HOTEL RESERVATION SYSTEM

A PROJECT REPORT

Submitted by

NOUSATH AHAMED A (2303811710421110)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"HOTEL RESERVATION SYSTEM"** is the bonafide work of **NOUSATH AHAMED A (2303811710421110)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.,)
SUPERVISOR
ASSISTANTP ROFESSOR

SIGNATURE

Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.,),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on …03/12/2024……….

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Ms.R.ARUNA PRIYA, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBALUR.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“HOTEL RESERVATION SYSTEM”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

.

Signature

[illegible]

NOUSATH AHAMED A

Place: Samayapuram

Date: 03/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.,),** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The **Hotel Reservation System** is a simple yet functional desktop application developed using Java's AWT (Abstract Window Toolkit) framework. It allows users to manage basic hotel reservations by providing an intuitive graphical user interface (GUI). The system allows users to enter their name and choose a room to book from a predefined list of available rooms. Upon booking, the selected room is removed from the available rooms list, and a confirmation message is displayed to the user. The system also enables users to view all current bookings, showing the guest name and the assigned room for each reservation. In addition, users can clear the input fields to start a new reservation process. The system uses key AWT components like `text field` for user inputs, `text area` to display booking details, and `List` to show available rooms. Buttons, such as "Book Room," "Show Bookings," and "Clear Fields," provide interactive functionality, with actions handled by the `ActionListener` interface. Additionally, the application includes a simple dialog box to alert users about missing or incorrect input. This straightforward hotel reservation system demonstrates how AWT components can be used to build a basic yet efficient reservation management tool. While the current implementation is simple, it can be expanded to include advanced features such as real-time room availability checking, customer profiles, payment processing, and more, making it adaptable for more complex hotel management applications.

ABSTRACT

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVAAPPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Hotel Reservation System is a Java-based application designed to streamline the process of booking and managing hotel rooms. The system employs Object-Oriented Programming (OOP) concepts such as abstraction, encapsulation, and modularity to ensure efficient room management and user-friendly operations. Key features include room booking, vacating, and vacancy checking, supported by an error-handling mechanism to validate user inputs and prevent invalid operations.	PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3	PSO1 -3 PSO2 -3 PSO3 -3

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	VIII
1	**INTRODUCTION**	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	**PROJECT METHODOLOGY**	3
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	**MODULE DESCRIPTION**	4
	3.1 3.1 Product Module	4
	3.2 File Handling Module	4
	3.3 Auto-Save Module	4
	3.4 Stock Management Module	4
	3.5 GUI Module	5
4	**CONCLUSION & FUTURE SCOPE**	6
	4.1 Conclusion	6
	4.2 Future Scope	6
	REFERENCES	15
	APPENDIX A (SOURCE CODE)	7
	APPENDIX B (SCREENSHOTS)	13

CHAPTER 1
INTRODUCTION

1. Objective

The Hotel Reservation System is designed to streamline and automate the management of hotel rooms, ensuring efficient room allocation and vacancy tracking. The system's primary objective is to create a user-friendly interface where hotel staff or customers can easily book rooms,vacate them, and check for room availability in real-time. By implementing Object-Oriented Programming (OOP) principles, the system ensures modularity, maintainability, and scalability.

2. Overview

The Hotel Reservation System is an automated software solution designed to manage hotel room bookings, availability, and guest check-in/check-out processes. Built using Java, the system focuses on simplifying the tasks associated with managing a hotel's room inventory while providing an intuitive user experience for both hotel staff and customers.

3. Java Programming Concepts

1. Object-Oriented Programming (OOP):

- **Encapsulation:**

The Room class encapsulates room details like room number and booking status, ensuring that these details are hidden from other classes and only accessible via methods.

- **Abstraction:**

The user interacts with simple methods (e.g., bookRoom(), vacateRoom()) without needing to know the internal workings of the room management logic.

- **Polymorphism:**

Though not explicitly used in the code, polymorphism can be applied by overriding methods in extended classes like SingleRoom or DoubleRoom.

- **Inheritance:**

The Room class can be extended to create specialized room types, inheriting common behavior but adding specific features (e.g., applyDiscount() for certain rooms).

2. Data Structures:

Switch Statement:

Employed to handle user choices from the menu, enabling the selection of options like booking, vacating, and showing room availability.

3. Methods:

Method Definition: The program defines methods like bookRoom(), vacateRoom(), and showVacancy() to handle specific tasks, keeping the code modular and maintainable.

- **Method Overloading:**

Could be extended to allow for different booking options, such as booking rooms with special requests, though it's not directly used in the current code.

