**Reflective Report on Hotel Management System**

**Submitted To Submitted By**

**Hari Sharan Shrestha Parash Mainali**

ST5007CEMWeb Development

# Introduction

This report presents a comprehensive overview of the Hotel Management System project developed using a React frontend and a Spring Boot backend. The system provides features such as room management, hotel booking, order management, and staff/user role access management. This report will reflect on the learning outcomes, design process, technologies used, quality assurance, design choices, and illustrative examples, providing a thorough insight into the project development journey.

# Learning Outcomes

Throughout the term, I have significantly advanced my understanding and practical skills in full-stack development. This project has facilitated learning in various key areas:

- **Designing and Implementing RESTful APIs**: I have learned how to create, secure, and document RESTful APIs, which form the backbone of modern web applications.

- **React State and Props Management**: Through extensive use of React Hooks and Context API, I have become proficient in managing component state and ensuring efficient data flow within the application.

- **Authentication and Authorization**: Implementing JWT-based authentication in Spring Boot and securing endpoints with Spring Security has deepened my understanding of web security principles.

- **Database Management**: Working with Spring Data JPA and PostgreSql has provided insights into efficient data management practices, including ORM (Object-Relational Mapping) and database migrations.

- **Unit Testing**: Conducting unit tests for both frontend and backend components has underscored the importance of testing in maintaining code quality and reliability.

- **Deployment**: Deploying a full-stack application has taught me about server configuration, environment management, and continuous integration/continuous deployment (CI/CD) practices.

# Scope and Design Process

## Scope

The Hotel Management System is designed to streamline hotel operations through a comprehensive suite of features, including:

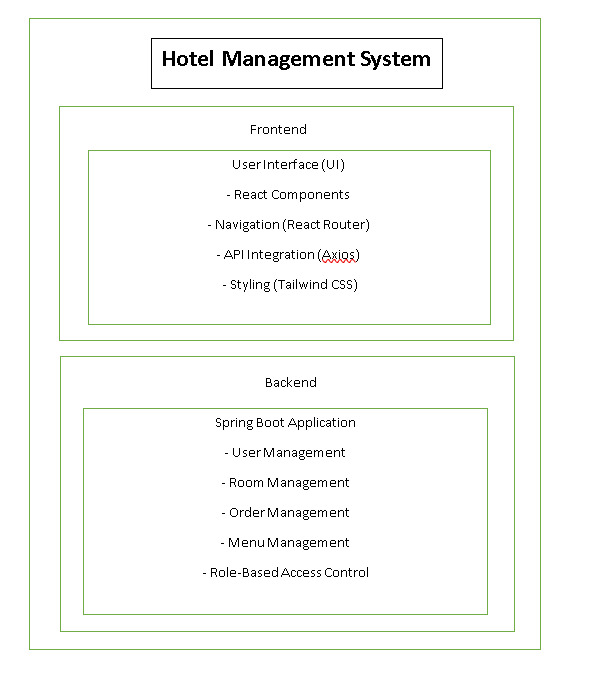
- Room Management: Add, update, delete, and view available rooms.

- Booking System: Book rooms, view booking details, and manage bookings.

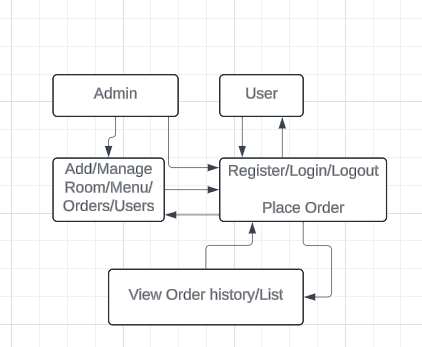
- Order Management: Handle orders related to room service, restaurant services, and more.

- User Management: Manage staff and user roles, ensuring secure access to different parts of the system.

