Grade Management System

Project Overview

Project Name: Grade Management System

Description:

The Grade Management System is a Java-based console application designed to automate and simplify the process of managing student grades in educational institutions. It allows teachers, administrators, or school staff to efficiently record, store, and analyze student marks across multiple subjects. The system supports essential academic operations such as adding student marks, calculating averages, assigning grades based on predefined criteria, and generating detailed performance reports. By leveraging arrays and loops, the application can store marks for multiple students and subjects while providing an interactive menu-driven interface for seamless navigation.

This project not only streamlines the tedious process of manual grade calculation but also ensures accuracy and consistency in reporting student performance. The system is capable of identifying top performers, computing subject-wise averages, and producing a comprehensive grade distribution summary. Input validation ensures that only valid marks (0–100) and menu choices are accepted, reducing the chance of errors. Additionally, the modular structure of the code—with dedicated methods for adding marks, calculating averages, assigning grades, and generating reports—makes it easy to maintain, extend, or integrate with other educational tools in the future.

Overall, the Grade Management System serves as a practical demonstration of fundamental programming concepts such as arrays, loops, conditionals, methods, and error handling in Java, while also providing a functional tool for real-world academic management. It is suitable for small to medium-sized classrooms and can be easily extended to handle more subjects, dynamic student records, or even persistent storage using files or databases.

Objectives

The primary objective of the Grade Management System is to automate the process of student grade management while ensuring accuracy, efficiency, and ease of use. Specifically, the project aims to:

1. Provide a reliable system for storing student names and marks across multiple subjects.
2. Automate grade calculations by computing averages and assigning letter grades according to predefined criteria.
3. Offer a user-friendly, interactive menu interface for seamless navigation and operation.
4. Identify top-performing students and generate detailed performance reports, including subject averages and grade distributions.
5. Implement input validation and error handling to ensure accurate and consistent data entry.
6. Demonstrate the application of core Java programming concepts such as arrays, loops, methods, conditional statements, and exception handling.
7. Serve as a foundational tool that can be easily extended or integrated with more advanced academic management systems in the future.

Setup Instructions

Requirements:

- Java JDK 8 or higher
- IntelliJ IDEA (Community or Ultimate Edition)
- Basic knowledge of running Java applications

Installation Steps:

1. Install **Java JDK** and set the PATH environment variable.
2. Download and install **IntelliJ IDEA**.
3. Open IntelliJ → Click **New Project → Java → Next → Finish**.
4. Create a new Java class named **GradeManagementSystem**.
5. Copy the provided **Grade Management System code** into the class.
6. Right-click the file → **Run 'GradeManagementSystem.main()'**.
7. The application menu will appear in the console, ready for use.

Code Structure

File Hierarchy:

GradeManagementSystem/

└─ src/

  └─ GradeManagementSystem.java

Explanation:

- **GradeManagementSystem.java**: Main program containing:
  - Arrays to store student names and marks (studentNames, studentMarks)
  - Methods to add marks, view students, calculate averages, determine grades, find top performers, and generate reports
  - Input validation for menu and marks
  - Console-based menu system

Core Methods:

Method	Purpose
main()	Launches menu system and handles user input
addStudentMarks()	Adds a new student's marks with validation
viewAllStudents()	Displays all students in a table with averages
calculateAverages()	Calculates each student's average and grade
findTopPerformers()	Identifies the highest-scoring student(s)
generateReport()	Displays subject averages and grade distribution
getGrade()	Determines grade based on average
getValidInt()	Validates integer input for menu
getValidMark()	Validates marks between 0-100

Source Code:

GradeManagementSystem.java

```
import java.util.Scanner;

public class GradeManagementSystem {

    // Constants
    private static final int MAX_STUDENTS = 100;
    private static final int SUBJECT_COUNT = 5;

    // Arrays to store student names and marks
    private static String[] studentNames = new String[MAX_STUDENTS];
    private static double[][] studentMarks = new double[MAX_STUDENTS][SUBJECT_COUNT];
    private static int studentCount = 0;

    // Subjects
    private static final String[] subjects = {"Mathematics", "Science", "English", "History",
"Computer"};

    // Scanner for input
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean running = true;

        while (running) {
            printMenu();
            int choice = getValidInt(1, 6);

            switch (choice) {
                case 1 -> addStudentMarks();
                case 2 -> viewAllStudents();
                case 3 -> calculateAverages();
                case 4 -> findTopPerformers();
                case 5 -> generateReport();
                case 6 -> {
                    running = false;
                    System.out.println("Thank you for using Grade Management System!");
                }
```

```
        }
    }
    scanner.close();
}

// Print menu
private static void printMenu() {
    System.out.println("\n=== GRADE MANAGEMENT SYSTEM ===");
    System.out.println("1. Add Student Marks");
    System.out.println("2. View All Students");
    System.out.println("3. Calculate Averages");
    System.out.println("4. Find Top Performers");
    System.out.println("5. Generate Report");
    System.out.println("6. Exit");
    System.out.print("Enter your choice: ");
}

// Add student marks
private static void addStudentMarks() {
    if (studentCount >= MAX_STUDENTS) {
        System.out.println("Maximum student limit reached!");
        return;
    }