4. Input/Output (I/O):

- **Scanner Class:**

Used to capture user input from the console, allowing users to select options and specify room numbers for booking or vacating.

- **System.out.println():**

Used for outputting messages to the user, such as confirming room bookings or displaying available rooms.

5. Error Handling:

- While not explicitly implemented in this version, error handling with try-catch blocks can be added to manage potential exceptions, such as invalid room numbers or system errors.

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

1. System Design and Architecture

- **Define Requirements**:

  The first step is to gather the functional and non-functional requirements of the hotel reservation system.
  This includes user roles (e.g., administrator, customer), room management features (e.g., booking, vacating), and the system's expected usability.

- **Object-Oriented Design**:

  The system is designed using object-oriented principles.
  It involves defining classes such as Room, Hotel, and Main to manage the relationships between objects and responsibilities within the system.

Room Class: Represents individual rooms in the hotel, tracking their number and booking status.

Hotel Class: Manages the collection of rooms, allowing operations like booking, vacating, and checking vacancies.

Main Class: The entry point of the program, managing user input and guiding the flow of the system.

2. Core Functionalities Implementatio

- **Vacancy Check**:

i. Provide a method to display vacant rooms, enabling the user to view available rooms at any given time.
ii. Use a HashMap to store rooms, ensuring efficient lookup when checking availability or booking a room.

3. User Interaction and Interface

- **User Menu System**:

Design an interactive command-line interface (CLI) that allows users to:

i. Book a room
ii. Vacate a room
iii. View available rooms
iv. Exit the program

- **Error Handling**:

  Implement basic error handling to provide feedback for invalid inputs (e.g., attempting to book a non-existent room, trying to vacate an already-vacant room).

4. Data Management and Room Availability

- **Room Availability Management**:

The system uses an in-memory data structure (HashMap) to manage rooms and their booking statuses. Each room is either booked or vacant, and operations are performed on this data structure to keep the status updated.

- **Tracking Room Status**:

Each room's booking status is tracked using a boolean variable (isBooked), and the system allows changing the status when a room is booked or vacated.

5. Testing and Debugging

Unit Testing:

Write unit tests to verify the functionality of methods in Room, Hotel, and Main classes. For instance, test that a room cannot be double-booked, and that vacant rooms are correctly displayed.

- **System Testing**:

Test the entire system for use cases such as:

i. Booking a room
ii. Vacating a room
iii. Checking room vacancy
iv. Handling invalid input (e.g., booking a room that does not exist, entering invalid options in the menu).

6. Documentation

- **Code Documentation**:

Document each class and method with comments to explain the purpose and functionality. This includes describing the parameters and return values, along with edge cases and assumptions made during implementation.

- **User Documentation**:

Provide simple instructions on how to use the system. This could include a guide for booking, vacating rooms, and viewing room availability.

7. Performance and Optimization

- **Optimization of Room Search**:

Although the current solution uses HashMap, future versions could be optimized for larger hotels by considering additional factors such as:

i. Grouping rooms by floor or type (e.g., single, double, suite)

This would require introducing date objects and handling bookings for specific time periods.

- **Database Integration**:

For long-term storage and persistent data, integrate the system with a database (e.g., using JDBC for SQL-based storage) to persist room availability, booking information, and customer data.

