ATTENDANCE MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

NESHANTH M (2303811710421105)

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 **" ATTENDANCE MANAGEMENT SYSTEM"** is the bonafide work of **NESHANTH M (2303811710421105)** 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
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.,)
SUPERVISOR
ASSISTANTP ROFESSOR

[illegible]

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mrs.K.Valli Priyadharshini, M.E.,(ph.D.,),
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 …03.12.2024………….

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

[illegible]

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

[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 ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

Signature

Neshanth.M

NESHANTH M

Place: Samayapuram
Data: 03/12/2024

ACKNOWLEDGEMENT

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

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

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

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

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

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

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

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

VISION OF THE INSTITUTION

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

MISSION OF THE INSTITUTION

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

VISION OF DEPARTMENT

To be a leading force in leveraging cutting-edge technology to drive students with innovation, efficiency and Strategic growth.

MISSION OF DEPARTMENT

M1: Academic Excellence: To provide high-quality education that equips students with the theoretical knowledge and practical skills necessary to excel in their careers and adapt to the evolving technological landscape.

M2: Innovative Research: To foster a research-oriented environment that encourages innovation, interdisciplinary collaboration, and the pursuit of cutting-edge solutions to complex problems in technology and related fields.

M3: Ethical and Inclusive Education: To promote ethical practices, diversity, and inclusion within the IT field, preparing students to be responsible and culturally aware professionals.

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

1.1 Objective:

The objective of this attendance management system is to:

1. **Manage Students**: Allow for the addition of students with unique IDs and names.
2. **Record Attendance**: Provide functionality to record the attendance status of students on specific dates.
3. **Store Attendance Records**: Maintain a list of all attendance records, including the student details, dates, and attendance status.
4. **Calculate Attendance Percentage**: Calculate and display the attendance percentage of each student based on the recorded attendance data.
5. **User Interaction**: Offer a user-friendly command-line interface for adding students, recording attendance, viewing attendance records, and calculating attendance percentages.

Overall, it aims to simplify the process of tracking and analyzing student attendance in an educational setting.

1.2 Overview:

An attendance management system using Java is designed to streamline the process of tracking attendance for students, employees, or participants in any organization. The system typically includes user authentication for secure login, attendance recording either manually or automatically, and the generation of attendance reports for analysis. It connects to a relational database like MySQL, using JDBC to manage data stored in tables for users and attendance records. The user interface, built with JavaFX or Swing, provides forms for logging in, marking attendance, and viewing reports. The backend involves creating Java classes for users and attendance records, implementing CRUD operations, and handling the business logic for attendance management. Comprehensive testing ensures the functionality and integration of all components before deploying the application for use, making the system efficient, accurate, and user-friendly.

1.3 Java Programming Concepts:

1. **Classes and Objects**:
    - **Student** and **AttendanceRecord**: These classes encapsulate data and behavior related to students and attendance records.
    - **AttendanceManagementSystem**: This class manages the overall system operations, such as adding students and marking attendance.
2. **Collections**:
    - **ArrayList**: Used to store lists of students and attendance records, providing dynamic array capabilities.
3. **Date and Time Handling**:
    - **Date** and **SimpleDateFormat**: Used for managing and formatting dates in attendance records.
4. **Event Handling**:
    - **ActionListener**: Interfaces implemented for handling button click events in the GUI.
5. **Graphical User Interface (GUI)**:
    - **Swing**: Used to create the graphical user interface, including **JFrame**, **JButton**, and **JOptionPane** components.
    - **GridLayout**: Utilized for arranging GUI components in a grid layout.
6. **Control Flow Statements**:
    - **For Loops** and **If Statements**: Used to iterate through collections and make decisions based on conditions, such as checking attendance and sending notifications.
7. **Exception Handling**:
    - **try-catch Blocks**: Employed to handle exceptions, such as **ParseException** when parsing date strings.
8. **Streams**:
    - **Stream API**: Utilized in methods like getAttendancePercentage() to filter and count attendance records.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

For your attendance management system using Java, here's a proposed plan of work to help you structure and execute the project efficiently:

1. Requirements Gathering

- Define the scope of the system.
- Identify key features: User authentication, attendance marking, report generation, notifications, etc.
- Determine user roles (e.g., student, teacher, admin).

2. System Design

- **Database Design**: Create ER diagrams and schema for the database.
    - Tables: Users, AttendanceRecords, Notifications.
- **UML Diagrams**: Design class diagrams and use case diagrams.
- **Interface Design**: Sketch or wireframe the user interface.