## Conceptual Diagram

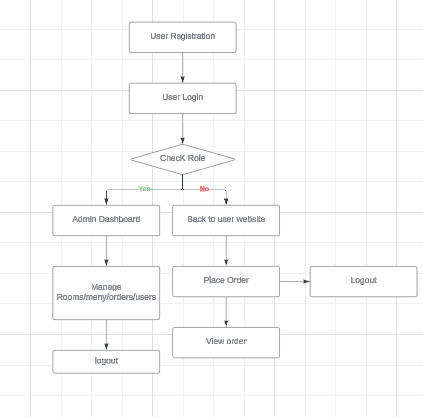


## Use Case Diagram



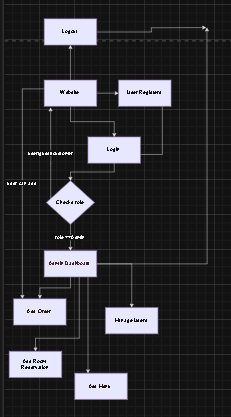
The use case diagram details the primary interactions users have with the system, highlighting functionalities like room booking, order management, and user role assignments.

## Activity Diagram



The activity diagram provides a step-by-step workflow for key processes, such as booking a room, from selecting dates to payment confirmation.

## Flow Diagram



The sequence diagram shows the interaction between frontend and backend components during a booking operation, detailing the flow of requests and responses.

# Technologies Used

## Frontend: React

**- React Hooks**: Enabled efficient state management and lifecycle handling within functional components.

**- Axios**: Used for making HTTP requests to the backend, simplifying API interactions.

**- React Router**: Provided robust navigation and routing capabilities within the single-page application.

**- Tailwind CSS**: Offered a utility-first approach to styling, ensuring a responsive and customizable UI design.

**- Chart.js**: Facilitated data visualization in the dashboard, presenting insights into hotel operations.

**- JSX**: Used for creating components in a syntax that closely resembles HTML, making the structure of the UI more intuitive.

## Backend: Spring Boot

- **Spring Data JPA**: Simplified database interactions through repositories, enabling CRUD operations and complex queries.

- **Spring Security**: Ensured robust authentication and authorization mechanisms, protecting sensitive endpoints.

- **JWT**: Provided secure token-based authentication, ensuring secure and stateless sessions.

- **H2 Database**: Used for in-memory database during development and testing phases, enabling rapid prototyping.

- **PostgreSql**: Deployed as the production database, offering reliability and scalability.

### Tools

- **Postman**: Utilized for API testing, allowing for efficient testing and debugging of backend endpoints.

- **JUnit**: Employed for unit testing backend services, ensuring functionality and reliability.

- **Yup resolver**: Used for unit testing frontend components, validating the integrity of the UI.

# Quality Assurance

## Unit Testing

Unit testing was integral to ensuring the reliability and functionality of both frontend and backend components. Tests were written to cover various scenarios, validating the correctness of each module.

## Test Cases and Results

### Frontend Test Cases

Test Case ID Description Expected Result Actual Result Status

TC-FRONT-01 Test login form validation Form validation works As expected Passed

TC-FRONT-02 Test room add/edit/delete Room is added,edted and deleted As expected Passed

TC-FRONT-03 Test booking functionality Booking is successful As expected Passed

TC-FRONT-04 Test order management Orders are managed As expected Passed

### Backend Test Cases

Test Case ID Description Expected Result Actual Result Status

TC-BACK-01 Test user authentication User is authenticated | As expected Passed

TC-BACK-02 Test room retrieval Rooms are retrieved As expected Passed

TC-BACK-03 Test booking endpoint Booking is created As expected Passed

TC-BACK-04 Test order retrieval Orders are retrieved As expected Passed

# Design Choices

## Frontend

- Responsive Design: Utilizing Tailwind CSS ensured that the application was fully responsive, providing a consistent user experience across various devices.

- State Management: React Hooks, such as `useState` and `useEffect`, were extensively used for managing state and side effects, promoting functional and declarative programming.

- Error Handling: Comprehensive error handling mechanisms were implemented to provide meaningful feedback to users and improve the application's robustness.

## Backend

- Spring Security: Spring Security was chosen for its robust and flexible security features, ensuring secure authentication and authorization.

- RESTful API: Designing RESTful APIs facilitated clear and efficient communication between the frontend and backend, adhering to modern web standards.

- Database Design: A normalized database schema was created to ensure data integrity and reduce redundancy, enhancing overall performance.

# Frontend Development

## Overview

The frontend of the Hotel Management System was built using React, a widely-used JavaScript library for building dynamic user interfaces. The primary goal was to create a responsive, user-friendly interface that would streamline hotel management tasks.

