VISITOR MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

SHARARASAN S (2303811710421143)

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 **"VISITOR MANAGEMENT SYSTEM "** is the bonafide work of **SHARARASAN S(2303811710421143)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

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.

SIGNATURE

Mr. A. Malarmannan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 6/12/2024

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **"VISITOR MANAGEMENT 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

SHARARASAN S

Place: Samayapuram

Date: 6/12/2024

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 Visitor Management System is a desktop-based application developed using Java's Abstract Window Toolkit (AWT), designed to simplify the process of managing visitor check-ins and check-outs. This system allows organizations to efficiently record visitor details, track their visit times, and maintain an accessible visitor history. The system features a user-friendly graphical interface where visitors can be registered with their name, contact information, and purpose of visit, and their check-in time is logged automatically. Additionally, visitors can check out, recording their departure time. For security and confidentiality, the "Visitor History" feature is protected with an admin password, ensuring only authorized personnel can access the complete historical records. Current visitors are displayed in real-time, helping organizations monitor active engagements effectively. The system employs in-memory data structures to manage records and uses Java's LocalDateTime class for accurate time logging. Designed for simplicity and efficiency, this application is ideal for corporate offices, educational institutions, and other organizations aiming to enhance visitor tracking. Future enhancements could include database integration for persistent storage, dynamic user authentication, and advanced reporting capabilities, making it a versatile and secure visitor management solution.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The Visitor Management System is a user-friendly Java-based application that streamlines visitor check-ins and check-outs, ensuring efficient record-keeping and enhanced security. Built using the Abstract Window Toolkit (AWT), it features intuitive modules for visitor registration, check-out management, and current visitor display. A password-protected history module ensures confidentiality while maintaining access control. The program leverages Java's event-driven programming and modular design principles to deliver a seamless and secure experience. With its simple yet powerful interface, the system is ideal for managing visitor logs in small to medium-sized organizations, combining functionality with ease of use	**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**

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

The primary objective of this project is to design and implement a Visitor Management System using Java's Abstract Window Toolkit (AWT). The system aims to streamline the process of managing visitors, ensuring both operational efficiency and security. It provides a user-friendly interface for visitor check-in and check-out, real-time tracking of current visitors, and access to historical data through a password protected module. This project showcases modular software design principles while addressing data accuracy, security, and accessibility in visitor management.

1.2 OVERVIEW

The Visitor Management System is a lightweight desktop application designed to simplify the tracking and management of visitors. Built using Java AWT, it provides an interactive graphical user interface (GUI) that handles key functionalities: visitor registration, check-out management, current visitor monitoring, and viewing visitor history. The system is structured to ensure secure data handling, including a password-protected feature for accessing visitor logs.Upon registration, a visitor's details and check-in time are recorded. The check-out module ensures accurate logging of exit times, while the current visitor display offers a real time overview of active visitors.

- **Ease of Use:** Intuitive buttons, input fields, and live updates displayed in a central text area.
- **Security:** Password-protected access to sensitive visitor history.
- **Modularity:** Distinct modules for registration, check-out, and historical data.

1.3 JAVA PROGRAMMING CONCEPTS

1. Object-Oriented Programming (OOP)

- **Classes and Objects:**The Visitor class encapsulates visitor data and methods such as registration, check-in, and check-out, demonstrating the principles of encapsulation and abstraction.
- **Inheritance and Polymorphism:** Java AWT components like Frame, Button, and TextField inherit from base classes, showcasing Java's rich library inheritance hierarchy.

2. Event-Driven Programming

- Java AWT's event-handling mechanism is used extensively. For example, buttons like "Register Visitor" and "Check-out Visitor" trigger specific actions through ActionListener interfaces, enabling interactive functionalities.

3. Graphical User Interface (GUI)

- Java AWT components such as TextField, Button, TextArea, and Dialog are used to create an interactive and visually appealing interface. This demonstrates Java's capability to build desktop applications with intuitive user interaction.

