SCHOOL MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

SIBI RAJ .A (2303811710421148)

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 **“SCHOOL MANAGEMENT SYSTEM”** is the bonafide work of **SIBI RAJ .A (2303811710421148)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
ASSISTANTP ROFESSOR

[illegible]

SIGNATURE	**SIGNATURE**
Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. A. Malarmannan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 06.12.2024

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

[illegible]

CGB1201-JAVA PROGRAMMING
Mr.R. KARTHIK, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8138-SCE, TRICHY.

[illegible]

INTERNAL EXAMINER	EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“SCHOOL 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

[illegible]

SIBI RAJ .A

Place: Samayapuram

Date: 06.12.2024

ACKNOWLEDGEMENT

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

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

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

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

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

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

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

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

VISION OF THE INSTITUTION

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

MISSION OF THE INSTITUTION

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

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The School Management System (SMS) is a software application designed to streamline the management of various administrative tasks in a school. Developed using Java programming, this system automates processes such as student enrollment, grade management, attendance tracking, timetable scheduling, and report generation. It provides a user-friendly interface for administrators, teachers, and students, ensuring efficient data management and easy access to information. The system supports multiple roles with different levels of access, offering features like student registration, fee management, academic performance tracking, and communication between teachers and parents. By centralizing school operations, the SMS enhances productivity, reduces paperwork, and ensures accurate and real-time data processing. The system is highly scalable and can be customized for different types of educational institutions. This system enables schools to manage student data, track academic performance, maintain attendance records, and facilitate communication between teachers, students, and parents. Key features of the system include user authentication, class scheduling, grade management, fee tracking, and report generation.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The School Management System (SMS) is a software application designed to streamline the management of various administrative tasks in a school. Developed using Java programming, this system automates processes such as student enrollment, grade management, attendance tracking, timetable scheduling, and report generation. It provides a user-friendly interface for administrators, teachers, and students, ensuring efficient data management and easy access to information. The system supports multiple roles with different levels of access, offering features like student registration, fee management, academic performance tracking, and communication between teachers and parents. By centralizing school operations, the SMS enhances productivity, reduces paperwork, and ensures accurate and real-time data processing. The system is highly scalable and can be customized for different types of educational institutions. This system enables schools to manage student data, track academic performance, maintain attendance records, and facilitate communication between teachers, students, and parents. Key features of the system include user authentication, class scheduling, grade management, fee tracking, and report generation.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of the project is to record and manage student personal details such as, (name, age, address, etc.), for maintaining the teacher profiles namely, (name, qualifications, contact details), to create, monitor, and manage courses for the students in the school, for generating and managing class schedules for students and teachers, and record student grades for exams and assignments and generate reports.

1.2 Overview

A School Management System (SMS) is a software solution designed to streamline and automate various administrative tasks within a school, ensuring efficiency, accuracy, and better management of resources. The system typically manages student information, attendance, grades, timetables, fee structures, library resources, and communication between students, teachers, and administrators. In Java, the School Management System can be developed using core Java concepts, object-oriented programming principles, and GUI (Graphical User Interface) frameworks such as Swing or JavaFX for a user-friendly interface. Java's rich ecosystem of libraries and frameworks also facilitates database integration (e.g., MySQL, SQLite) to store and manage large datasets.

1.3 Java Programming Concepts

1. Object – Oriented Programming (OOP) :

You'll create classes to represent entities like Student, Teacher, Course, ClassRoom, etc. Each class will have attributes (variables) and methods (functions) that define the behavior of that entity.

2. Control Flow Statement :

Loops are used to iterate over collections like lists of students or teachers. A for loop to display all student names or to calculate the average grade.

3. Exception Handling :

Try-catch blocks: Exception handling helps to deal with runtime errors such as invalid input, missing files, or database connection errors. This ensures that your program doesn't crash unexpectedly.

4. File (I/O) (Input/Output) :

Saving student data, grades, or class schedules to a file or reading data from a file can be done using FileReader, BufferedReader, FileWriter, and BufferedWriter.

