COLLEGE HOSTEL BOOKING SYSTEM

A PROJECT REPORT

Submitted by

DILANI R (8115U23EC023)

in partial fulfillment of requirements for the award of the course

EGB1201 - JAVA PROGRAMMING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

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

SAMAYAPURAM – 621 112

DECEMBER - 2024

K. RAMAKRISHNAN COLLEGE OF ENGINEERING (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **“ COLLEGE HOSTEL BOOKING SYSTEM ”** is the bonafide work of **DILANI R (8115U23EC023)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Dr. T. M. NITHYA, M.E.,Ph.D.,	Mr.V.KUMARARAJA, M.E.,(Ph.D.,),
HEAD OF THE DEPARTMENT	**SUPERVISOR**
ASSOCIATE PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Engineering (Autonomous)	K.Ramakrishnan College of Engineering (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 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 **EGB1201 - JAVA PROGRAMMING.**

.

Signature

R. Dilani

DILANI R

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 Engineering (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. D. SRINIVASAN, B.E, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, 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.V.KUMARARAJA, 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 achieve a prominent position among the top technical institutions.

MISSION OF THE INSTITUTION

- M1: To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

VISION OF DEPARTMENT

To create eminent professionals of Computer Science and Engineering by imparting quality education.

MISSION OF DEPARTMENT

M1: To provide technical exposure in the field of Computer Science and Engineering through state of the art infrastructure and ethical standards.

M2: To engage the students in research and development activities in the field of Computer Science and Engineering.

M3: To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

PROGRAM EDUCATIONAL OBJECTIVES

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.

- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

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 College Hostel Booking System is a software solution designed to streamline the process of hostel room allocation for students in educational institutions. This system provides an intuitive and user-friendly graphical interface, enabling students to register, log in, and book hostel rooms efficiently. The primary goal of the system is to simplify the administrative task of room allocation while accommodating specific needs, such as special room requirements. The application leverages Java Swing for its graphical user interface, CardLayout for seamless navigation between modules, and validation mechanisms to ensure secure and reliable operation. By automating the room booking process, this system reduces administrative workload, minimizes human errors, and enhances the student experience.

This project serves as a foundation for a scalable hostel management system that can be further enhanced with database integration, secure password encryption, and additional features like payment processing and advanced reporting.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
1. Streamlines the booking process, making it more efficient. 2. Ensures accurate tracking of student information and room availability. 3. Provides immediate booking and cancellation confirmations to students. 4. Reduces administrative overhead for hostel management. 5. Improves overall user experience with minimal manual intervention.	**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**

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the **College Hostel Booking System** is to streamline and automate the process of hostel room allocation for students, making it efficient, user-friendly, and accessible. The system aims to reduce administrative workload by automating room selection based on user eligibility, such as prioritizing special rooms for differently-abled students. It ensures secure authentication, dynamic room management, and accurate data validation to enhance reliability and user experience. Additionally, the system is designed for scalability, allowing future integration with databases and advanced features like payment processing, ensuring it can adapt to evolving requirements.

1.2 Overview

The **College Hostel Booking System** is a desktop application designed to automate and simplify the process of hostel room allocation for students. It provides a user-friendly interface that allows students to register, log in, and book rooms efficiently. The system dynamically manages room availability, ensuring that students with special requirements are prioritized for designated rooms. It features secure authentication, real-time validation, and a clear navigation structure using Java Swing's CardLayout.

By automating the room booking process, the system reduces administrative burden, minimizes errors, and enhances the overall user experience. Designed for scalability, it can be extended with advanced features like database integration and payment gateways, making it a robust solution for hostel management.

1.3 Java Programming Concepts

The **College Hostel Booking System** leverages several core Java programming concepts, including:

➢ **Object-Oriented Programming (OOP) Principles**

- **Encapsulation**: Data fields like nameField, usernameField, and methods like showMessage() are encapsulated within the class, ensuring modularity and reusability.
- **Inheritance**: The system extends the Frame class, inheriting properties and methods for building GUI applications.
- **Polymorphism**: Overriding methods like windowClosing() from WindowAdapter demonstrates runtime polymorphism.

➢ **Java Swing and AWT for GUI**

- **Swing Components**: Use of TextField, Button, Label, Choice, and Checkbox for creating user interfaces.
- **AWT Layout Managers**: Employs GridLayout and BorderLayout to arrange components in the application.
- **Dialogs**: Modal Dialog boxes are used for user feedback and error messages.

