ATTENDANCE MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

HEMCHUDESH A(2303811720521017)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

INFORMATION TECHNOLOGY

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 **“ATTENDANCE MANAGEMENT SYSTEM”** is the bonafide work of **HEMCHUDESH A (230311720521017)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr. C. SHYAMALA, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Ms. S. UMA MAGESHWARI, M.E.,
SUPERVISOR
ASSISTANTP ROFESSOR

[illegible]

SIGNATURE	**SIGNATURE**
Dr. C. Shyamala, M.E.,Ph.D.,	Ms. S. Uma Mageshwari, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of IT	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 04.12.2024

CGB1201-JAVA PROGRAMMING
Ms. S. UMA MAGESHWARI, M.E.,
INTERNAL EXAMINER
ASSISTANTP ROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Mr.N.M.K.RAMALINGAMSAKTHIVELAN
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
PAAVAI ENGINEERING COLLEGE.NAMAKKAL.

[illegible]

INTERNAL EXAMINER	EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“ATTENDANCE 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 TECHNOLOGY**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

.

Signature

[illegible]

HEMCHUDESH A

Place: Samayapuram

Date:04.12.2024

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. C. SHYAMALA, M.E.,Ph.D.,** Head of the department, **INFORMATION TECHNOLOGY** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Ms. S. UMA MAGESHWARI, 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 Attendance Monitoring and Notification System is an automated software solution aimed attracking student attendance and providing notifications when attendance falls below a specified threshold. The system is designed to help both students and advisors keep track of attendance in real-time, reducing the burden of manual attendance recording and enabling timely intervention when necessary. Built using core Java concepts such as classes, objects, loops, and HashMaps, this system is flexible and scalable, ensuring easy management of student records and attendance data. By integrating features like attendance tracking and notification automation, the system makes attendance management more efficient and helps improve student participation.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The Attendance Monitoring and Notification System is an automated software solution aimed attracking student attendance and providing notifications when attendance falls below a specified threshold. The system is designed to help both students and advisors keep track of attendance in real-time, reducing the burden of manual attendance recording and enabling timely intervention when necessary. Built using core Java concepts such as classes, objects, loops, and HashMaps, this system is flexible and scalable, ensuring easy management of student records and attendance data. By integrating features like attendance tracking and notification automation, the system makes attendance management more efficient and helps improve student participation.	**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

1.1Objective:

The primary objective of the Attendance Management System using Java is to provide an efficient, user-friendly platform for managing student attendance records. This system aims to streamline the process of recording, tracking, and analyzing attendance data. By automating attendance marking, report generation, and notifications, it reduces the administrative burden on educators, ensuring accurate and timely attendance records. Additionally, the system facilitates transparent communication with students and their guardians about attendance performance, promoting accountability and encouraging better attendance habits. Ultimately, this system enhances the overall educational experience by fostering a disciplined and well-informed academic environment.

1.2 Overview:

An Attendance Management System using Java efficiently tracks and manages student or employee attendance through a combination of backend logic and user interface components. It utilizes Java Swing or JavaFX to create a graphical user interface (GUI) that includes forms and dialogs for user interactions, such as adding new students, marking attendance, and viewing reports. The core logic is implemented in Java, handling various operations like marking attendance, calculating attendance percentages, generating attendance reports, and sending notifications for low attendance. Attendance data is stored using Java's collection framework, typically with classes and data structures like HashMap and ArrayList. If needed, the system can integrate with a database (e.g., MySQL) for persistent storage of attendance records, using JDBC (Java Database Connectivity) to connect and interact with the database, performing CRUD (Create, Read, Update, Delete) operations. The system allows for marking attendance either for individual students or in bulk for all students on a given date, supporting manual input through the GUI and handling different attendance statuses (e.g., present, absent). It generates detailed attendance reports, showing records for specific dates and calculating overall attendance percentages, and can send notifications when attendance falls below a certain threshold.