3. Project Setup

- Initialize a new Java project in an IDE (IntelliJ IDEA, Eclipse).
- Set up version control using Git.

4. Database Setup

- Install and configure the database (MySQL).
- Create the database schema and tables.

5. Backend Development

- **Database Connectivity**: Implement JDBC to connect Java with the database.
- **CRUD Operations**: Create methods for Create, Read, Update, Delete operations.
- **Business Logic**: Implement the core logic for marking attendance, calculating percentages, and generating reports.

6. User Interface Development

- **Forms and Dialogs**: Create forms for login, attendance marking, and viewing reports using JavaFX or Swing.
- **Event Handling**: Implement action listeners for button clicks and other interactions.

7. Integration

- Integrate the backend with the user interface.
- Ensure seamless communication between the UI and the database.

8. Testing

- **Unit Testing**: Test individual components and methods.
- **Integration Testing**: Ensure all components work together smoothly.
- **User Acceptance Testing**: Conduct testing with real users to gather feedback.

9. Deployment

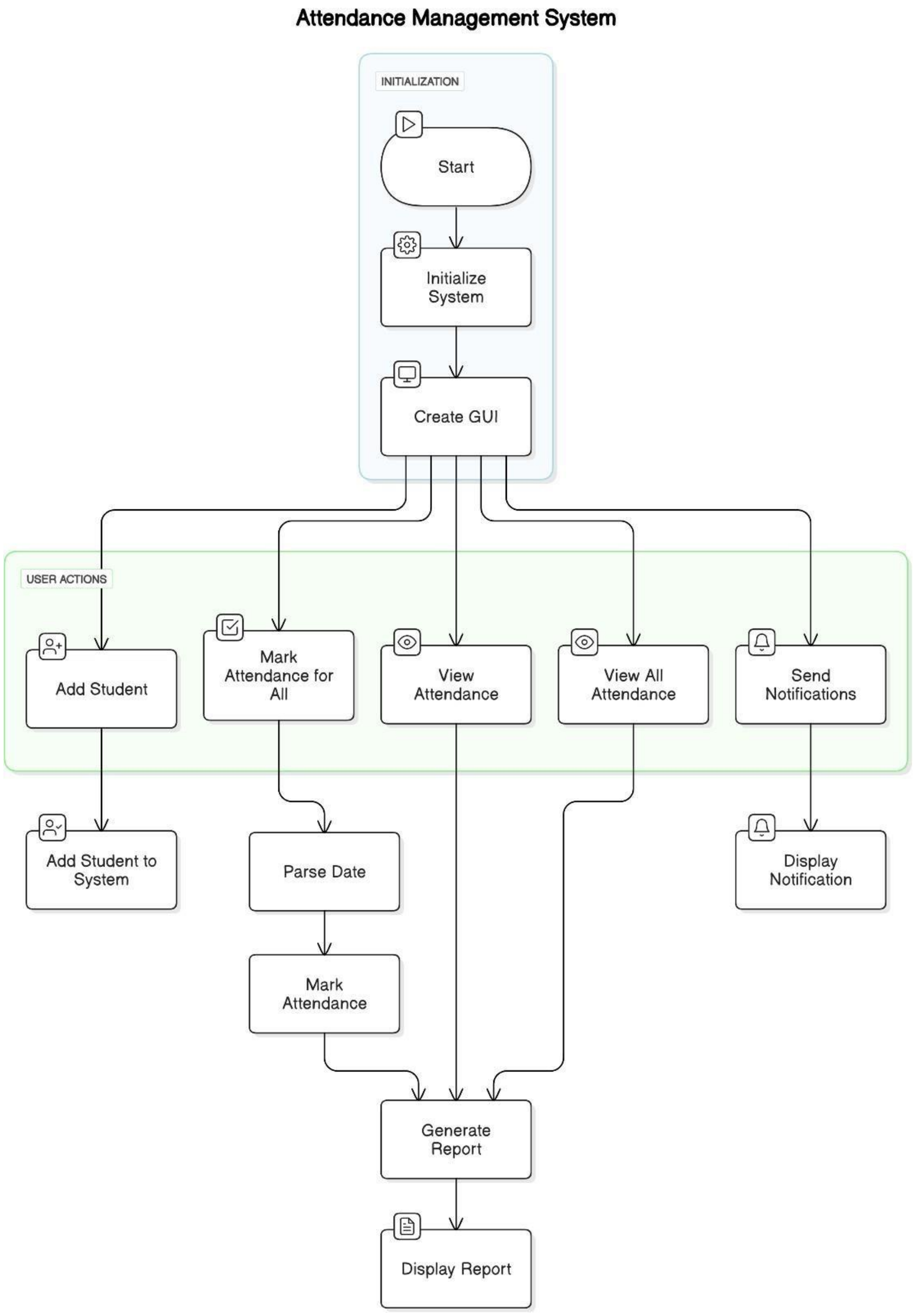
- Package the application for distribution (e.g., JAR file).
- Deploy the application in the intended environment (e.g., desktop, server).

