TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Proposal

on

C-Programming for Student Grade Management System

Submitted by:

Krishna Kandel	(THA081BEI014)
Nishanta Poudel	(THA081BEI025)
Pranish Pokhrel	(THA081BEI029)
Prateek Chaulagain	(THA081BEI030)

Submitted to:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

February, 2025

ABSTRACT

This project is about building an efficient Grade Management System using C-programming to accommodate educational sectors. The system operates by feeding scores as input which are then processed to calculate GPA and storing student records individually. Under the rays of arrays, structures and functions, it organizes student data plus grades. It handles the tasks of sorting students in ranking order as soon as data entry completes. This prototype lays the groundwork for future improvements in grade management systems.

Keywords: Student Grade Management System, GPA

Table of Content

LIST OF FIGURES

LIST OF ABBREVIATIONS

CSV	Comma-Separated Values
GPA	Grade Point Average
GCC	GNU Compiler Collection
I/O	Input/Output
IDE	Integrated Development Environment
OS	Operating System
PHP	Hypertext Preprocessor
SGMS	Student Grade Management System
SQL	Structured Query Language
UI	User Interface

1 INTRODUCTION

1.1 Background Introduction

The main institution from which everyone goes through is definitely academic institutions. The movement of pursuing advancement in education affiliation leads to educational institutions. The term educational institutions includes each and every kind of institution which provides education either formal or informal. The students being attended there have to appear in different kinds of examinations and assessments. The management of grade results seems too slow in nature and seems overwhelming when calculating each and every mark to corresponding GPA. This an attempt to create a Student Grade Management System using C-programming language. It will allow the user to input important student data regarding exams and store it in a systematic order.

In educational institutes, managing student data is crucial for ensuring seamless academic administration. The Student Grade Management System (SGMS) is designed to effectively store, manage, evaluate, and categorize students' results, along with tracking other important information. SGMS makes it easy to quickly find and update information, and it helps everyone stay on the same page. Whether it's tracking how a student is doing or pulling up a report, the system makes it all simpler and less stressful. It's like having a reliable partner that takes care of the admin work, so everyone—students, teachers, and staff—can do their best.

1.2 Motivation

The idea behind creating the SGMS is to promote paperless functioning in administration offices for calculating grade results more precisely, conveniently plus fastly. To create a system that helps to minimize the redundancy of same functioning for long iterations. The time consuming- act of grade calculating manually should be discarded and as a replacement SGMS should be the best choice for this.

1.3 Project Objectives

- To develop a system that efficiently records, processes, and manages student grades, reducing manual effort and errors.
- To provide accurate grade calculation.

2 LITERATURE REVIEW

This literature review explores existing studies, projects and technologies. Numerous studies highlight the importance of automated grading systems in enhancing academic administration. Most existing systems rely on web-based solutions using technologies like PHP, Python, and SQL databases. However, these systems may have performance overhead due to web server dependencies.

2.1 Student Management System by *ishan-cse*:

This system was innovated by Md. Harun Aur Rashid Khan Ishan, is a console-based application created using C programming language. This system was compiled in Code::Blocks IDE using GCC compiler. [1]

The system is designed as a basic structure for every choice the whole interface got changed. Firstly, the user is given choices for either teacher or student information. On the requirements, the interface proceeds. Either for the teacher or student's section. The interface asks if an individual is registered or new on the server. It provides the feature of updates like deleting individuals from the server database. The server stores the salary details about teachers and tuition fees for students based on a predefined database.

The system uses the verification for the existing students and teacher specimen by their assigned id and password at the time of registration process as a new user while accessing the database. This procedure of verification seems difficult as the administrators would need to remember id and password for every teacher and students. The logic behind every function seems scattered across different parts of the program. Instead of using a clean, modular way, it seems written separately instead of being generalized into reusable functions.

The system lacks vulnerability and portability which are the basic prerequisites for a system development. Using system-specific commands make the program dependent on a particular environment rather than being flexible across. The decision of using structures students and teachers under a single structure is lame as they design makes the system inflexible, wasting memory when it’s not fully used and causing limitations when more data needs to be stored.

