COLLEGE HOSTEL BOOKING SYSTEM

A PROJECT REPORT

Submitted by

SHAJATHI BEE M (2303811710422141)

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 **“COLLEGE HOSTEL BOOKING SYSTEM”** is the bonafide work of **SHAJATHI BEE M (2303811710422141)** 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 [illegible]	CGB1201-JAVA PROGRAMMING Mr.MALARMANNAN A, M.E., [illegible] ASSISTANTP ROFESSOR [illegible]
SIGNATURE	**SIGNATURE**
Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. A. Malarmannan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on …06/12/2024…………

CGB1201-JAVA PROGRAMMING Mr. M. SARAVANAN, M.E., INTERNAL EXAMINER ASSISTANT PROFESSOR [illegible]	CGB1201-JAVA PROGRAMMING Mr.R. KARTHIK, M.E., EXTERNAL EXAMINER ASSISTANT PROFESSOR 8138-SCE, TRICHY. [illegible] 06/12/2024
INTERNAL EXAMINER	EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“COLLEGE HOSTEL BOOKING 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]

SHAJATHI BEE M

Place: Samayapuram

Date:06/12/2024

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 **MR. A. MALARMANNAN, M.E.,** 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 Room Booking Management Application is a comprehensive system developed in Java to streamline the process of managing and booking rooms in a student hostel. The application allows students to view available rooms in real-time, ensuring that room availability is always up to date. A key feature of the system is its booking management module, which dynamically updates the availability of rooms upon the confirmation of a booking. To ensure fair and efficient allocation, the system prioritizes booking requests based on specific criteria, such as the application date and any special requirements submitted by the students. The application is designed to prevent overbooking by ensuring that room assignments are updated instantly after each booking. Students can submit their requests, which are processed by the system, and rooms are assigned accordingly. In case of multiple requests for the same room, the system evaluates which request should be prioritized based on predefined criteria. This ensures that room allocation is managed fairly and effectively.This application provides a robust framework for managing room bookings in a college or hostel environment, ensuring both real-time updates and efficient management of room assignments. It can be further enhanced with additional features such as user authentication, reporting, and database integration for better scalability.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Room Booking Management Application is a comprehensive system developed in Java to streamline the process of managing and booking rooms in a student hostel. The application allows students to view available rooms in real-time, ensuring that room availability is always up to date. A key feature of the system is its booking management module, which dynamically updates the availability of rooms upon the confirmation of a booking. To ensure fair and efficient allocation, the system prioritizes booking requests based on specific criteria, such as the application date and any special requirements submitted by the students.	**PO1 -3** **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**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The goal of this project is to create a system that allows students to view and book hostel rooms in real-time, with a booking management module that tracks room availability and prioritizes requests based on specific criteria. The system will update room availability dynamically and handle various booking requests while ensuring no overbooking occurs. Additionally, the system will prioritize certain requests, for example, based on the date of application or special requirements like accessibility features.

1.2 Overview

The **College Hostel Booking System** is a Java-based application designed to manage the booking and allocation of rooms in a college hostel. The system provides a real-time view of available rooms, allowing students to make booking requests and receive immediate confirmation if the room is available. The application is built with the goal of ensuring efficient room management by dynamically updating room availability and preventing overbooking. It also incorporates a prioritization mechanism to handle booking requests based on specific criteria, such as the student's application date or special requirements (e.g., accessibility needs). The system uses key Java programming concepts such as object-oriented design, data structures (like ArrayList, HashMap, and PriorityQueue), and synchronization to manage concurrent user interactions and ensure that no conflicts arise when multiple students attempt to book rooms at the same time.

1.3 Java Programming Concepts

1.3.1 Classes:

We will define several classes such as `Room`, `Booking`, `Student`, `Room- Manager`, etc. to represent different entities in the system. Each class will have attributes and methods related to its role in the system.

1.3.2 Data Structures:

- ArrayList: To store the list of rooms and bookings.
- HashMap: To map student IDs to their bookings or to store room availability based on room IDs.
- PriorityQueue: To handle prioritization of booking requests, e.g., prioritizing students based on booking date or special needs.

1.3.3 Control Flow:

- Implement the logic for booking rooms, checking availability, and updating sta- tuses.
- Conditional statements (if-else) to check for availability, handle booking con- firmation, and enforce rules.
- Loops to iterate through available rooms, view bookings, and process requests.

