**HOSTEL MANAGEMENT SYSTEM**

**MINI PROJECT REPORT**

SUBMITTED BY

230701005 Abhishek Robin S A

230701003 Abhilash G R

*In partial fulfilment for the award of the degree*

BACHELOR OF ENGINEERING

IN COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE(AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024- 2025

**BONAFIDE CERTIFICATE**

Certified that this project report

**“HOSTEL MANAGEMENT SYSTEM”**

is the bonafide work of

**“Abhishek Robin S A (230701005),**

**Abhilash G R (230701003)”**

who carried out the project work under my supervision.

Submitted for the Practical Examination held on 23.11.2024\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ABSTRACT

This project on **Hostel Management System** automates the registration and management processes of a hostel. The system provides information like resident details, room information, a list of all residents, and facilitates data storage and retrieval for the hostel environment, enabling transactions related to resident management and room allocation. It allows the addition of records when a resident is registered or a room is reserved. For data storage and retrieval, we use a MySQL Database, which enables us to add any number of records in our database.

The project **"Hostel Management System"** manages various entities associated with hostel operations. The system includes:

* **Hostel**
* **Hostel Staff**
* **Residents**

Each of these accesses a database schema with corresponding tables for efficient data handling. This system is designed to streamline the management of residents, rooms, staff, and reservations in a hostel environment.

Language Used - Java Core

Concept Used - Swing

IDE Used - NetBeans

Database Used - MySQL

CONTENTS

CHAPTERS

Chapter 1 Introduction

1.1Problem Definition

1.2 Need

Chapter 2 Requirements

2.1 Software Requirement Specifications

2.2 Hardware Requirement Specifications

Chapter 3 Entity Relationship Diagram

3.1 Entity relationship diagram

Chapter 4 Schema Diagram

4.1 Schema diagram

Chapter 5 Implementation

5.1 Backend Implementation

5.2 Frontend implemenatation

5.3 Creating mainframe class

Chapter 6 Snapshots

Conclusion

References

CHAPTER 1

INTRODUCTION

The **Hostel Management System** is designed to streamline and automate the management of hostel operations, particularly focusing on tracking room information, monitoring room availability, handling bookings, and managing rooms. In a hostel environment, where numerous residents and rooms need to be organized effectively, this system provides a comprehensive solution to optimize resource management, enhance record-keeping, and improve overall efficiency.

This system serves as a digital solution for hostel administrators to manage various aspects of hostel operations. It enables administrators to track room assignments, monitor room occupancy, and update room availability in real time. With an intuitive interface, the system facilitates the booking process, allowing hostel staff to easily allocate rooms based on availability and resident requirements.

**Key Features:**

* **Room Information Tracking**: Maintains detailed records of each room, including room type, capacity, amenities, and current occupancy status.
* **Room Availability Monitoring**: Provides real-time updates on room availability, helping administrators efficiently allocate resources and optimize room usage.
* **Room Bookings**: Manages the process of booking rooms for new residents, ensuring smooth check-ins and a clear overview of assigned rooms.
* **Room Management**: Allows for easy reallocation of rooms, handling room changes, and managing maintenance schedules to ensure the upkeep of facilities.

Overall, this **Hostel Management System** is intended to simplify hostel operations, reduce administrative workload, and provide a reliable way to manage resident accommodations.

# 1.1Problem Definition

The **Hostel Management System** addresses several challenges faced by hostel administrators in managing hostel operations, particularly around room assignments, tracking resident information, and maintaining facility records. Manual management of these processes is time-consuming, prone to errors, and inefficient, often resulting in issues like double bookings, poor room utilization, and incomplete resident records.

The **Hostel Management System** is essential in streamlining various administrative and operational processes in hostel environments. Below are some key needs that the system fulfills:

# 1.2 Need

## 1. Efficient Room Allocation and Availability Tracking

* Assigning rooms to residents based on availability, preferences, and capacity becomes much easier.
* Provides real-time updates on room occupancy, helping to reduce confusion and ensure each resident has an appropriate room assignment.
* Reduces the risk of overbooking and provides a clear view of vacant and occupied rooms.

## 2. Resident Record Management

* Stores detailed information about each resident, including personal information, duration of stay, payment status, and assigned room.
* Centralized records allow hostel staff to quickly access resident information, ensuring effective communication and support.

## 3. Streamlined Check-In and Check-Out Process

* Speeds up the check-in process by storing booking information and pre-assigned rooms, making it convenient for both staff and residents.
* Facilitates smooth check-outs by providing details on room conditions and outstanding balances, if any.

## 4. Billing and Payments Management

* Tracks room fees, deposits, and any additional charges (e.g., for damages or extra services).
* Keeps a detailed payment history for each resident, ensuring transparent and accurate billing.