1.3 Java Programming Concepts:

The Java Attendance Management System employs several fundamental Java programming concepts. It utilizes classes and objects to represent entities like students and their attendance records. Collections, specifically ArrayList, manage lists of students and attendance data. The system handles errors gracefully using exception handling, particularly for date parsing. It leverages streams and lambda expressions for efficient data processing. The graphical user interface (GUI) is crafted using Swing components like JFrame and JButton, with event handling to manage user interactions. Dates are formatted and parsed using SimpleDateFormat, ensuring consistency. StringBuilder is utilized for efficient string manipulation, especially in generating reports. Control structures like loops and conditional statements direct the program flow, while encapsulation protects data within classes. The code employs polymorphism through method overriding in the ActionListener interface, facilitating dynamic method execution based on user interaction. Additionally, constructors are used to initialize objects with default values, and data encapsulation is achieved by declaring class fields as private and providing public methods for data access. These concepts collectively create a functional and robust system for managing attendance**.**

CHAPTER 2
PROJECT METHODOLOGY

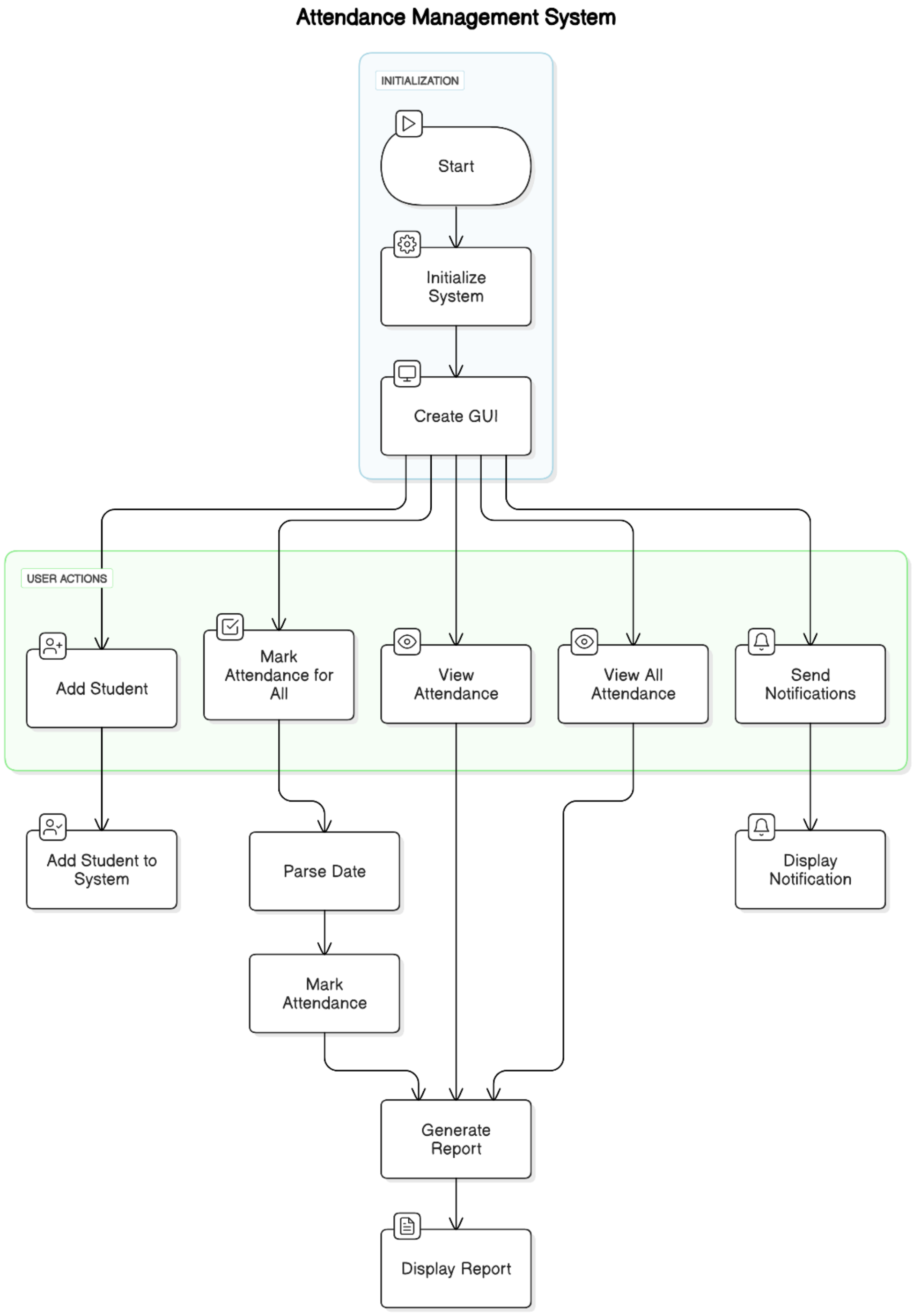
2.1Proposed Work

Developing an Attendance Management System in Java involves several key steps. First, requirements gathering involves defining the system's scope, identifying key features such as user authentication, attendance marking, report generation, and notifications, and determining user roles like students, teachers, and admins. System design includes creating ER diagrams and database schema for tables like Users, AttendanceRecords, and Notifications, designing UML class and use case diagrams, and sketching the user interface.

Project setup requires initializing a new Java project in an IDE like IntelliJ IDEA or Eclipse and setting up version control with Git. Database setup involves installing and configuring MySQL, and creating the schema and tables. Backend development includes implementing JDBC for database connectivity, creating CRUD operations, and implementing business logic for attendance marking, percentage calculations, and report generation.

User interface development involves creating forms for login, attendance marking, and viewing reports using JavaFX or Swing, and implementing event handling for interactions. Integration ensures seamless communication between the backend and user interface. Testing includes unit testing, integration testing, and user acceptance testing to gather feedback. Deployment involves packaging the application as a JAR file and deploying it in the intended environment, such as a desktop or server. Maintenance involves monitoring the system for issues and implementing updates based on user feedback.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1Student Module