5. GUI (GRAPHICAL USER INTERFACE) :

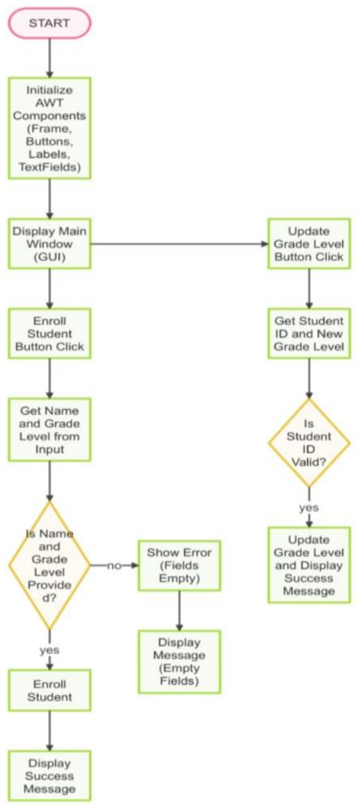
Used to provide a user-friendly interface, you can use Swing or JavaFX to create windows, buttons, tables, forms, and other UI components.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the School Management System to provide an efficient way for school administrators, teachers, and students to interact with the system, enabling smoother workflows and better organization are, Student details include personal information, contact details, enrollment number, and grade. Admin can add new students to the system, update personal details, and remove students when necessary. Teachers can view their class schedules, assignments, and daily tasks. Admin can add, remove, or edit teacher details, such as personal information, subject specialization, and assigned classes. Admin can create new courses (subjects) and assign them to relevant teachers. Teachers can update their course details. Students, teachers, and administrators can access and view schedules for classes and exams. Teachers can input grades for students based on their performance in assignments, tests, and exams.

2.2 Block Diagram


START
Initialize AWT Components (Frame, Buttons, Labels, TextFields)
Display Main Window (GUI)
Update Grade Level Button Click
Enroll Student Button Click
Get Student ID and New Grade Level
Get Name and Grade Level from Input
Is Student ID Valid?
yes
Update Grade Level and Display Success Message
Is Name and Grade Level Provided?
no
Show Error (Fields Empty)
yes
Display Message (Empty Fields)
Enroll Student
Display Success Message

CHAPTER 3
MODULE DESCRIPTION

3.1 Student Management Module

The Student Management module is the core of a School Management System, responsible for maintaining a comprehensive database of all student information. This includes personal details such as name, address, date of birth, and contact information, along with academic details like enrollment status, grades, and subject selection. The module allows administrators or staff to register new students, update existing records, track academic performance, and monitor progress over time. Additionally, it can generate student reports, such as progress cards, and help in the overall management of student demographics, making it easier for school officials to access and manage student information efficiently.

3.2 Staff Management Module

The Staff Management module is designed to handle all aspects of teacher administration. It stores essential teacher information, including personal details, qualifications, subjects taught, work schedule, and salary information. The module enables administrators to assign teachers to specific subjects or classes, manage their workloads, and ensure compliance with the teaching curriculum. It can also include functionalities for tracking teacher performance, evaluating teaching methods, and organizing professional development opportunities. This module streamlines the management of teaching staff, ensuring that the right teachers are assigned to the right subjects and courses.

3.3 Course Management Module

The Course and Subject Management module helps in organizing and structuring academic courses and classrooms. It allows the creation of various classes, courses, and subjects offered by the school. Administrators can assign specific teachers to these subjects and organize them according to the school's curriculum. This module ensures that all academic activities are aligned with the school's educational standards and helps in generating class schedules. It also facilitates easy updates if changes in the curriculum or subject offerings are required. It provides an efficient system for managing subjects, ensuring that each student's academic path is well-structured and easy to follow.

3.4 Time Table Management Module

The Timetable Management module is essential for scheduling and organizing school events, classes, and exams. It enables administrators to create daily, weekly, or monthly timetables for students and teachers. This module ensures that there are no conflicts in scheduling, such as overlapping subjects or teacher availability issues. It also helps in allocating classrooms, optimizing space, and making adjustments based on availability or special events. The system can be flexible, allowing for quick updates or changes, and can send automatic reminders to both teachers and students about upcoming classes or activities.

3.5 Grade And Examination Module

The Grade and Examination Management module is integral for managing academic assessments. It handles the creation, scheduling, and grading of exams, both internal (midterms, quizzes) and external (final exams, standardized tests). Teachers can input grades, calculate results, and generate report cards or transcripts. The module allows students to view their results, track their academic progress, and identify areas for improvement. It can also include functionality for managing different grading systems, weightings for assignments or exams, and setting up rubrics for grading. This system not only simplifies the exam and grading process but also ensures transparency and accessibility for students and parents.