4. Date and Time API

- The program uses Java's LocalDateTime and DateTimeFormatter classes for managing and formatting check-in and check-out times. This ensures precise and standardized time tracking.

5. Data Validation and Error Handling

- The system validates user inputs to ensure all necessary fields are filled. It also handles scenarios like duplicate check-outs or incorrect passwords gracefully, demonstrating robust input validation and error management.

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed Visitor Management System is designed to improve visitor tracking processes by leveraging Java's AWT framework for an intuitive and secure application.

Visitor Registration Module:

- Visitors will be registered using their name, contact information, and purpose of visit.
- Each registration will automatically log the current date and time as the check-in time.
- The data will be stored and displayed for real-time monitoring.

Visitor Check-Out Module:

- A dedicated feature to record visitor exit times.
- The system ensures that visitors who have already checked out cannot check out again, eliminating redundancy and enhancing data accuracy.

Current Visitor Monitoring:

- A live view of all visitors currently checked in will be available, allowing administrators to know the occupancy status at any given moment.
- This feature is critical for ensuring safety compliance and managing on premise resources effectively.

Visitor History Access:

- The system includes a password-protected module for administrators to view the complete history of visitor data.
- This ensures sensitive information is safeguarded and accessible only to authorized personnel.

Enhanced Security Measures:

- Admin-level access to visitor logs will require password authentication, preventing unauthorized data access.
- The password system is integrated seamlessly into the GUI, enhancing usability without compromising security.