This class represents individual students within the system. It includes attributes like the student's name and an attendance record, which is a list of AttendanceRecord objects. The class provides methods to mark attendance, calculate the attendance percentage, and generate a report of the student's attendance. The methods ensure that each student's attendance data is accurately recorded and easily accessible for generating reports or calculating statistics.

3.2AttendanceRecord Module

This class holds the details of an individual attendance entry. Each AttendanceRecord object contains a date and a boolean value indicating whether the student was present on that date. This class serves as a data container, ensuring that attendance entries are consistently structured and easily manageable within the Student class.

3.3AttendanceManagementSystem Module

This class manages the entire attendance system. It maintains a list of students and provides methods to add new students, mark attendance for individual students or all students, view attendance records, and send notifications for low attendance. It uses Java's collection framework to handle student data and incorporates error handling for invalid date formats. This class acts as the core of the system, coordinating various tasks and ensuring smooth operation and data integrity.

3.4AttendanceApp Module

This is the main class that sets up the graphical user interface (GUI) using Swing components like JFrame and JButton. It includes action listeners for handling user interactions, such as adding a new student, marking attendance, viewing attendance, and sending notifications. This class serves as the entry point of the application, initializing the GUI and integrating all functionalities of the system into a user-friendly interface. It ensures that users can interact with the system efficiently through a visually appealing and intuitive interface.

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION:

The Attendance Monitoring and Notification System offers an efficient, automated solution for tracking student attendance and sending notifications when attendance falls below a predefined threshold. Using core Java concepts, the system is both effective and flexible, allowing for easy management of student attendance records. By utilizing key programming concepts such as encapsulation, collections, and exception handling, your system is both robust and modular. The use of a command-line interface offers basic functionality, but there is potential for improvement with the introduction of a graphical user interface (GUI) for better user experience.

4.2 FUTURE SCOPE:

The future scope for an Attendance Management System (AMS) in Java is very promising. Integrating AI and machine learning can help analyze attendance patterns and provide actionable insights for better workforce management. Incorporating biometric authentication, like fingerprint scanning and facial recognition, will enhance security and accuracy. Developing mobile applications will allow employees to clock in and out from anywhere, adding flexibility and convenience. Real-time analytics and dashboards will help managers monitor trends and make quick decisions. Moving to cloud-based solutions will improve scalability, data security, and remote access.

REFERENCES:

1. "Java Programming: A Beginner's Guide" by Herbert Schildt.

2. "Data Structures and Algorithms in Java" by Robert Lafore.

