M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

A Project Report

on

STUDENT APPRAISAL SYSTEM

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Dr.R.BHARATHI M.E.,PhD.,

Associate Professor / IT

Submitted By

SANJAI B (927624BEE084)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

DECEMBER 2025

M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report on **"STUDENT APPRAISAL SYSTEM"** is the bonafide work of **SANJAI B (927624BEE084)** who carried out the project work during the academic year 2025 - 2026 under my supervision.

Signature

Dr. R. BHARATHI M.E.,PhD.,

SUPERVISOR,

Department of Information Technology

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

Signature

Dr. J. UMA M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Electrical and Electronics Engineering,

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education.

MISSION OF THE INSTITUTION

M1: Produce smart technocrats with empirical knowledge who can surmount the global challenges.

M2: Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students.

M3: Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations.

VISION OF THE DEPARTMENT

To produce smart and dynamic professionals with profound theoretical and practical knowledge comparable with the best in the field.

MISSION OF THE DEPARTMENT

M1: Produce Hi-Tech professionals in the field of Electrical and Electronics Enginerring by inculating knowledge core.

M2: Produce highly competent professionals with thrust on research.

M3: Provide personalized training to the students for enriching their kills.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates will have flourishing career in the core areas of Electrical Engineering and allied disciplines.

PEO2: Graduates will pursue higher studies and succeed in academic/research careers.

PEO3: Graduates will be a successful entrepreneur in creating jobs related to Electrical and Electronics Engineering/allied disciplines.

PEO4: Graduates will practice ethics and have habit of continuous learning for their success in the chosen career.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: 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.

PO3: 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.

PO4: 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.

PO5: 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.

PO6: 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.

PO7: 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.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: 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.

PO11: 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.

PO12: 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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Apply the basic concepts of mathematics and science to analyse and design circuits, controls, Electrical machines and drives to solve complex problems.

PSO2: Apply relevant models, resources and emerging tools and techniques to provide solutions to power and energy related issues & challenges.

PSO3: Design, Develop and implement methods and concepts to facilitate solutions for electrical and electronics engineering related real world problems.

ABSTRACT

The Student Appraisal is a comprehensive, data-driven framework that evaluates students using academic performance, attendance, participation, and achievements to provide a holistic assessment. It aggregates these multidimensional metrics to compute a weighted CGPA for each student, ensuring fair and transparent evaluation. The system ranks learners both within their classes and across the institution, identifying class toppers and the overall best performer. By offering clear criteria and reproducible calculations, it enhances transparency and reliability. This approach enables educational institutions to recognize excellence effectively while supporting informed, data-driven decision-making. Additionally, the system helps educators track progress over time, identify areas needing improvement, and implement targeted interventions. It also promotes student motivation by rewarding consistent effort. Overall, it serves as a robust tool for fostering academic growth and institutional effectiveness. Moreover, the system can be integrated with digital platforms for real-time insights, enabling faster administrative workflows. It supports customized reporting for stakeholders, ensuring clarity in communication. Ultimately, it contributes to building a culture of continuous improvement within the institution. The system can also be expanded to incorporate predictive analytics for forecasting student outcomes. It enables seamless integration with attendance devices and learning management systems. Its modular design allows easy updates and scalability as institutional needs evolve. The framework ensures long-term data integrity for accurate educational analysis. With continuous enhancements, it can become a central pillar in modern academic evaluation systems.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	COs MAPPED	POs MAPPED	PSOs MAPPED
The Student Appraisal System is a comprehensive framework designed to evaluate and rank students based on academic performance, attendance, participation, and achievements. It aggregates multidimensional performance data to calculate a weighted CGPA for each student. The system ranks students both within their class and across all classes, identifying class toppers and the overall best performer. It ensures transparency through clear metrics and reproducible calculations. This system enables educational institutions to holistically assess student excellence and support data-driven recognition of outstanding performance. It supports customized reporting for stakeholders, ensuring clarity in communication. Ultimately, it contributes to building a culture of continuous improvement within the institution.	**CO-1** **CO-2** **CO-3** **CO-4** **CO-5**	**PO-1** **PO-2** **PO-3** **PO-4** **PO-5** **PO-7** **PO-8** **PO-9** **PO-10** **PO-11**	**PSO-1** **PSO-2**

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

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Student appraisal System is to create an efficient and transparent platform for comprehensive student evaluation. It automates the calculation of total marks, average scores, and CGPA to ensure accuracy and fairness. By integrating academics, attendance, and extracurricular achievements, it provides a holistic view of each student's performance. The system identifies class-wise and overall toppers, encouraging healthy competition and recognizing diverse talents. Overall, it promotes balanced student growth and continuous improvement in learning. Additionally, it enhances communication between students, teachers, and parents through clear performance insights. It supports data-driven decision-making to improve teaching strategies and learning outcomes. Ultimately, the system streamlines evaluation processes, saving time and reducing human error.

