Project Report

**Project Title:** Student Grading System  
**Technology Used:** Java, Console-Based  
**Prepared By:** Sarthak Yadav

1. **Introduction**

The Student Grading System provides a console-based user interface to manage students, subjects, record grades, and calculate performance statistics. It demonstrates Object-Oriented Programming (OOP) principles and robust exception handling in Java. Data is stored in memory or optionally persisted to simple files, and the system is modular for scalability.

**2. Objectives**

* Apply OOP concepts such as classes, inheritance, abstraction, and encapsulation to model students, subjects, and grades.
* Implement exception handling to manage invalid grades, duplicate entries, and incorrect input formats.
* Create a menu-driven console UI for user operations.
* Enable calculation of grade point averages (GPA), report generation, and summary statistics.

**3. System Design**

**Modules**

* Student Class: Stores details including student ID, name, and a list of grades.
* Subject Class: Holds subject code, name, and credit information.
* Grade Class: Associates a student with a subject and their respective mark.
* Grade Manager Class: Manages all operations (add student, record grade, calculate GPA).
* Main UI Class: Menu interface for all user interactions.

**Data Flow Diagram (simplified)**

User Interface  
→ Grade Manager  
→ Student, Subject, Grade Entities  
→ In-Memory Storage OR File Storage (students.txt, grades.txt, subjects.txt).

**4. Features**

* Student Management: Add, list, update, and remove students.
* Subject Management: Define and list available subjects.
* Grade Recording: Input, validate, and store grades for students per subject.
* GPA Calculation: Calculate and display GPA for individual students.
* Exception Handling: Prevent and handle duplicate student IDs, invalid marks, and missing entities.
* Menu-Driven Console Input: User-friendly command-line operation.

**5. Example Data Files**

**students.txt**

S001,John Doe  
S002,Jane Smith  
S003,Sam Patel

**subjects.txt**

SUB01,Mathematics,3  
SUB02,Physics,4

**grades.txt**

S001,SUB01,89  
S001,SUB02,76  
S002,SUB01,93  
S003,SUB02,67

**6. Technologies Used**

* Programming Language: Java
* Exception Handling: Custom and standard Java exceptions
* Data Structures: Map, List, Set
* Storage: In-memory; optionally file-based persistence

**7. Output (Console)**

Main Menu

1. Add Student
2. Add Subject
3. Record Grade
4. Display Student GPA
5. List All Records
6. Exit

Example GPA Calculation:  
Enter student ID: S001  
Calculated GPA: 3.45

**8. Advantages**

* Portable and lightweight design works with basic Java.
* Easily extendable to support new features.
* Robust error-resistance using exception handling.
* Clear separation of concerns and modular design.