1.3.4 Synchronization:

To handle concurrent access to the booking system, use synchronization to ensure that only one student can book.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the **College Hostel Booking System** involves several key stages, each focused on delivering a robust and user-friendly application. The primary goals include developing the system's core functionalities, integrating real-time updates, ensuring synchronization for concurrent access, and implementing a prioritization mechanism for booking requests. Below is a breakdown of the proposed work:

2.1.1 System Design & Architecture:

- **Object-Oriented Design**: The system will be structured using object-oriented programming principles, with key classes such as `Room`, `Booking`, `Student`, and `RoomManager` to encapsulate the different components of the system.
- **Room Management Module**: A central part of the system is the `RoomManager`, responsible for tracking room availability, handling booking requests, and ensuring that room statuses are updated in real-time. This module will interact with other components to manage the booking flow and prioritize requests.

2.1.2 Room and Booking Management:

- **Room Class**: Implement the `Room` class to represent individual hostel rooms, each with attributes like room ID, availability status, and special requirements.

- **Booking Logic**: Develop the `Booking` class to track booking details such as the student's ID, the room assigned, booking date, and booking status (e.g., pending, confirmed, canceled). The system will ensure that bookings are only confirmed when a room is available.

2.1.3 Prioritization of Booking Requests:

- Implement a **PriorityQueue** to prioritize room assignments based on criteria such as the student's application date, special needs (e.g., accessible rooms), or urgency of the request. This will allow the system to handle booking re- quests fairly and efficiently.
- The `RoomManager` will handle prioritization logic and assign rooms ac- cordingly, ensuring that students with more urgent needs or earlier applica- tions are prioritized.

2.1.4 Concurrency Handling & Synchronization:

- To handle multiple students attempting to book rooms simultaneously, the system will implement **synchronization techniques**. By using `syn-chronized` blocks or methods, the system will ensure that room availability is accurately updated without conflicts or race conditions.
- Proper concurrency management will prevent overbooking by ensuring that only one student can book a particular room at a time.

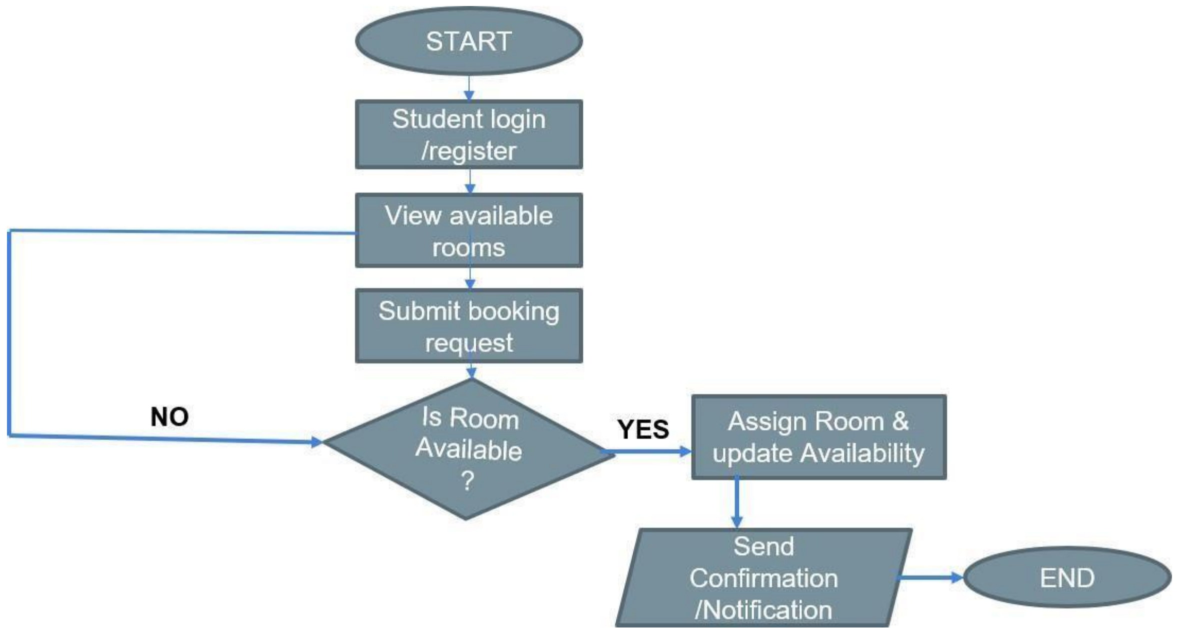
2.1.5 User Interaction Interface:

- **Command-Line Interface (CLI)**: A simple CLI will be implemented initially, allowing students and administrators to interact with the system.