3. Oracle Java Documentation: https://docs.oracle.com/javase/

4. Stack Overflow: Java-related discussions on HashMaps, Java classes, and loops.

APPENDIX

(Project Source Code)

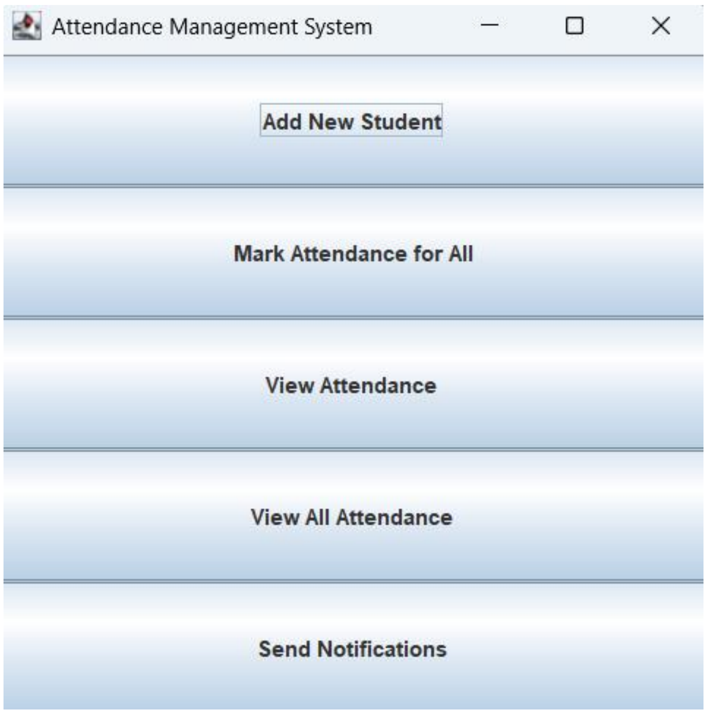
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

class Student {
   String name;
   ArrayList<AttendanceRecord> attendance;

   Student(String name) {
      this.name = name;
      this.attendance = new ArrayList<>();
   }

   void markAttendance(Date date, boolean present) {
      attendance.add(new AttendanceRecord(date, present));
   }

   double getAttendancePercentage() {
      long totalClasses = attendance.size();
      long attendedClasses = attendance.stream().filter(record -> record.present).count();
      return (attendedClasses / (double) totalClasses) * 100;
   }

   String getAttendanceReport() {
      StringBuilder report = new StringBuilder();
      SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
      for (AttendanceRecord record : attendance) {
         report.append("Date: ").append(sdf.format(record.date)).append(", Present: 
").append(record.present).append("\n");
      }
      report.append("Attendance Percentage: ").append(String.format("%.2f", 
getAttendancePercentage())).append("%\n");
      return report.toString();
   }
}

class AttendanceRecord {
   Date date;
   boolean present;

   AttendanceRecord(Date date, boolean present) {
      this.date = date;
```

```
        this.present = present;
    }
}

class AttendanceManagementSystem {
    ArrayList<Student> students;
    int attendanceThreshold = 75;
    SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");

    AttendanceManagementSystem() {
        students = new ArrayList<>();
    }

    void addStudent(String name) {
        students.add(new Student(name));
    }

    void markAttendance(String name, String dateString, boolean present) throws
ParseException {
        Date date = dateFormat.parse(dateString);
        for (Student student : students) {
            if (student.name.equalsIgnoreCase(name)) {
                student.markAttendance(date, present);
                return;
            }
        }
        System.out.println("Student not found.");
    }

    void markAttendanceForAll(String dateString) throws ParseException {
        Date date = dateFormat.parse(dateString);
        StringBuilder reports = new StringBuilder();
        for (Student student : students) {
            int present = JOptionPane.showConfirmDialog(null, "Is " + student.name + "
present?", "Mark Attendance for All", JOptionPane.YES_NO_OPTION);
            student.markAttendance(date, present == JOptionPane.YES_OPTION);
            reports.append("Attendance Report for ").append(student.name).append(":\n")
                    .append(student.getAttendanceReport()).append("\n");
        }
        JOptionPane.showMessageDialog(null, reports.toString());
    }
    void viewAttendance(String name) {
        for (Student student : students) {
            if (student.name.equalsIgnoreCase(name)) {
                double percentage = student.getAttendancePercentage();
                String report = "Student Name: " + student.name + "\nAttendance: " +
student.attendance.size() + " (" + String.format("%.2f", percentage) + "%)\n";
                JOptionPane.showMessageDialog(null, report, "View Attendance",
JOptionPane.INFORMATION_MESSAGE);
                return;
            }
        }
        JOptionPane.showMessageDialog(null, "Student not found.", "Error",
JOptionPane.ERROR_MESSAGE);
    }