2.2 BLOCK DIAGRAM

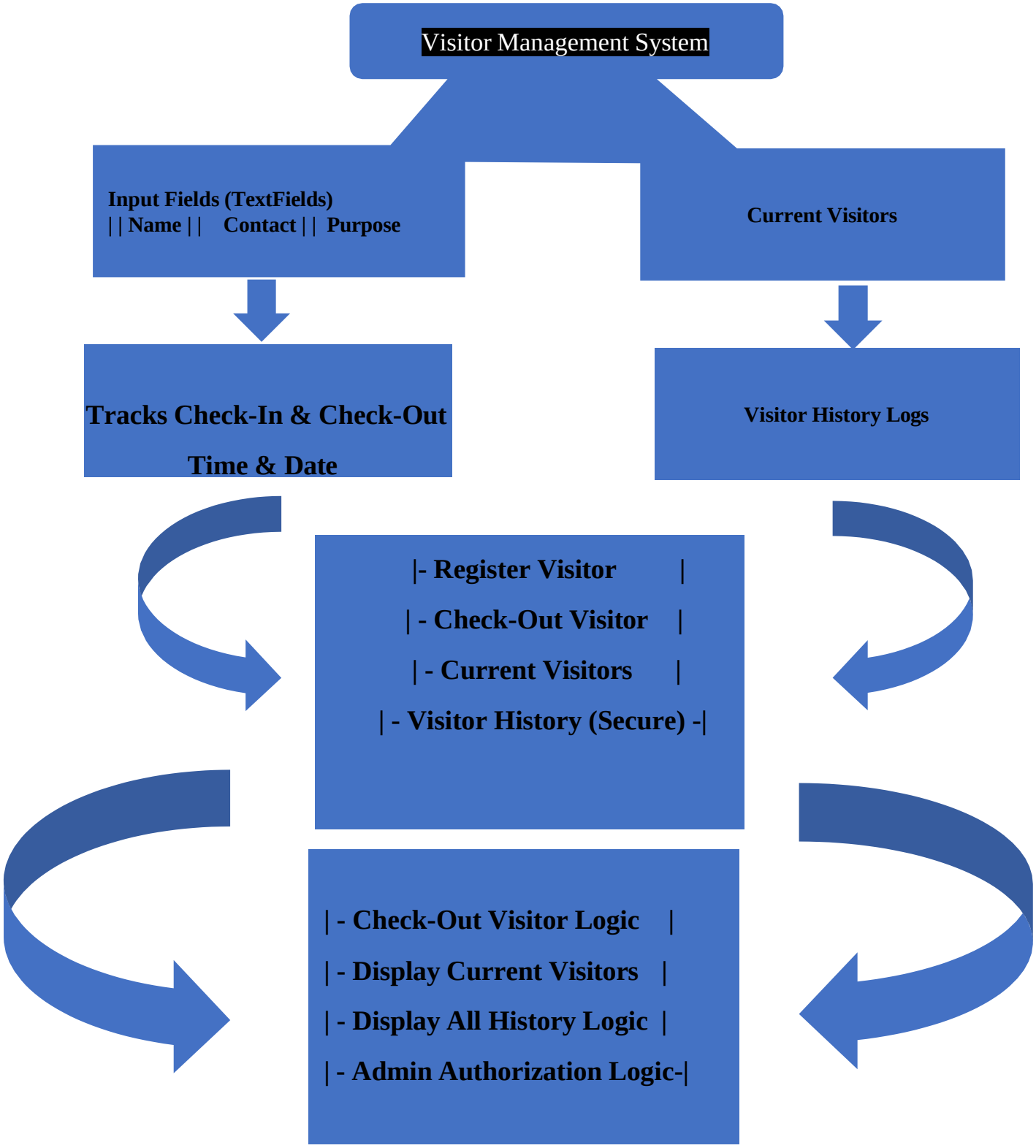


Figure 2.2.1 : Visitor Management System.

CHAPTER 3

MODULE DESCRIPTION

3.1 VISITOR REGISTRATION MODULE.

The Visitor Registration Module in the code is a key feature that facilitates the systematic and efficient registration of visitors. This module captures essential visitor details and records their entry into the premises. It ensures that all relevant information is documented accurately, enabling effective visitor tracking and management.

Input Fields for Visitor Details:

- **Name Field:** Captures the name of the visitor.
- **Contact Field:** Stores the visitor's contact information.
- **Purpose Field:** Records the reason for the visit.

Automatic Check-In Time Logging:

- Upon registration, the system automatically logs the current date and time as the visitor's check-in time.
- This timestamp is generated using the LocalDateTime.now() method from Java's time API, ensuring precision and reliability.

Registration Validation:

- The module validates that all fields (Name, Contact, Purpose) are filled before registering a visitor.
- If any field is left empty, a message is displayed in the text area, prompting the user to provide complete details.

Data Storage:

- Each registered visitor's information is stored as an instance of the Visitor class.
- The visitor details are added to a List collection, which maintains a record of all visitors for further operations, such as check-out or history retrieval.

3.2 CHECK-OUT MANAGEMENT MODULE.

The Check-Out Management Module is a critical feature of the visitor management system that ensures efficient and precise tracking of when visitors leave the premises. This module complements the registration process by providing a complete record of visitor activities, enhancing the accountability and security of the environment.

Visitor identification for check-out:

- The visitor's name is entered into the system to initiate the check-out process.
- The system cross-verifies the provided name with the list of registered visitors.

Automatic check-out time logging:

- When a match is found, the system automatically records the check-out time using the LocalDateTime.now() method.
- The check-out timestamp is added to the visitor's record, ensuring a complete log of entry and exit times**.**

Validation and error handling:

- If the visitor name does not match any registered visitor or the visitor has already checked out, an appropriate message is displayed.
- This prevents duplication or erroneous data entry.

3.3 CURRENT VISITORS DISPLAY MODULE.

The Current Visitors Display Module serves as a real-time monitoring system that lists all visitors currently on the premises. It provides an easily accessible overview of active visitors who have checked in but not yet checked out, enhancing the transparency and security of visitor tracking.

1. Real-time visitor tracking:

- The system dynamically filters the list of visitors to display only those who have not checked out.
- This ensures that the displayed data reflects the current state of the premises.

2. **Clear and concise output:**

    - The module generates a formatted list of current visitors, including essential details such as their name and check-in time.