## 5. Maintenance and Housekeeping Management

* Schedules room cleaning, repairs, and maintenance to maintain the quality and safety of facilities.
* Provides alerts or reminders for regular maintenance activities, ensuring a clean and functional environment for residents.

## 6. Enhanced Security and Access Control

* Keeps records of resident IDs and visitor logs to ensure hostel security.
* Tracks access to rooms and common areas, improving safety by monitoring who is on the premises.

## 7. Reporting and Analytics

* Generates reports on occupancy rates, financial summaries, and maintenance needs, aiding in informed decision-making.
* Provides insights on trends, like peak occupancy periods and preferred room types, allowing the hostel management to plan accordingly.

## 8. Improved Communication

* Facilitates communication between hostel staff and residents, especially for updates regarding payments, maintenance schedules, and announcements.
* Provides a platform for residents to submit requests or complaints, which can be addressed efficiently by management.

## 9. Resource Optimization

* Reduces administrative workload by automating repetitive tasks, allowing staff to focus on more critical functions.
* Optimizes the use of hostel resources, such as room allocation and maintenance scheduling, improving overall operational efficiency.

In summary, the **Hostel Management System** addresses the need for an organized, efficient, and secure way to manage hostel operations, improving the experience for both residents and staff.

CHAPTER 2

REQUIREMENTS

# 2.1 Software Requirement Specifications

Operating System Front End Back End Server Documentation : Windows 10

Frontend Software: Java NetBeans 8.2 : JDK 8

Backend Software: MySQL

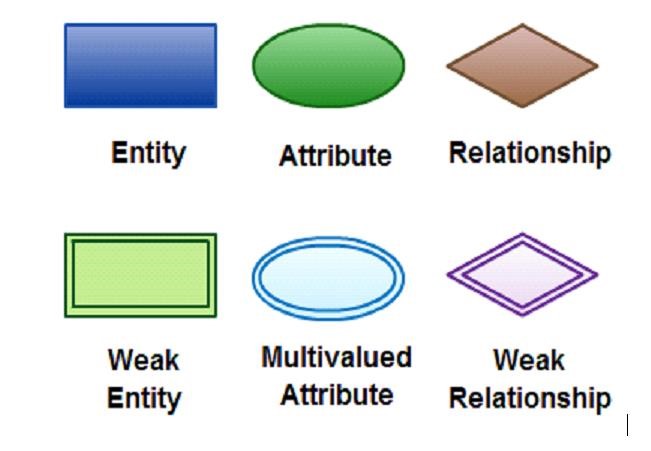
# 2.2 Hardware Requirement Specifications

Computer Processor Core i3 Processor Speed 2.3 GHz Processor Hard Disk 400 GB or more RAM Min 2GB

# CHAPTER 3

## ENTITY RELATIONSHIP DIAGRAM

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. If the application is primarily a database application, the entityrelationship approach can be used effectively for modeling some parts of the problem. The main focus in ER modeling is the Data Items in the system and the relationship between them. It aims to create conceptual scheme for the Data from the user’s perspective. The model thus created is independent of any database model. The ER models are frequently represented as ER diagram. Here we present the ER diagram of the above mentioned project.



# CHAPTER 4

4.1 SCHEMA DIAGRAM

A database schema is the skeleton structure that represents the logical view of the entire database. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It’s the database designers who design the schema to help programmers understand the database and make it useful.

CHAPTER 5

## IMPLEMENTATION

5.1 Backend Implementation

### MYSQL

MySQL is an open-source relational database management system (RDBMS). A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

-- Create the hostel\_management database

CREATE DATABASE IF NOT EXISTS hostel\_management;

-- Use the hostel\_management database

USE hostel\_management;

-- Room Table: Stores details of each room CREATE TABLE IF NOT EXISTS Room ( room\_id INT AUTO\_INCREMENT PRIMARY KEY, room\_type VARCHAR(50) NOT NULL, status ENUM('Available', 'Booked', 'Maintenance') DEFAULT 'Available', price DECIMAL(10, 2) NOT NULL

);

-- Booking Table: Tracks room bookings and guest information CREATE TABLE IF NOT EXISTS Booking ( booking\_id INT AUTO\_INCREMENT PRIMARY KEY, room\_id INT, guest\_name VARCHAR(100) NOT NULL,

check\_in\_date DATE NOT NULL, check\_out\_date DATE NOT NULL,

FOREIGN KEY (room\_id) REFERENCES Room(room\_id)

);

-- Optional: User Table (for login and role-based management) CREATE TABLE IF NOT EXISTS User ( user\_id INT AUTO\_INCREMENT PRIMARY KEY, username VARCHAR(50) UNIQUE NOT NULL, password VARCHAR(100) NOT NULL, role ENUM('Admin', 'Receptionist') NOT NULL

