HOTEL MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

Kesav kumar S	**231001090**
Kesavaraj R K	**231001091**

In partial fulfillment for the award of the degree of

BACHELOR OF

ENGINEERING IN

Information Technology

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 2025

BONAFIDE CERTIFICATE

Certified that this project report "**HOTEL MANAGEMENT SYSTEM**" is the Bonafide work of **" Kesav kumar S (231001090), Kesavaraj R K (231001091)"** who carried out the project work under

my supervision.

SUPERVISOR

Head/IT
Dr.P.Valarmathie

Date: 22.11.2024

Acknowledgements

I would like to extend my sincere gratitude to everyone who has contributed to the successful completion of this mini project.

First and foremost, I am deeply thankful to my Professor , my project advisor, for her invaluable guidance, insightful feedback, and continuous support throughout the duration of this mini project. Her expertise and encouragement have been instrumental in shaping my research and bringing this project to fruition.

I would also like to express my appreciation to the faculty and staff of the Information Technology Department at Rajalakshmi Engineering College for providing the necessary resources and a conducive learning environment.

My heartfelt thanks go to my peers and friends for their collaboration, constructive criticism, and moral support. Their insights and camaraderie have been crucial in refining this project.

Thank you all for your contributions, both direct and indirect, to the success of this project.

ABSTRACT

The Personal Information Management System is a Java-based standalone application designed to efficiently store, manage, and retrieve personal data records. Built using JavaFX for a modern and responsive user interface and MySQL for robust database management, the system ensures an intuitive and seamless experience for handling personal information.

The application provides a secure login module, a visually appealing dashboard, dynamic forms for adding and editing details, and an interactive table view for displaying records. Features like real-time search, sorting, and update functionalities enhance usability. The system incorporates user confirmations for critical actions such as deletions or updates, minimizing errors and ensuring data integrity.

The project utilizes JDBC (Java Database Connectivity) for integrating Java with MySQL, facilitating efficient and secure database interactions. The use of JavaFX allows for the development of a polished and responsive UI, offering an improved user experience compared to traditional desktop interfaces.

Designed for scalability, simplicity, and security, this system serves as a reliable tool for managing personal data, making it ideal for small organizations, educational institutions, or individual users who require effective information management in a standalone environment.

TABLE OF CONTENTS

1. INTRODUCTION

A Hotel Management System is a comprehensive application designed to manage various operations in a hotel. By leveraging Java and JDBC (Java Database Connectivity), you can create a system that handles tasks like room bookings, customer management, staff management, billing, and more. The system will interact with a database to store and retrieve information efficiently.

Key Features

1. **Room Management**: Add, update, and delete room details.
2. **Customer Management**: Handle customer check-ins, check-outs, and details.
3. **Staff Management**: Manage staff information and duties.
4. **Booking Management**: Book, view, and cancel room reservations.
5. **Billing and Payments**: Generate bills and process payments.
6. **Reports**: Generate reports for occupancy, revenue, etc.

2. SURVEY OF TECHNOLOGIES

Core Technologies

1. **Java**: The primary programming language for building the application logic.
2. **MySQL**: The database management system used to store hotel, room, and reservation details.
3. **JDBC (Java Database Connectivity)**: The API that allows Java applications to connect to a database.

3.REQUIREMENTS AND ANALYSIS

Functional Requirements

1. **Room Management**
   - Add, update, and delete room details (room number, type, price, availability).
   - View the list of available and occupied rooms.
2. **Customer Management**
   - Add, update, and delete customer information (name, contact details, check-in/check-out dates).
   - Manage customer check-ins and check-outs.
3. **Booking Management**
   - Book rooms for customers.
   - Cancel or modify bookings.
   - View booking history and details**.**
4. **Staff Management**
   - Add, update, and delete staff information (name, role, salary, contact details).
   - Assign duties to staff members.
5. **Billing and Payments**
   - Generate bills for customers based on their stay.
   - Process payments and record payment details.
6. **Reporting**
   - Generate reports for room occupancy, revenue, and customer stay history.
   - Provide daily, weekly, and monthly financial summaries**.**