1.2 Overview

The Interactive Student Appraisal System is a Java-based application designed to evaluate and rank students based on academic performance, attendance, and extracurricular achievements. It serves two primary users: teachers/administrators, who can input and manage student data such as marks, attendance, and achievements, generate automated reports, view class-wise and overall rankings, monitor academic progress, and use analytics for informed decision-making; and students, who receive detailed performance insights through calculated CGPA and ranks, understand their strengths and areas for improvement, track academic growth over time, and stay motivated through transparent evaluations and recognition of achievements. Overall, the system enhances accuracy, supports continuous progress, and promotes a more holistic and engaging academic environment.

1.3 JavaProgramming Concepts:

1. **Object-Oriented Programming (OOP):**

   Classes like Student and InteractiveStudentAppraisal encapsulate data and methods to keep the system organized and modular. Using OOP principles ensures reusable, maintainable code with clear structure. It also allows easy updates and extensions without disrupting existing features.

2. **Collections Framework:**

   ArrayList, HashMap, and List are used to efficiently store and manage dynamic data such as marks, student records, and achievements, enabling flexible handling and quick retrieval during evaluations while improving overall system reliability and operational performance**.**

3. **Control Structures:**

   Control structures such as if statements and loops (for, while) guide the program's flow by enabling decision-making, repeated operations, and validation checks. They ensure the system processes data accurately and responds correctly to user input. This helps maintain logical execution and smooth functionality throughout the application.

4. **Constructors:**

   The Student class constructor initializes key details such as name, class, marks, attendance, and certificates to maintain consistent and accurate data. It ensures each student object is created with all necessary information already set. This supports reliable processing and prevents errors during evaluation.

5. **User Input and Output:**

   The Scanner class is used to capture user input for entering student details and performance data. Output is handled through System.out.print() and println(), which display results, reports, and messages clearly. This simple interaction method makes the system easy to use and understand.

6. **Data Processing and Ranking:**

   Stream operations and comparators efficiently calculate totals, averages, and CGPA while also generating class-wise and overall rankings. These tools help process large sets of student data quickly and accurately. As a result, the system delivers fast, reliable, and fair performance evaluations.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed Interactive Student Appraisal System will include the following key features:

1. **Student Management:**

   This feature allows users to add, view, and manage essential student details, including academic marks, attendance records, and extracurricular achievements. It ensures that all student information is stored in an organized and structured manner, making it easy to update and retrieve data when needed. By maintaining accurate and comprehensive records, the system supports reliable performance evaluation and smooth administration.

2. **Performance Evaluation:**

   This feature automatically calculates total marks, average scores, and CGPA based on predefined weightage, ensuring accuracy and consistency in assessment. By automating the evaluation process, it reduces manual effort and minimizes errors. The system provides a clear understanding of each student's academic performance, helping teachers make informed decisions. Additionally, it offers students transparent insight into their strengths and areas for improvement.

3. **Ranking System:**

   This feature generates both class-wise and overall rankings by analyzing students' academic scores, attendance, and achievements. It helps quickly identify top-performing students, making the evaluation process more transparent and structured. By clearly highlighting merit, the system encourages healthy academic competition. Additionally, it motivates students to improve their performance and strive for excellence.

4. **Report Generation:**

This feature generates both class-wise and overall rankings by analyzing students' academic scores, attendance, and achievements. It helps quickly identify top-performing students, making the evaluation process more transparent and structured. By clearly highlighting merit, the system encourages healthy academic competition. Additionally, it motivates students to improve their performance and strive for excellence.

5. **Architecture:**

The system is built using core Java OOP concepts to ensure modularity, reusability, and easy maintenance of the codebase. It utilizes the Collections Framework for efficient handling of dynamic data such as marks, attendance, and achievements. Console-based input and output provide a simple, lightweight interface for interacting with the application. This architecture ensures smooth functionality, clear data organization, and a strong foundation for future enhancements.