- **Event Handling**

- Implements **ActionListeners** for button clicks (e.g., registerButton and loginButton) to define specific behaviors when user actions occur.
- **ComponentAdapter**: Handles the dynamic population of room choices when the "Room Allocation" panel is shown.

- **CardLayout for Navigation**

- Utilizes CardLayout to switch between different screens (Registration, Login, Room Allocation), creating a multi-page application structure.

- **Conditional Statements**

- **If-Else**: Used to validate passwords during registration, authenticate user credentials during login, and allocate rooms based on user type (special/general).

- **String Handling**

- Compares and validates string inputs (e.g., checking if passwords match or if login credentials are correct).

- **Component Lifecycle Management**

- The application manages the lifecycle of GUI components, such as updating the roomChoice dynamically when the Room Allocation panel becomes visible.

- **Exception Handling (Implicit)**

- While not explicitly shown, the application relies on Java's robustness to handle exceptions like invalid component states or null pointers during GUI rendering.

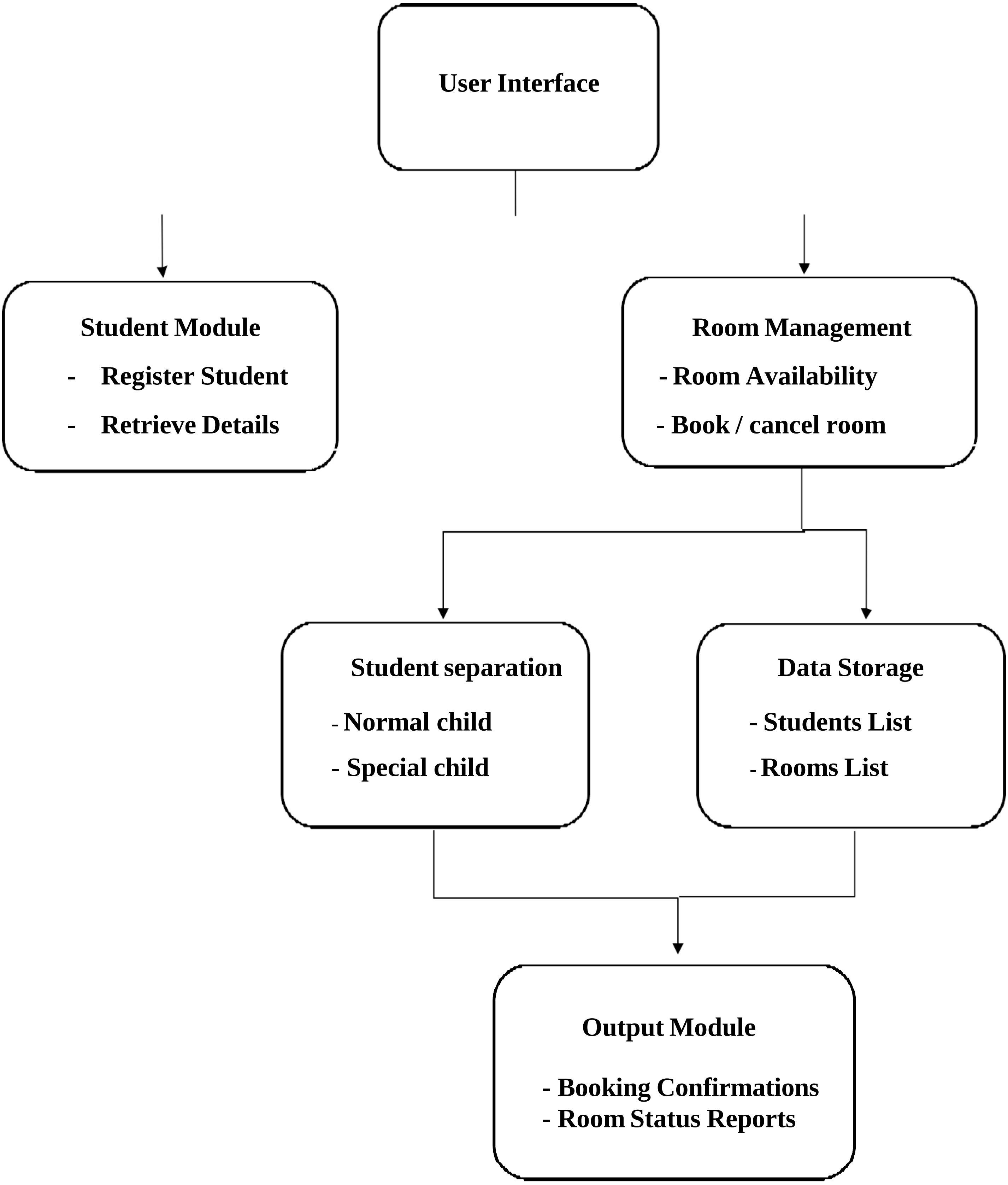
CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the **College Hostel Booking System** involves developing a structured and efficient software solution to automate hostel room allocation for students. The methodology begins with a comprehensive requirement analysis to identify key functionalities such as user registration, secure login, and dynamic room allocation. The system is designed using Java’s Swing and AWT libraries, with a focus on intuitive navigation through a CardLayout and responsive GUI components.