    System.out.println("\n=== ADD STUDENT MARKS ===");
    System.out.print("Enter Student Name: ");
    String name = scanner.nextLine();
    studentNames[studentCount] = name;

    System.out.println("\nEnter marks for 5 subjects (0-100):");
    for (int i = 0; i < SUBJECT_COUNT; i++) {
        System.out.print(subjects[i] + ": ");
        studentMarks[studentCount][i] = getValidMark();
    }

    studentCount++;
    System.out.println("✅ Student marks added successfully!");
}

// View all students
```

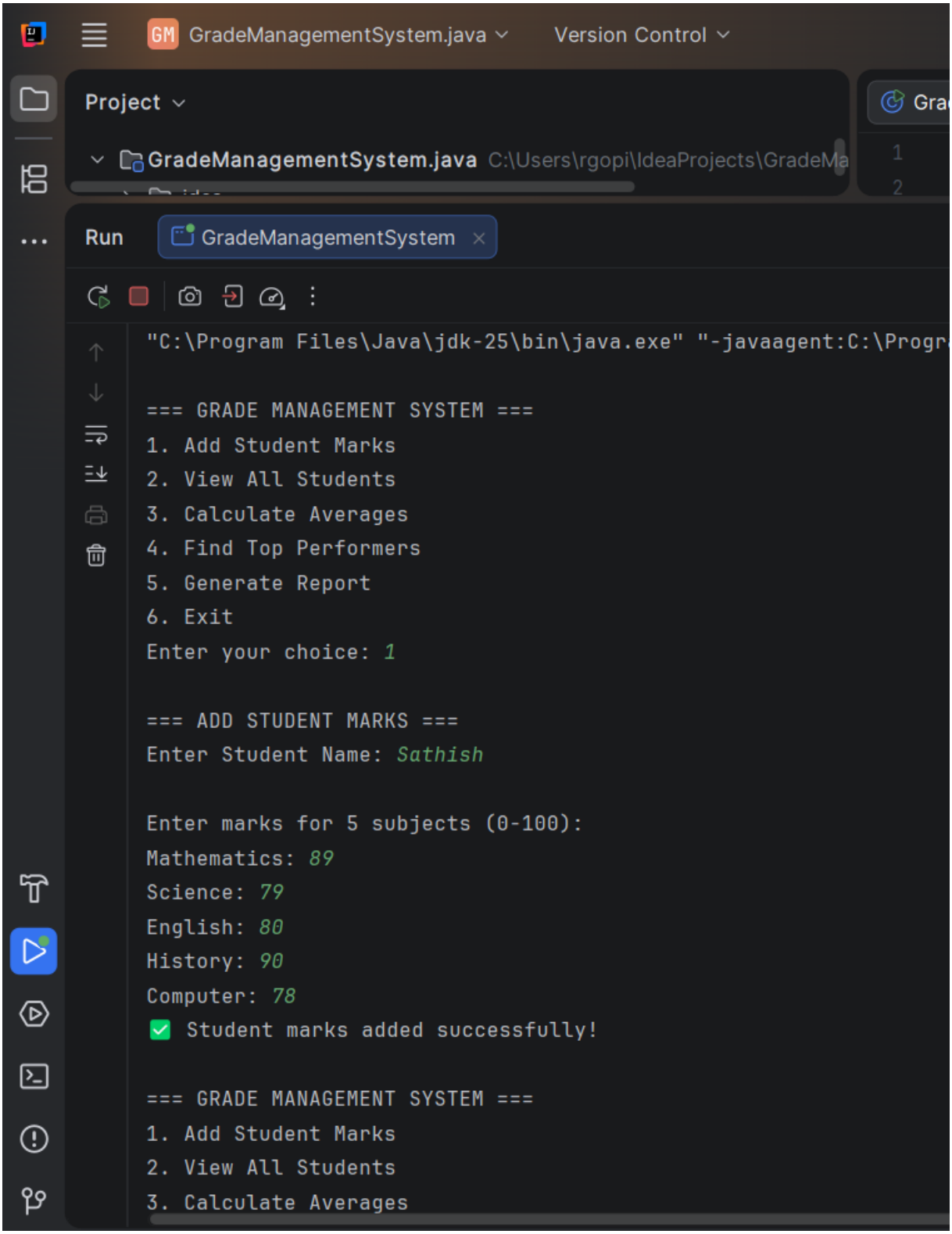
```
private static void viewAllStudents() {
    if (studentCount == 0) {
        System.out.println("No students found!");
        return;
    }

    System.out.printf("%-20s %-12s %-12s %-12s %-12s %-12s %-12s\n",
            "Student Name", "Math", "Science", "English", "History", "Computer", "Average");
    System.out.println("-".repeat(100));

    for (int i = 0; i < studentCount; i++) {
        double avg = calculateStudentAverage(i);
        System.out.printf("%-20s %-12.2f %-12.2f %-12.2f %-12.2f %-12.2f %-12.2f\n",
                studentNames[i],
                studentMarks[i][0], studentMarks[i][1], studentMarks[i][2],
                studentMarks[i][3], studentMarks[i][4],
                avg);
    }
}

// Calculate average for one student
private static double calculateStudentAverage(int index) {
    double sum = 0;
    for (int i = 0; i < SUBJECT_COUNT; i++) {
        sum += studentMarks[index][i];
    }
    return sum / SUBJECT_COUNT;
}

// Calculate averages and show grade
private static void calculateAverages() {
    if (studentCount == 0) {
        System.out.println("No students found!");
        return;
    }

    System.out.println("\n=== STUDENT AVERAGES ===");
    for (int i = 0; i < studentCount; i++) {
        double avg = calculateStudentAverage(i);
        String grade = getGrade(avg);
```