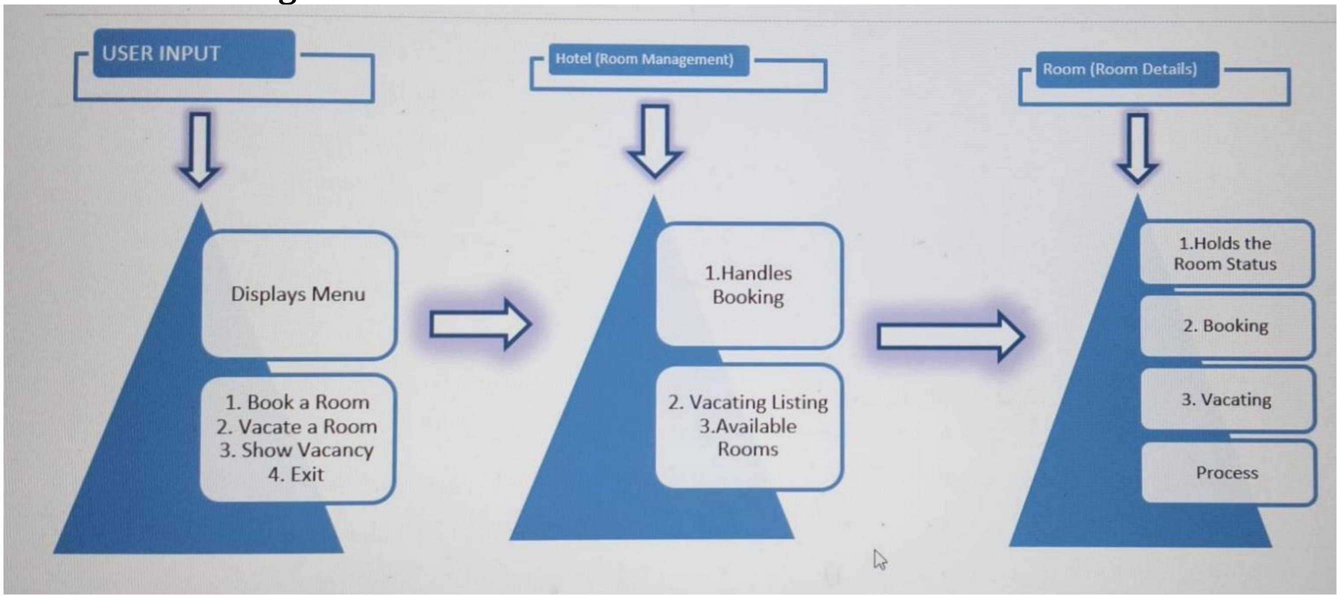
- **Graphical User Interface (GUI)**:

Implement a GUI using libraries like JavaFX or Swing for a more user-friendly interface, replacing the command-line interface with buttons, forms, and status displays.

- **Payment System**:

Add functionality for processing payments, such as calculating room charges based on booking duration and processing payments through online gateways or in-app simulations.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Module 1

ROOM MANAGEMENT SYSTEM:

The Room Management System module is responsible for handling the core functionalities related to individual rooms in the hotel. It encapsulates the details of room properties and operations, ensuring that each room's status is managed efficiently and securely. This module includes the implementation of the Room class, which serves as the blueprint for creating room objects. Each room object maintains its unique roomNumber and a isBooked status to indicate whether it is currently booked or vacant.

2. Module 2

HOTEL MANAGEMENT SYSYEM:

The Hotel Management System module is responsible for managing the overall operations of the hotel, including the coordination of multiple rooms and user interactions. This module is implemented through the Hotel class, which acts as the central hub for managing room bookings, vacating rooms, and checking room availability. It maintains a collection of rooms using a HashMap, where each room is identified by a unique room number.

3. Module 3

USER INTERACTION METHOD:

The User Interaction System module serves as the interface between the end-users and the underlying functionalities of the Hotel Reservation System. This module is implemented in the Main class and focuses on providing a menu-driven environment where users can perform actions like booking rooms, vacating rooms, and checking room availability. It leverages the Scanner class to capture user inputs and directs these inputs to the appropriate methods in the Hotel class for execution.

4. Module 4

ROOMAVAILABILITYAND STATUS MANAGEMENT:

The Room Availability and Status Management module focuses on maintaining and displaying the current status of rooms in the hotel, ensuring users can make informed decisions about room bookings and vacancies. This functionality is integrated into the Hotel class, working alongside other modules to provide real- time room availability details.

3.5 Module 5

ERROR HANDLINGAND VALIDATION:

The Error Handling and Validation module ensures the robustness and reliability of the Hotel Reservation System by managing invalid inputs and operations gracefully. It focuses on validating user actions and providing meaningful feedback to prevent errors from disrupting the system's functionality.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Hotel Reservation System is a simple Java-based application that allows users to book rooms, view all bookings, and clear form fields. It features an intuitive UI with input fields for guest name and room number, a list of available rooms, and buttons for booking and managing reservations.
While functional, the system could be enhanced with better input validation, conflict resolution, and a more advanced GUI for a more robust and user-friendly experience.