## Key Components

- Dashboard: Provides an overview of hotel operations, including room availability, bookings, and recent orders. It features visual representations of data using Chart.js for better insights.

- Room Management: Allows administrators to perform CRUD operations on room entities, facilitating efficient room management.

- Booking System: Enables users to book rooms, view available rooms, and manage their bookings through an intuitive interface.

- Order Management: Manages orders placed by guests, such as room service and restaurant orders, ensuring seamless service delivery.

- User Management: Provides functionalities for adding and managing staff and user roles, ensuring secure access management.

## State Management

State management in React was handled using React Hooks. The `useState` hook was used for managing local component state, while the `useEffect` hook managed side effects such as data fetching. This approach promoted a functional programming style and improved code readability.

## Styling

Tailwind CSS was employed for styling the frontend, providing a utility-first CSS framework that facilitated rapid and responsive design. Custom components were created to maintain a consistent look and feel throughout the application. Tailwind's responsive utilities ensured that the application was mobile-friendly and accessible on various devices.

# Backend Development

## Overview

The backend of the Hotel Management System was developed using Spring Boot, a powerful framework for building Java applications. The backend handles the core business logic, data persistence, and communication with the frontend through RESTful APIs.

## Key Components

**- User Authentication and Authorization**: Implemented using Spring Security and JWT for secure token-based authentication. This ensures that only authenticated users can access protected resources.

- **Room Management**: Handles CRUD operations for room entities, providing endpoints for adding, updating, deleting, and retrieving room information.

- **Booking Management**: Manages the booking process, including creating, retrieving, and canceling bookings. The booking service ensures that room availability is accurately tracked and updated.

- **Order Management**: Manages orders placed by guests, such as room service and restaurant orders. This component ensures efficient order tracking and management.

- **Role-Based Access Control**: Implements role based access control to ensure that users have appropriate permissions based on their roles (e.g., admin, guest, staff).

**- Security:** Security was a primary concern in the backend development. Spring Security was used to secure endpoints, ensuring that only authorized users could access specific resources. JWT tokens provided a stateless authentication mechanism, enhancing security and scalability.

**- Data Management:** Spring Data JPA was used for data management, providing a powerful and flexible ORM solution. This allowed for efficient database interactions, including complex queries and transactions. The database schema was designed to ensure data integrity and reduce redundancy, optimizing performance.

# Illustrative Examples

## Room Booking Process

The room booking process involves several steps, from selecting dates to payment confirmation. The following sequence diagram illustrates the interaction between the frontend and backend during a booking operation.

![Sequence Diagram](path\_to\_sequence\_diagram)

1. **User selects dates**: The user selects check-in and check-out dates from the calendar component on the frontend.

2. **Availability check**: The frontend sends a request to the backend to check room availability for the selected dates.

3. **Room selection**: The backend responds with a list of available rooms, which is displayed to the user.

4**. Booking details**: The user selects a room and provides booking details, such as guest information and payment method.

5. **Booking confirmation**: The frontend sends a booking request to the backend, which processes the booking and returns a confirmation response.

## Order Management

The order management process involves tracking and managing orders placed by guests. The following activity diagram illustrates the workflow for managing orders.

![Activity Diagram](path\_to\_activity\_diagram)

1. **Order placement:** The guest places an order through the frontend, specifying the items and quantity.

2. **Order processing:** The frontend sends the order details to the backend, which processes the order and updates the order status.

3. **Order tracking**: The backend sends order status updates to the frontend, allowing the guest to track the order progress.

4. **Order delivery**: Once the order is ready, the backend notifies the frontend, and the order is delivered to the guest.

# Links

- GitHub Repository: [Link to GitHub Classroom Repository](https://github.com/YourUsername/YourRepository)

- Video Screencast: [Link to YouTube Screencast](https://youtu.be/YourVideoLink)

# Conclusion

The Hotel Management System project has been a valuable learning experience, providing insights into full-stack development, security, database management, and testing. The project has equipped me with the skills and knowledge necessary to develop robust and scalable web applications. Through this project, I have learned the importance of planning, designing, and testing in software development, ensuring that the final product meets the requirements and provides a seamless user experience.