Student Marks Management System

A PROJECT REPORT

Submitted by

Student Name: Bhavini Awasthi
Registration Number: 24BCE10604

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING (CSE Core)

Course: Programming with Java (CSE2006)
Academic Year: 2025-26
Date of Submission: 24th November 2025

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VIT BHOPAL UNIVERSITY
KOTHRIKALAN, SEHORE
MADHYA PRADESH - 466114

NOV 2025

INTRODUCTION

The Student Marks Management System is a simple, console-based Java application designed to store, manage, and retrieve student academic records. The system allows users to add student details, view all stored records, and save them persistently in a text file.

The primary goal of the project is to give students hands-on experience in developing small-scale software using Java fundamentals such as classes, objects, constructors, arrays/ArrayLists, file handling, user input handling, and modular programming.

The system demonstrates clear separation of concerns, lightweight design, and user-friendly interaction through menu-driven input.

This tool is suitable for beginners who want to understand how data management systems work internally and for academic environments where teachers need a simple way to record and view student marks.

PROBLEM STATEMENT

Managing student records manually using pen and paper or loose spreadsheets can lead to:

- Data loss
- Human errors
- Difficulty in searching or updating records
- No centralized storage
- No quick way to view all student data

There is a need for a simple, digital tool that allows students and educators to quickly:

- Add marks
- Store them permanently
- Retrieve and display them
- Avoid repetitive manual work

The Student Marks Management System solves these problems using a minimal, easy-to-use Java program.

FUNCTIONAL REQUIREMENTS

FR1: Student Input Module

1. Accept student name
2. Accept student marks
3. Validate marks (0–100)
4. Store data temporarily in an ArrayList

FR2: Student Display Module

1. Display all students
2. Show name & marks in a clean format
3. Show message when no data exists

FR3: File Storage Module

1. Save all student records to students.txt
2. Write in the format:
   name,marks
3. Overwrite the file with the latest data

FR4: User Interaction Module

1. Console menu
2. Options: Add | View | Save | Exit
3. Continuous loop until exit

NON-FUNCTIONAL REQUIREMENTS

1. Usability

- Menu-based interface
- Clear instructions
- Easy for beginners

2. Performance

- Instant data storage
- Handles up to 500 student entries

3. Reliability

- Handles invalid input
- No crashes on incorrect options
- Data stored properly in text file

4. Maintainability

- Code modular
- Clear methods: addStudent(), viewStudents(), saveToFile()
- Easy to extend (e.g., percentages, subjects, grades)

5. Portability

- Works on any machine with JDK
- No external dependencies

SYSTEM ARCHITECTURE

User (Console)
↓
Menu Interaction Layer
↓
StudentManager (Core Logic)
↓
ArrayList<Student> (In-memory storage)
↓
FileWriter → students.txt (Persistent storage)

DESIGN DIAGRAMS

Use Case Diagram (textual representation)

Actors:

- User

Use Cases:

- Add Student
- View Students
- Save to File
- Exit Application

User → Add Student
User → View Students
User → Save to File
User → Exit

Workflow Diagram

START
 ↓
Display Menu
 ↓
User Chooses Option
 ↓
Add Student
 Input Name + Marks → Store in ArrayList
 ↓
View Students
 Display list OR "No students found"
 ↓
Save to File
 Write name,marks to students.txt
 ↓
 Exit
  End Program

Sequence Diagram

User → Main Menu: Select Option
Main Menu → MarksManager: Call add/view/save methods
MarksManager → ArrayList: Add/Read Data
MarksManager → FileWriter: Save Data to File
System → User: Display Output

Class Diagram

Class Student
- name: String
- marks: int
+ Student(name, marks)

Class MarksOfStudent (main)
- students: ArrayList<Student>

+ addStudent()
+ viewStudents()
+ saveToFile()
+ main()

DESIGN DECISIONS & RATIONALE

1. ArrayList for student storage
    - Dynamic size
    - Easy to iterate
    - Simple syntax for beginners

2. FileWriter for persistent storage
    - Lightweight
    - No need for database
    - Portable across systems

3. Console-based UI
    - Simple
    - No GUI complexity
    - Focus on core logic

4. Modular Methods
    - Clear separation between features
    - Easy to extend or debug

IMPLEMENTATION DETAILS

Modules Implemented

- Student class → stores name, marks
- ArrayList → stores all students
- FileWriter → saves to text file