    void viewAllAttendance() {
        StringBuilder reports = new StringBuilder();
        for (Student student : students) {
            double percentage = student.getAttendancePercentage();
            reports.append("Student Name: ").append(student.name).append("\nAttendance:
").append(student.attendance.size()).append(" (").append(String.format("%.2f",
percentage)).append("%)\n");
        }
        JOptionPane.showMessageDialog(null, reports.toString(), "View All Attendance",
JOptionPane.INFORMATION_MESSAGE);
    }

    void sendLowAttendanceNotifications() {
        StringBuilder notifications = new StringBuilder();
        for (Student student : students) {
            double percentage = student.getAttendancePercentage();
            if (percentage < attendanceThreshold) {
                notifications.append("Notification: ").append(student.name).append(" has low
attendance (").append(String.format("%.2f", percentage)).append("%)\n");
            }
        }
        JOptionPane.showMessageDialog(null, notifications.toString(), "Low Attendance
Notifications", JOptionPane.WARNING_MESSAGE);
    }
}

public class AttendanceApp {
    static AttendanceManagementSystem ams = new AttendanceManagementSystem();

    public static void main(String[] args) {
        JFrame frame = new JFrame("Attendance Management System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);
        frame.setLayout(new GridLayout(5, 1));

        JButton addStudentButton = new JButton("Add New Student");
        JButton markAttendanceForAllButton = new JButton("Mark Attendance for All");
        JButton viewAttendanceButton = new JButton("View Attendance");
        JButton viewAllAttendanceButton = new JButton("View All Attendance");
        JButton sendNotificationsButton = new JButton("Send Notifications");

        addStudentButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

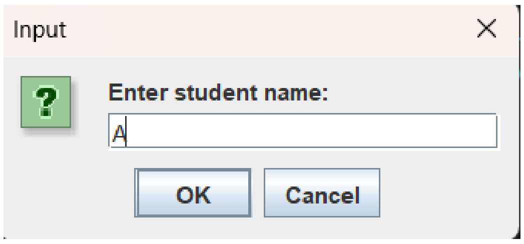
```
            String name = JOptionPane.showInputDialog("Enter student name:");
            ams.addStudent(name);
            JOptionPane.showMessageDialog(frame, "Student added successfully!");
        }
    });

    markAttendanceForAllButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String date = JOptionPane.showInputDialog("Enter date (dd-MM-yyyy):");
            try {
                ams.markAttendanceForAll(date);
                JOptionPane.showMessageDialog(frame, "Attendance marked for all
students!");
            } catch (ParseException ex) {
                JOptionPane.showMessageDialog(frame, "Invalid date format.");
            }
        }
    });

    viewAttendanceButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String name = JOptionPane.showInputDialog("Enter student name:");
            ams.viewAttendance(name);
        }
    });

    viewAllAttendanceButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            ams.viewAllAttendance();
        }
    });

    sendNotificationsButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            ams.sendLowAttendanceNotifications();
        }
    });
    frame.add(addStudentButton);
    frame.add(markAttendanceForAllButton);
    frame.add(viewAttendanceButton);
    frame.add(viewAllAttendanceButton);
    frame.add(sendNotificationsButton);

    frame.setVisible(true);
  }
}
```

APPENDIX B

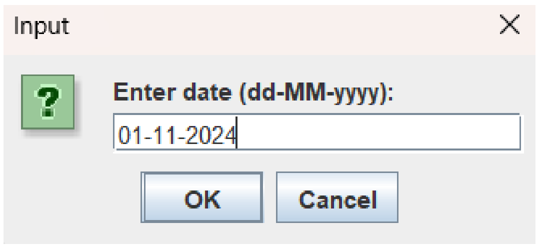
(SCREENSHOTS)