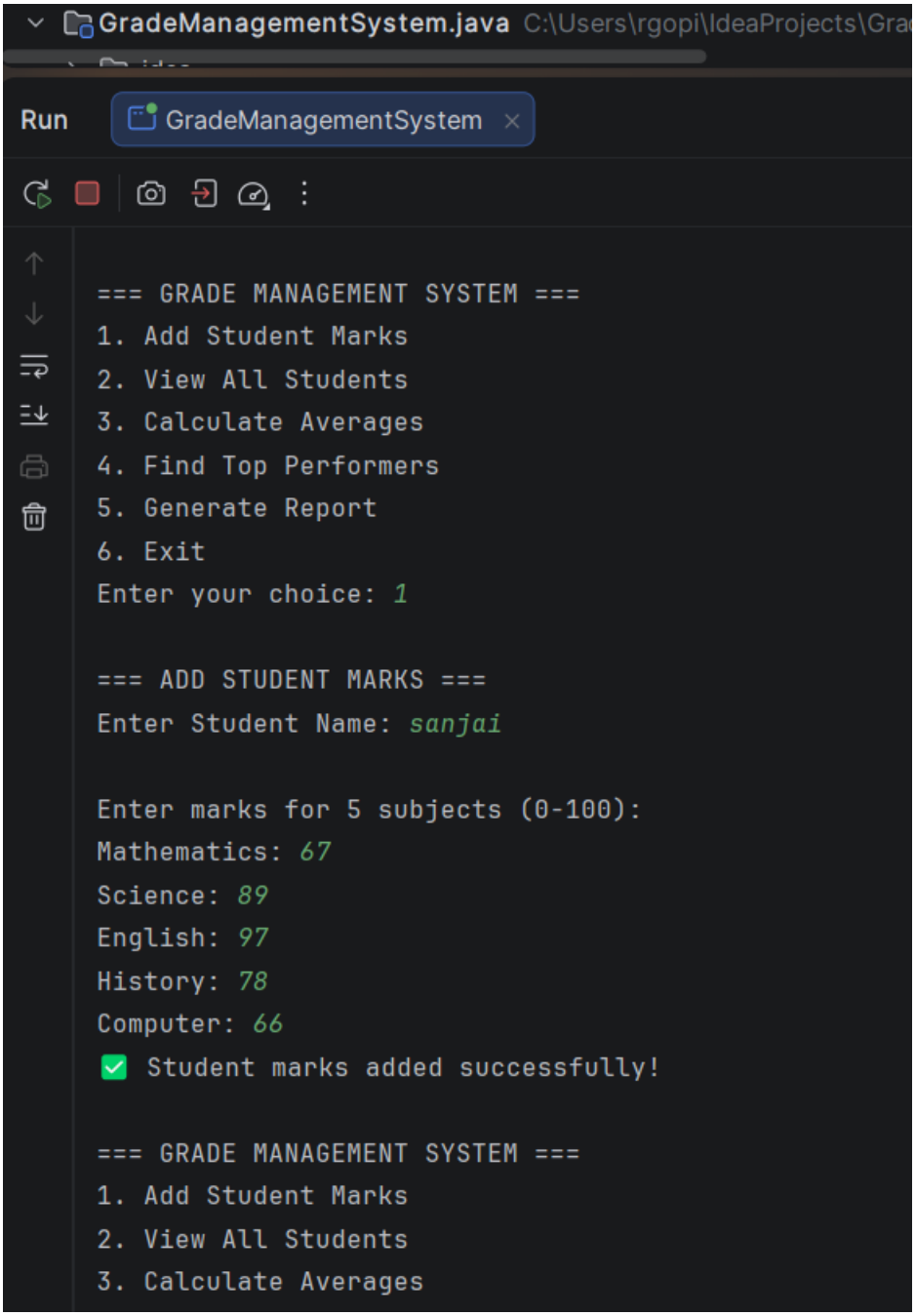
```
            System.out.printf("%-20s: Average = %.2f, Grade = %s\n",
                    studentNames[i], avg, grade);
        }
    }

    // Get grade based on average
    private static String getGrade(double average) {
        if (average >= 90) return "A+";
        else if (average >= 80) return "A";
        else if (average >= 70) return "B";
        else if (average >= 60) return "C";
        else if (average >= 50) return "D";
        else return "F";
    }

    // Find top performers
    private static void findTopPerformers() {
        if (studentCount == 0) {
            System.out.println("No students found!");
            return;
        }

        double highestAverage = -1;
        for (int i = 0; i < studentCount; i++) {
            double avg = calculateStudentAverage(i);
            if (avg > highestAverage) highestAverage = avg;
        }

        System.out.println("\n=== TOP PERFORMERS ===");
        for (int i = 0; i < studentCount; i++) {
            double avg = calculateStudentAverage(i);
            if (avg == highestAverage) {
                System.out.printf("%s - Average: %.2f\n", studentNames[i], avg);
            }
        }
    }

    // Generate performance report
    private static void generateReport() {
        if (studentCount == 0) {
```