3.4 USER INTERFACE (UI) MODULE.

The User Interface (UI) Module in the Visitor Management System is designed to provide a seamless and intuitive experience for users. This module acts as the interaction layer between the system and the user, ensuring all features are easily accessible and visually organized.

1. **Simplicity:** Ensure that users can interact with the system without technical expertise.
2. **Clarity:** Present information and actions in a well-structured, visually appealing manner.
3. **Efficiency:** Reduce the time required for actions like visitor registration, check-out, and data retrieval.

3.5 PASSWORD-PROTECTED VISITOR HISTORY MODULE.

This module restricts access to the complete visitor history using an admin password, safeguarding private information from unauthorized viewing**.**

1. **Secure Access:**
    - Only users with the correct admin password can access the full visitor history.
    - Prevents misuse of data and ensures compliance with privacy standards.
2. **Complete Visitor Records:**
    - Displays the history of all visitors, including check-in and check-out details.
    - Provides a comprehensive log of visitor activities for administrative or audit purposes.
3. **User Authentication:**
    - Prompts the user for a password when the history view button is clicked.
    - Denies access if the password is incorrect, displaying an appropriate

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Visitor Management System developed using Java and AWT is a comprehensive solution designed to streamline the process of managing visitor records in an organized and efficient manner. With its modular structure, the system addresses core functionalities such as visitor registration, check-out management, and secure access to visitor history, all integrated within a user-friendly interface.The system’s emphasis on simplicity and security ensures that even non-technical users can operate it with ease while safeguarding sensitive information through features like password-protected history access. The real-time display of current visitors and detailed logging of visitor activities enhance transparency and accountability.

By leveraging Java's robust programming capabilities and the AWT framework for a cross-platform graphical user interface, this program provides a reliable tool for environments such as offices, schools, and organizations where visitor tracking is essential. The modular design also allows for easy scalability and future enhancements, making it a versatile and valuable solution.
In conclusion, this Visitor Management System not only simplifies visitor tracking but also ensures data integrity and security, making it a practical and efficient tool for modern-day visitor management needs.

4.2 FUTURE SCOPE

The Visitor Management System has significant potential for future enhancements. Integrating biometric features such as fingerprint or facial recognition can streamline the check-in and check-out process while enhancing security. Cloud integration would enable centralized data storage, providing accessibility across multiple locations and ensuring data backup.

- **Biometric Integration:** Enable check-in and check-out using fingerprint or facial recognition for enhanced security.
- **Cloud Integration:** Store visitor data on the cloud for centralized management and accessibility across locations.
- **Mobile App Support:** Develop a companion mobile app for self-check-in, QR code-based entry, and real-time notifications.