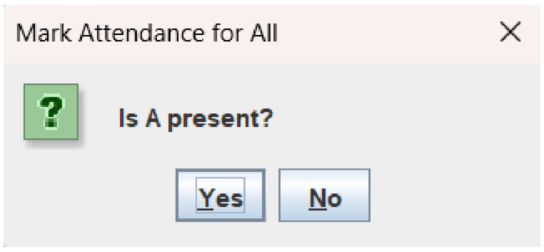


ADDING STUDENTS:



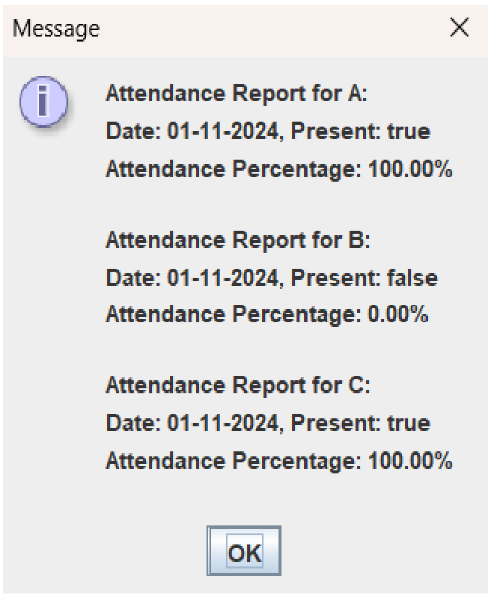
MARKING ATTENDANCE:





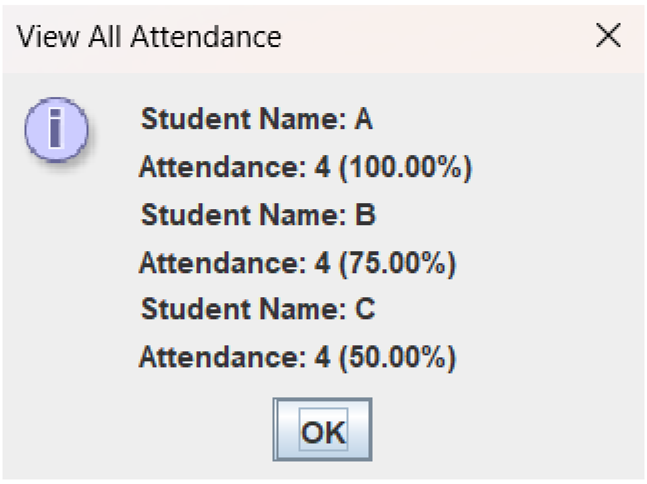
```
Is A present?
```

DISPLAYING ATTENDANCE RECORDS:



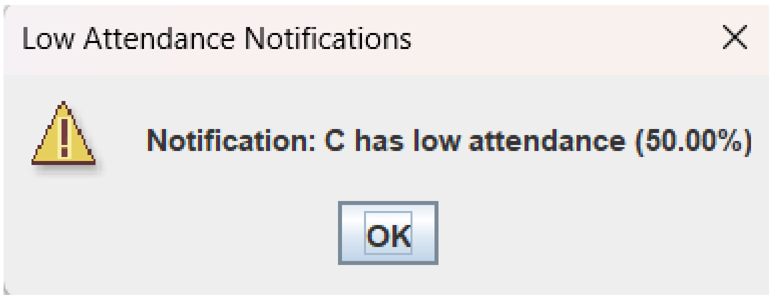
```
Attendance Report for A:
Date: 01-11-2024, Present: true
Attendance Percentage: 100.00%

Attendance Report for B:
Date: 01-11-2024, Present: false
Attendance Percentage: 0.00%

Attendance Report for C:
Date: 01-11-2024, Present: true
Attendance Percentage: 100.00%
```

VIEWING ATTENDANCE:



```
Student Name: A
Attendance: 4 (100.00%)
Student Name: B
Attendance: 4 (75.00%)
Student Name: C
Attendance: 4 (50.00%)
```

NOTIFICATIONS:



Low Attendance Notifications

Notification: C has low attendance (50.00%)