📄 Project Requirement Document

Project Title: Real-Time Student Marksheet Generator in C

□ Overview

This project involves building a real-time, console-based Student Marksheet Generator using the C programming language. Students will design a modular program that accepts student details and subject-wise marks, calculates total marks, percentage, and grade, and stores the formatted marksheet in a file. The project emphasizes real-time data entry, file handling, and structured programming.

🎯 Objective

To develop a real-time, file-based marksheet generator that:

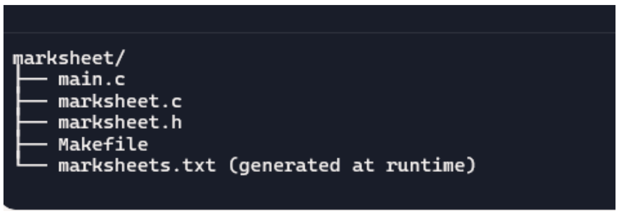
- Accepts student information and subject marks
- Calculates total, percentage, and grade
- Displays and stores the marksheet in a structured format
- Demonstrates modular programming and file I/O in C
- Encourages real-time interaction and formatted output similar to professional systems

📌 Functional Requirements

Feature	**Description**
Student Input	Accept student name, roll number, class, and number of subjects
Subject-wise Marks Entry	Accept subject names and marks for each subject
Grade Calculation	Compute total marks, percentage, and assign grade based on defined criteria
File Storage	Save each marksheet to a file named marksheets.txt
Formatted Output	Display a clean, readable marksheet on the console
Multi-Student Support	Allow multiple student entries in a single session
Modular Design	Use header files and separate source files for logic and main program
Real-Time Feedback	Immediately display calculated results and confirmation of file save

□ Technical Specifications

- Language: C (GCC Compiler)
- File Structure:



```
marksheet/
├── main.c
├── marksheet.c
├── marksheet.h
├── Makefile
└── marksheets.txt (generated at runtime)
```

- Compilation: Use the provided Makefile
- Grade Criteria:

Percentage Range	**Grade**
90–100%	A+
80–89%	A
70–79%	B
60–69%	C
50–59%	D
Below 50%	F

 Implementation Guidelines

1. Define a Student structure to hold:
    - Name, roll number, class
    - Subject names and marks
    - Number of subjects
2. Create a function get_grade(float percentage) that returns a string grade.
3. Create a function generate_marksheet(struct Student s) that:
    - Calculates total and percentage
    - Calls get_grade()
    - Displays and writes the marksheet to a file
4. In main.c, use a loop to:
    - Accept input for multiple students
    - Call generate_marksheet() for each
5. Use fgets() for string input and handle newline characters with strcspn().
6. Use fopen(), fprintf(), and fclose() for file operations.
7. Organize your code into:

- main.c: handles input and program flow

- marksheet.c: contains logic for grade and marksheet generation
- marksheet.h: contains structure and function declarations

🛠 Build Instructions

1. Open terminal in the project directory
2. Run: make ./MarksheetApp
3. To clean build files: make clean

🖥 Sample Console Output

```
===== STUDENT MARKSHEET GENERATOR =====

Enter student name: Sanjeet Prasad

Enter roll number: 105

Enter class: 11B

Enter number of subjects: 3

Enter subject 1 name: English

Enter marks in English: 88

Enter subject 2 name: Physics

Enter marks in Physics: 91

Enter subject 3 name: Chemistry

Enter marks in Chemistry: 84

----------- MARKSHEET -----------

Name      : Sanjeet Prasad

Roll No.  : 105

Class     : 11B

Subject       Marks

---------------------

English       88

Physics       91

Chemistry     84

Total Marks : 263

Percentage  : 87.67%

Grade       : A

--------------------------------

Marksheet saved to 'marksheets.txt'

Do you want to enter another student? (y/n): n
```

📄 Sample File Output (marksheets.txt)

```
----------- MARKSHEET -----------

Name      : Sanjeet Prasad
```

```
Roll No.   : 105
Class      : 11B
Subject       Marks
----------------------
English       88
Physics       91
Chemistry      84
Total Marks : 263
Percentage  : 87.67%
Grade      : A
--------------------------------
```

📤 Deliverables

Students must submit the following:

- main.c, marksheet.c, marksheet.h
- Makefile
- A sample output file marksheets.txt with at least 3 student entries
- A short README file explaining how to compile and run the project

✅ Evaluation Criteria

Criteria	**Weight**
Correctness of Output	30%
Code Modularity & Structure	20%
File Handling Implementation	15%
Grade Calculation Logic	10%
Use of Header Files & Makefile	10%
Code Readability & Comments	10%
Bonus: Input Validation	5%

💡 Future Enhancements (Optional for Extra Credit)

Enhancement Idea	**Description**
📊 Graphical Grade Distribution	Display bar charts using ASCII for grade visualization
📁 Export to CSV	Allow exporting marksheets in CSV format for Excel compatibility
🔍 Search Functionality	Search marksheets by roll number or name

🏆 Class Average & Topper Highlight	Calculate and display class average and top scorer
🖨 Print-Ready Formatting	Format output for printing physical report cards

🎓 Learning Outcomes

This project reinforces the following C programming concepts:

Concept	**Application in Project**
Structures (struct)	To store student and subject data
File Handling	To persistently store marksheets using fopen, fprintf, fclose
Arrays	To manage subject names and marks
Loops & Conditionals	For input, processing, and grade logic
Functions	For modularizing logic and improving code reusability
Header Files	For separating declarations and promoting modular design
Makefile	For automating compilation and enforcing clean builds

👨‍🏫 Instructor Notes

This project is ideal for reinforcing:

- Structs and arrays
- File I/O operations
- Modular programming with header/source separation
- Real-time interaction and formatted output
- Scalable design thinking for future enhancements

👤 Author & Contact Information

- 👨‍💻 Author: Sanjeet Prasad
- 📧 Email: sanjeet8.23@gmail.com
- 📱 Mobile: +91 9958217807