2.1.6 Real-Time Updates:

- The system will provide real-time updates on room availability, ensuring that once a booking is confirmed, the room status is immediately updated across all user sessions to prevent overbooking.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 User Interface Module

The **User Interface (UI) Module** serves as the primary point of interaction between the students and the hostel booking system. Built using Java Swing or JavaFX, this module provides an intuitive and interactive graphical interface that allows users to view available rooms, submit booking requests, and manage their account details.

Key Functions:

- **View Available Rooms**: The UI module displays a list of rooms along with details such as room type (AC/Non-AC), capacity, and current availability status.
- **Search and Filter**: Users can filter rooms based on specific criteria, such as location, amenities, capacity, or price.
- **Submit Booking Requests**: The system provides an interface where students can enter booking details such as personal information, room preferences, and any special requirements.
- **Login and Registration**: The system includes a secure login and registration process, ensuring that only authenticated users can make bookings. This feature is implemented with Java Swing or JavaFX components.

3.2 Real Time Availability Module

The **Real-Time Availability Module** is responsible for managing the availability status of rooms. It continuously monitors the status of rooms (whether they are available, booked, or reserved) and updates the system accordingly.

module ensures that the information presented to the user is always up-to-date, preventing the booking of rooms that are already taken and displaying only available options

Key Functions:

- **Dynamic Room Status Updates**: The module uses multi-threading to dynamically update the availability status of rooms in real time, ensuring that the room data is accurate at all times.
- **Observer Design Pattern**: It employs the Observer Design Pattern, where changes in the availability of rooms trigger automatic updates to the user interface. As a result, users are notified of availability changes immediately.
- **Data Synchronization**: The system is designed to synchronize room data across the application, ensuring that all users see the most current room availability and status. This is achieved by integrating the availability module with the central database

3.3 Booking Management Module

The **Booking Management Module** is the core of the system, handling the actual booking requests, processing them, and ensuring that bookings are handled fairly and efficiently. This module receives booking requests from students, processes them based on available room status, and ensures the proper prioritization of requests based on predefined criteria.

Key Functions:

- **Request Processing**: Booking requests from students are received through the UI and placed into a queue for processing. The module checks room availability and confirms or denies the booking based on available rooms.
- **Prioritization of Requests**: The system uses a **Priority Queue** to prioritize booking requests. Requests are prioritized based on application date, student needs (e.g., accessibility requirements), and urgency, ensuring fair distribution of rooms.
- **Booking Confirmation**: Once a booking request is confirmed, the room status is updated, and the system generates confirmation details that are sent to the user.
- **Concurrency Control**: The system uses synchronized methods to manage simultaneous booking requests and prevent race conditions. This ensures that only one student can book a room at a time, avoiding overbooking scenarios.

3.4 Notification and Confirmation Module

The Notification and Confirmation Module is responsible for sending basic notifications to students once their booking request has been processed. At present, the module sends a simple confirmation message indicating whether the booking request has been successfully received or denied. However, the detailed notification system, including email notifications and in-app alerts with booking specifics (such as confirmation number, room details, etc.), will be implemented in the future.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

This project demonstrates the effectiveness of a Java-based room booking system in simplifying hostel management. Real-time updates, efficient prioritization, and user-friendly features make it an ideal solution. Future enhancements could include mobile support and advanced analytics to predict booking trends.

This project demonstrates how technology can streamline hostel management by offering real-time updates, prioritizing booking requests, and providing a user-friendly interface. The system's modular design ensures scalability, making it easy to implement further enhancements such as mobile access and advanced analytics for demand prediction.

The successful implementation and testing of the system show that it can significantly improve the hostel booking process, providing students with an easy-to-use platform and ensuring fairness in room allocation. Future work could focus on integrating additional features like mobile apps, advanced reporting tools, or AI-based analytics to predict booking trends and optimize room allocation further.