);

-- Insert Sample Data into Room Table

INSERT INTO Room (room\_type, status, price) VALUES

('Single', 'Available', 50.00),

('Single', 'Available', 50.00),

('Double', 'Available', 75.00),

('Double', 'Booked', 75.00),

('Suite', 'Available', 120.00),

('Suite', 'Maintenance', 120.00);

-- Insert Sample Data into Booking Table (linked to a booked room)

INSERT INTO Booking (room\_id, guest\_name, check\_in\_date, check\_out\_date) VALUES (4, 'John Doe', '2024-11-01', '2024-11-05');

-- Optional: Insert Sample Data into User Table

INSERT INTO User (username, password, role) VALUES

('admin', 'admin\_password', 'Admin'),

('reception', 'reception\_password', 'Receptionist');

-- Check initial data

SELECT \* FROM Room;

SELECT \* FROM Booking;

SELECT \* FROM User;

5.2 Frontend Implementation

## Java Core

Core Java is the part of Java programming language that is used for creating or developing a general-purpose application. It uses only one tier architecture that is why it is called as ‘stand alone’ application.Core java programming covers the swings, socket, awt, thread concept, collection object and classess.

Swings

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

### 5.3 Creating mainframe class

import javax.swing.\*; import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener; import java.sql.Connection; import java.sql.DriverManager; import java.sql.SQLException; import java.sql.Date;

public class App { public static class MainFrame extends JFrame { private Connection connection;

public MainFrame() { setTitle("Hostel Management System"); setSize(500, 400);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLocationRelativeTo(null);

// Initialize the database connection try {

connection = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/hostel\_management", "root", "12345678");

System.out.println("Connected to the database successfully!");

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Database connection failed: " + e.getMessage(),

"Error", JOptionPane.ERROR\_MESSAGE);

System.exit(1);

}

// Set up the layout setLayout(new GridLayout(4, 1, 10, 10));

// Add buttons for each action

JButton viewRoomsButton = new JButton("View All Rooms");

JButton checkAvailabilityButton = new JButton("Check Room Availability");

JButton bookRoomButton = new JButton("Book a Room");

JButton cancelBookingButton = new JButton("Cancel a Booking");

// Add action listeners for buttons viewRoomsButton.addActionListener(e -> viewAllRooms()); checkAvailabilityButton.addActionListener(e -> checkRoomAvailability()); bookRoomButton.addActionListener(e -> bookRoom()); cancelBookingButton.addActionListener(e -> cancelBooking());

// Add buttons to the frame add(viewRoomsButton); add(checkAvailabilityButton); add(bookRoomButton); add(cancelBookingButton);

}

private void viewAllRooms() {

// Implement the code to retrieve and display room details

try (var stmt = connection.createStatement(); var rs = stmt.executeQuery("SELECT \* FROM Room")) {

StringBuilder roomList = new StringBuilder("Room List:\n"); while (rs.next()) {

roomList.append("ID: ").append(rs.getInt("room\_id"))

.append(", Type: ").append(rs.getString("room\_type"))

.append(", Status: ").append(rs.getString("status"))

.append(", Price: $").append(rs.getDouble("price"))

.append("\n");

}

JOptionPane.showMessageDialog(this, roomList.toString(),

"View All Rooms", JOptionPane.INFORMATION\_MESSAGE);

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Error retrieving rooms: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void checkRoomAvailability() {

String roomType = JOptionPane.showInputDialog(this, "Enter room type (e.g., Single, Double):");

if (roomType != null && !roomType.isEmpty()) {

String query = "SELECT \* FROM Room WHERE room\_type = ? AND status = 'Available' LIMIT 1";

try (var stmt = connection.prepareStatement(query)) { stmt.setString(1, roomType); var rs = stmt.executeQuery();

if (rs.next()) {

String message = "Available Room:\nID: " + rs.getInt("room\_id") +

", Type: " + rs.getString("room\_type") +

", Price: $" + rs.getDouble("price");

JOptionPane.showMessageDialog(this, message,

"Room Availability", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "No available rooms of type: " + roomType,

"Room Availability", JOptionPane.INFORMATION\_MESSAGE);

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Error checking availability: " + e.getMessage(),

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

private void bookRoom() {

String guestName = JOptionPane.showInputDialog(this, "Enter guest name:");

String roomType = JOptionPane.showInputDialog(this, "Enter room type to book (e.g., Single, Double):");

if (guestName != null && roomType != null && !guestName.isEmpty() && !roomType.isEmpty()) {

String findRoomQuery = "SELECT \* FROM Room WHERE room\_type = ? AND status = 'Available' LIMIT 1";

String updateRoomQuery = "UPDATE Room SET status = 'Booked' WHERE room\_id = ?";