10. Maintenance

- Monitor the system for any issues.
- Implement updates and improvements based on user feedback.

By following this structured approach, you'll be able to develop a robust and user-friendly attendance management system.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Student Module

Purpose: Represents each student and their attendance records.

Components:

- **Attributes**: name (String) and attendance (ArrayList<AttendanceRecord>).
- **Methods**:
    - markAttendance(Date date, boolean present): Adds a new attendance record.
    - getAttendancePercentage(): Calculates the attendance percentage.
    - getAttendanceReport(): Generates a detailed attendance report.

3.2 AttendanceRecord Module

Purpose: Represents an individual attendance record for a student.

Components:

- **Attributes**: date (Date) and present (boolean).
- **Constructor**: Initializes the date and attendance status.

3.3 AttendanceManagementSystem Module

Purpose: Manages students and their attendance records.

Components:

- **Attributes**: students (ArrayList<Student>), attendanceThreshold (int), dateFormat (SimpleDateFormat).
- **Methods**:
    - addStudent(String name): Adds a new student to the system.
    - markAttendance(String name, String dateString, boolean present): Marks attendance for a specific student.
    - markAttendanceForAll(String dateString): Marks attendance for all students on a given date.

    - viewAttendance(String name): Views attendance details for a specific student.
    - viewAllAttendance(): Views attendance details for all students.
    - sendLowAttendanceNotifications(): Sends notifications for students with low attendance.

3.4 User Interface Module

Purpose: Provides a graphical interface for interacting with the attendance management system. **Components**:

- **Main Frame**: A JFrame that serves as the main window.
- **Buttons**:
    - addStudentButton: For adding new students.
    - markAttendanceButton: For marking attendance for a specific student.
    - markAttendanceForAllButton: For marking attendance for all students.
    - viewAttendanceButton: For viewing attendance of a specific student.
    - viewAllAttendanceButton: For viewing attendance of all students.
    - sendNotificationsButton: For sending low attendance notifications.

3.5 Main Module

Purpose: Initializes the attendance management system and sets up the user interface.
Components:

- **Static Instance**: AttendanceManagementSystem ams.
- **Main Method**: Sets up the main frame and buttons, and attaches action listeners to the buttons.

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

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.

FUTURE SCOPE

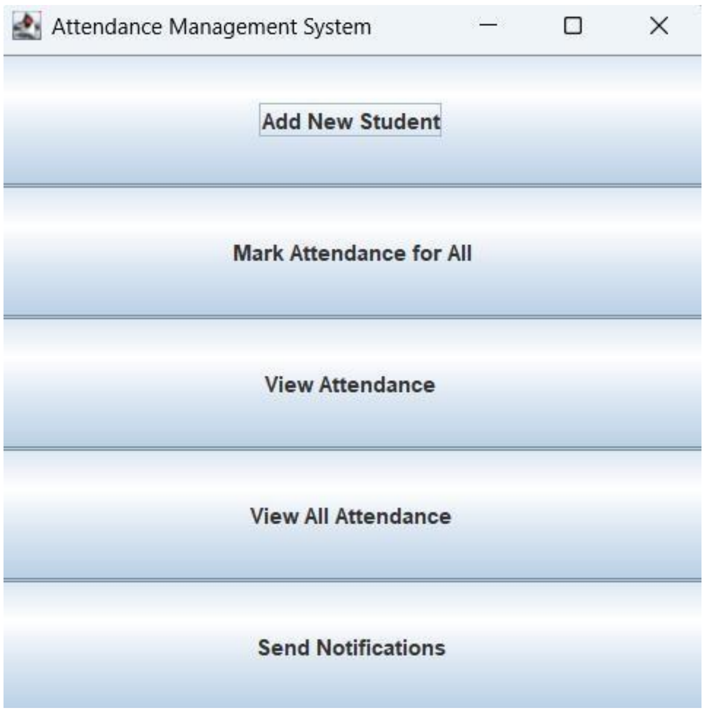
The future scope of an attendance management system is expansive, driven by advancements in technology and evolving organizational needs. Integrating artificial intelligence (AI) and machine learning (ML) can enable predictive analytics to monitor attendance trends, detect anomalies, and enhance employee engagement. Biometric technologies like facial recognition and iris scanning will improve accuracy and security, while cloud-based systems will offer greater scalability and remote accessibility. Mobile app integration can streamline real-time attendance tracking, even for remote or hybrid work environments. Additionally, automation of compliance reporting and seamless integration with payroll and HR systems will enhance operational efficiency, making attendance management systems more intelligent, adaptive, and indispensable in the future workplace.
4o