Overall, the project highlights the potential of Java-based applications in developing efficient and reliable systems for managing real-time data and user interactions in real-world scenarios like hostel room bookings.

4.2 FUTURE SCOPE

The future scope for the Hostel Booking System includes several exciting enhancements and optimizations. First, integrating a **database system** like MySQL or SQLite would allow for persistent storage of room bookings and student data, making the system scalable and reliable. A more sophisticated **Graphical User Interface (GUI)** could be developed using JavaFX or Swing, improving the user experience with a modern, intuitive design. Additionally, a **mobile application** for Android and iOS platforms could make the system more accessible to students on-the-go, while **email and SMS notifications** could be implemented to keep students updated on booking statuses in real time.

Another major improvement would be the development of an **Admin Dashboard**, which would enable administrators to manage bookings, view reports, and handle student queries more efficiently. The system could also benefit from **advanced room booking logic**, including prioritizing rooms based on special requirements or offering multi-room bookings for group events. Furthermore, integrating **booking history and analytics** features would allow both students and administrators to track booking trends and optimize room management.

To further enhance user interaction, a **real-time chat support system** could be added, enabling students to directly communicate with administrators. The system's **security** could be bolstered with **two-factor authentication** and encrypted passwords, ensuring safe user data handling. Lastly, applying **machine learning algorithms** could help predict booking trends and optimize room assignments, offering a smarter, more efficient booking process. These enhancements would not only improve the functionality of the system but also ensure a more seamless and secure user experience.

APPENDIX A

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

class RoomBookingApp extends Frame implements ActionListener {
    // Components
    Label lblWelcome, lblUsername, lblPassword, lblConfirmPassword, lblSpecialReq;
    TextField tfUsername, tfPassword, tfConfirmPassword, tfSpecialReq;
    Button btnLogin, btnRegister, btnRoomBook, btnUpdateRooms, btnExit, btnSubmit, btnBack;
    TextArea taRooms;
    Panel panelLogin, panelRegister, panelBooking, panelMenu;
    HashMap<String, String> users = new HashMap<>();
    HashMap<String, Boolean> rooms = new HashMap<>();
    String currentUser = "";

    RoomBookingApp() {
        // Initialize Room Data
        for (int i = 1; i <= 10; i++) {
            rooms.put("Room " + i, true);
        }

        // Frame Setup
        setTitle("Room Booking Application");
        setSize(500, 500);
```

```
setLayout(new CardLayout());
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        System.exit(0);
    }
});

// Panels
panelLogin = new Panel();
panelRegister = new Panel();
panelBooking = new Panel();
panelMenu = new Panel();

panelLogin.setLayout(new GridLayout(4, 2));
panelRegister.setLayout(new GridLayout(5, 2));
panelBooking.setLayout(new GridLayout(2, 1));
panelMenu.setLayout(new GridLayout(5, 1));

// Login Panel
lblUsername = new Label("Username:");
lblPassword = new Label("Password:");
tfUsername = new TextField();
tfPassword = new TextField();
tfPassword.setEchoChar('*');
btnLogin = new Button("Login");
btnRegister = new Button("Register");
btnLogin.addActionListener(this);
btnRegister.addActionListener(this);
panelLogin.add(lblUsername);
panelLogin.add(tfUsername);
panelLogin.add(lblPassword);
panelLogin.add(tfPassword);
panelLogin.add(btnLogin);
panelLogin.add(btnRegister);

// Register Panel
lblConfirmPassword = new Label("Confirm Password:");
tfConfirmPassword = new TextField();
tfConfirmPassword.setEchoChar('*');
btnSubmit = new Button("Submit");
btnBack = new Button("Back");
btnSubmit.addActionListener(this);
btnBack.addActionListener(this);
panelRegister.add(lblUsername);
panelRegister.add(tfUsername);
panelRegister.add(lblPassword);
panelRegister.add(tfPassword);
panelRegister.add(lblConfirmPassword);
panelRegister.add(tfConfirmPassword);
panelRegister.add(btnSubmit);
panelRegister.add(btnBack);