CHAPTER 4
CONCLUSION

In conclusion, the School Management System in Java provides a comprehensive, efficient, and secure solution for managing the various administrative and academic functions within a school. By integrating multiple modules such as student management, attendance tracking, grade management, timetable scheduling, fee management, and communication, this system streamlines day-to-day operations, reduces manual workloads, and enhances overall productivity. The School Management System also enhances communication between students, teachers, and parents by providing real-time updates on attendance, grades, and important school events. This promotes transparency, facilitates faster decision-making, and allows stakeholders to stay informed and engaged in the educational process.

FUTURE SCOPE

The future scope of a School Management System (SMS) is vast, with continuous advancements in technology offering opportunities to enhance its functionality, efficiency, and user experience. One promising development is the integration of Artificial Intelligence (AI) and Machine Learning (ML) to automate administrative tasks, predict student performance trends, and provide personalized learning pathways. AI can help in grading, identifying learning gaps, and offering tailored recommendations to students based on their strengths and weaknesses. Additionally, the incorporation of Blockchain for secure and transparent record-keeping of student achievements, grades, and credentials could revolutionize how academic records are stored and accessed.

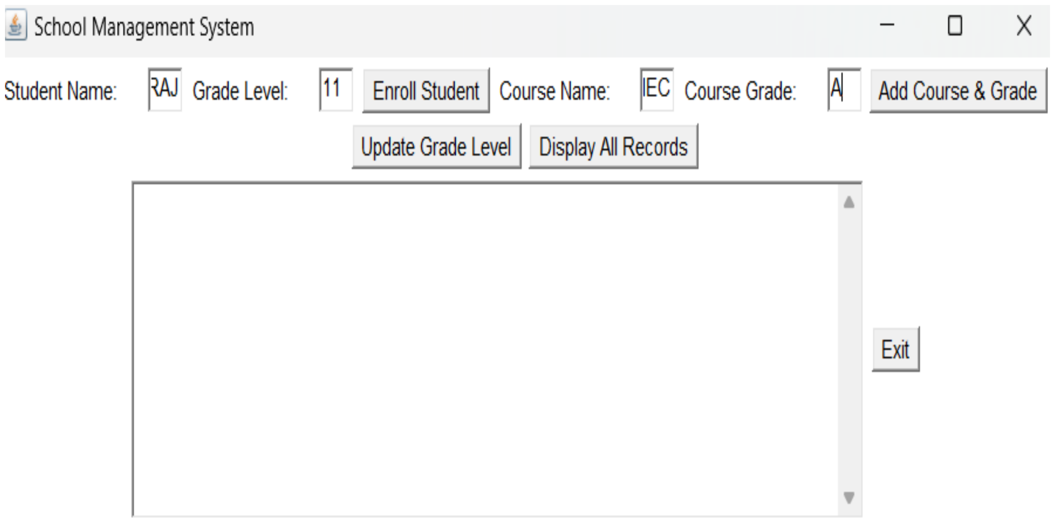
APPENDIX A
(SOURCE CODE)