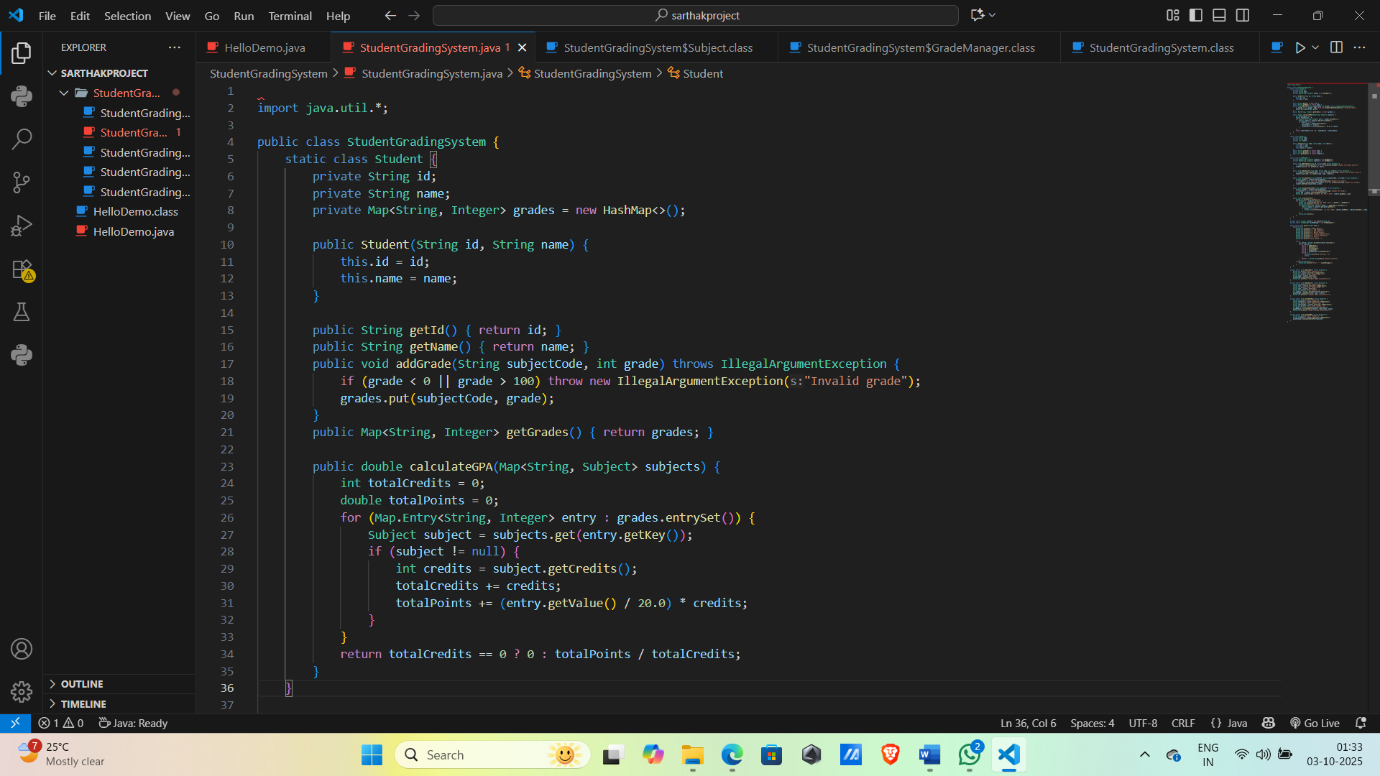
**9. Future Enhancements**

* Add persistent storage using CSV or a database system.
* Implement GUI interface for enhanced user experience.
* Enable import/export of data for batch operations.
* Role-based authentication (admin/teacher/student).
* Advanced analytics (subject averages, grade distribution, etc.).