Implementation includes modules for user authentication, real-time input validation, and dynamic room assignment based on user eligibility (special/general). Rigorous testing ensures functionality and seamless integration between modules, while user feedback is incorporated through clear dialogs and error messages. The system will initially be deployed as a standalone desktop application, with provisions for future enhancements such as database integration, multi-user support, and payment gateway inclusion. This methodology ensures a robust, scalable, and user-friendly solution that automates manual processes and enhances the overall efficiency of hostel management.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 User Interface module

PURPOSE:

- This module provides the graphical interface that allows users (students and administrators) to interact with the system. It organizes the navigation between Registration, Login, and Room Allocation pages and ensures smooth transitions using CardLayout.

Key Features:

- CardLayout for Navigation:
  - The CardLayout class is used to switch between panels, including Registration, Login, and Room Allocation modules.
  - Simplifies the interface by presenting one functional screen at a time.
- Graphical Components:
  - Labels: Used to display prompts and messages to the user.
  - Text Fields: Captures user inputs such as Name, Roll Number, Username, and Password.
  - Checkboxes: Allows students to indicate whether they are a Special Child.
  - Buttons: Triggers actions like "Submit," "Login," "Allocate Room," or "Log Off".

- Choice Component: Provides dropdowns for room selection and AC/Non-AC preferences.

3.2 Student Module

PURPOSE:

- ➢ This module is responsible for student registration and login functionalities. It gathers personal information and validates inputs for a secure user experience.

Key Features:

- ➢ Registration:
  - Data Input: Captures Name, Roll Number, Username, Password, and re-entered Password for validation.
  - Password Validation: Compares Password and Re-enter Password fields to ensure consistency. If mismatched, an error message is displayed using a dialog box.
  - Special Child Checkbox: Tracks whether a student requires a special room.
- ➢ Retrieve Details:
  - Stores the entered details temporarily in memory for subsequent login and room allocation steps.

3.3 Room Management Module

PURPOSE :

- Handles room selection, availability, and allocation based on student status (Special or General)

Key Features:

- **Dynamic Room List**:
  - Displays room choices based on the student’s status (Special or General).
  - Uses arrays (specialRooms and generalRooms) to populate available options dynamically.
- **Room Booking and Cancellation**:
  - Allows students to select an available room and choose between AC or Non-AC options.
  - Ensures rooms are allocated uniquely and adjusts availability accordingly.
- **Dynamic Updates**:
  - When the "Room Allocation" panel is displayed, it dynamically updates the list of available rooms based on the student’s type (Special or General)

3.4 Data Storage Module

PURPOSE:

- Manages temporary storage of student and room data during the runtime of the application.

Key Features:

- Student List:
  - Temporarily holds details such as Name, Roll Number, Username, and Password for all registered students.
  - Facilitates data retrieval for login and room allocation processes.

- Room List:
  - Tracks the availability and assignment of rooms in the hostel.
  - Updates dynamically when a room is booked or canceled.

3.5 Output Module

PURPOSE:

- Handles the display of results and feedback for both students and administrators.

Key Features:

- **Booking Confirmations**:
  - Displays a success message with the assigned room details after allocation.
  - If the user is a Special Child, it specifies that a special room was assigned.
- **Room Status Reports**:
  - Provides details on room availability and assignments for better monitoring by the administrator.
- **Error Messages**:
  - Alerts students if passwords don’t match during registration or if login credentials are invalid.
  - Notifies users if no rooms are available or a required field is left blank.