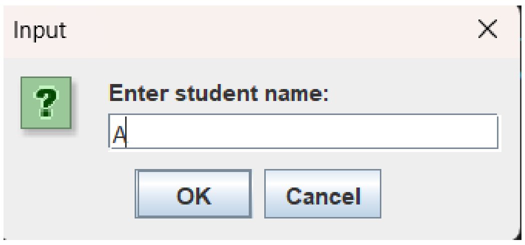
APPENDIX
(Project Source Code)

```
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();
}
```

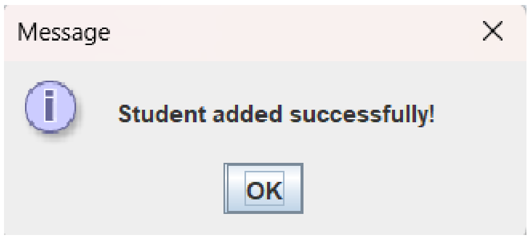
```
}

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 + "\n Attendance: " +
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
(SCREEN SHOT)

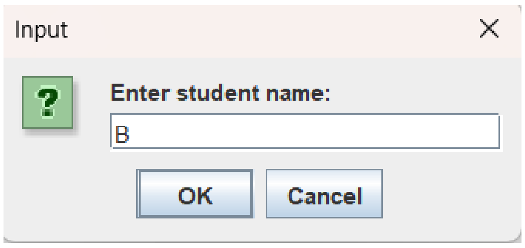




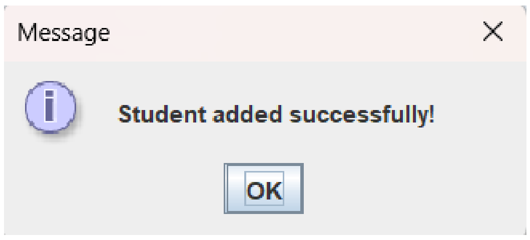
Enter student name: A



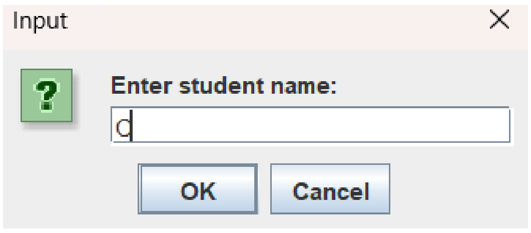
Student added successfully!