**10. Conclusion**

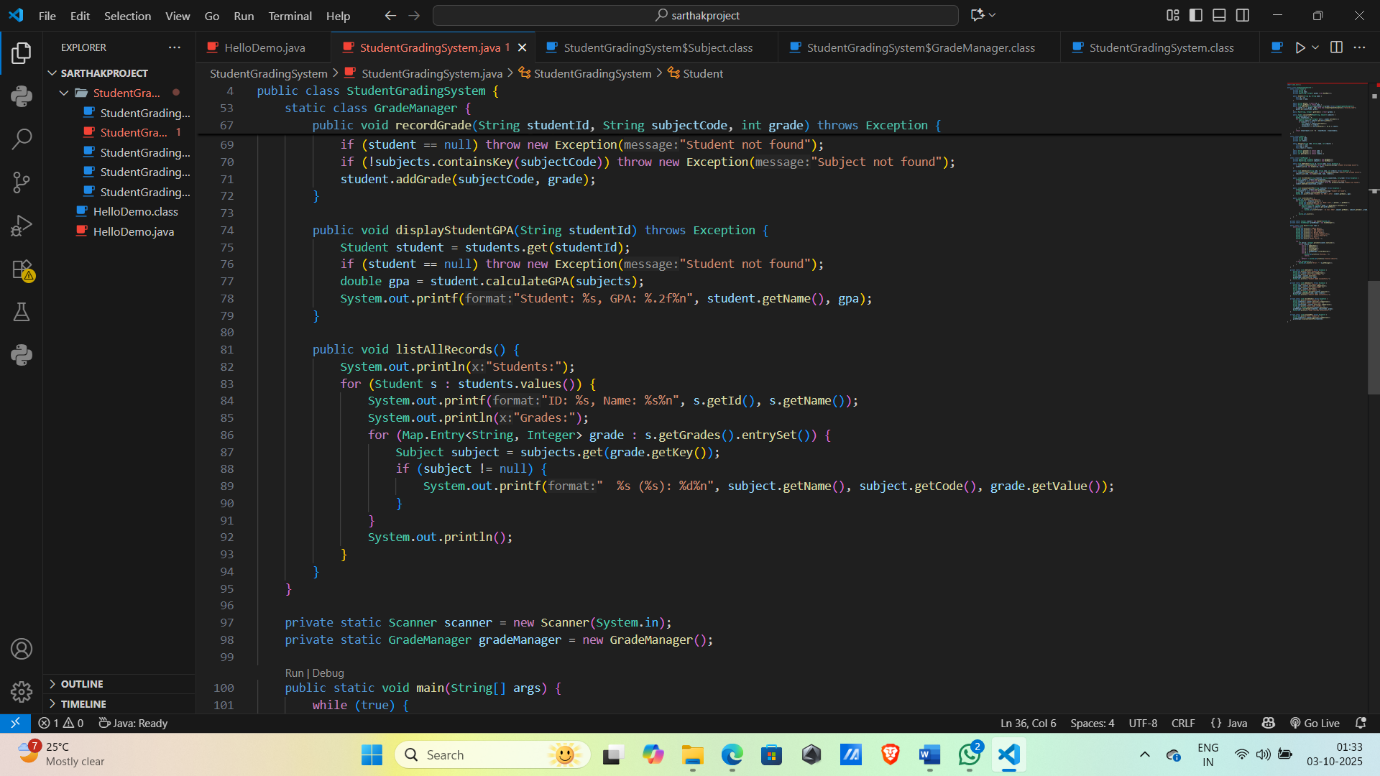
The Student Grading System project effectively demonstrates Java OOP principles and exception handling in a practical, console-based application. The system is suitable for academic, training, and prototype envirod analytics, and a graphical user interface.

# Complete Code:



A screen shot of a computer

AI-generated content may be incorrect.



A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

Code:

import java.util.\*;

public class StudentGradingSystem {

    static class Student {

        private String id;

        private String name;

        private Map<String, Integer> grades = new HashMap<>();

        public Student(String id, String name) {

            this.id = id;

            this.name = name;

        }

        public String getId() { return id; }

        public String getName() { return name; }

        public void addGrade(String subjectCode, int grade) throws IllegalArgumentException {

            if (grade < 0 || grade > 100) throw new IllegalArgumentException("Invalid grade");

            grades.put(subjectCode, grade);

        }

        public Map<String, Integer> getGrades() { return grades; }

        public double calculateGPA(Map<String, Subject> subjects) {

            int totalCredits = 0;

            double totalPoints = 0;

            for (Map.Entry<String, Integer> entry : grades.entrySet()) {

                Subject subject = subjects.get(entry.getKey());

                if (subject != null) {

                    int credits = subject.getCredits();

                    totalCredits += credits;

                    totalPoints += (entry.getValue() / 20.0) \* credits;

                }

            }

            return totalCredits == 0 ? 0 : totalPoints / totalCredits;

        }

    }

    static class Subject {

        private String code;

        private String name;

        private int credits;

        public Subject(String code, String name, int credits) {

            this.code = code;

            this.name = name;

            this.credits = credits;

        }

        public String getCode() { return code; }

        public String getName() { return name; }

        public int getCredits() { return credits; }

    }

    static class GradeManager {

        private Map<String, Student> students = new HashMap<>();

        private Map<String, Subject> subjects = new HashMap<>();

        public void addStudent(String id, String name) throws Exception {

            if (students.containsKey(id)) throw new Exception("Student ID already exists");

            students.put(id, new Student(id, name));