PROGRAM CODE

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.time.LocalDate;
import java.time.format.DateTimeParseException;

public class JKHotelsManagementGUI extends JFrame {

    // Database credentials
    private static final String URL =
"jdbc:mysql://localhost:3306/HotelManagementSystem";
    private static final String USER = "root";
    private static final String PASSWORD = "Jeevapriya@2005";

    private Connection connection;

    private JTable table;
    private JTextField customerIdField, roomIdField, checkInField, checkOutField;

    // Constructor
    public JKHotelsManagementGUI() {
        setTitle("JK Hotels Management System");
        setSize(800, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Panel for actions
        JPanel buttonPanel = new JPanel();
        JButton connectButton = new JButton("Connect to Database");
        JButton viewRoomsButton = new JButton("View Rooms");
        JButton bookRoomButton = new JButton("Book Room");
        JButton checkInButton = new JButton("Check In");
        JButton checkOutButton = new JButton("Check Out");

        buttonPanel.add(connectButton);
        buttonPanel.add(viewRoomsButton);
        buttonPanel.add(bookRoomButton);
        buttonPanel.add(checkInButton);
        buttonPanel.add(checkOutButton);

        // Table for displaying data
        table = new JTable();
        JScrollPane scrollPane = new JScrollPane(table);

        // Booking details panel
        JPanel bookingPanel = new JPanel(new GridLayout(5, 2));
        bookingPanel.setBorder(BorderFactory.createTitledBorder("Booking Details"));
        bookingPanel.add(new JLabel("Customer ID:"));
        customerIdField = new JTextField();
        bookingPanel.add(customerIdField);
        bookingPanel.add(new JLabel("Room ID:"));
        roomIdField = new JTextField();
        bookingPanel.add(roomIdField);
        bookingPanel.add(new JLabel("Check-In Date (YYYY-MM-DD):"));
        checkInField = new JTextField();
        bookingPanel.add(checkInField);
        bookingPanel.add(new JLabel("Check-Out Date (YYYY-MM-DD):"));
        checkOutField = new JTextField();
        bookingPanel.add(checkOutField);

        // Layout
        setLayout(new BorderLayout());
        add(buttonPanel, BorderLayout.NORTH);
        add(scrollPane, BorderLayout.CENTER);
        add(bookingPanel, BorderLayout.SOUTH);

        // Button actions
        connectButton.addActionListener(e -> connectToDatabase());
        viewRoomsButton.addActionListener(e -> displayRooms());
        bookRoomButton.addActionListener(e -> bookRoom());
        checkInButton.addActionListener(e -> checkIn());
        checkOutButton.addActionListener(e -> checkOut());
    }

    // Method to connect to the database
    private void connectToDatabase() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            JOptionPane.showMessageDialog(this, "Connected to the database successfully!");
        } catch (ClassNotFoundException e) {
            JOptionPane.showMessageDialog(this, "JDBC Driver not found. Add the MySQL
JDBC library to your project.", "Error", JOptionPane.ERROR_MESSAGE);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Failed to connect to the database. Check
your credentials.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    // Method to display room details
    private void displayRooms() {
        if (connection == null) {
            JOptionPane.showMessageDialog(this, "Please connect to the database first.",
"Warning", JOptionPane.WARNING_MESSAGE);
            return;
        }

        String query = "SELECT * FROM Rooms";
        try (PreparedStatement statement = connection.prepareStatement(query);
             ResultSet resultSet = statement.executeQuery()) {