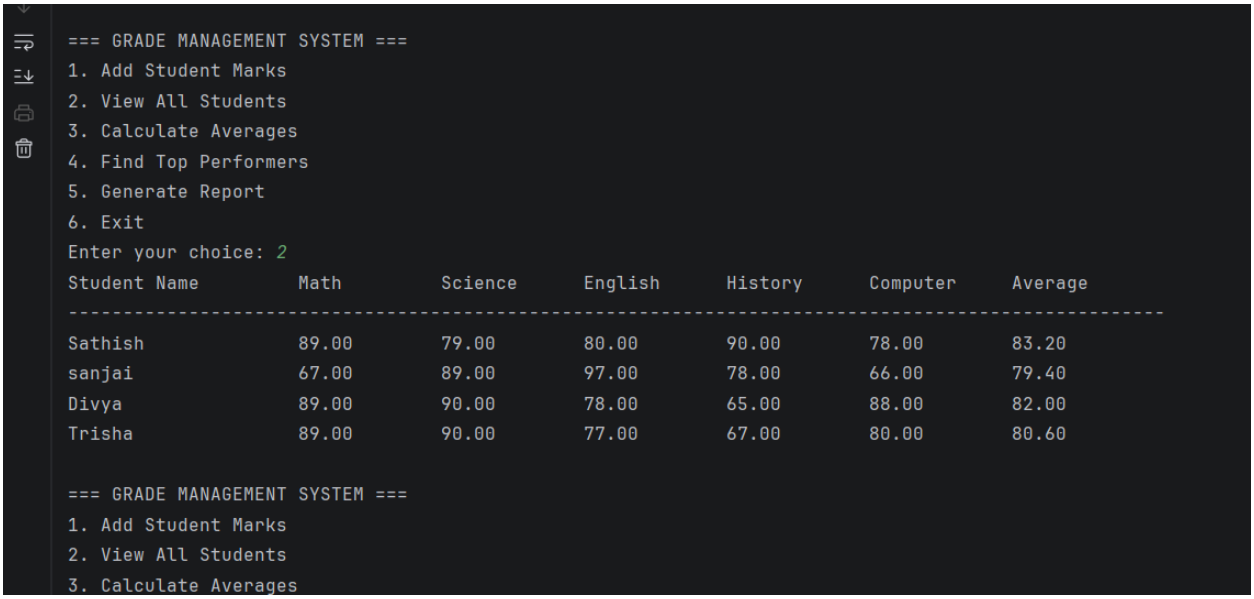
```
            System.out.println("No students found!");
            return;
        }

        System.out.println("\n=== PERFORMANCE REPORT ===");
        System.out.println("Total Students: " + studentCount);

        // Subject averages
        System.out.println("\n📊 SUBJECT AVERAGES:");
        for (int s = 0; s < SUBJECT_COUNT; s++) {
            double sum = 0;
            for (int i = 0; i < studentCount; i++) {
                sum += studentMarks[i][s];
            }
            double avg = sum / studentCount;
            System.out.printf("• %s: %.2f\n", subjects[s], avg);
        }

        // Grade distribution
        int[] gradeCounts = new int[6]; // A+, A, B, C, D, F
        for (int i = 0; i < studentCount; i++) {
            String grade = getGrade(calculateStudentAverage(i));
            switch (grade) {
                case "A+" -> gradeCounts[0]++;
                case "A"  -> gradeCounts[1]++;
                case "B"  -> gradeCounts[2]++;
                case "C"  -> gradeCounts[3]++;
                case "D"  -> gradeCounts[4]++;
                case "F"  -> gradeCounts[5]++;
            }
        }

        System.out.println("\n📈 GRADE DISTRIBUTION:");
        System.out.printf("• A+ Grade: %d\n• A Grade: %d\n• B Grade: %d\n• C Grade: %d\n• D
Grade: %d\n• F Grade: %d\n",
                gradeCounts[0], gradeCounts[1], gradeCounts[2], gradeCounts[3], gradeCounts[4],
gradeCounts[5]);
    }

    // Validate menu input
```

```
    private static int getValidInt(int min, int max) {
        int value;
        while (true) {
            try {
                value = scanner.nextInt();
                scanner.nextLine(); // consume newline
                if (value >= min && value <= max) return value;
                else System.out.printf("Enter number between %d and %d: ", min, max);
            } catch (Exception e) {
                System.out.print("Invalid input! Enter a number: ");
                scanner.nextLine(); // clear invalid input
            }
        }
    }

    // Validate marks input
    private static double getValidMark() {
        double mark;
        while (true) {
            try {
                mark = scanner.nextDouble();
                scanner.nextLine(); // consume newline
                if (mark >= 0 && mark <= 100) return mark;
                else System.out.print("Marks must be 0-100. Re-enter: ");
            } catch (Exception e) {
                System.out.print("Invalid input! Enter a number: ");
                scanner.nextLine();
            }
        }
    }
}
```