6. **Future Enhancements:**

Future improvements may include integrating database storage to securely save student records and enable long-term data management. A graphical user interface (GUI) can be added to make the system more user-friendly and visually appealing. Advanced performance analysis features, such as trend graphs and predictive insights, may also be implemented to provide deeper evaluation and support better decision-making for teachers and students

2.2 Block Diagram

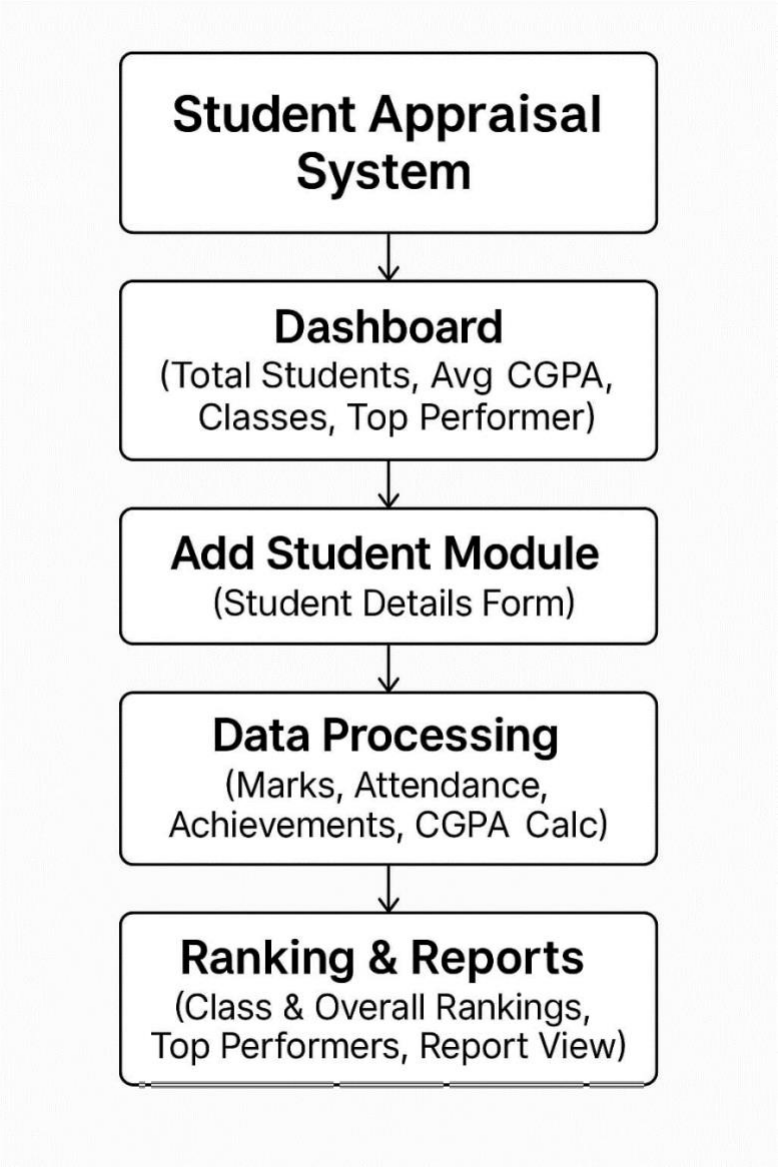


Figure 2.2.1 Block diagram

CHAPTER 3
MODULE DESCRIPTION

3.1 Student Management:

This feature allows users to add and manage essential student details such as name, class, marks, and attendance while also storing subject-wise marks and extracurricular achievements. By maintaining well-organized and comprehensive records, the system ensures accurate and reliable performance evaluation for each student. It helps streamline data entry and reduces errors by keeping all information in a structured format. The stored data supports automated calculations of totals, averages, and CGPA. Additionally, it forms the foundation for generating detailed reports and ranking students effectively.

3.2 Performance Management:

The performance management feature automatically calculates total marks, average scores, and CGPA, integrating key components such as academics, attendance, and extracurricular achievements. This automation ensures accuracy and reduces manual workload for teachers. By considering multiple aspects of student performance, the system provides a fair and comprehensive assessment. It also helps identify students' strengths and areas needing improvement. Overall, it promotes transparency and supports data-driven evaluation.