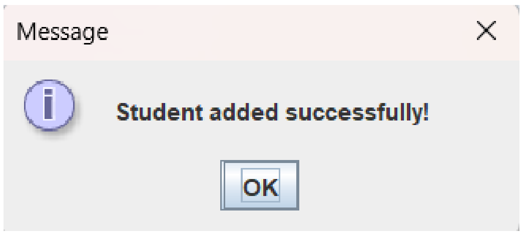
Enter student name: B



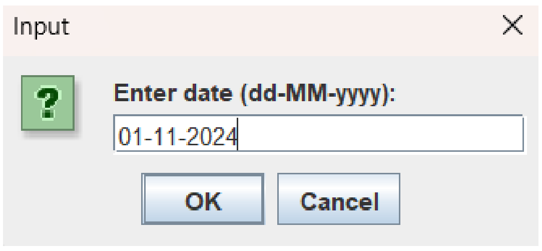
Student added successfully!

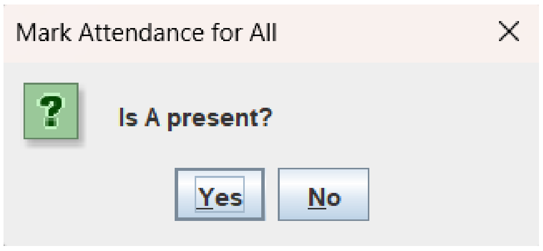


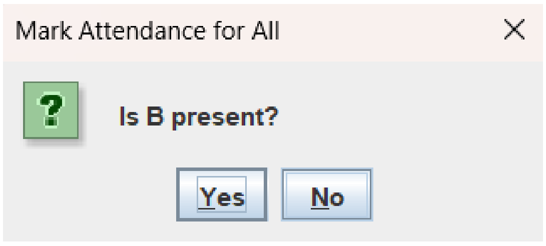
Enter student name: C

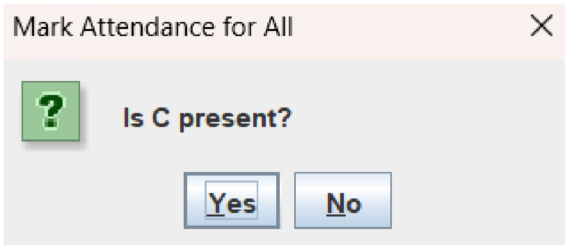


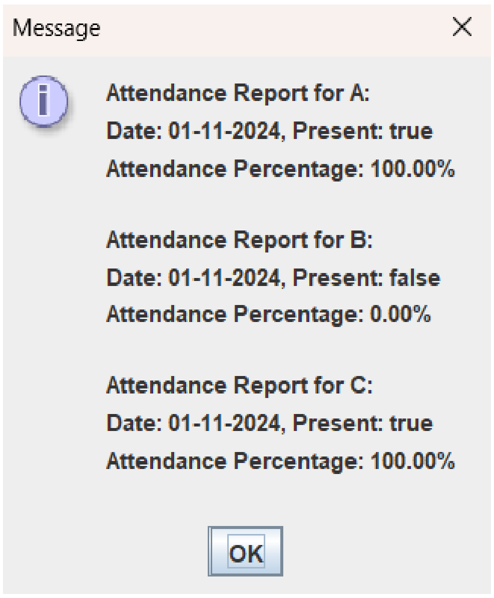
Student added successfully!

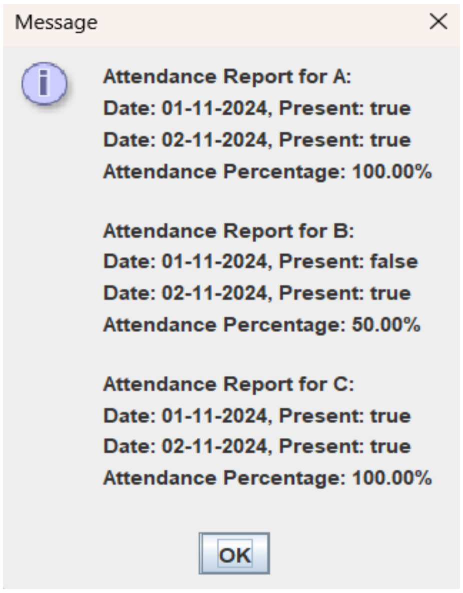


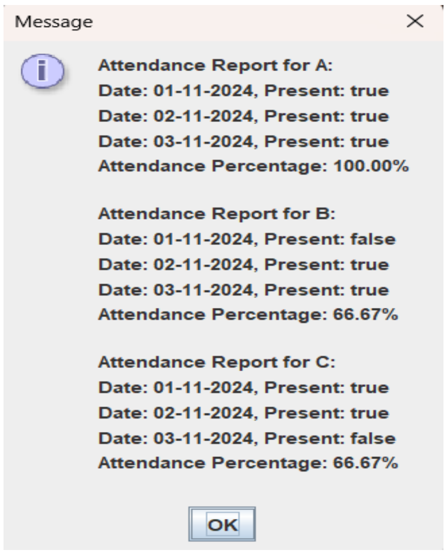


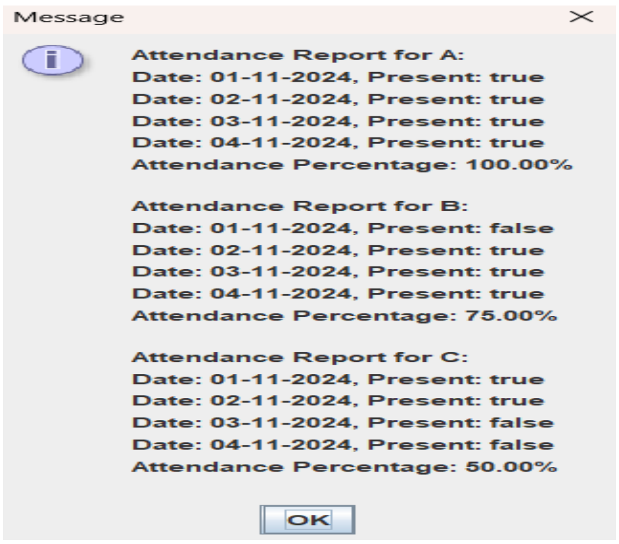


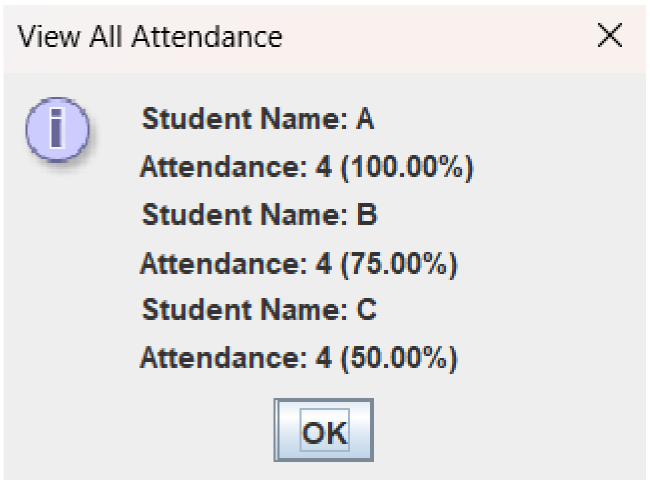