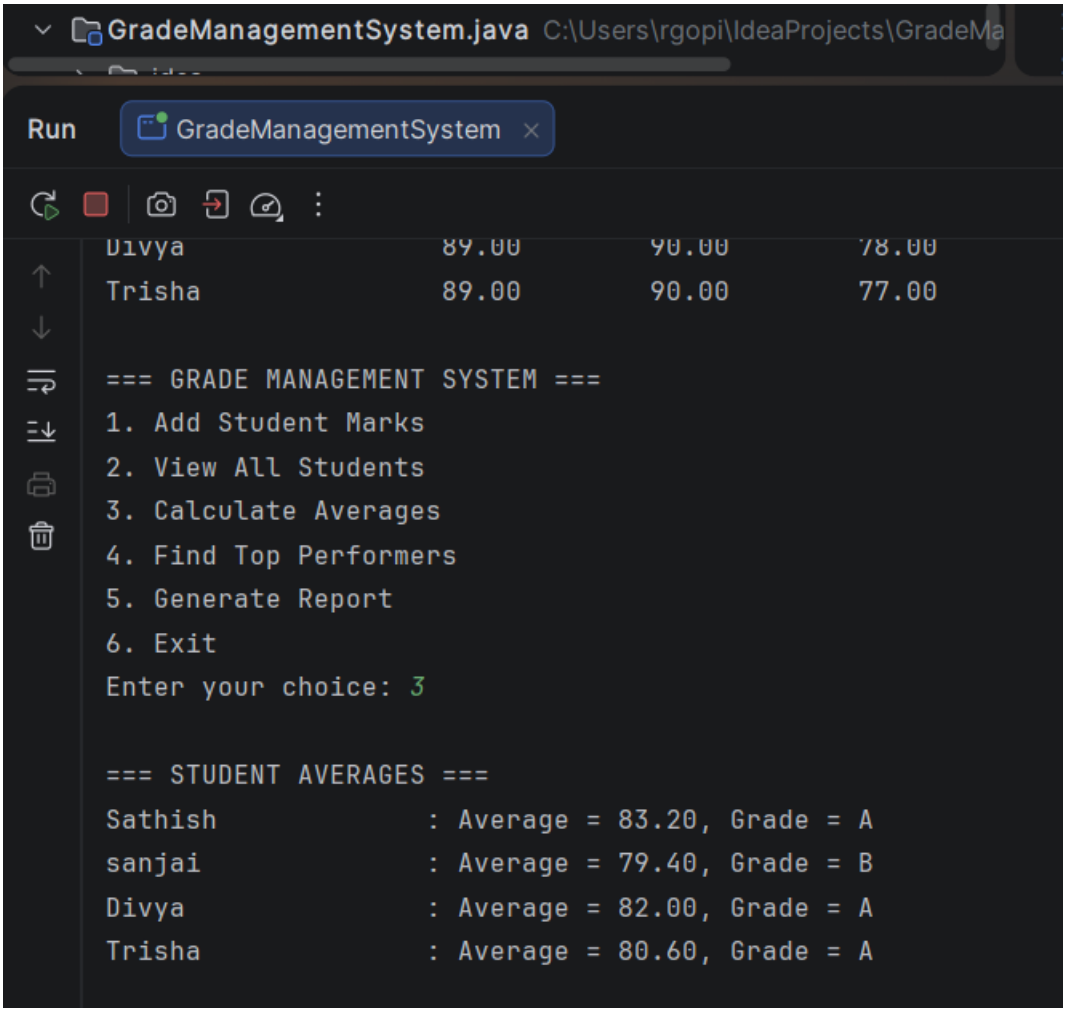
Visual Documentation




GradeManagementSystem.java C:\Users\rgopi\IdeaProjects\Gra
Run
GradeManagementSystem
=== GRADE MANAGEMENT SYSTEM ===
1. Add Student Marks
2. View All Students
3. Calculate Averages
4. Find Top Performers
5. Generate Report
6. Exit
Enter your choice: 1
=== ADD STUDENT MARKS ===
Enter Student Name: sanjai
Enter marks for 5 subjects (0-100):
Mathematics: 67
Science: 89
English: 97
History: 78
Computer: 66
Student marks added successfully!
=== GRADE MANAGEMENT SYSTEM ===
1. Add Student Marks
2. View All Students
3. Calculate Averages