APPENDIX A

```
import java.awt.*;
import java.awt.event.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;

class Visitor {
    private String name;
    private String contact;
    private String purpose;
    private LocalDateTime checkInTime;
    private LocalDateTime checkOutTime;

    public Visitor(String name, String contact, String purpose) {
        this.name = name;
        this.contact = contact;
        this.purpose = purpose;
        this.checkInTime = LocalDateTime.now();
        this.checkOutTime = null;
    }

    public void checkOut() {
        this.checkOutTime = LocalDateTime.now();
    }
```

```
    public String getName() {
        return name;
    }

    public LocalDateTime getCheckInTime() {
        return checkInTime;
    }

    public boolean isCheckedOut() {
        return checkOutTime != null;
    }

    public String getFormattedCheckInTime() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
        return checkInTime.format(formatter);
    }

    public String getFormattedCheckOutTime() {
        if (checkOutTime != null) {
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
            return checkOutTime.format(formatter);
        }
        return "Not available";
    }

    @Override
    public String toString() {
```

```
        return "Name: " + name + ", Check-in: " + getFormattedCheckInTime() +
                ", Check-out: " + getFormattedCheckOutTime();
    }
}

public class VisitorManagementSystemAWT extends Frame {
    private List<Visitor> visitors = new ArrayList<>();
    private final String adminPassword = "sharkesh"; // Admin password

    private TextField nameField, contactField, purposeField;
    private Button registerButton, checkOutButton, currentVisitorsButton,
allHistoryButton, exitButton;
    private TextArea displayArea;

    public VisitorManagementSystemAWT() {
        setLayout(new FlowLayout());
        setTitle("Visitor Management System");
        setSize(600, 400);

        // Labels and Input Fields
        add(new Label("Name:"));
        nameField = new TextField(20);
        add(nameField);

        add(new Label("Contact:"));
        contactField = new TextField(20);
        add(contactField);

        add(new Label("Purpose:"));
```

```
purposeField = new TextField(20);
add(purposeField);

// Buttons
registerButton = new Button("Register Visitor");
add(registerButton);

checkOutButton = new Button("Check-out Visitor");
add(checkOutButton);

currentVisitorsButton = new Button("Current Visitors");
add(currentVisitorsButton);

allHistoryButton = new Button("Visitor History");
add(allHistoryButton);

exitButton = new Button("Exit");
add(exitButton);

// Display Area
displayArea = new TextArea(15, 50);
displayArea.setEditable(false);
add(displayArea);

// Button Actions
registerButton.addActionListener(e -> registerVisitor());
checkOutButton.addActionListener(e -> checkOutVisitor());
currentVisitorsButton.addActionListener(e -> displayCurrentVisitors());
allHistoryButton.addActionListener(e -> promptForPassword());
```

```
        exitButton.addActionListener(e -> System.exit(0));

        // Close Window Action
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });

        setVisible(true);
    }

    private void registerVisitor() {
        String name = nameField.getText().trim();
        String contact = contactField.getText().trim();
        String purpose = purposeField.getText().trim();

        if (name.isEmpty() || contact.isEmpty() || purpose.isEmpty()) {
            displayArea.setText("All fields are required for registration.\n");
            return;
        }

        Visitor newVisitor = new Visitor(name, contact, purpose);
        visitors.add(newVisitor);
        displayArea.setText("Visitor " + name + " successfully checked in at " +
newVisitor.getFormattedCheckInTime() + ".\n");

        clearFields();
    }
```

```
    private void checkOutVisitor() {
        String name = nameField.getText().trim();

        if (name.isEmpty()) {
            displayArea.setText("Please enter the name of the visitor to check out.\n");
            return;
        }

        boolean found = false;
        for (Visitor visitor : visitors) {
            if (visitor.getName().equalsIgnoreCase(name) && !visitor.isCheckedOut()) {
                visitor.checkOut();
                displayArea.setText("Visitor " + name + " checked out at " +
visitor.getFormattedCheckOutTime() + ".\n");
                found = true;
                break;
            }
        }

        if (!found) {
            displayArea.setText("Visitor not found or already checked out.\n");
        }

        clearFields();
    }

    private void displayCurrentVisitors() {
        StringBuilder output = new StringBuilder("Current Visitors:\n");
```