String insertBookingQuery = "INSERT INTO Booking (room\_id, guest\_name, check\_in\_date, check\_out\_date) VALUES (?, ?, ?, ?)";

try (var findRoomStmt = connection.prepareStatement(findRoomQuery); var updateRoomStmt = connection.prepareStatement(updateRoomQuery); var insertBookingStmt = connection.prepareStatement(insertBookingQuery)) {

findRoomStmt.setString(1, roomType); var rs = findRoomStmt.executeQuery();

if (rs.next()) { int roomId = rs.getInt("room\_id");

String checkInDate = JOptionPane.showInputDialog(this, "Enter check-in date (YYYY-MM-DD):");

String checkOutDate = JOptionPane.showInputDialog(this, "Enter check-out date (YYYY-MM-DD):");

updateRoomStmt.setInt(1, roomId); updateRoomStmt.executeUpdate(); insertBookingStmt.setInt(1, roomId); insertBookingStmt.setString(2, guestName); insertBookingStmt.setDate(3, Date.valueOf(checkInDate)); insertBookingStmt.setDate(4, Date.valueOf(checkOutDate)); insertBookingStmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Room booked successfully for " + guestName,

"Booking Confirmation", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "No available rooms of type: " + roomType,

"Booking Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Error booking room: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

private void cancelBooking() {

String bookingIdInput = JOptionPane.showInputDialog(this, "Enter booking ID to cancel:");

if (bookingIdInput != null && !bookingIdInput.isEmpty()) { int bookingId = Integer.parseInt(bookingIdInput); String getBookingQuery = "SELECT \* FROM Booking WHERE booking\_id = ?";

String deleteBookingQuery = "DELETE FROM Booking WHERE booking\_id = ?";

String updateRoomStatusQuery = "UPDATE Room SET status = 'Available' WHERE room\_id = ?";

try (var getBookingStmt = connection.prepareStatement(getBookingQuery); var deleteBookingStmt = connection.prepareStatement(deleteBookingQuery); var updateRoomStatusStmt =

connection.prepareStatement(updateRoomStatusQuery)) {

getBookingStmt.setInt(1, bookingId); var rs = getBookingStmt.executeQuery();

if (rs.next()) { int roomId = rs.getInt("room\_id");

deleteBookingStmt.setInt(1, bookingId); deleteBookingStmt.executeUpdate();

updateRoomStatusStmt.setInt(1, roomId); updateRoomStatusStmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Booking canceled successfully.",

"Cancellation Confirmation", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "Booking ID not found.",

"Cancellation Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Error canceling booking: " + e.getMessage(),

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

MainFrame mainFrame = new MainFrame(); mainFrame.setVisible(true);