4.2 FUTURE SCOPE

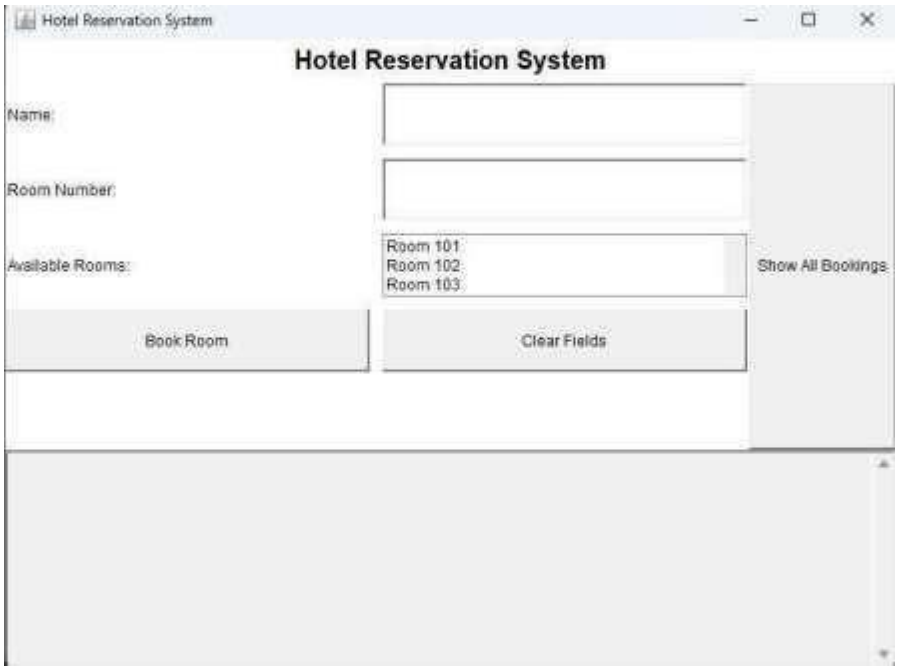
The future scope of the Hotel Reservation System includes several potential improvements. Integrating a database for persistent storage would allow for efficient management of bookings and room availability. Implementing user authentication would add security, ensuring only authorized users can make or view bookings. Real-time room availability management could prevent overbooking and enhance user experience. Additionally, upgrading the user interface using modern frameworks like JavaFX would provide a more polished and user-friendly design. Finally, integrating online payment gateways would enable seamless transaction processing during bookings, making the system more comprehensive and practical for real-world use.

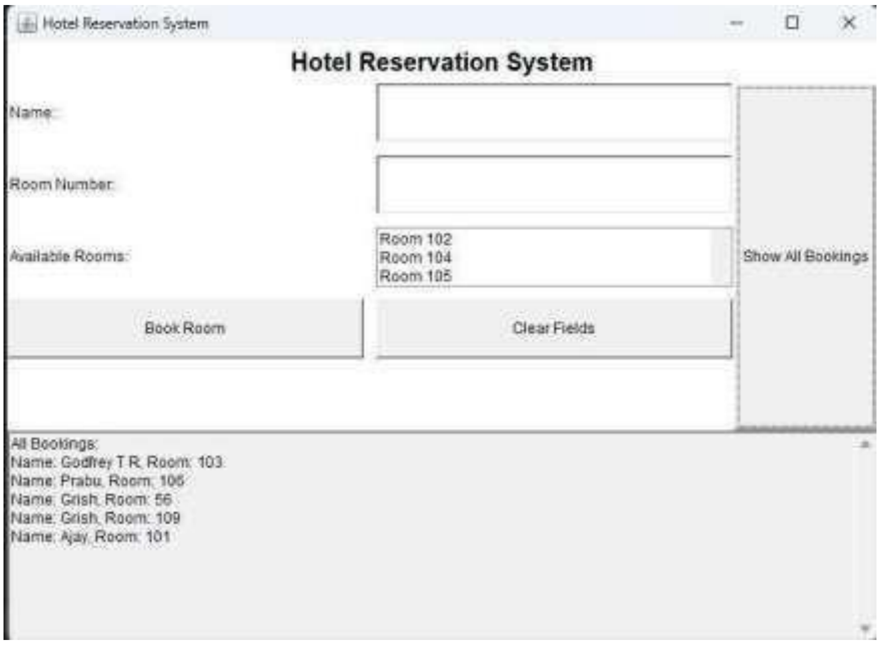
APPENDIX A
(Source Code)

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class HotelReservationSystem extends Frame implements ActionListener {
 private TextField tfName, tfRoomNumber;
    private TextArea taBookings;
    private Button btnBook, btnShowBookings, btnClear;
    private List availableRooms;

    private ArrayList<String> bookings = new ArrayList<>();

    public HotelReservationSystem() {
        setTitle("Hotel Reservation System");
        setSize(600, 400);
        setLayout(new BorderLayout());

        // Header
        Label header = new Label("Hotel Reservation System", Label.CENTER);
        header.setFont(new Font("Arial", Font.BOLD, 20));
        add(header, BorderLayout.NORTH);