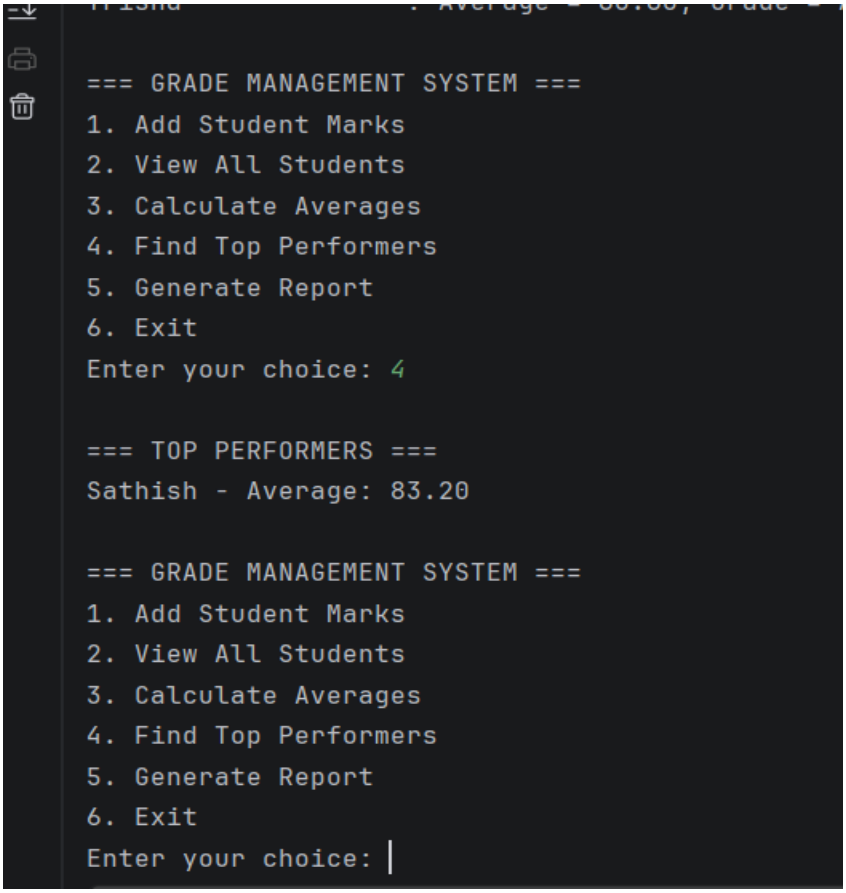
```
=== GRADE MANAGEMENT SYSTEM ===
1. Add Student Marks
2. View All Students
3. Calculate Averages
4. Find Top Performers
5. Generate Report
6. Exit
Enter your choice: 2
Student Name         Math         Science      English      History      Computer     Average
-------------------------------------------------------------------------------------------------
Sathish              89.00        79.00        80.00        90.00        78.00        83.20
sanjai               67.00        89.00        97.00        78.00        66.00        79.40
Divya                89.00        90.00        78.00        65.00        88.00        82.00
Trisha               89.00        90.00        77.00        67.00        80.00        80.60

=== GRADE MANAGEMENT SYSTEM ===
1. Add Student Marks
2. View All Students
3. Calculate Averages
```