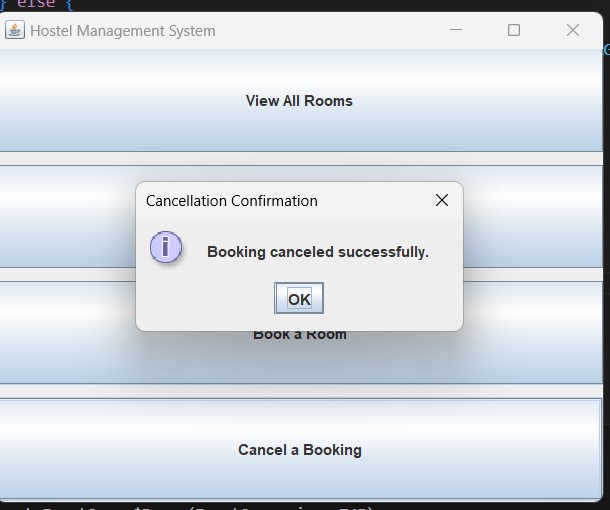
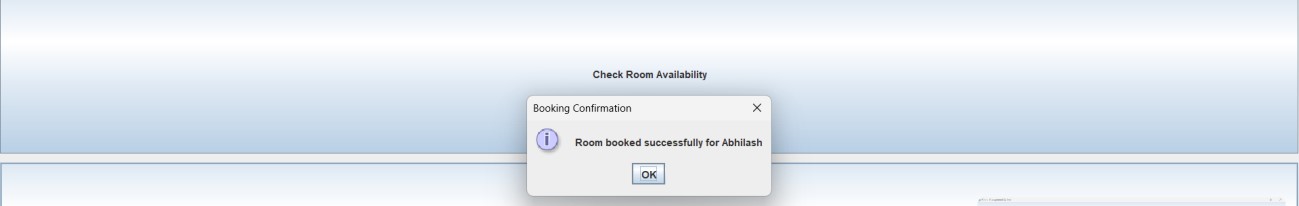
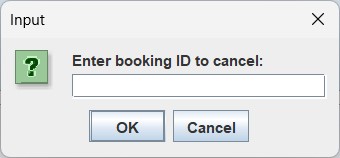
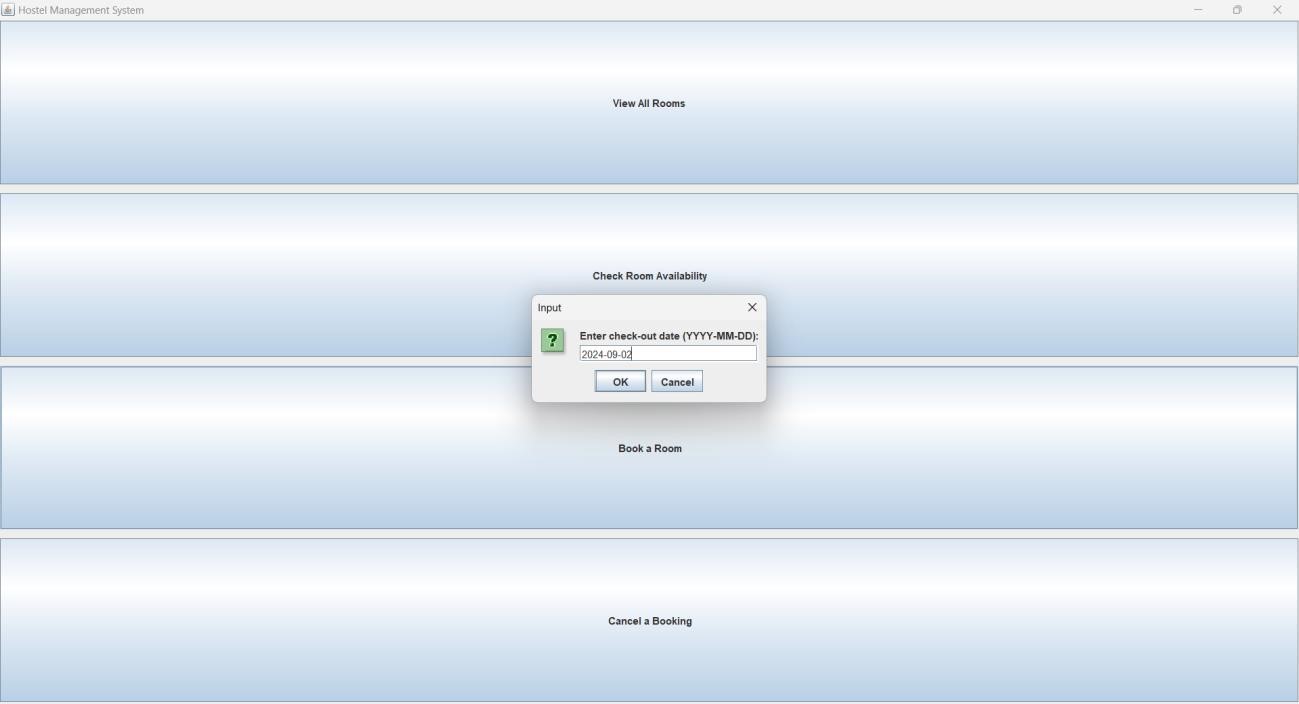
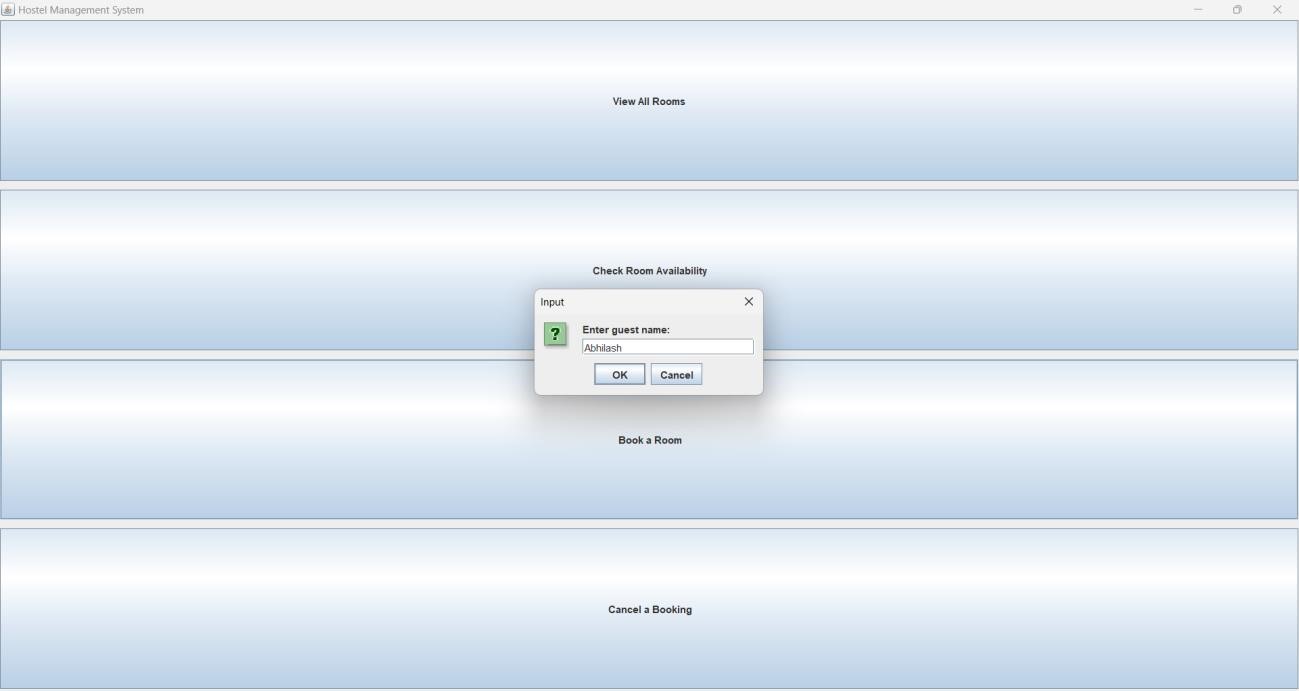
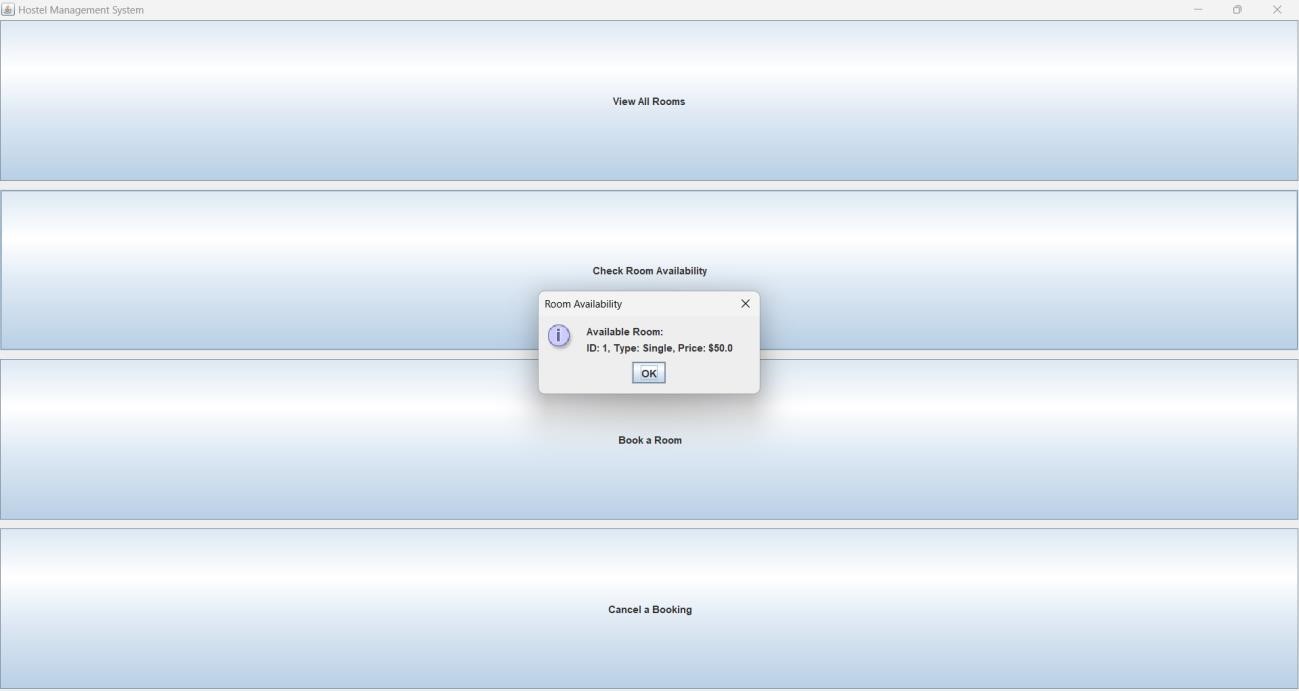
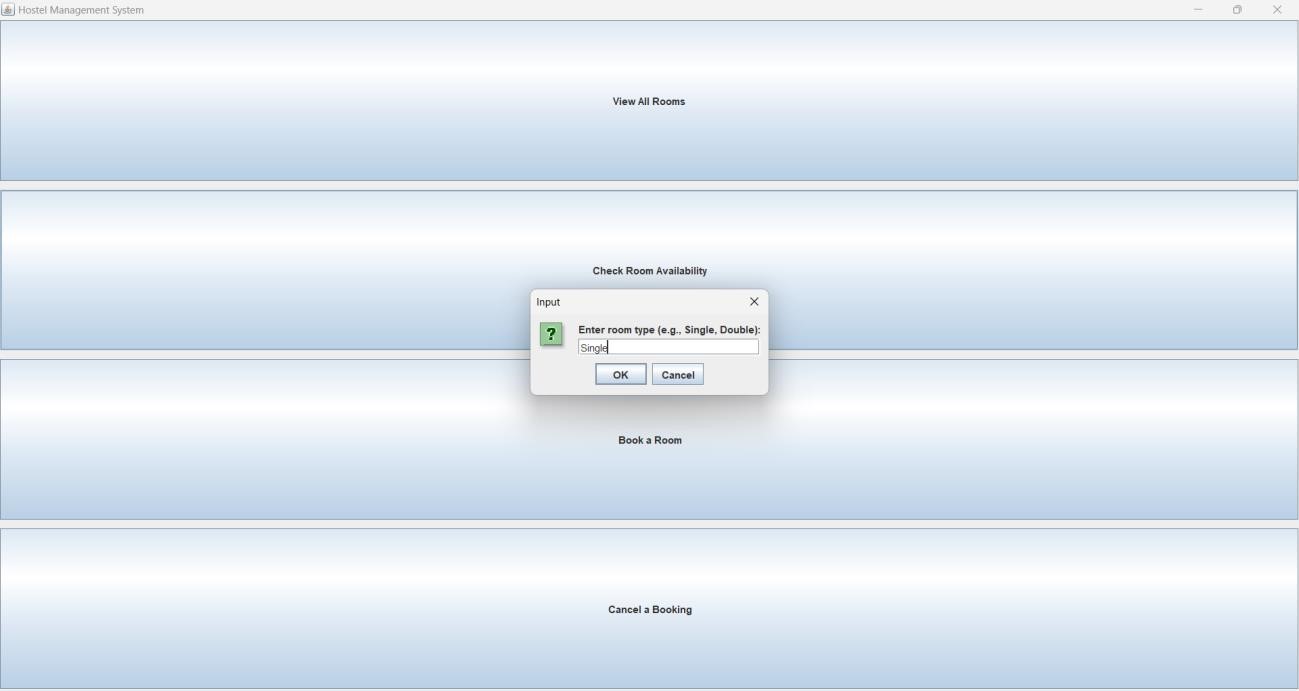
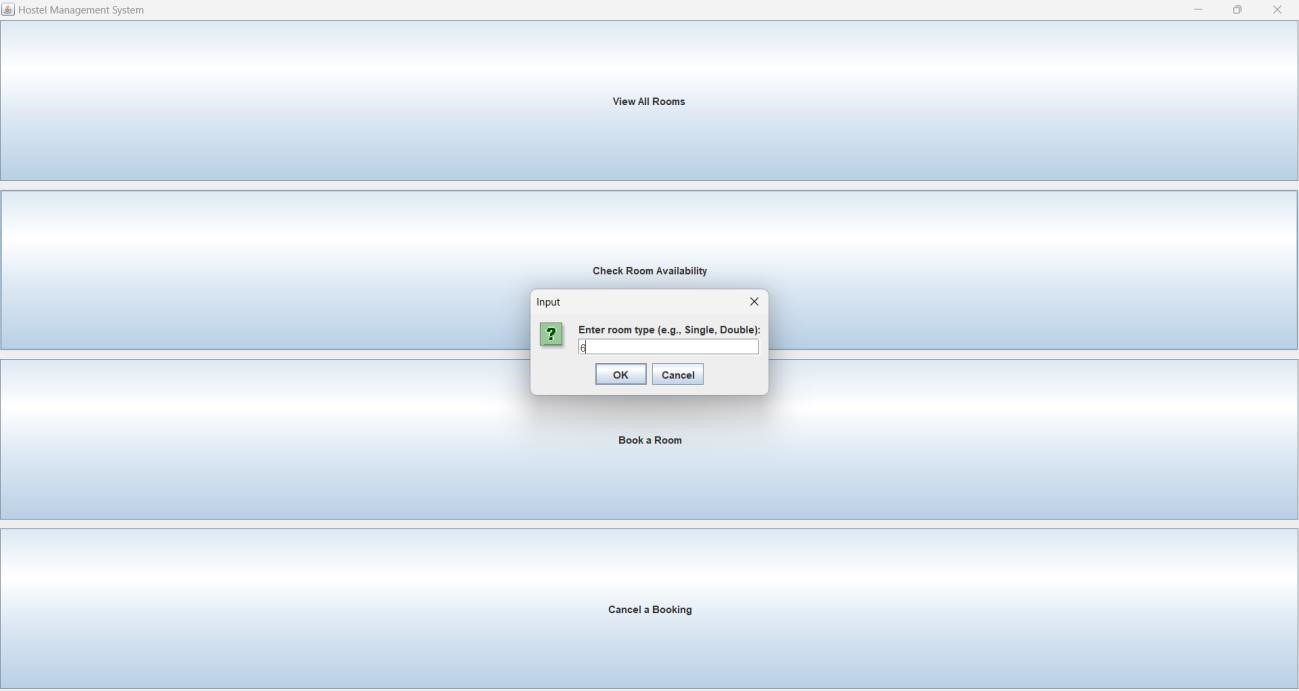
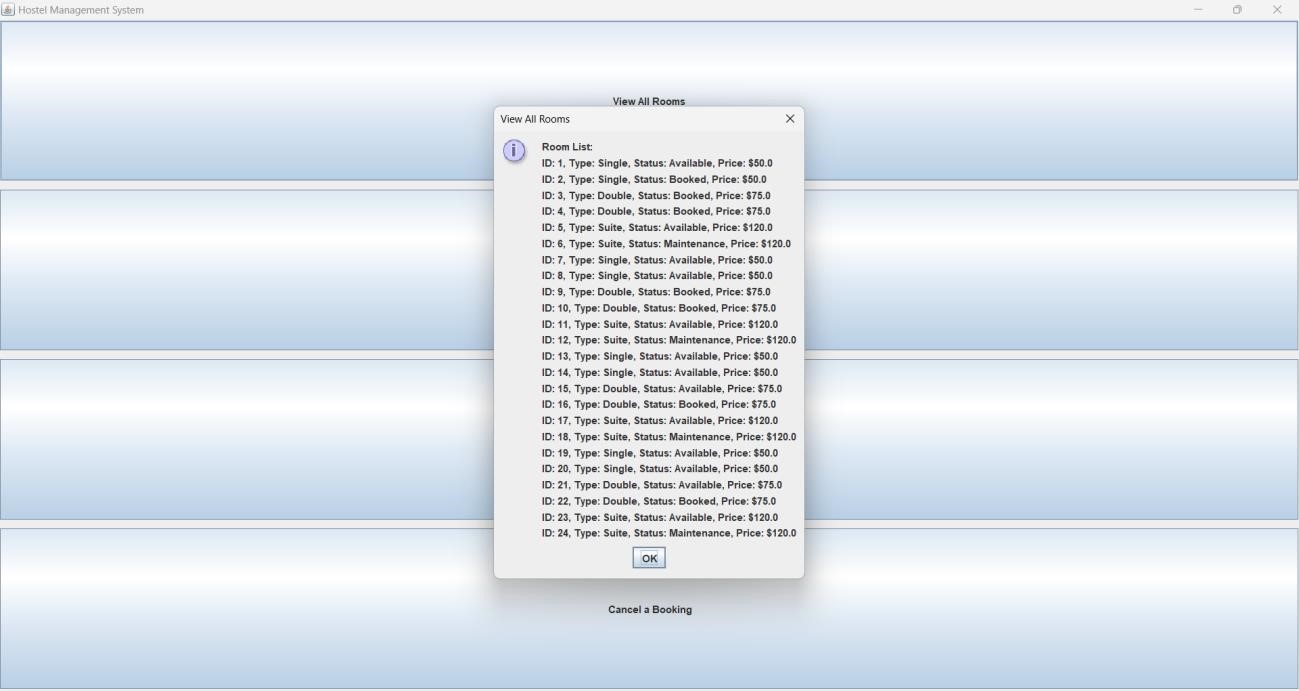
});

}

}

}

### CHAPTER 6



CONCLUSION

In conclusion, the **Hostel Management System** is an invaluable tool for efficiently handling room information, availability, and bookings, thus significantly streamlining the room management process. This system not only improves the organization of resident data and room allocations but also enhances the overall operational effectiveness of hostel facilities. By providing real-time tracking of room occupancy, facilitating seamless room bookings, and ensuring smooth check-in and check-out processes, the system meets the essential needs of both hostel administrators and residents.

Moreover, with automated billing, maintenance scheduling, and easy access to detailed reports, the Hostel Management System reduces the administrative burden on staff, allowing them to focus on enhancing the resident experience and maintaining a safe, comfortable living environment. This system plays a crucial role in creating a wellorganized, transparent, and secure hostel environment that meets the demands of modern-day residents and the operational needs of the hostel industry.

REFERENCES

1. https://developers.openshift.com/database/mysql.html
2. Web References- https://youtu.be/UbIIFLsEeiM