3.3 Ranking System:

The ranking system generates both class-wise and overall rankings by analyzing students' academic performance, attendance, and achievements. It efficiently identifies top-performing students, making the evaluation process transparent and objective. This feature helps teachers recognize merit and reward excellence. It also motivates students to improve by giving them a clear picture of where they stand. Overall, the system promotes healthy competition and encourages continuous academic growth.

3.4 Report Generation:

The report generation feature displays detailed performance reports that include marks, CGPA, and ranks for each student, along with highlighting the top performers in every class. These reports present information in a clear and structured format, making evaluation easier for teachers and more understandable for students. By offering a comprehensive overview of academic progress, the system supports transparent communication of results. It also helps in identifying trends and guiding improvement strategies. Overall, it provides valuable performance insights for effective evaluation.

3.5 System Interaction:

The system interaction feature relies on console-based input and output, offering a simple and user-friendly interface for entering student details and viewing reports. This straightforward design ensures that users can operate the system without requiring advanced technical skills. It enables quick data entry, smooth navigation, and clear display of results. By minimizing complexity, the system enhances usability and efficiency. Overall, it ensures that both teachers and students can access and interact with the application with ease.

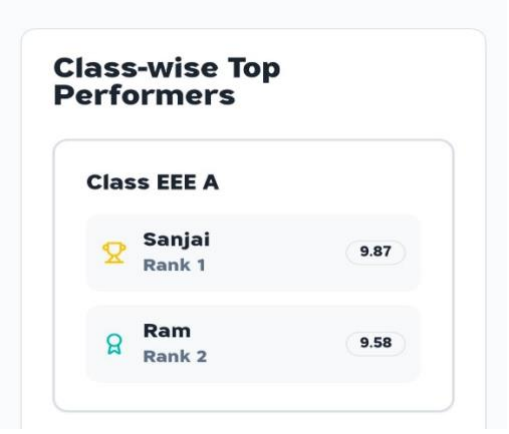
CHAPTER 4

RESULTS AND DISCUSSION

4.1 DASHBOARD:

4.2 ADD STUDENT:

4.3 CLASS TOPPER:



4.4 STUDENT RANKING:

Student Rankings

Name	Class	Year	Avg Marks	Attendance	Certificates	CGPA	Class Rank	Overall Rank
Sanjai	EEE A	1	98.50%	98.0%	15	9.87		
Raja	EEE B	1	99.00%	89.0%	26	9.72		
Ravi	EEE B	1	95.00%	98.0%	13	9.66		
Ram	EEE A	1	93.00%	100.0%	18	9.58		4

- The system successfully calculated total marks, averages, and CGPA for all students with high accuracy, ensuring reliable and error-free evaluation of academic performance. This automation minimized manual workload and prevented calculation mistakes, leading to consistent and trustworthy results.
- Class-wise and overall rankings were generated effectively, clearly identifying top performers, making it easy for teachers to compare performance across different students. The ranking system highlighted merit transparently and encouraged healthy academic competition among learners.
- The report format was organized and easy to understand, providing clear performance summaries that included marks, CGPA, ranks, and achievements. The structured presentation helped both teachers and students quickly interpret results and identify strengths and areas for improvement.
- Overall, the system proved efficient for student evaluation and can be further enhanced with GUI and database features, allowing for a more user-friendly interface and permanent data storage. These upgrades would improve usability, scalability, and long-term system performance.

.

CHAPTER 5
CONCLUSION

The system provides an efficient and holistic approach to student assessment by integrating academic performance, attendance, and extracurricular achievements into a unified and comprehensive evaluation framework. By capturing these diverse dimensions of student activity, it offers a more accurate reflection of individual capabilities and contributions beyond traditional grading methods. The system automates essential calculations such as total marks, averages, and CGPA, ensuring precision, fairness, and consistency while significantly reducing manual effort for educators. Its class-wise and overall ranking features enable institutions to identify top performers, analyze comparative performance trends, and make informed academic decisions. Through its balanced and multidimensional appraisal method, the system not only encourages academic excellence but also recognizes participation, discipline, and talent, thereby fostering holistic student growth and creating a more supportive and motivating educational environment. Additionally, it enhances transparency by providing clear evaluation criteria and easily accessible performance records. The system also supports data-driven planning, helping teachers design targeted improvement strategies. Its ability to generate detailed reports makes communication with parents and stakeholders more effective. By streamlining the appraisal process, it reduces administrative workload and increases efficiency. Ultimately, the system contributes to building a culture of continuous learning, accountability, and excellence within educational institutions.