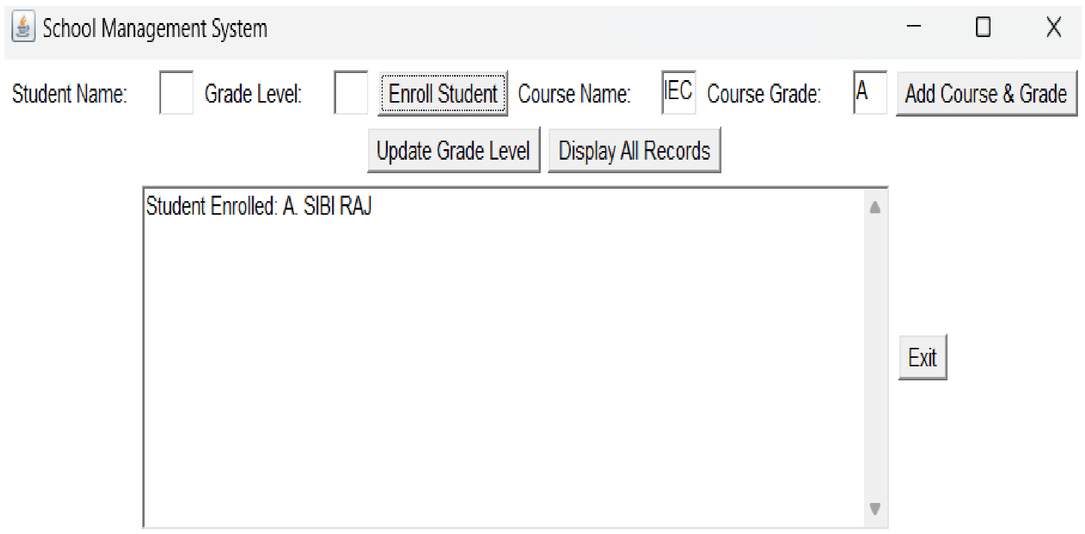
```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

// Define the Student class
class Student {
    private String name;
    private int id;
    private String gradeLevel;
    private ArrayList<String> courses;
    private ArrayList<String> grades;

    public Student(String name, int id, String gradeLevel) {
        this.name = name;
        this.id = id;
        this.gradeLevel = gradeLevel;
        this.courses = new ArrayList<>();
        this.grades = new ArrayList<>();
    }

    // Getters and setters
    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public String getGradeLevel() {
        return gradeLevel;
    }

    public void setGradeLevel(String gradeLevel) {
        this.gradeLevel = gradeLevel;
    }

    public void addCourse(String course, String grade) {
        this.courses.add(course);
        this.grades.add(grade);
    }

    public void updateCourseGrade(String course, String newGrade) {
        int index = courses.indexOf(course);
        if (index != -1) {
            grades.set(index, newGrade);
        }
    }

    public String displayStudentRecord() {
        StringBuilder record = new StringBuilder();
        record.append("Student Name: ").append(name)
              .append("\nStudent ID: ").append(id)
              .append("\nGrade Level: ").append(gradeLevel)
              .append("\nCourses & Grades: \n");
        for (int i = 0; i < courses.size(); i++) {
            record.append(courses.get(i)).append(": ").append(grades.get(i)).append("\n");
        }
        return record.toString();
    }
}

// School Management System class
class SchoolManagementSystem {
    private ArrayList<Student> students;
    private int nextId;

    public SchoolManagementSystem() {
        students = new ArrayList<>();
        nextId = 1;  // Start student IDs from 1
    }

    public void enrollStudent(String name, String gradeLevel) {
        Student newStudent = new Student(name, nextId, gradeLevel);
        students.add(newStudent);
        nextId++;  // Increment ID for the next student
    }

    public Student findStudentById(int id) {
        for (Student student : students) {
            if (student.getId() == id) {
                return student;
            }
        }
        return null;  // If student not found
    }

    public void updateStudentGradeLevel(int id, String newGradeLevel) {
        Student student = findStudentById(id);
        if (student != null) {
            student.setGradeLevel(newGradeLevel);
        }
    }

    public void addCourseToStudent(int id, String course, String grade) {
        Student student = findStudentById(id);
        if (student != null) {
            student.addCourse(course, grade);
        }
    }

    public String displayStudentRecords() {
        StringBuilder records = new StringBuilder();
        for (Student student : students) {
            records.append(student.displayStudentRecord()).append("\n----------------------------
------\n");
        }
        return records.toString();
    }
}

// Main AWT Application
public class SchoolSystemAppAWT {
    public static void main(String[] args) {
        Frame frame = new Frame("School Management System");

        // School management system instance
        SchoolManagementSystem sms = new SchoolManagementSystem();

        // Components for the AWT GUI
        Label labelName = new Label("Student Name: ");
        Label labelGradeLevel = new Label("Grade Level: ");
        Label labelCourse = new Label("Course Name: ");
        Label labelGrade = new Label("Course Grade: ");
        TextField textName = new TextField();
        TextField textGradeLevel = new TextField();
        TextField textCourse = new TextField();
        TextField textGrade = new TextField();
        TextArea textAreaDisplay = new TextArea();
        Button btnEnroll = new Button("Enroll Student");
        Button btnUpdateGradeLevel = new Button("Update Grade Level");
        Button btnAddCourse = new Button("Add Course & Grade");
        Button btnDisplayRecords = new Button("Display All Records");
        Button btnExit = new Button("Exit");

        // Layout setup
        frame.setLayout(new FlowLayout());
        frame.setSize(400, 400);

        // Add components to frame
        frame.add(labelName);
        frame.add(textName);
        frame.add(labelGradeLevel);
        frame.add(textGradeLevel);
        frame.add(btnEnroll);
frame.add(labelCourse);
frame.add(textCourse);
frame.add(labelGrade);
frame.add(textGrade);
frame.add(btnAddCourse);

frame.add(btnUpdateGradeLevel);
frame.add(btnDisplayRecords);
frame.add(textAreaDisplay);
frame.add(btnExit);

// Action Listeners for buttons
btnEnroll.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String name = textName.getText();
        String gradeLevel = textGradeLevel.getText();
        if (!name.isEmpty() && !gradeLevel.isEmpty()) {
            sms.enrollStudent(name, gradeLevel);
            textName.setText("");
            textGradeLevel.setText("");
            textAreaDisplay.setText("Student Enrolled: " + name);
        }
    }
});

btnUpdateGradeLevel.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String gradeLevel = textGradeLevel.getText();
        try {
            int id = Integer.parseInt(textName.getText());
            sms.updateStudentGradeLevel(id, gradeLevel);
            textAreaDisplay.setText("Grade level updated for student ID: " + id);
        } catch (NumberFormatException ex) {
            textAreaDisplay.setText("Invalid student ID");
        }
    }
});

btnAddCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String course = textCourse.getText();
        String grade = textGrade.getText();
        try {
            int id = Integer.parseInt(textName.getText());
            sms.addCourseToStudent(id, course, grade);
            textAreaDisplay.setText("Course " + course + " added for student ID: " + id);
        } catch (NumberFormatException ex) {
            textAreaDisplay.setText("Invalid student ID");
        }
    }
```

```
 });

 btnDisplayRecords.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
       textAreaDisplay.setText(sms.displayStudentRecords());
    }
 });

 btnExit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
       System.exit(0);  // Exit the application
    }
 });

 // Frame setup
 frame.setVisible(true);
 }
}
```