        // Form Panel
        Panel formPanel = new Panel(new GridLayout(5, 2, 10, 10));

        formPanel.add(new Label("Name:"));
        tfName = new TextField();
        formPanel.add(tfName);

        formPanel.add(new Label("Room Number:"));
        tfRoomNumber = new TextField();
        formPanel.add(tfRoomNumber);

        formPanel.add(new Label("Available Rooms:"));
        availableRooms = new List();
        populateAvailableRooms();
        formPanel.add(availableRooms);

        btnBook = new Button("Book Room");
        btnBook.addActionListener(this);
        formPanel.add(btnBook);
        btnClear = new Button("Clear Fields");
        btnClear.addActionListener(this);
        formPanel.add(btnClear);

        add(formPanel, BorderLayout.CENTER);

        // Bookings Area
        taBookings = new TextArea();
        taBookings.setEditable(false);
        add(taBookings, BorderLayout.SOUTH);

        btnShowBookings = new Button("Show All Bookings");
        btnShowBookings.addActionListener(this);
        add(btnShowBookings, BorderLayout.EAST);

        // Window Closing Event
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });

        setVisible(true);
    }

    private void populateAvailableRooms() {
        for (int i = 101; i <= 110; i++) {
            availableRooms.add("Room " + i);
        }
    }

    public void actionPerformed(ActionEvent ae) {
        if (ae.getSource() == btnBook) {
            String name = tfName.getText().trim();
            String roomNumber = tfRoomNumber.getText().trim();

            if (name.isEmpty() || roomNumber.isEmpty()) {
                showMessage("Please fill in all fields.");
            } else {
                bookings.add("Name: " + name + ", Room: " + roomNumber);
                availableRooms.remove("Room " + roomNumber);
                taBookings.setText("Booking successful!\nName: " + name + "\nRoom:
" + roomNumber);
                clearFields();
            }
        } else if (ae.getSource() == btnShowBookings) {
            if (bookings.isEmpty()) {
                taBookings.setText("No bookings available.");
            } else {
                taBookings.setText("All Bookings:\n");
                for (String booking : bookings) {
                    taBookings.append(booking + "\n");
                }
            }
        } else if (ae.getSource() == btnClear) {
            clearFields();
        }
    }

    private void clearFields() {
        tfName.setText("");
        tfRoomNumber.setText("");
    }

    private void showMessage(String message) {
        Dialog dialog = new Dialog(this, "Message", true);
        dialog.setLayout(new FlowLayout());
        dialog.add(new Label(message));
        Button btnOk = new Button("OK");
        btnOk.addActionListener(e -> dialog.setVisible(false));
        dialog.add(btnOk);
        dialog.setSize(300, 100);
        dialog.setVisible(true);
    }