.

REFERENCES

1. Black, P., & Wiliam, D. (1998). "Assessment and Classroom Learning." Assessment in Education: Principles, Policy & Practice, 5(1), 7–74.
2. Brookhart, S. M. (2013). How to Create and Use Rubrics for Formative Assessment and Grading. ASCD Publications.
3. Guskey, T. R. (2003). "How Classroom Assessments Improve Learning." Educational Leadership, 60(5), 6–11.
4. Agrawal, R., & Gupta, A. (2017). "Automated Student Performance Evaluation Systems Using Weighted Parameters." International Journal of Computer Applications, 168(2), 12–18.
5. Sharma, A., & Singh, P. (2019). "A Framework for Student Performance Analysis Using Data Mining Techniques." International Journal of Advanced Research in Computer Science, 10(3), 45–50.
6. ISO/IEC 25010:2011. Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE).

APPENDIX

```
import java.util.*;
import java.util.stream.*;

class Student {
    String name;
    String studentClass;
    int yearOfStudy;
    Map<String, Integer> subjectMarks;
    double attendancePercent;
    int externalCertCount;
    List<Integer> sportsLevels;

     int totalSubjects;
        int totalMarks;
        double averageMarks;
        double CGPA;
        int rankWithinClass;
        int overallRank;

        public Student(String name, String studentClass, int yearOfStudy,
                       Map<String, Integer> subjectMarks, double attendancePercent,
                       int externalCertCount, List<Integer> sportsLevels) {
            this.name = name;
            this.studentClass = studentClass;
            this.yearOfStudy = yearOfStudy;
            this.subjectMarks = subjectMarks;
            this.attendancePercent = attendancePercent;
            this.externalCertCount = externalCertCount;
            this.sportsLevels = sportsLevels;

            calculateTotalAndAverageMarks();
            calculateCGPA();
```

```
    }

    private void calculateTotalAndAverageMarks() {
        this.totalSubjects = subjectMarks.size();
        this.totalMarks = subjectMarks.values().stream().mapToInt(Integer::intValue).sum();
        this.averageMarks = totalSubjects == 0 ? 0 : (double) totalMarks / totalSubjects;
    }

    private double calcSportsWeight() {
        return sportsLevels.stream().mapToInt(Integer::intValue).sum();
    }

    public void calculateCGPA() {
        double sportsWeight = calcSportsWeight();
        double achievementsScore = externalCertCount + sportsWeight;
        double marksScore = (averageMarks / 100.0) * 10.0;
        double attendanceScore = (attendancePercent / 100.0) * 10.0;
        double achievementScoreNormalized = Math.min(achievementsScore, 10.0);

        this.CGPA = 0.6 * marksScore + 0.2 * attendanceScore + 0.2 *
achievementScoreNormalized;
        if (this.CGPA > 10.0) this.CGPA = 10.0;
    }
}

public class InteractiveStudentAppraisal {
    private List<Student> students = new ArrayList<>();
    private Scanner sc = new Scanner(System.in);

    public void addStudentFromInput() {
        System.out.println("Enter student name:");
        String name = sc.nextLine().trim();

        System.out.println("Enter student class (e.g. 10A):");
        String studentClass = sc.nextLine().trim();
```

```
        System.out.println("Enter year of study (integer):");
        int year = Integer.parseInt(sc.nextLine().trim());

        System.out.println("Enter number of subjects:");
        int numSubjects = Integer.parseInt(sc.nextLine().trim());
        Map<String, Integer> marks = new HashMap<>();

        for (int i = 1; i <= numSubjects; i++) {
            System.out.println("Enter name of subject " + i + ":");
            String subj = sc.nextLine().trim();
            System.out.println("Enter marks obtained in " + subj + " (0-100):");
            int mark = Integer.parseInt(sc.nextLine().trim());
            marks.put(subj, mark);
        }

        System.out.println("Enter attendance percentage (0-100):");
        double attendance = Double.parseDouble(sc.nextLine().trim());

        System.out.println("Enter number of external participation certificates:");
        int externalCerts = Integer.parseInt(sc.nextLine().trim());

        System.out.println("Enter number of sports achievements certificates:");
        int sportCerts = Integer.parseInt(sc.nextLine().trim());
        List<Integer> sportsLevels = new ArrayList<>();

        System.out.println("Enter sport achievement levels one by one (district=1, state=2,