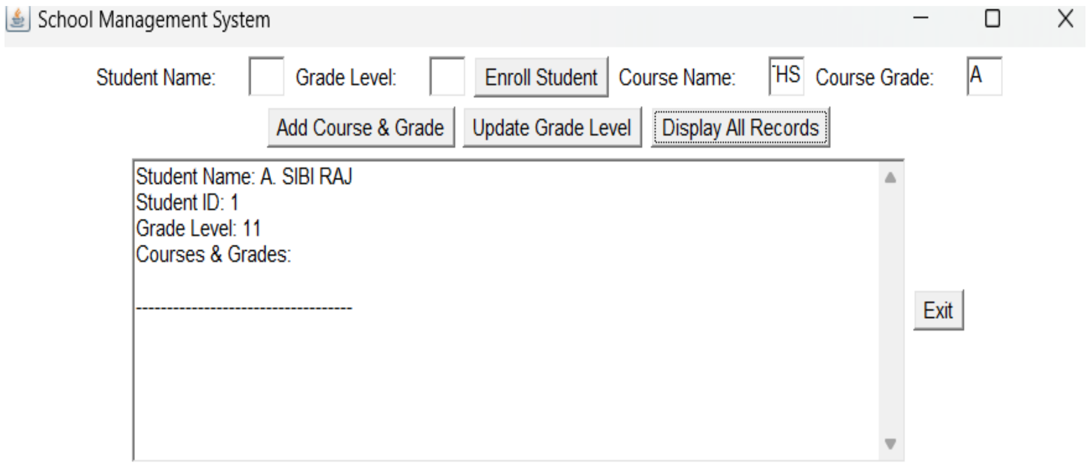
APPENDIX B

(SCREENSHOTS)





```
Student Enrolled: A. SIBI RAJ
```



```
Student Name: A. SIBI RAJ
Student ID: 1
Grade Level: 11
Courses & Grades:

------------------------------------
```

REFERENCES

1. Steve Holmes, E. and Reachel Adams, J. (2019) ‘Management System in Java – MSJ’ , Oswald – Journal of Computing Vol.31, No.7, pp.124 – 168.

2. Benjamin R.W. and Kristina, C. (2020) ‘Applications of Java System for Management in University Theory’ , Micah Moore, N., Vol.13, pp.87 – 95.

3. Jin Sakai and Nathan George, N.D. (2021) ‘Open World Java and it’s Applications’ , North California, pp. 1234 – 1241.