        }

        public void addSubject(String code, String name, int credits) throws Exception {

            if (subjects.containsKey(code)) throw new Exception("Subject code already exists");

            subjects.put(code, new Subject(code, name, credits));

        }

        public void recordGrade(String studentId, String subjectCode, int grade) throws Exception {

            Student student = students.get(studentId);

            if (student == null) throw new Exception("Student not found");

            if (!subjects.containsKey(subjectCode)) throw new Exception("Subject not found");

            student.addGrade(subjectCode, grade);

        }

        public void displayStudentGPA(String studentId) throws Exception {

            Student student = students.get(studentId);

            if (student == null) throw new Exception("Student not found");

            double gpa = student.calculateGPA(subjects);

            System.out.printf("Student: %s, GPA: %.2f%n", student.getName(), gpa);

        }

        public void listAllRecords() {

            System.out.println("Students:");

            for (Student s : students.values()) {

                System.out.printf("ID: %s, Name: %s%n", s.getId(), s.getName());

                System.out.println("Grades:");

                for (Map.Entry<String, Integer> grade : s.getGrades().entrySet()) {

                    Subject subject = subjects.get(grade.getKey());

                    if (subject != null) {

                        System.out.printf("  %s (%s): %d%n", subject.getName(), subject.getCode(), grade.getValue());

                    }

                }

                System.out.println();

            }

        }

    }

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

    private static GradeManager gradeManager = new GradeManager();

    public static void main(String[] args) {

        while (true) {

            System.out.println("\nMain Menu");

            System.out.println("1. Add Student");

            System.out.println("2. Add Subject");

            System.out.println("3. Record Grade");

            System.out.println("4. Display Student GPA");

            System.out.println("5. List All Records");

            System.out.println("6. Exit");

            System.out.print("Enter choice: ");

            try {

                int choice = Integer.parseInt(scanner.nextLine());

                switch (choice) {

                    case 1 -> addStudent();

                    case 2 -> addSubject();

                    case 3 -> recordGrade();

                    case 4 -> displayGPA();

                    case 5 -> gradeManager.listAllRecords();

                    case 6 -> {

                        System.out.println("Exiting...");

                        return;

                    }

                    default -> System.out.println("Invalid choice");

                }

            } catch (Exception e) {

                System.out.println("Error: " + e.getMessage());

            }

        }

    }

    private static void addStudent() throws Exception {

        System.out.print("Enter Student ID: ");

        String id = scanner.nextLine().toUpperCase();

        System.out.print("Enter Student Name: ");

        String name = scanner.nextLine();

        gradeManager.addStudent(id, name);

        System.out.println("Student added successfully.");

    }

    private static void addSubject() throws Exception {

        System.out.print("Enter Subject Code: ");

        String code = scanner.nextLine().toUpperCase();

        System.out.print("Enter Subject Name: ");

        String name = scanner.nextLine();

        System.out.print("Enter Credits: ");

        int credits = Integer.parseInt(scanner.nextLine());

        gradeManager.addSubject(code, name, credits);

        System.out.println("Subject added successfully.");

    }

    private static void recordGrade() throws Exception {

        System.out.print("Enter Student ID: ");

        String studentId = scanner.nextLine().toUpperCase();

        System.out.print("Enter Subject Code: ");

        String subjectCode = scanner.nextLine().toUpperCase();

        System.out.print("Enter Grade (0-100): ");

        int grade = Integer.parseInt(scanner.nextLine());

        gradeManager.recordGrade(studentId, subjectCode, grade);

        System.out.println("Grade recorded successfully.");

    }

    private static void displayGPA() throws Exception {

        System.out.print("Enter Student ID: ");

        String studentId = scanner.nextLine().toUpperCase();

        gradeManager.displayStudentGPA(studentId);

    }

}