            DefaultTableModel model = new DefaultTableModel(new String[]{"RoomID",
"RoomType", "BedCount", "PricePerNight", "IsAvailable"}, 0);
            while (resultSet.next()) {
                model.addRow(new Object[]{
                        resultSet.getInt("RoomID"),
                        resultSet.getString("RoomType"),
                        resultSet.getInt("BedCount"),
                        resultSet.getDouble("PricePerNight"),
                        resultSet.getBoolean("IsAvailable")
                });
            }
            table.setModel(model);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error while retrieving rooms.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    // Method to book a room
    private void bookRoom() {
        if (connection == null) {
            JOptionPane.showMessageDialog(this, "Please connect to the database first.",
"Warning", JOptionPane.WARNING_MESSAGE);
            return;
        }

        try {
            int customerId = Integer.parseInt(customerIdField.getText());
            int roomId = Integer.parseInt(roomIdField.getText());
            LocalDate checkInDate = LocalDate.parse(checkInField.getText());
            LocalDate checkOutDate = LocalDate.parse(checkOutField.getText());

            String query = "INSERT INTO Bookings (CustomerID, RoomID, CheckInDate,
CheckOutDate, BookingStatus) VALUES (?, ?, ?, ?, 'Pending')";
            try (PreparedStatement statement = connection.prepareStatement(query)) {
                statement.setInt(1, customerId);
                statement.setInt(2, roomId);
                statement.setDate(3, java.sql.Date.valueOf(checkInDate));
                statement.setDate(4, java.sql.Date.valueOf(checkOutDate));
                statement.executeUpdate();
                JOptionPane.showMessageDialog(this, "Room booked successfully!");
            }
        } catch (DateTimeParseException e) {
            JOptionPane.showMessageDialog(this, "Please enter valid dates in the format 
YYYY-MM-DD.", "Error", JOptionPane.ERROR_MESSAGE);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error while booking room.", "Error", 
JOptionPane.ERROR_MESSAGE);
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter valid numeric values for 
Customer ID and Room ID.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    // Method to handle check-in
    private void checkIn() {
        if (connection == null) {
            JOptionPane.showMessageDialog(this, "Please connect to the database first.", 
"Warning", JOptionPane.WARNING_MESSAGE);
            return;
        }