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

4.1 Conclusion

The College Hostel Booking System provides an efficient, user-friendly, and automated solution for managing hostel room bookings. By integrating key features such as student registration, room management, booking, and cancellation, the system simplifies the traditional manual process and improves operational efficiency. The program employs Object-Oriented Programming (OOP) concepts such as classes, encapsulation, and methods to create modular and maintainable code. The use of ArrayLists for dynamic data storage ensures scalability, while input validation and error handling enhance the system's robustness and user experience. Additionally, the console-based interface allows for easy interaction with students and administrators, ensuring a seamless process for booking and managing rooms.This system streamlines hostel room allocation, reduces administrative workload, and ensures transparency in the booking process. As a result, it provides a reliable solution for educational institutions to manage hostel facilities efficiently. In the future, this system can be expanded with features such as database integration, a graphical user interface (GUI), and enhanced security to support a larger number of users and further improve its usability.Overall, the College Hostel Booking System is a functional, scalable, and effective tool that meets the needs of students and hostel administrators, paving the way for smoother management of hostel accommodations.

4.2 Future Scope

The future scope of the **College Hostel Booking System** involves significant advancements to enhance its functionality, scalability, and user experience.

Integrating a relational database will enable persistent data storage and historical record management, while transitioning to a web-based or mobile application will allow remote access and convenience for users. Advanced security measures like OTP verification, two-factor authentication, or biometric access can improve data protection. Features like automated notifications, feedback modules, and reporting tools will enhance communication and provide valuable insights into booking patterns and room utilization. Room allocation algorithms and IoT integration can optimize space management and enable real-time occupancy tracking. Scaling the system for multiple hostels and providing multilingual support will broaden its applicability. Additionally, integrating payment gateways for deposits or fines and AI-powered recommendations for room allocation will streamline operations. Cloud- based implementation ensures scalability, data backup, and disaster recovery, transforming the system into a comprehensive, modernized hostel management platform.

APPENDIX A

(Coding)

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

class CollegeHostelBookingSystem extends Frame {
    CardLayout card;
    Panel mainPanel;

    // Global fields for user data
    TextField nameField, rollField, usernameField, passwordField, rePasswordField;
    Checkbox specialChildCheckbox;
    Choice roomChoice, acChoice;

    // Special and general rooms
    String[] specialRooms = {"201", "202", "203"};
    String[] generalRooms = {"101", "102", "103", "104"};

    // Login fields
    TextField loginUsername, loginPassword;

    // Constructor
    public CollegeHostelBookingSystem() {
        setTitle("College Hostel Booking System");
        setSize(600, 400);
        setLayout(new BorderLayout());

        card = new CardLayout();
```

```
mainPanel = new Panel(card);

// Registration Page
Panel registrationPanel = new Panel(new GridLayout(8, 2, 10, 10));
registrationPanel.add(new Label("Name:"));
nameField = new TextField();
registrationPanel.add(nameField);

registrationPanel.add(new Label("Roll Number:"));
rollField = new TextField();
registrationPanel.add(rollField);

registrationPanel.add(new Label("Username:"));
usernameField = new TextField();
registrationPanel.add(usernameField);

registrationPanel.add(new Label("Password:"));
passwordField = new TextField();
passwordField.setEchoChar('*');
registrationPanel.add(passwordField);

registrationPanel.add(new Label("Re-enter Password:"));
rePasswordField = new TextField();
rePasswordField.setEchoChar('*');
registrationPanel.add(rePasswordField);

registrationPanel.add(new Label("Special Child (Yes/No):"));
specialChildCheckbox = new Checkbox();
registrationPanel.add(specialChildCheckbox);
```

```
Button registerButton = new Button("Submit");
registrationPanel.add(registerButton);

registerButton.addActionListener(e -> {
    if (passwordField.getText().equals(rePasswordField.getText())) {
        card.show(mainPanel, "Login");
    } else {
        showMessage("Passwords do not match!");
    }
});

mainPanel.add(registrationPanel, "Registration");

// Login Page
Panel loginPanel = new Panel(new GridLayout(4, 2, 10, 10));
loginPanel.add(new Label("Username:"));
loginUsername = new TextField();
loginPanel.add(loginUsername);

loginPanel.add(new Label("Password:"));
loginPassword = new TextField();
loginPassword.setEchoChar('*');
loginPanel.add(loginPassword);