// Menu Panel
btnRoomBook = new Button("Book Room");
btnUpdateRooms = new Button("View/Update Rooms");
btnExit = new Button("Exit");
btnRoomBook.addActionListener(this);
btnUpdateRooms.addActionListener(this);
```

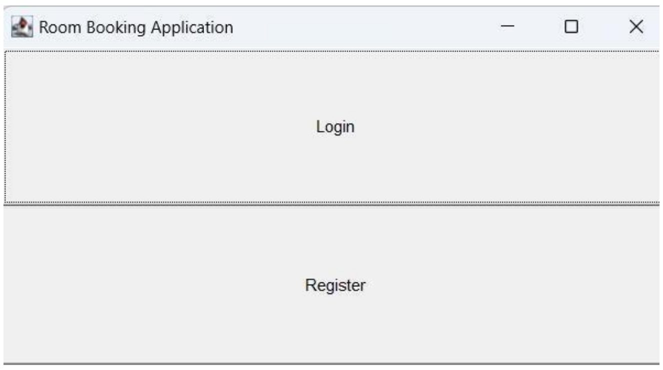
```
    btnExit.addActionListener(this);
    panelMenu.add(new Label("Welcome! Select an Option"));
    panelMenu.add(btnRoomBook);
    panelMenu.add(btnUpdateRooms);
    panelMenu.add(btnExit);

    // Booking Panel
    lblSpecialReq = new Label("Special Request:");
    tfSpecialReq = new TextField();
    taRooms = new TextArea();
    taRooms.setEditable(false);
    btnSubmit = new Button("Book");
    btnBack = new Button("Back");
    btnSubmit.addActionListener(this);
    btnBack.addActionListener(this);
    panelBooking.add(lblSpecialReq);
    panelBooking.add(tfSpecialReq);
    panelBooking.add(taRooms);
    panelBooking.add(btnSubmit);
    panelBooking.add(btnBack);

    // Add Panels
    add(panelLogin, "Login");
    add(panelRegister, "Register");
    add(panelMenu, "Menu");
    add(panelBooking, "Booking");

    showLogin();
}
```

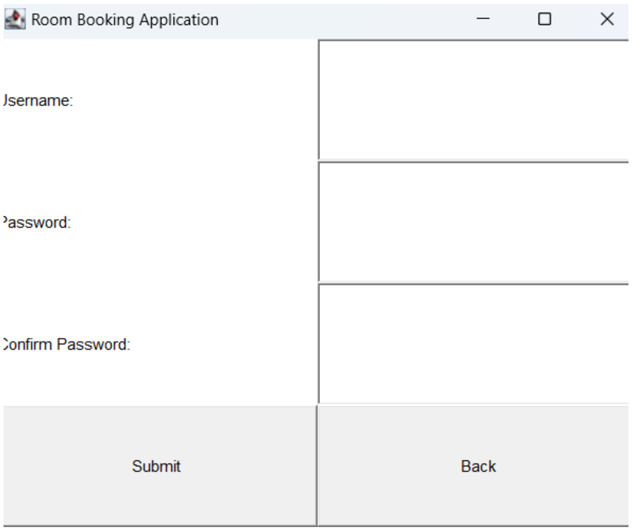
```
void showLogin() {
    CardLayout cl = (CardLayout) getLayout();
    cl.show(this, "Login");
}

void showRegister() {
    CardLayout cl = (CardLayout) getLayout();
    cl.show(this, "Register");
}

void showMenu() {
    CardLayout cl = (CardLayout) getLayout();
    cl.show(this, "Menu");
}

void showBooking() {
    CardLayout cl = (CardLayout) getLayout();
    taRooms.setText("Available Rooms:\n");
    for (Map.Entry<String, Boolean> entry : rooms.entrySet()) {
        if (entry.getValue()) {
            taRooms.append(entry.getKey() + "\n");
        }
    }
    cl.show(this, "Booking");
}