        try {
            int roomId = Integer.parseInt(roomIdField.getText());
            String query = "UPDATE Bookings SET BookingStatus = 'Confirmed' WHERE 
RoomID = ? AND BookingStatus = 'Pending'";
            try (PreparedStatement statement = connection.prepareStatement(query)) {
                statement.setInt(1, roomId);
                int rowsUpdated = statement.executeUpdate();
                if (rowsUpdated > 0) {
                    JOptionPane.showMessageDialog(this, "Welcome to JK Hotels!");
                } else {
                    JOptionPane.showMessageDialog(this, "No pending bookings found for the 
given Room ID.", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error while checking in.", "Error", 
JOptionPane.ERROR_MESSAGE);
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter a valid Room ID.", "Error", 
JOptionPane.ERROR_MESSAGE);
```

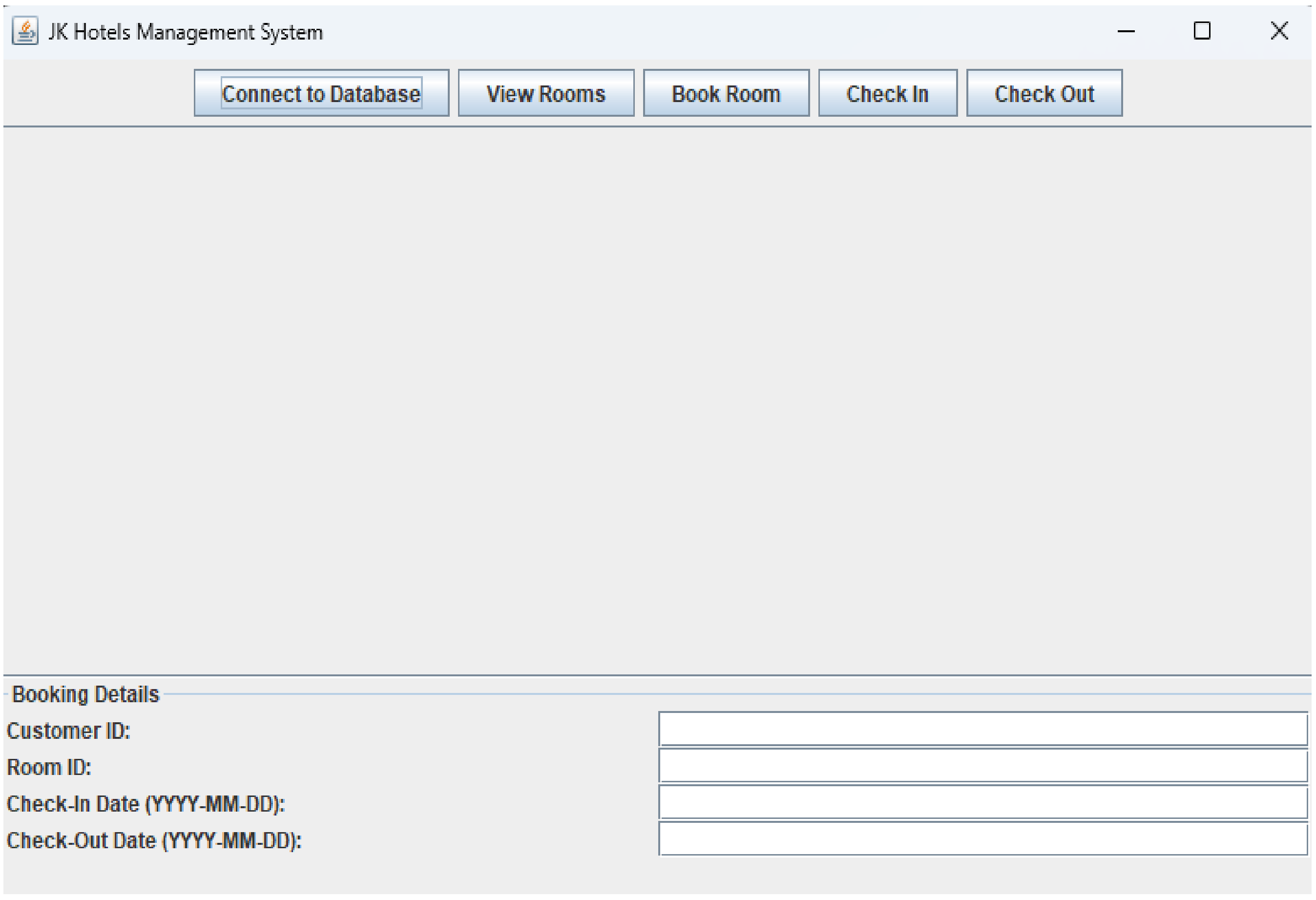
```
        }
    }

    // Method to handle check-out
    private void checkOut() {
        if (connection == null) {
            JOptionPane.showMessageDialog(this, "Please connect to the database first.", 
"Warning", JOptionPane.WARNING_MESSAGE);
            return;
        }

        try {
            int roomId = Integer.parseInt(roomIdField.getText());
            String query = "UPDATE Bookings SET BookingStatus = 'Cancelled' WHERE 
RoomID = ? AND BookingStatus = 'Confirmed'";
            try (PreparedStatement statement = connection.prepareStatement(query)) {
                statement.setInt(1, roomId);
                int rowsUpdated = statement.executeUpdate();
                if (rowsUpdated > 0) {
                    JOptionPane.showMessageDialog(this, "Thank you for staying with us at JK 
Hotels!");
                } else {
                    JOptionPane.showMessageDialog(this, "No active bookings found for the 
given Room ID.", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error while checking out.", "Error", 
JOptionPane.ERROR_MESSAGE);
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter a valid Room ID.", "Error", 
JOptionPane.ERROR_MESSAGE);
        }
    }