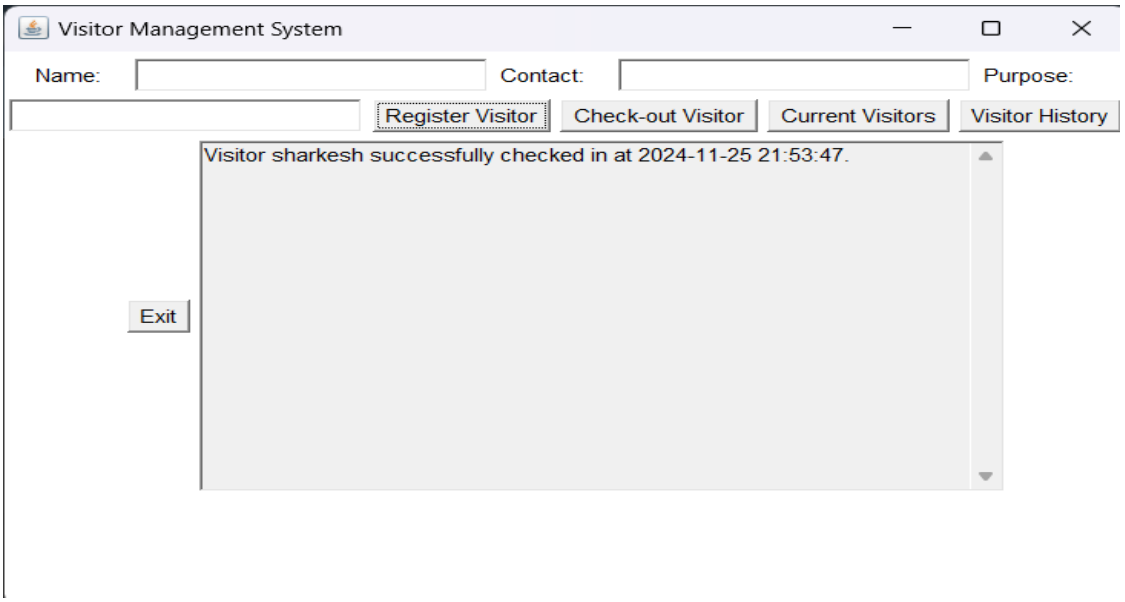
```
    for (Visitor visitor : visitors) {
        if (!visitor.isCheckedOut()) {
            output.append(visitor.toString()).append("\n");
        }
    }
    if (output.length() == 16) { // No visitors found
        output.append("No visitors are currently checked in.\n");
    }
    displayArea.setText(output.toString());
}

private void promptForPassword() {
    String inputPassword = getPasswordFromUser();
    if (inputPassword.equals(adminPassword)) {
        displayAllVisitorHistory();
    } else {
        displayArea.setText("Incorrect password. Access denied.\n");
    }
}

private String getPasswordFromUser() {
    Frame passwordFrame = new Frame("Enter Password");
    Dialog passwordDialog = new Dialog(passwordFrame, "Password", true);
    passwordDialog.setLayout(new FlowLayout());
    passwordDialog.setSize(300, 150);

    Label passwordLabel = new Label("Enter Admin Password:");
    TextField passwordField = new TextField(20);
    passwordField.setEchoChar('*');
```

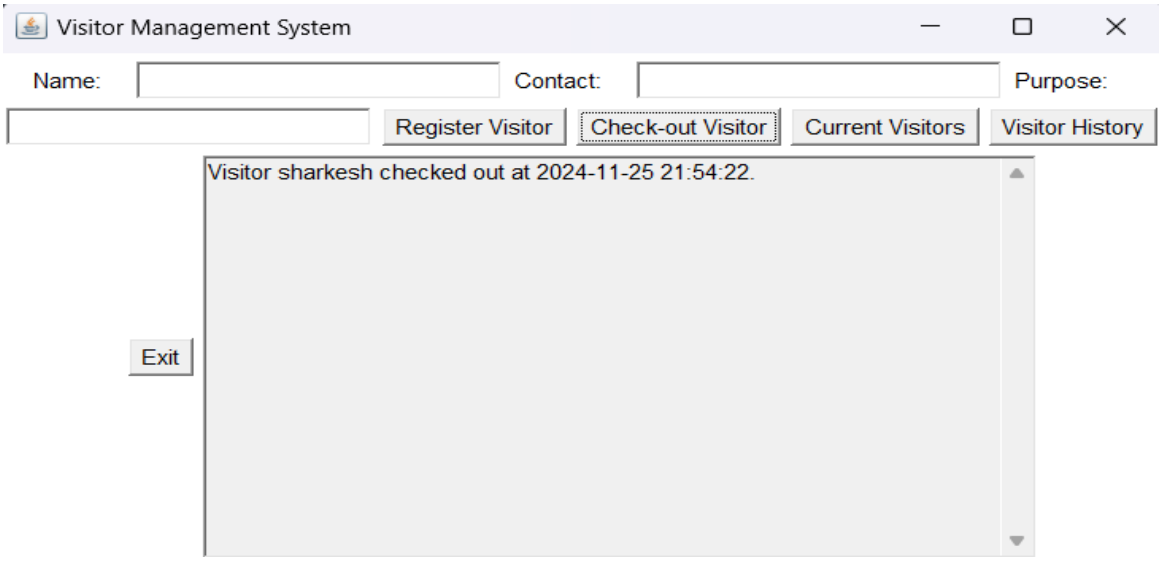
```
    Button submitButton = new Button("Submit");

    passwordDialog.add(passwordLabel);
    passwordDialog.add(passwordField);
    passwordDialog.add(submitButton);

    final String[] passwordInput = {null};

    submitButton.addActionListener(e -> {
        passwordInput[0] = passwordField.getText();
        passwordDialog.setVisible(false);
    });

    passwordDialog.setVisible(true);

    return passwordInput[0] != null ? passwordInput[0] : "";
}

private void displayAllVisitorHistory() {
    StringBuilder output = new StringBuilder("All Visitor History:\n");
    for (Visitor visitor : visitors) {
        output.append(visitor.toString()).append("\n");
    }
    if (output.length() == 20) { // No visitors found
        output.append("No visitor history found.\n");
    }
    displayArea.setText(output.toString());
}
```

