitor system-wide activity, add/edit users, change roles.
- **Bulk Email Notification**: Reminders before appointments.
- **ICS Import Functionality**: Read external calendar inputs.
- **Email Templates**: Beautify current plain-text messages.
- **Mobile Responsiveness**: Improve UI for smaller screens.
- **Token Expiry Notification**: Alert users when session is close to expiry.

Lessons Learned

- Emphasized best practices in Spring Boot like separation of layers, stateless APIs, and dependency injection.
- Learned to integrate full-stack technologies—React, Bootstrap, Spring Security.
- Solved real-world challenges in routing, authentication, and concurrent access.
- Appreciated test-first development and modular component design.
- Realized the importance of exception-safe coding in backend and frontend.
- Improved GitHub repository hygiene by separating backend and frontend and adding clear READMEs.
- Gained confidence in working with cloud services (JavaMail/Gmail App Password).

User Guide and Setup Instructions

GitHub Repository

https://github.com/neszh24/student-counselling-system

A. Prerequisites

- Java 17+
- Node.js 18+
- MySQL 8+
- Maven

B. Backend Setup

1. Open / student-counselling-system-backend in IntelliJ
2. Import Maven project

3. Run appointments_db.sql in MySQL
4. Edit application.yml:

   spring.datasource.username: root

   spring.datasource.password: root
5. Run StudentCounsellingApplication.java

C. Frontend Setup

1. Open / student-counselling-system-frontend in VS Code
2. Run npm install
3. Start app with npm run dev
4. Visit http://localhost:5173

D. Login Credentials

Username	Password	Role
student1	pass123	STUDENT
student2	pass234	STUDENT
counselor1	pass1234	COUNSELOR
counselor2	pass1234	COUNSELOR

E. Usage Guide

- Students: Login → Book Appointment → View/Download ICS
- Counselors: Login → View Appointments
- Confirmation email is sent instantly to configured Gmail ID.

References

1. W3Schools. (2024). Spring Boot Tutorial. https://www.w3schools.com/spring/spring_boot_intro.php
2. Baeldung. (2024). JWT in Spring Security. https://www.baeldung.com/spring-security-oauth-jwt
3. React Documentation. (2024). https://reactjs.org/docs/getting-started.html
4. Bootstrap Docs. (2024). https://getbootstrap.com/docs/5.3/getting-started/introduction/
5. Vite.js Docs. (2024). https://vitejs.dev/guide/
6. Spring.io Guides. (2024). https://spring.io/guides
7. MySQL 8.0 Reference Manual. (2024). https://dev.mysql.com/doc/
8. Stack Overflow Discussions. https://stackoverflow.com
9. GitHub Docs. (2024). https://docs.github.com/en
10. Mailtrap Blog - Sending Emails. https://mailtrap.io/blog/spring-boot-send-email/