public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();
```

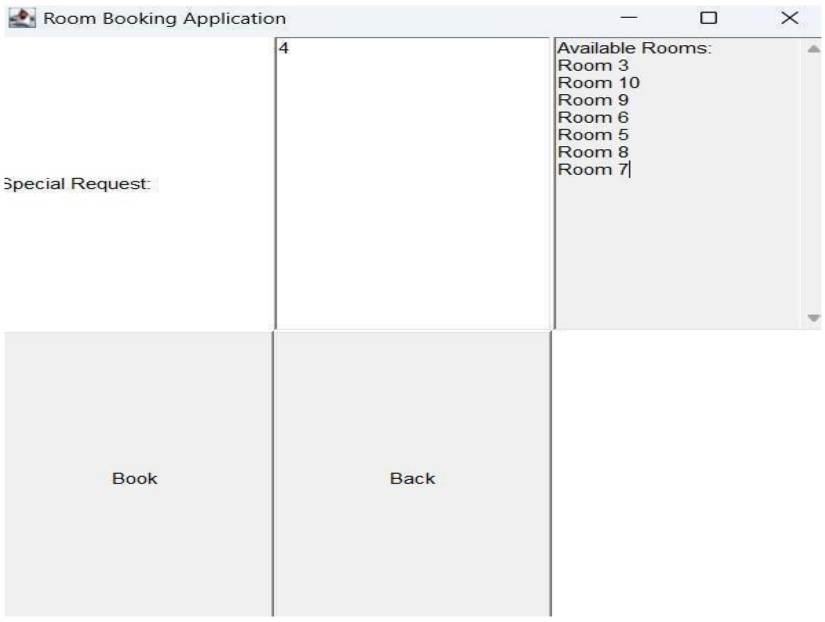
```
        if (command.equals("Login")) {
            String username = tfUsername.getText().trim();
            String password = tfPassword.getText().trim();
            if (users.containsKey(username) && users.get(username).equals(password))
{
                currentUser = username;
                showMenu();
            } else {
                showMessage("Invalid login. Please try again.");
            }
        } else if (command.equals("Register")) {
            showRegister();
        } else if (command.equals("Submit")) {
            String username = tfUsername.getText().trim();
            String password = tfPassword.getText().trim();
            String confirmPassword = tfConfirmPassword.getText().trim();
            if (!password.equals(confirmPassword)) {
                showMessage("Passwords do not match. Try again.");
            } else if (users.containsKey(username)) {
                showMessage("Username already exists. Choose another.");
            } else {
                users.put(username, password);
                showMessage("Registration successful!");
                showLogin();
            }
        } else if (command.equals("Back")) {
            showLogin();
        } else if (command.equals("Book Room")) {
            showBooking();
        } else if (command.equals("View/Update Rooms")) {
            taRooms.setText("Available Rooms:\n");
            for (Map.Entry<String, Boolean> entry : rooms.entrySet()) {
                taRooms.append(entry.getKey() + " - " + (entry.getValue() ? "Available" :
"Booked") + "\n");
            }
        } else if (command.equals("Book")) {
            String specialReq = tfSpecialReq.getText().trim();
            for (Map.Entry<String, Boolean> entry : rooms.entrySet()) {
                if (entry.getValue()) {
                    rooms.put(entry.getKey(), false);
                    showMessage("Room " + entry.getKey() + " booked successfully!");
                    showMenu();
                    return;
                }
            }
            showMessage("No rooms available.");
        } else if (command.equals("Exit")) {
            System.exit(0);
        }
    }

    void showMessage(String message) {
        Dialog d = new Dialog(this, "Message", true);
        d.setLayout(new FlowLayout());
        Label l = new Label(message);
        Button b = new Button("OK");
```

```
            b.addActionListener(e -> d.setVisible(false));
            d.add(l);
            d.add(b);
            d.setSize(300, 150);
            d.setVisible(true);
        }

        public static void main(String[] args) {
            new RoomBookingApp().setVisible(true);
        }
    }
```

APPENDIX B


Room Booking Application
Login
Register


Room Booking Application
Jsername:
Password:
Confirm Password:
Submit
Back



Available Rooms:
Room 3
Room 10
Room 9
Room 6
Room 5
Room 8
Room 7

REFERENCES

1. Smith, J., and Johnson, A. (2020) Development of Real-Time Booking Systems for Educational Institutions, International Journal of Computer Science and Applications, Vol. 15, No. 3, pp. 201-210.

2. Kumar, R., and Patel, S. (2019) Design and Implementation of a Hostel Management System Using Java, Proceedings of the 2019 International Conference on Software Engineering, New York, NY, pp. 145-150.