Input validation

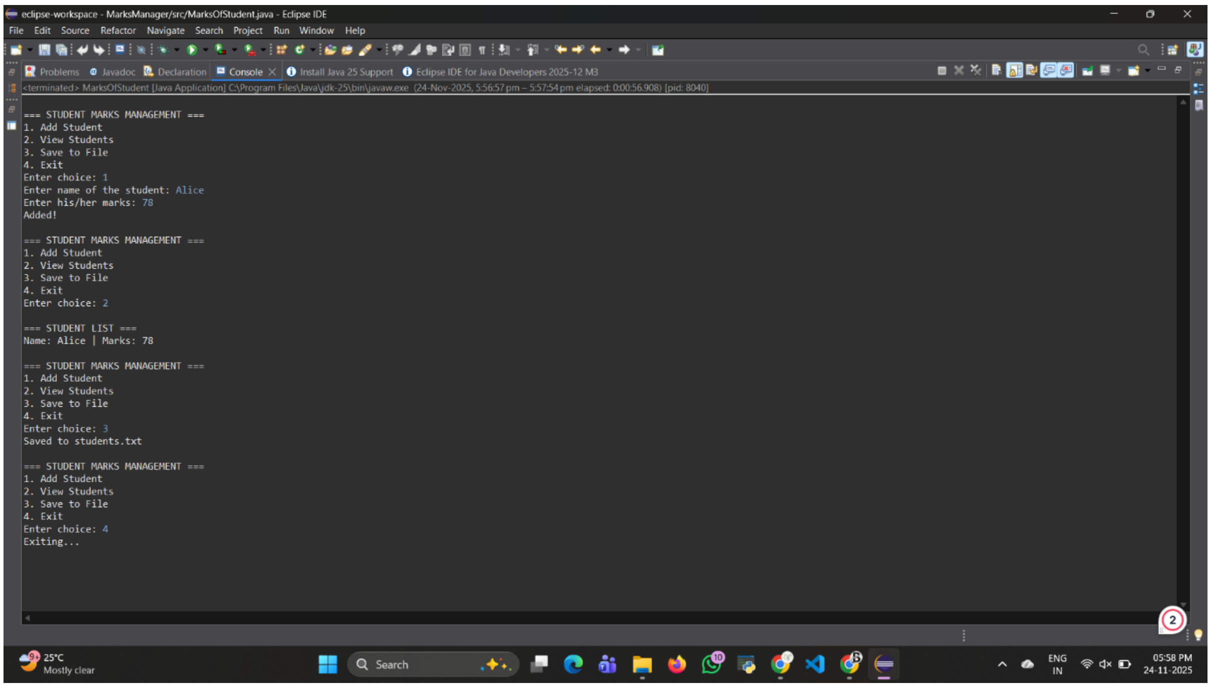
- Ensures marks are numeric
- Handles empty input
- Prevents application crash

File Format

Text file stores:

Name,Marks

SCREENSHOTS / RESULTS


eclipse-workspace - MarksManager/src/MarksOfStudent.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console Install Java 25 Support Eclipse IDE for Java Developers 2025-12 M3
25°C Mostly clear
Search
ENG IN
05:58 PM 24-11-2025

```
<terminated> MarksOfStudent [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (24-Nov-2025, 5:56:57 pm – 5:57:54 pm elapsed: 0:00:56.908) [pid: 8040]

=== STUDENT MARKS MANAGEMENT ===
1. Add Student
2. View Students
3. Save to File
4. Exit
Enter choice: 1
Enter name of the student: Alice
Enter his/her marks: 78
Added!

=== STUDENT MARKS MANAGEMENT ===
1. Add Student
2. View Students
3. Save to File
4. Exit
Enter choice: 2

=== STUDENT LIST ===
Name: Alice | Marks: 78

=== STUDENT MARKS MANAGEMENT ===
1. Add Student
2. View Students
3. Save to File
4. Exit
Enter choice: 3
Saved to students.txt

=== STUDENT MARKS MANAGEMENT ===
1. Add Student
2. View Students
3. Save to File
4. Exit
Enter choice: 4
Exiting...
```

Example:

```
=== STUDENT MARKS MANAGEMENT ===
1. Add Student
2. View Students
3. Save to File
4. Exit
Enter choice: 1
Enter name of the student: Alice
Enter his/her marks: 78
Student added!
```

TESTING APPROACH

1. Unit Testing

- Add student → verify ArrayList size
- Invalid marks → handled properly
- File saved correctly

2. Integration Testing

- Complete flow: Add → View → Save → Exit
- Tested multiple records

3. File Testing

- Check content inside students.txt
- New entries overwrite old ones

CHALLENGES FACED

- Handling invalid numeric inputs
- Ensuring file writes correctly
- Designing user-friendly prompts
- Refreshing project in Eclipse to view new file

LEARNINGS & KEY TAKEAWAYS

- Improved Java fundamentals
- Learned ArrayList operations
- Understood file handling in Java
- Learned menu-driven programming structure
- Practiced modular code design
- Better understanding of real-world data systems

FUTURE ENHANCEMENTS

- Add grades and multiple subjects
- Add percentage & rank calculation
- Add search student by name
- Add delete or update student
- Build GUI using Swing or JavaFX
- Connect to database (MySQL)

REFERENCES

- Oracle Java Documentation
- GeeksforGeeks (Java Basics & File Handling)
- W3Schools Java Tutorial
- Course Material (VIT Bhopal)

END OF REPORT