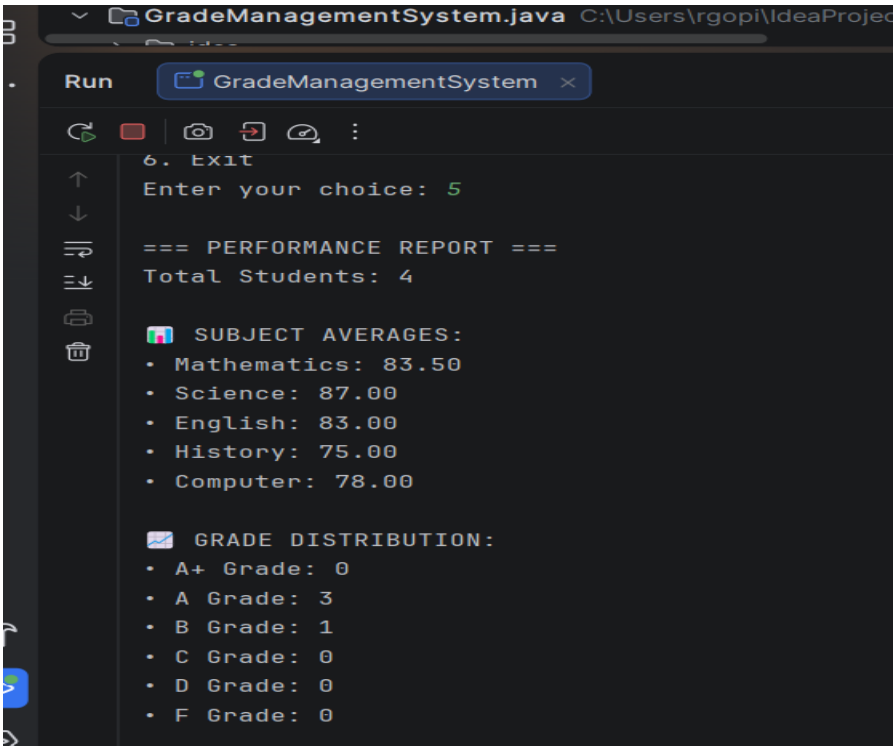


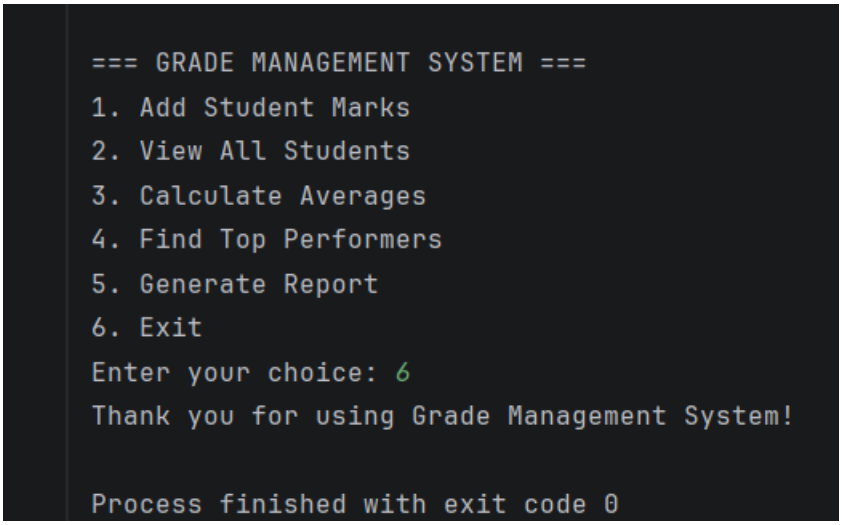
GradeManagementSystem.java C:\Users\rgopi\IdeaProjects\GradeMa

Run GradeManagementSystem

```
Divya                89.00        90.00        78.00
Trisha               89.00        90.00        77.00

=== GRADE MANAGEMENT SYSTEM ===
1. Add Student Marks
2. View All Students
3. Calculate Averages
4. Find Top Performers
5. Generate Report
6. Exit
Enter your choice: 3

=== STUDENT AVERAGES ===
Sathish              : Average = 83.20, Grade = A
sanjai               : Average = 79.40, Grade = B
Divya                : Average = 82.00, Grade = A
Trisha               : Average = 80.60, Grade = A
```


=== GRADE MANAGEMENT SYSTEM ===
1. Add Student Marks
2. View All Students
3. Calculate Averages
4. Find Top Performers
5. Generate Report
6. Exit
Enter your choice: 4
=== TOP PERFORMERS ===
Sathish - Average: 83.20
=== GRADE MANAGEMENT SYSTEM ===
1. Add Student Marks
2. View All Students
3. Calculate Averages
4. Find Top Performers
5. Generate Report
6. Exit
Enter your choice:


GradeManagementSystem.java
Run
GradeManagementSystem
6. Exit
Enter your choice: 5
=== PERFORMANCE REPORT ===
Total Students: 4
SUBJECT AVERAGES:
• Mathematics: 83.50
• Science: 87.00
• English: 83.00
• History: 75.00
• Computer: 78.00
GRADE DISTRIBUTION:
• A+ Grade: 0
• A Grade: 3
• B Grade: 1
• C Grade: 0
• D Grade: 0
• F Grade: 0



Technical Details

Data Structures Used:

- **Arrays**:
  - String[] studentNames → stores student names
  - double[][] studentMarks → stores marks for each student and subject
- **Constants:**
  - MAX_STUDENTS = 100 → maximum student capacity
  - SUBJECT_COUNT = 5 → fixed number of subjects

Algorithms Implemented:

- **Average Calculation:**
- sum = marks[subject1] + ... + marks[subjectN]
- average = sum / SUBJECT_COUNT
- **Grade Categorization:**
- A+ : 90-100
- A : 80-89
- B : 70-79
- C : 60-69

- D : 50-59
- F : <50
- **Top Performer Identification:**
    - Calculate average for all students
    - Compare to find the highest average
- **Performance Report:**
    - Subject averages → sum marks for each subject / total students
    - Grade distribution → count number of students in each grade category

Testing Evidence

Test Case Examples:

Student Name	Math	Science	English	History	Computer	Expected Average	Expected Grade
Sathish	89	79	80	90	78	83.20	A
Sanjai	67	89	97	78	66	79.40	B
Divya	89	90	78	65	88	82.00	A
Trisha	89	90	77	67	80	80.60	A

Validation Cases:

- **Invalid Marks**: Input 105 → Prompt: *"Marks must be 0-100. Re-enter:"*
- **Invalid Menu Choice**: Input 9 → Prompt: *"Enter number between 1 and 6:"*
- **Maximum Students Reached**: Adding 101st student → Message: *"Maximum student limit reached!"*

Student Name	Math	Science	English	History	Computer	Expected Average	Expected Grade
Sathish	89	79	80	90	78	83.20	A
Sanjai	67	89	97	78	66	79.40	B
Divya	89	90	78	65	88	82.00	A
Trisha	89	90	77	67	80	80.60	A