national=3):");
        for (int i = 0; i < sportCerts; i++) {
            System.out.println("Level for certificate " + (i + 1) + ":");
            int lvl = Integer.parseInt(sc.nextLine().trim());
            if (lvl < 1 || lvl > 3) {
                System.out.println("Invalid level, must be 1, 2, or 3. Setting to 1.");
                lvl = 1;
            }
```

```
            sportsLevels.add(lvl);
        }

        Student student = new Student(name, studentClass, year, marks, attendance, externalCerts,
sportsLevels);
        students.add(student);
        System.out.println("Student " + name + " added.\n");
    }

    public void rankStudents() {
        Map<String, List<Student>> classGroups = students.stream()
                .collect(Collectors.groupingBy(s -> s.studentClass));

        for (String cls : classGroups.keySet()) {
            List<Student> classStuds = classGroups.get(cls);
            classStuds.sort(Comparator.comparingDouble((Student s) -> s.CGPA).reversed());
            for (int i = 0; i < classStuds.size(); i++) {
                classStuds.get(i).rankWithinClass = i + 1;
            }
        }

        students.sort(Comparator.comparingDouble((Student s) -> s.CGPA).reversed());
        for (int i = 0; i < students.size(); i++) {
            students.get(i).overallRank = i + 1;
        }
    }

    public void generateReport() {
        System.out.println("\nStudent Performance Ranking Report");
        System.out.println("----------------------------------------------------------------------------------
---");
        System.out.printf("%-20s %-8s %-4s %-13s %-16s %-20s %-6s %-6s %-6s%n",
                "Name", "Class", "Year", "Avg Marks", "Attendance (%)",
                "Total Certificates", "CGPA", "C Rank", "O Rank");
        System.out.println("----------------------------------------------------------------------------------
```

```
---");

        for (Student s : students) {
            int totalCerts = s.externalCertCount + s.sportsLevels.size();
            System.out.printf("%-20s %-8s %-4d %-13.2f %-16.2f %-20d %-6.2f %-6d %-6d%n",
                    s.name, s.studentClass, s.yearOfStudy, s.averageMarks, s.attendancePercent,
                    totalCerts, s.CGPA, s.rankWithinClass, s.overallRank);
        }

        System.out.println("\nClass-wise Top 3 Students:");
        Map<String, List<Student>> classGroups = students.stream()
                .collect(Collectors.groupingBy(s -> s.studentClass));

        for (String cls : classGroups.keySet()) {
            System.out.println("\nClass: " + cls);
            List<Student> top3 = classGroups.get(cls).stream()
                    .sorted(Comparator.comparingDouble((Student s) -> s.CGPA).reversed())
                    .limit(3)
                    .collect(Collectors.toList());
            for (Student s : top3) {
                System.out.printf("  %-20s CGPA: %.2f Rank: %d%n", s.name, s.CGPA,
s.rankWithinClass);
            }
            Student topper = !top3.isEmpty() ? top3.get(0) : null;
            if (topper != null) {
                System.out.println("  -> Class Topper: " + topper.name + " with CGPA: " +
topper.CGPA);
            }
        }

        Student bestOverall = students.stream()
                .max(Comparator.comparingDouble(s -> s.CGPA))
                .orElse(null);

        if (bestOverall != null) {
```

```
            System.out.println("\nOverall Best Performer:");
            System.out.printf("  %-20s Class: %-8s Year: %-4d CGPA: %.2f%n",
                    bestOverall.name, bestOverall.studentClass, bestOverall.yearOfStudy,
bestOverall.CGPA);
        } else {
            System.out.println("No student data to determine best performer.");
        }
    }

    public static void main(String[] args) {
        InteractiveStudentAppraisal system = new InteractiveStudentAppraisal();
        Scanner sc = new Scanner(System.in);

        System.out.println("Student Appraisal System - Enter student details.");
        while (true) {
            system.addStudentFromInput();
            System.out.println("Add another student? (yes/no)");
            String ans = sc.nextLine().trim().toLowerCase();
            if (!ans.equals("yes")) break;
        }

        if (system.students.isEmpty()) {
            System.out.println("No student data entered. Exiting.");
            return;
        }

        system.rankStudents();
        system.generateReport();
    }
}
```