Button loginButton = new Button("Login");
loginPanel.add(loginButton);

loginButton.addActionListener(e -> {
    if (loginUsername.getText().equals(usernameField.getText())
            && loginPassword.getText().equals(passwordField.getText())) {
```

```
            card.show(mainPanel, "RoomAllocation");
          } else {
            showMessage("Invalid login credentials!");
          }
        });

        mainPanel.add(loginPanel, "Login");

        // Room Allocation Page
        Panel roomAllocationPanel = new Panel(new GridLayout(6, 2, 10, 10));
        roomAllocationPanel.add(new Label("Select Room:"));

        roomChoice = new Choice();
        roomAllocationPanel.add(roomChoice);

        roomAllocationPanel.add(new Label("AC or Non-AC:"));
        acChoice = new Choice();
        acChoice.add("AC");
        acChoice.add("Non-AC");
        roomAllocationPanel.add(acChoice);

        Button allocateButton = new Button("Allocate Room");
        roomAllocationPanel.add(allocateButton);

        allocateButton.addActionListener(e -> {
          if (specialChildCheckbox.getState()) {
            showMessage("Special Room Allocated: Room " +
roomChoice.getSelectedItem());
          } else {
            showMessage("Room Allocated: Room " + roomChoice.getSelectedItem());
```

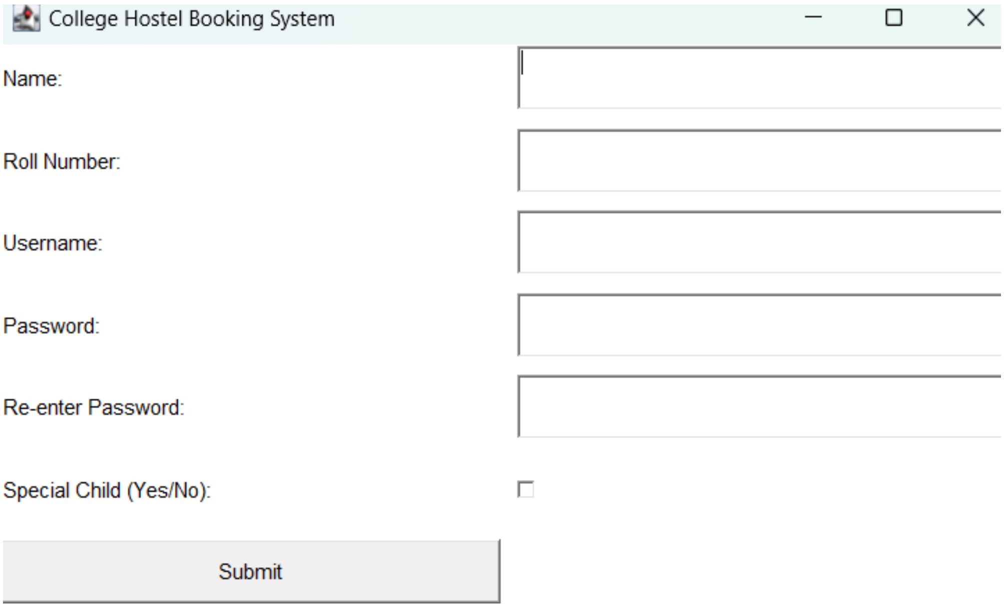
```
    }
});

Button logoffButton = new Button("Log Off");
roomAllocationPanel.add(logoffButton);

logoffButton.addActionListener(e -> card.show(mainPanel, "Login"));

mainPanel.add(roomAllocationPanel, "RoomAllocation");

// Add main panel to frame
add(mainPanel, BorderLayout.CENTER);

// Adjust available rooms based on user type
roomAllocationPanel.addComponentListener(new ComponentAdapter() {
    @Override
    public void componentShown(ComponentEvent e) {
        roomChoice.removeAll();
        if (specialChildCheckbox.getState()) {
            for (String room : specialRooms) {
                roomChoice.add(room);
            }
        } else {
            for (String room : generalRooms) {
                roomChoice.add(room);
            }
        }
    }
});
```

```
        card.show(mainPanel, "Registration");