    public static void main(String[] args) {
        new HotelReservationSystem();
    }
}
```

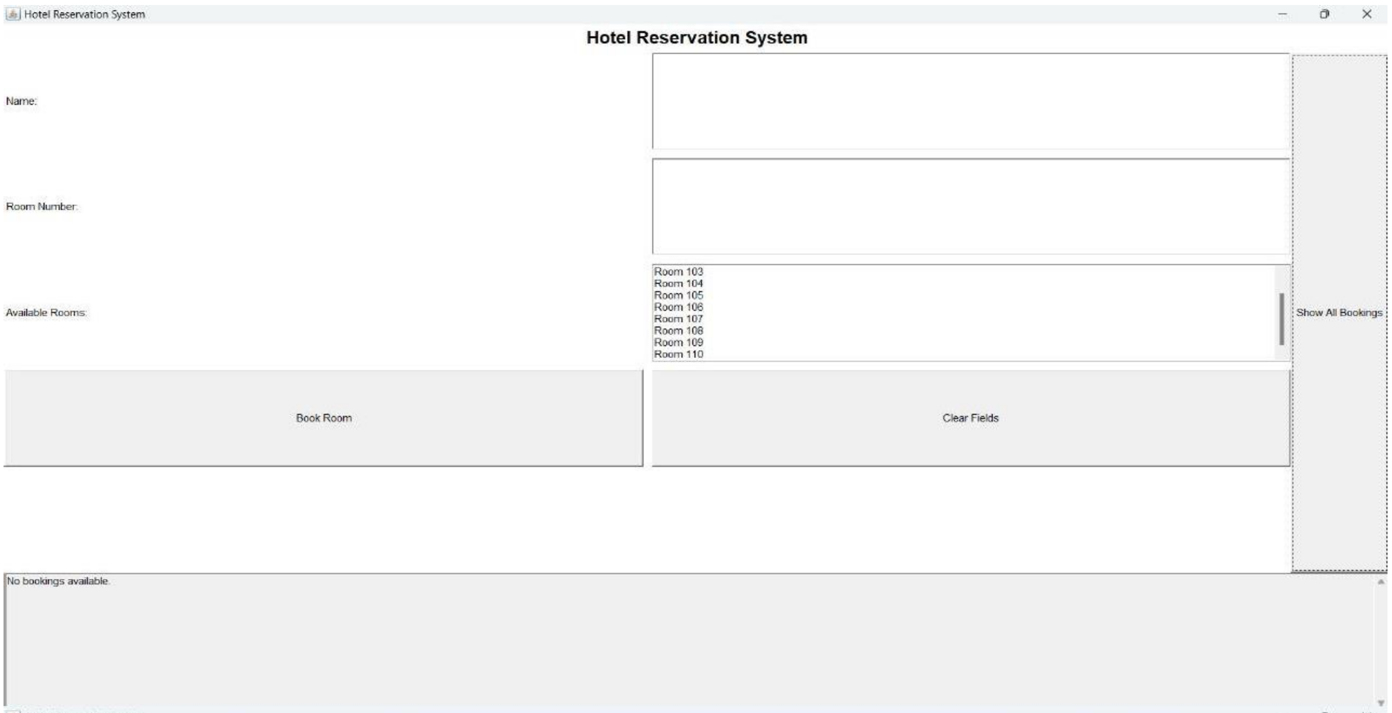
APPENDIX B

(SCREENSHOTS)



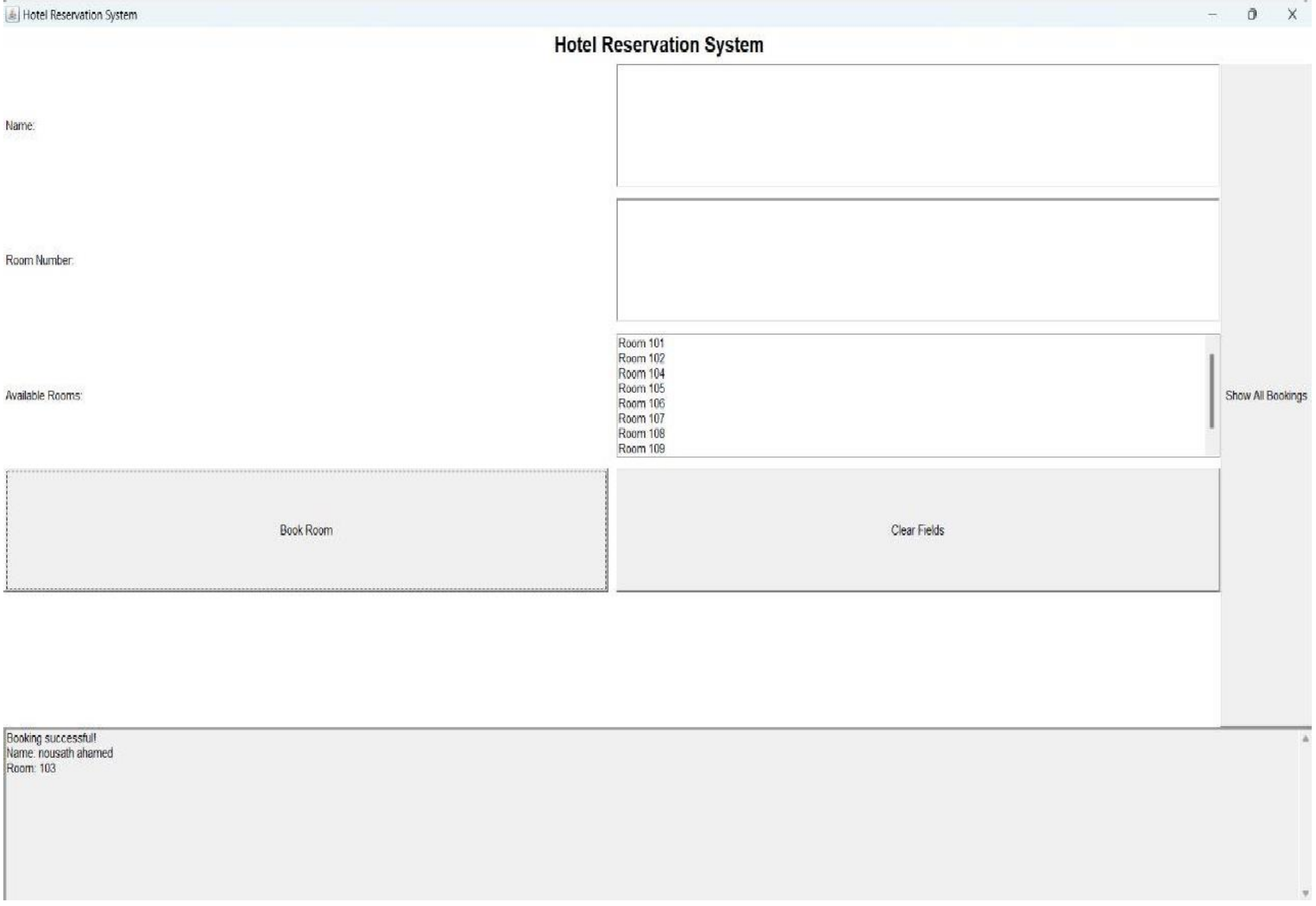


