**Hotel Management System Project Report**

**Course: COMP371**

**University of the Fraser Valley (UFV)**

**Term: Spring 2025**

**Prepared by: Ekamjot Singh**

1. Introduction

This report describes the development and implementation of a Java-based Hotel Management System. The purpose of the system is to streamline hotel operations through a graphical user interface (GUI) integrated with a MySQL database. It supports role-based access, enabling employees to efficiently manage hotel resources, guests, and finances.

2. Project Objectives

- Develop a user-friendly GUI using Java Swing.
- Integrate the system with a MySQL/MariaDB database.
- Implement role-based access control (RBAC) with multiple user roles (Admin, Receptionist, Housekeeping).
- Provide CRUD (Create, Read, Update, Delete) operations for key entities including reservations, rooms, inventory, bills, and reports.
- Apply software engineering design patterns such as Singleton and Factory.

- Use UML diagrams for system documentation and planning.

3. Technology Used

- Programming Language: Java 17+
- GUI Framework: Java Swing
- Database: MySQL (MariaDB) hosted on AWS Cloud9
- Database Connector: mysql-connector-j-9.2.0.jar
- UML Diagrams: PlantUML
- Version Control: GitLab

4. Methodology

4.1 Planning and Design

Initially, we planned our system by outlining the primary user interactions and functionalities. Use-case scenarios were created for major operations such as logging in, managing reservations, and billing processes. We selected Java Swing for GUI due to its ease of use and compatibility with Java.

4.2 Database Design

A relational database schema was designed to accommodate essential data such as user accounts, guest reservations, room statuses, inventory management, billing, and reporting information. We utilized MySQL for its widespread usage and robust SQL support.

The schema comprises the following tables:

- users
- reservations
- rooms
- inventory
- bills
- reports

### 4.3 Development and Implementation

We followed an incremental development approach, gradually implementing features:

- Login/Register functionality with different user roles.
- CRUD operations for managing reservations, rooms, inventory, bills, and reports.
- Singleton pattern for managing database connections efficiently.
- Factory pattern for creating role-specific dashboards dynamically.

### 4.4 Testing and Deployment

Extensive testing was performed manually to ensure reliability across all GUI components and database operations. Errors and exceptions, particularly database connectivity issues, were identified and resolved during this phase.

5. Features and Functionalities

## 5.1 User Authentication and Role-Based Access

- Secure login and registration system.
- Role-based dashboards: Admins have full access, receptionists handle reservations and billing, and housekeeping manages rooms and inventory.

## 5.2 Reservation Management

- Allows creation, modification, deletion, and viewing of guest reservations.

## 5.3 Room Management

- Tracks room numbers, types, and occupancy status.

## 5.4 Inventory Management

- Keeps detailed records of hotel inventory items and their quantities.

## 5.5 Billing Management

- Manages financial transactions associated with reservations, including the capability to mark bills as paid or unpaid.

## 5.6 Reporting

- Provides a reporting module for generating basic reports and executing custom SQL queries to analyze data.

6. UML Diagrams and Documentation

To ensure clear documentation and system planning, we created several UML diagrams using PlantUML:

- Use Case Diagram: Represents user interactions with the system.
- Class Diagram: Illustrates class structure and relationships.
- Sequence Diagram: Demonstrates flow during user login.
- Activity Diagram: Outlines the reservation-making process.

These diagrams have been included to support understanding of system functionality and design.

7. Challenges Faced

- Maintaining database connections: Resolved by implementing connection validation checks (isClosed()) to prevent connection-related exceptions.
- GUI responsiveness: Addressed by using appropriate layout managers and Swing threading best practices.

8. Future Plans

To further enhance the Hotel Management System, we plan the following future improvements:

- Implement Observer pattern for real-time low inventory notifications.
- Introduce dynamic pricing and billing for additional services such as dining and spa treatments.
- Develop automated email or SMS notifications for guest reservations.
- Add guest history tracking and loyalty metrics.
- Perform unit testing using JUnit to automate testing.
- Explore integration with web or mobile applications for remote management.

9. Conclusion

The Hotel Management System successfully integrates a robust backend with an intuitive GUI, satisfying the key requirements outlined in COMP371. Design patterns and systematic planning contributed significantly to our project's organization and maintainability. The system demonstrates both software engineering best practices and practical, industry-related functionality.

10. References

- Oracle Java Documentation, Swing Tutorial:

  https://docs.oracle.com/javase/tutorial/uiswing/
- MySQL Connector/J documentation: https://dev.mysql.com/doc/connector-j/en/
- PlantUML Documentation: https://plantuml.com/

11. Screenshots

Provided as a separate folder in the .zip submission!