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });

        setVisible(true);
    }

    private void showMessage(String message) {
        Dialog dialog = new Dialog(this, "Message", true);
        dialog.setLayout(new FlowLayout());
        dialog.add(new Label(message));
        Button okButton = new Button("OK");
        dialog.add(okButton);
        okButton.addActionListener(e -> dialog.dispose());
        dialog.setSize(300, 150);
        dialog.setVisible(true);
    }

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

APPENDIX B

(SCREENSHOTS)

Student registration:



College Hostel Booking System

Name:

Roll Number:

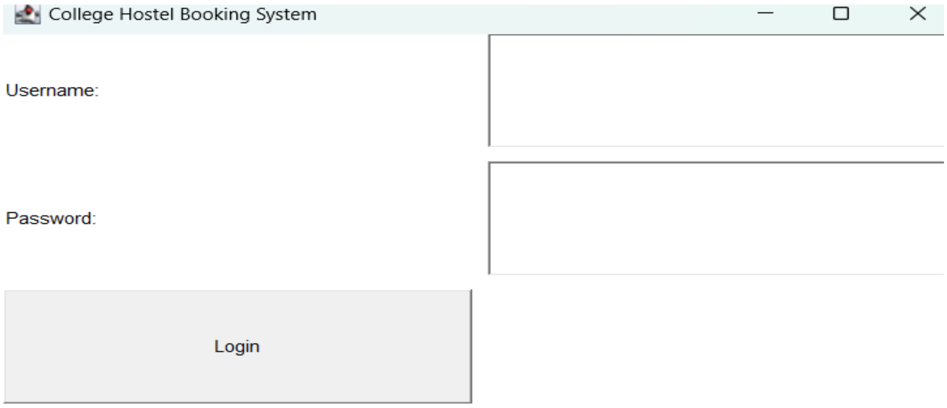
Username:

Password:

Re-enter Password:

Special Child (Yes/No): □

Login:



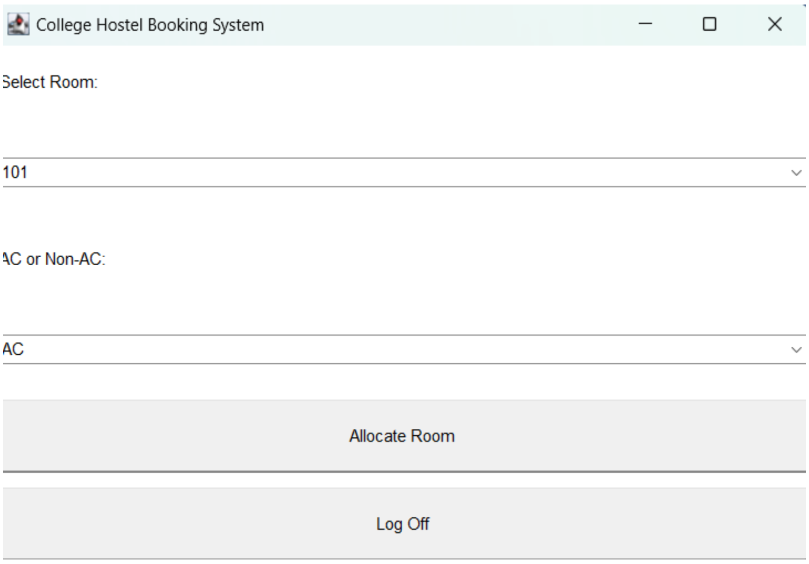
College Hostel Booking System

Username:

Password:

Book a room:

(normal child)



College Hostel Booking System

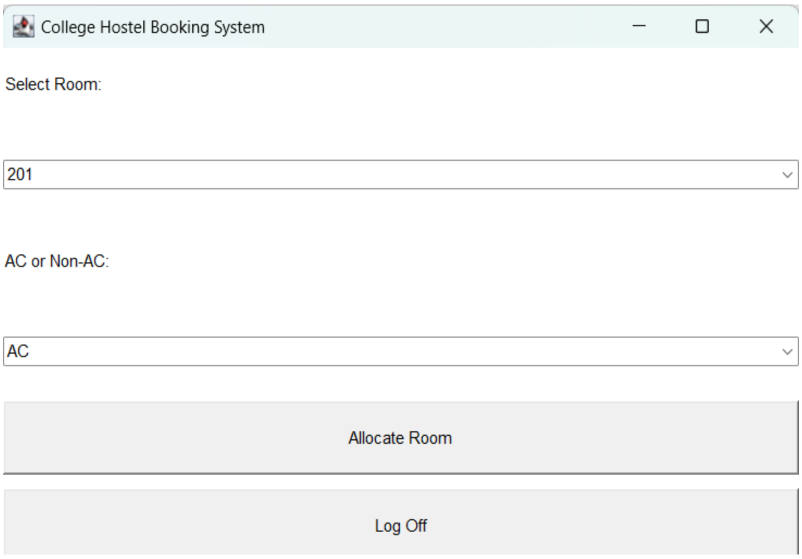
Select Room:

101

AC or Non-AC:

AC

(special child)



College Hostel Booking System

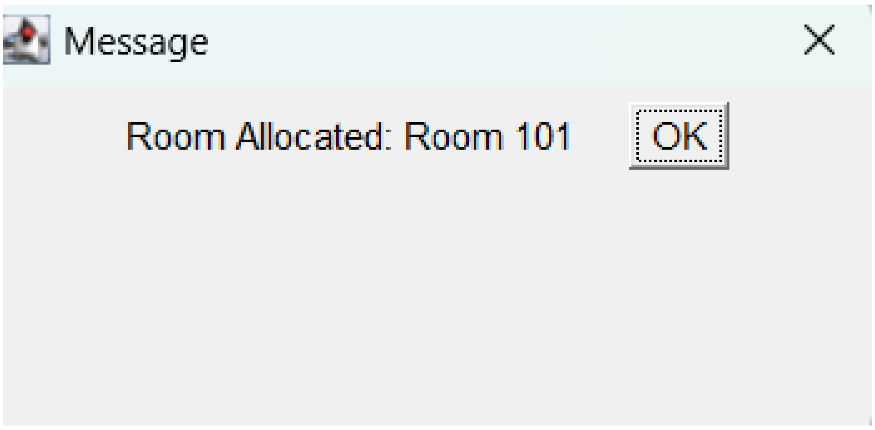
Select Room:

201

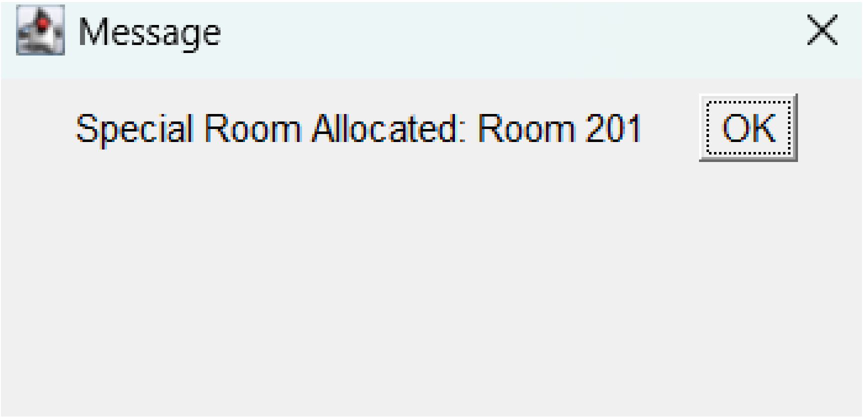
AC or Non-AC:

AC

Room allocation :



Room Allocated: Room 101



Special Room Allocated: Room 201

REFERENCES:

1. Schildt, Herbert. (2018). *Java: The Complete Reference*. 10th Edition. ISBN: 978-1259589313. Publisher: McGraw-Hill Education. Retrieved from https://www.mhprofessional.com/

2. Bloch, Joshua. (2008). *Effective Java*. 2nd Edition. ISBN: 978-0321356680. Publisher: Addison-Wesley. Retrieved from https://www.pearson.com/

3. GeeksforGeeks. (n.d.). *Java Programming Language Tutorials*. Retrieved from https://www.geeksforgeeks.org/java/

4. Oracle Corporation. (n.d.). *The Java Tutorials: Core Java*. Oracle Docs. Retrieved from https://docs.oracle.com/javase/tutorial/

5. Spring.io. (n.d.). *Spring Framework Documentation*. Retrieved from https://spring.io/

6. Wiley. (2019). *Data Structures and Algorithms in Java* by Ellis, David. ISBN: 978-1-118-71832-4. Publisher: Wiley. Retrieved from https://www.wiley.com/en-us/Data+Structures+and+Algorithms+in+Java-p-9781118718324