```
Message

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%

OK
```

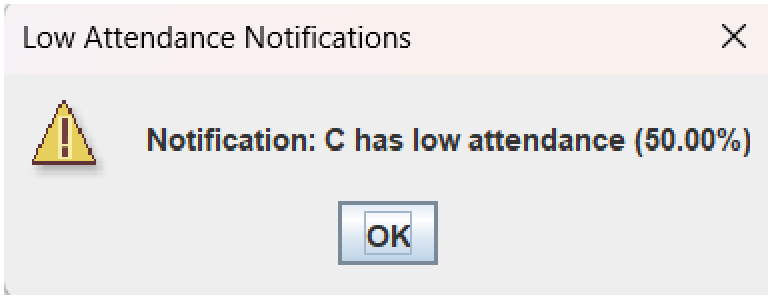

```
Message

Attendance Report for A:
Date: 01-11-2024, Present: true
Date: 02-11-2024, Present: true
Attendance Percentage: 100.00%

Attendance Report for B:
Date: 01-11-2024, Present: false
Date: 02-11-2024, Present: true
Attendance Percentage: 50.00%

Attendance Report for C:
Date: 01-11-2024, Present: true
Date: 02-11-2024, Present: true
Attendance Percentage: 100.00%

OK
```


```
Message

Attendance Report for A:
Date: 01-11-2024, Present: true
Date: 02-11-2024, Present: true
Date: 03-11-2024, Present: true
Attendance Percentage: 100.00%

Attendance Report for B:
Date: 01-11-2024, Present: false
Date: 02-11-2024, Present: true
Date: 03-11-2024, Present: true
Attendance Percentage: 66.67%

Attendance Report for C:
Date: 01-11-2024, Present: true
Date: 02-11-2024, Present: true
Date: 03-11-2024, Present: false
Attendance Percentage: 66.67%

OK
```


```
Message

Attendance Report for A:
Date: 01-11-2024, Present: true
Date: 02-11-2024, Present: true
Date: 03-11-2024, Present: true
Date: 04-11-2024, Present: true
Attendance Percentage: 100.00%

Attendance Report for B:
Date: 01-11-2024, Present: false
Date: 02-11-2024, Present: true
Date: 03-11-2024, Present: true
Date: 04-11-2024, Present: true
Attendance Percentage: 75.00%

Attendance Report for C:
Date: 01-11-2024, Present: true
Date: 02-11-2024, Present: true
Date: 03-11-2024, Present: false
Date: 04-11-2024, Present: false
Attendance Percentage: 50.00%

OK
```


View All Attendance

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


Low Attendance Notifications

```
Notification: C has low attendance (50.00%)
```

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.