

STUDENT MANAGEMENT SYSTEM USING JAVA

Table of Contents

1. Problem Statement
 2. Introduction
 3. Objectives
 4. Features
 5. System Design
 6. Class Diagrams
 7. Implementation Details
 8. Limitations
 9. Future Enhancements
 10. Conclusion
-

1. Problem Statement

Managing academic records manually in schools and colleges leads to inefficiencies, delayed evaluations, and errors in performance tracking. Without automation, tracking progress and generating reports becomes tedious and time-consuming.

Key Issues Identified:

- Difficult to maintain updated student profiles.
- Grade and percentage calculation errors due to manual work.
- No integrated system to display or analyze performance.
- Limited access to data in real time.

Proposed Solution:

A **Java-based CLI Student Management System (SMS)** that:

- Stores student profiles with academic information.
 - Manages subject-wise marks and calculates performance.
 - Automatically assigns grades based on defined criteria.
 - Uses a menu-driven console interface for ease of access.
-

2. Introduction

The project is a **console-based application** built using Java and OOP concepts. It provides a structured and user-friendly way to manage student records and examinations. The system uses in-memory data storage, making it lightweight and fast for small-scale academic environments.

Highlights:

- Built using **Object-Oriented Programming (OOP)**
 - Works without external storage (file/database)
 - Focuses on automation and simplicity
 - Encourages logical structuring and modular coding
-

3. Objectives

- Record and maintain student data like name, roll number, email, course, etc.
- Allow users to enter subject-wise marks for each student.
- Automate the process of calculating grades and performance percentage.
- Display a complete academic report for students.
- Eliminate human error and reduce time in report generation.

4. Features

Feature	Description
Student Registration	Capture and store student personal and academic details
Exam Entry	Record marks for 5 subjects and store them
Result Generation	Display marks, calculate percentage, and assign grades
Grade System	Uses predefined rules to convert marks to grades (A+ to F)
In-Memory Storage	Uses arrays for temporary data handling without persistence
CLI-based Interface	Interactive menu for operations without GUI complexity

5. System Design

5.1 Class Breakdown

1. Student

- Stores student information like name, DOB, email, ID, etc.
- Method: displayDetails()

2. Subject

- Holds subject name, marks, grade, and percentage.
- Method: calculateGradeAndPercentage()

3. Examination

- Maintains multiple subjects and computes overall grade/percentage.
- Methods: addSubject(), calculateOverallPerformance()

4. StudentManagementSystem

- Main class containing menu and logic for interaction.

- Handles student and exam coordination.

5.2 Flow of Control

pgsql

Copy Edit

Start → Show Menu → User selects an option
→ Perform action (Add/View) → Return to Menu → Repeat until Exit

6. Class Diagrams

Student Class

java

Copy Edit

```
class Student {  
    - name: String  
    - rollNumber: int  
    - dob: String  
    - email: String  
    - studentId: String  
    - course: String  
    + displayDetails(): void  
}
```

Examination Class

java

Copy Edit

```
class Examination {  
    - subjects: Subject[]  
    - subjectCount: int  
    - overallPercentage: double  
    - overallGrade: String  
    + addSubject(): void  
    + calculateOverallPerformance(): void  
    + displayExamResults(): void  
}
```

Class Relationships

- StudentManagementSystem uses both Student and Examination
 - Examination internally contains multiple Subject instances
-

7. Implementation Details

7.1 Data Handling

- All student and exam data is stored using arrays (Student[100], Subject[5]).
- Data exists only during runtime (no saving after exit).

7.2 Key Functions

- findStudentByRollNumber() → Returns the index of a student based on input
- calculateGradeAndPercentage() → Assigns grade based on marks
- displayExamResults() → Outputs marks, grades, and overall result







7.3 User Interaction

- Uses Scanner for input from console
 - All menus are text-based for lightweight navigation
 - Provides user-friendly prompts and validations
-

8. Limitations

- ❌ No data saving: Records are lost when program is closed
 - ❌ Fixed capacity: Only 100 students and 5 subjects supported
 - ❌ No GUI: Completely text-based, less intuitive for some users
 - ❌ Basic Validation: Limited checks on input duplication or format
-

9. Future Enhancements

-  Integrate with file handling or a relational database for persistence
 -  Add a graphical interface using **Swing** or **JavaFX**
 -  Include support for attendance and fee tracking
 -  Role-based access (e.g., Admin, Teacher, Student)
 -  Add input validation for cleaner and error-free data
 -  Export results as PDF or CSV
-

10. Conclusion

This project demonstrates a practical application of **Java and Object-Oriented Programming** to solve a real-world problem in the education sector. It streamlines student performance management through an efficient command-line interface.

Why this project matters:

- Reinforces Java concepts in a structured format
- Useful as a portfolio or mini project
- Lays the groundwork for larger educational software

 **End of Documentation**