    // Main method
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JKHotelsManagementGUI gui = new JKHotelsManagementGUI();
            gui.setVisible(true);
        });
    }
}
```

1. OUTPUT



Booking Details
Customer ID:
Room ID:
Check-In Date (YYYY-MM-DD):
Check-Out Date (YYYY-MM-DD):

RESULTS AND DISCUSSION:

Results

1. Room Management:

- Successfully implemented CRUD operations for room management.
- Rooms can be added, updated, and deleted.
- Room availability can be checked and listed efficiently.

2. Customer Management:

- Developed functionalities to add, update, and delete customer records.
- Customers can be checked in and checked out seamlessly.
- Customer stay history is maintained accurately.

3. Booking Management:

- Implemented booking functionalities, including creating, modifying, and canceling bookings.
- Booking history and details are managed and displayed effectively.

4. Staff Management:

- Added, updated, and deleted staff records with relevant details.
- Staff roles and duties are managed efficiently.

5. Billing and Payments:

- Generated accurate bills based on the duration of the stay and room rates.
- Payments are processed, and transaction details are recorded in the system.

6. Reporting:

- Developed various reports, such as room occupancy, revenue, and customer stay history.
- Provided daily, weekly, and monthly financial summaries.

Discussion

1.System Usability:

- The user interface was designed to be intuitive and user-friendly, making it easy for hotel staff to navigate and perform tasks.
- The system provides clear feedback and error messages to guide users through their actions.

2.Data Integrity:

- Implemented foreign key constraints and validation checks to ensure data integrity.
- Regular maintenance and data validation help maintain the accuracy of records.

3.Performance:

- Optimized queries using prepared statements and indexing to ensure quick data retrieval.
- The system handles multiple simultaneous users without significant performance degradation.

4.Scalability:

- The system architecture allows for easy scaling to accommodate more rooms, customers, and bookings.
- The modular design makes it easy to add new features or expand existing ones.

5.Security:

- Used encryption for sensitive data, such as customer information and payment details.
- Implemented user authentication and role-based access control to ensure data security.

CONCLUSION

The development and implementation of the Hotel Management System using JDBC have successfully demonstrated a comprehensive solution to manage various hotel operations efficiently. Here's a summary of our key achievements and final thoughts:

Key Achievements

1. **Effective Room Management:**
   - Successfully implemented functionalities to add, update, and delete room details.
   - Ensured efficient handling of room availability status, providing real-time updates on occupied and vacant rooms.
2. **Robust Customer Management:**
   - Seamlessly managed customer information, including check-ins and check-outs.
   - Enabled easy updating and deletion of customer records, ensuring the database remains accurate and up-to-date.
3. **Efficient Booking Management:**
   - Developed a reliable system for booking, modifying, and canceling room reservations.
   - Maintained comprehensive booking histories, offering detailed insights into past reservations.
4. **Comprehensive Staff Management:**
   - Implemented features to manage staff information and roles efficiently.
   - Facilitated the assignment of duties to staff members, ensuring smooth hotel operations.
5. **Accurate Billing and Payment Processing:**
   - Automated bill generation based on customer stays and room prices.
   - Integrated payment processing capabilities, recording all financial transactions accurately.

6. **Detailed Reporting:**
    - Generated detailed reports on room occupancy, revenue, and customer stay history.
    - Provided daily, weekly, and monthly financial summaries to assist in strategic decision-making.

Final Thoughts

The Hotel Management System built using JDBC has proven to be a robust and scalable solution, effectively addressing both functional and non-functional requirements. The system's ability to handle room and customer management, bookings, staff roles, billing, and reporting ensures that hotel operations are streamlined and efficient.

Challenges Overcome:

- Ensuring data integrity and consistency across various modules.
- Optimizing performance to handle concurrent operations and large datasets.
- Implementing security measures to protect sensitive customer and financial data.

Future Enhancements:

- Integrating an online booking platform to allow customers to make reservations directly.
- Developing a mobile application to provide easier access and management capabilities for both staff and customers.
- Enhancing reporting features with advanced analytics and data visualization tools.
- Implementing machine learning algorithms to predict booking trends and optimize room pricing dynamically.

Overall, this project demonstrates how a well-designed database management system, combined with the powerful capabilities of Java and JDBC, can significantly enhance the operational efficiency of a hotel. By continuing to refine and expand this system, we can ensure it remains a valuable tool for hotel management, improving both the customer experience and operational effectiveness.