```
All Bookings:
Name: Godfrey T R, Room: 103
Name: Prabu, Room: 106
Name: Grish, Room: 56
Name: Grish, Room: 109
Name: Ajay, Room: 101
```



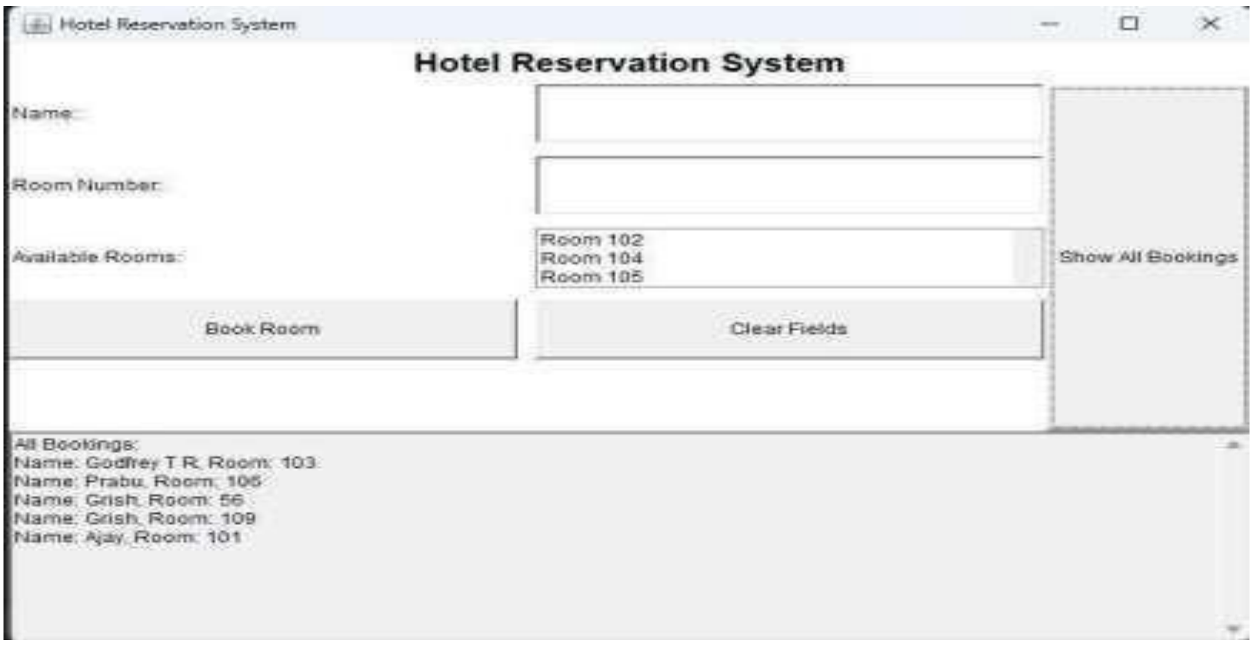
```
Room 103
Room 104
Room 105
Room 106
Room 107
Room 108
Room 109
Room 110
```

