**Software Requirements Specification (SRS) for Student Grade Management System**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the requirements for a **Student Grade Management System**. This console-based application will help students practice programming fundamentals like variables, if conditions, loops, functions, structs, and pointers while managing a list of students and their grades.

**1.2 Scope**

The application will allow users to add and update student records, calculate grades based on scores, and display summaries like the highest score, average score, and students who passed or failed. It will reinforce programming concepts by implementing logic using control structures, functions, structs, and enums.

**1.3 Definitions, Acronyms, and Abbreviations**

* **Student Record**: A data structure containing student information such as ID, name, subjects and scores for each subject.
* **Grade**: A letter-based representation of a student’s performance (e.g., A, B, C).
* **Pass/Fail**: Determination based on a minimum grade threshold.

**2. Overall Description**

**2.1 Product Perspective**

The **Student Grade Management System** is a standalone console application written in C. It will simulate basic record-keeping and grading operations, demonstrating practical use of programming concepts.

**2.2 Product Functions**

* **Add Student**: Add new student records with ID, name, subjects and scores.
* **Update Student**: Update the scores of an existing student.
* **Calculate Grades**: Automatically calculate and assign letter grades based on scores.
* **Display Results**: Display summaries like average score, highest score, and pass/fail lists.

**2.3 User Classes and Characteristics**

* **Student Users**: Users are students practicing programming fundamentals. They should understand variables, if conditions, loops, functions, structs, and pointers.

**3. System Features**

**3.1 Add Student**

**3.1.1 Description and Priority**

Users can add a new student record with their ID, name, and scores. This is a core feature with high priority.

**3.1.2 Functional Requirements**

* FR1: The system shall allow users to input a student ID, name, and scores for their subjects (user can input any number of subjects and scores).
* FR2: The system shall store the student record in memory using a struct.

**3.2 Update Student**

**3.2.1 Description and Priority**

Users can update the scores of an existing student. This feature has medium priority.

**3.2.2 Functional Requirements**

* FR3: The system shall allow users to search for a student by ID.
* FR4: The system shall allow users to update the scores of the selected student.

**3.3 Calculate Grades**

**3.3.1 Description and Priority**

The system will calculate and assign a letter grade to each student based on their average score. This feature is a high priority.

**3.3.2 Functional Requirements**

* FR5: The system shall calculate the average score for each student.
* FR6: The system shall assign a grade using the following criteria:
  + A: Average >= 90
  + B: 80 <= Average < 90
  + C: 70 <= Average < 80
  + D: 60 <= Average < 70
  + F: Average < 60

**3.4 Display Results**

**3.4.1 Description and Priority**

Users can view summaries of student records, such as pass/fail status, average score, and top performer. This feature is a medium priority.

**3.4.2 Functional Requirements**

* FR8: The system shall display all students’ records, including ID, name, average score, and grade.
* FR9: The system shall display the student with the highest average score.
* FR10: The system shall display the average score of all students.
* FR11: The system shall display a list of students who passed (grade >= D).
* FR12: The system shall display a list of students who failed (grade = F).

Notes

* Using coloured menu is a Bonus
* Saving data to a file and reading it again once the app is opened is a Bonus