Requirements Document

Project Title: Student Performance Management System (SPMS)
Course: CS118 – Programming Fundamentals (Fall 2025)
Team Formation: Pairs (2 students per team)
Weightage: 10% of total course grade

---

1. Project Overview

The Student Performance Management System (SPMS) is a console-based C++ application that allows storing, managing, and analyzing students' academic performance data (marks of quizzes, assignments, and exams).
The goal is to let students apply the concepts learned in Programming Fundamentals to build a complete, functioning application.

By completing this project, students will:

- Practice problem-solving and translating requirements into code.
- Work collaboratively in pairs, simulating teamwork.
- Handle input validation and edge cases.
- Learn to structure programs with arrays, functions, and file handling.

---

2. Functional Requirements

The system should provide the following features:

Core Features (Must-Have)

1. **Student Record Management**
    - Add a new student (Name, Roll No, Section).

    - Store marks for at least 3 quizzes, 2 assignments, Midterm, and Final exam.
    - Update or delete an existing student record.
2. **Data Storage & Retrieval**
    - Store records in **arrays** during runtime.
    - Save records to a **text file** for persistence.
    - Load records from file at program start.
3. **Performance Calculations**
    - Calculate total marks and percentage.
    - Assign a grade (A, B, C, D, F) based on percentage.
    - Identify topper(s) of the class.
4. **Reports**
    - Display all student records in a tabular format.
    - Search student by Roll No.
    - Show class average, highest, and lowest scores.

Additional Features (Good-to-Have)

- Sort students by percentage (descending order).
- Generate subject-wise statistics (e.g., average quiz score).
- Export report to a text file.

---

3. Non-Functional Requirements

- **Language:** C++ (must be compiled and run in Dev-C++/CodeBlocks or similar).
- **Program Design:** Code must use functions to avoid repetition.
- **Data Structures:** Use arrays (1D and 2D as required).
- **File Handling:** For saving/loading records.
- **Error Handling:** Input validation (e.g., no negative marks, valid roll numbers).
- **Code Style:** Indentation and comments required.

---

4. Constraints

- No use of advanced libraries beyond standard C++ (iostream, fstream, cctype, cstring, etc.).
- No use of STL containers (like vector, map) since scope is Programming Fundamentals.
- Console-based only (no GUI).

---

5. Deliverables

Each team must submit:

1. **Project Report** (PDF) including:
    - Cover page (Project title, team members, roll numbers).
    - Problem description.
    - Features implemented.
    - Sample input/output screenshots.
    - Known limitations (if any).

2. **Source Code Files** (.cpp files + input data file).
3. **Final Demonstration** (Week 15).

---

6. Evaluation Rubric (10% Weightage)

Criteria	Marks
Correctness of Features	3
Use of Arrays & Functions	2
File Handling Implementation	2
Code Quality (indentation, comments, naming)	1
Report & Presentation	2