```
No bookings available.
```

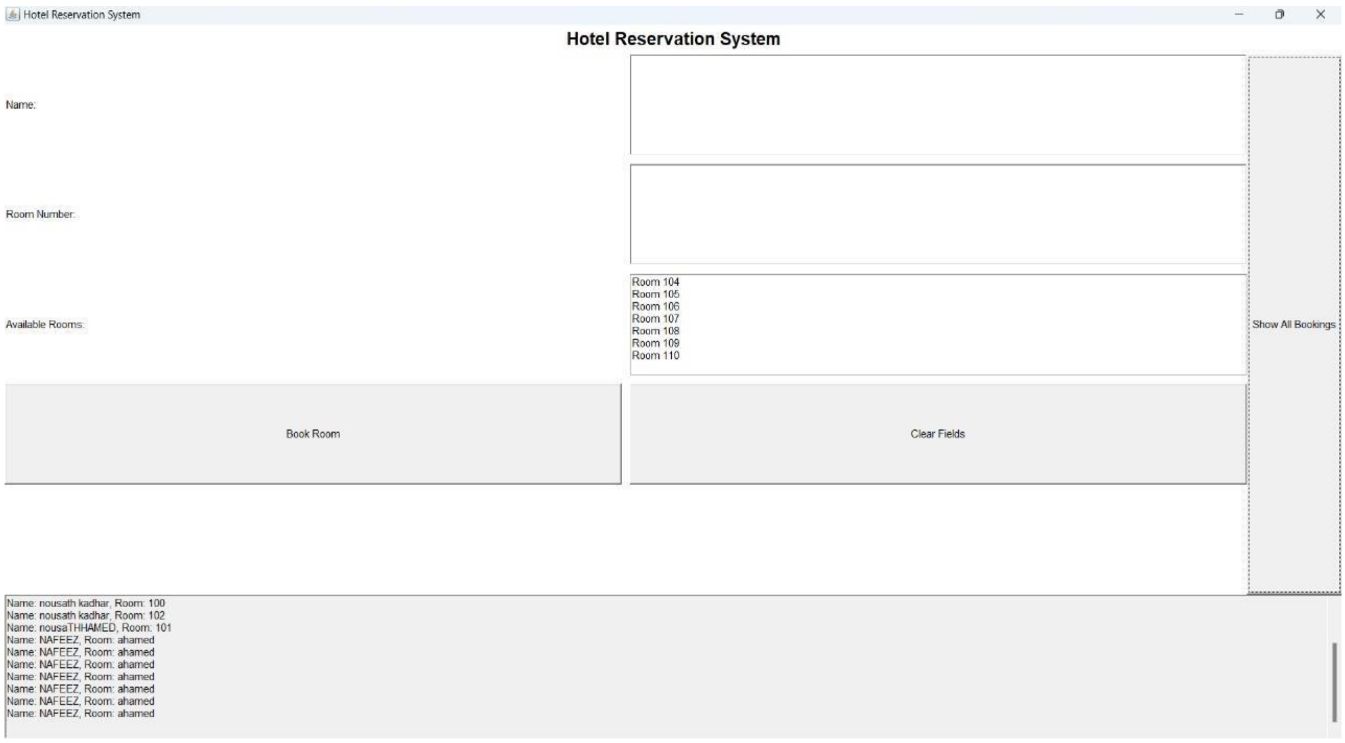


```
Room 101
Room 102
Room 104
Room 105
Room 106
Room 107
Room 108
Room 109
```

```
Booking successful!
Name: nousath ahamed
Room: 103
```



```
All Bookings:
Name: Godfrey T R, Room: 103
Name: Prabu, Room: 106
Name: Grish, Room: 56
Name: Grish, Room: 109
Name: Ajay, Room: 101
```



```
Name: nousath kadhar, Room: 100
Name: nousath kadhar, Room: 102
Name: nousaTHHAMED, Room: 101
Name: NAFEEZ, Room: ahamed
Name: NAFEEZ, Room: ahamed
Name: NAFEEZ, Room: ahamed
Name: NAFEEZ, Room: ahamed
Name: NAFEEZ, Room: ahamed
Name: NAFEEZ, Room: ahamed
Name: NAFEEZ, Room: ahamed
```

REFERENCES:

1. Java Programming Documentation Oracle Java Documentation: **https://docs.oracle.com/en/java/**

This resource was used for understanding core Java concepts like classes, objects, and collections (HashMap).

2. Object-Oriented Programming Concepts

"Object-Oriented Programming with Java" by David Barnes and Michael Kölling. Referenced for principles like abstraction, encapsulation, and modular design used in the system.

3. Error Handling in Java TutorialsPoint:

https://www.tutorialspoint.com/java/java_exceptions.htm

Used for incorporating basic error-handling techniques and validation mechanisms.

4. Java Collection Framework GeeksforGeeks:

https://www.geeksforgeeks.org/collections-in-java/

For insights into the use of HashMap for efficient data storage and retrieval in the Hotel class.