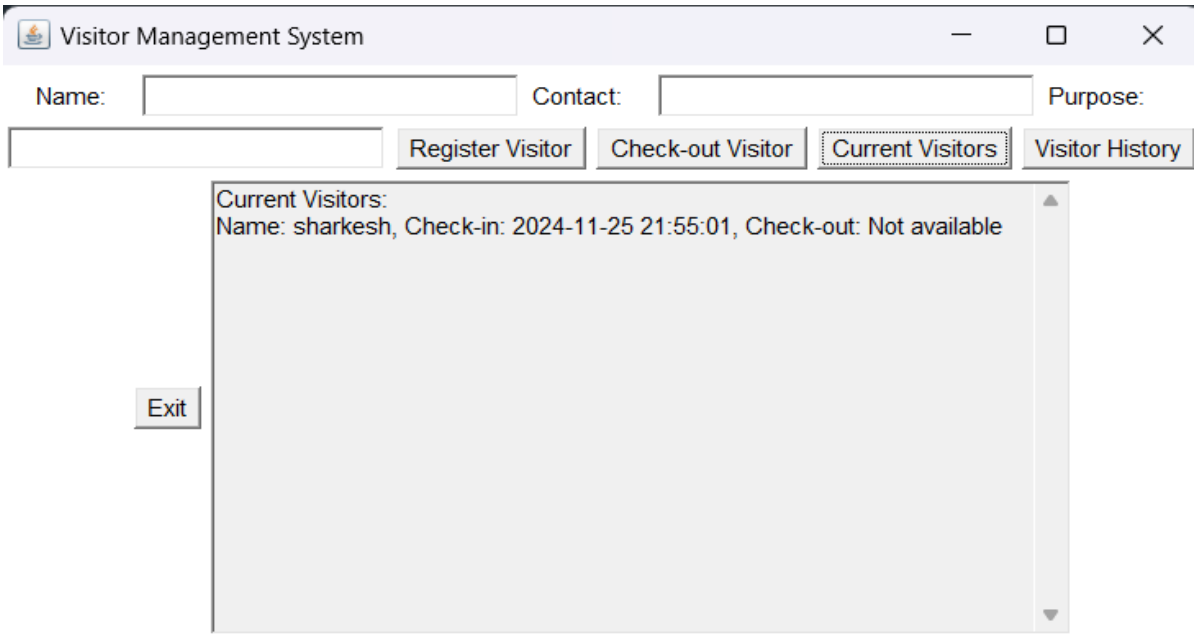
```
    private void clearFields() {
        nameField.setText("");
        contactField.setText("");
        purposeField.setText("");
    }

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

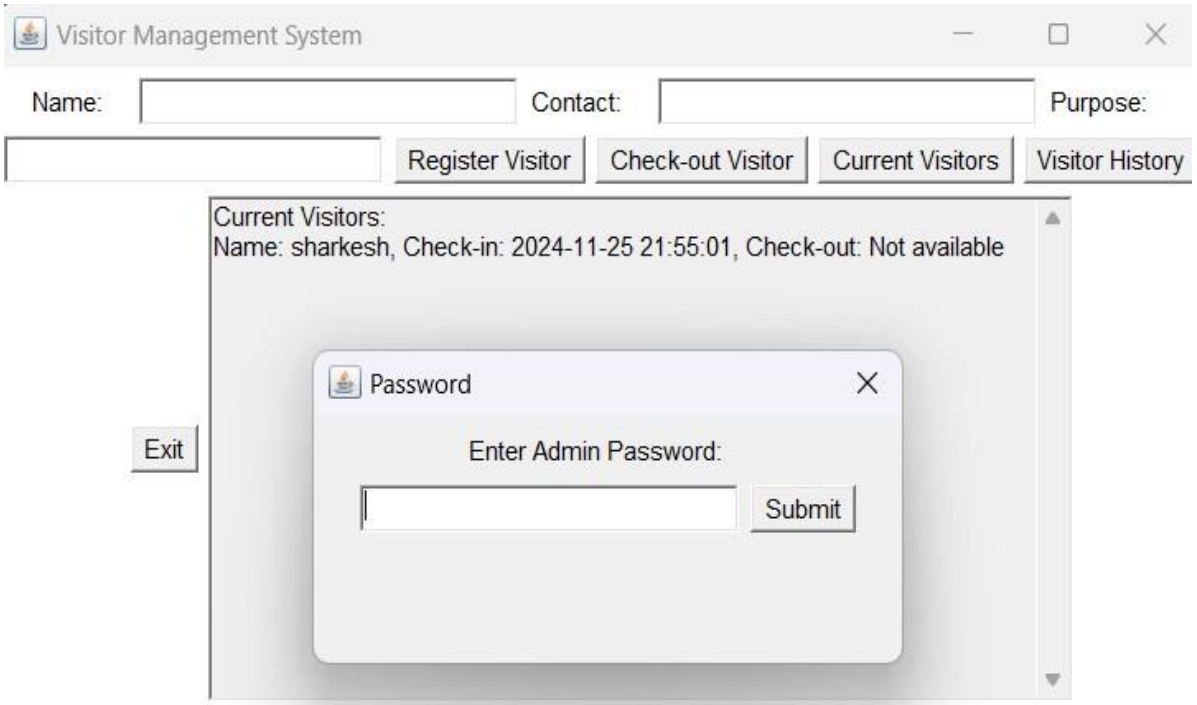
APPENDIX B

Output






Visitor Management System
Name:
Contact:
Purpose:
Register Visitor
Check-out Visitor
Current Visitors
Visitor History
Current Visitors:
Name: sharkesh, Check-in: 2024-11-25 21:55:01, Check-out: Not available
Exit


Visitor Management System
Name:
Contact:
Purpose:
Register Visitor
Check-out Visitor
Current Visitors
Visitor History
Current Visitors:
Name: sharkesh, Check-in: 2024-11-25 21:55:01, Check-out: Not available
Exit
Password
Enter Admin Password:
Submit

REFERENCES

1. "Java: A Beginner's Guide" by Herbert Schildt ,8th Edition, 2018. Covers basic Java programming and includes chapters on GUI development with AWT and Swing.
2. "Java 2D Graphics" by Jonathan Knudsen ,1st Edition,2004. This book provides advanced graphics and GUI concepts, which could be useful if you want to enhance your system’s interface.
3. "Head First Java" by Kathy Sierra and Bert Bates ,2nd Edition, 2005.A beginner-friendly book that includes concepts of object-oriented programming and GUI development in Java.