The strong point of using user credentials for logging into the database seems weak due to unuse of strong security. Using a logout feature would have solved this issue which will be the backbone for the system. Using structured prompts for data input of teachers, students, and their attendances seems good to initiate. The process of data modification is kind of similar to what we are using for our system SGMS.

The system should feature the use of dynamic memory instead of static arrays, ensuring efficiency and scalability. Security should be taken seriously by implementing password encryption and decryption models instead of storing raw values. Lastly, replacing system-dependent functions with standard C functions would improve portability, ensuring the program can run across different operating systems without modification.

3 PROPOSED SYSTEM ARCHITECTURE

3.1 Overview of the architecture:

This project is mainly based on file handling in C. It takes the data of students, sorts the data alphabetically, calculates GPA and rank of the students. The information is then stored in a text file.

3.2 Modules

Modular approach has been adopted for this program. The program consists of various modules as discussed below:

3.2.1 Input Module

This module receives the necessary input required from the user. It takes the name of students and their marks in the individual subjects.It stores the input data in .txt file for further processing.

3.2.2 Sorting Module

This module uses a sorting algorithm to sort the names of all students and generate the roll number based on the ascending order of alphabets.

3.2.3 Processing Module

This module calculates the GPA of the students and filters the failed students from the list. Those who are unable to get at least 40 percent of full marks are identified as failed by this program. Amongst the passed students, the best scoring ones are now sorted in numerical order and the assigned numeral is saved as the rank of that student.

3.2.4 File handling Module

This module interacts with the text file. We will be using a user readable .txt format for this program. This module will be storing and retrieving the GPA and ranking of the students.

3.2.5 Output Module

This module will display the result of all the students in the alphabetically sorted manner. It will display the roll number, name, GPA and rank of the students.

3.3 Data flow

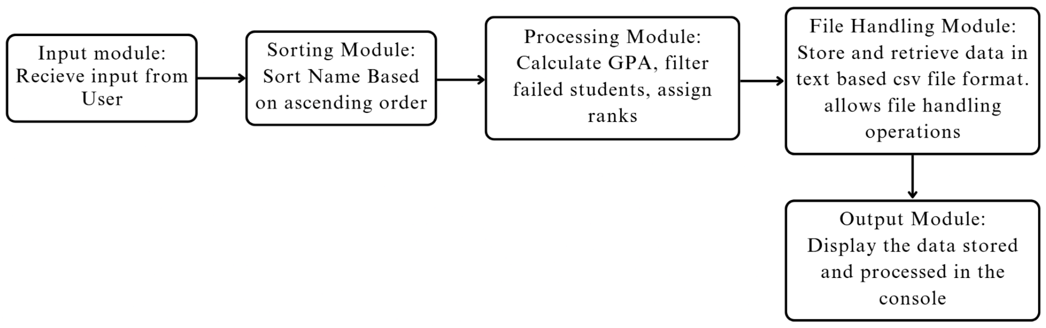
Input module: Recieve input from User
Sorting Module: Sort Name Based on ascending order
Processing Module: Calculate GPA, filter failed students, assign ranks
File Handling Module: Store and retrieve data in text based csv file format. allows file handling operations
Output Module: Display the data stored and processed in the console

Figure 1: flowchart diagram for modular data flow in the program

3.4 Tools and Environment

- Programming Language: C
- Compiler: GCC
- IDE: Code: Visual Studio Code
- Data Storage: .txt file
- Operating System: Windows
- Version Control and open source: Git for tracking changes and collaboration
- Libraries Used: Standard C libraries like stdio.h, stdlib.h, and string.h

4 METHODOLOGY

4.1 Problem Definition

At first we must have a clear idea of what we are trying to tackle by this project. We have declared the low efficiency and accuracy in record keeping the students' exam data in educational institutions as our problem statement. Our program aims to help the academic officials store the data digitally in an efficient way. This is our sole intention that we expect happens after the completion of this project.

4.2 System Design

To begin thinking about how we are going to achieve it, we brainstormed about the proper way to execute our plan. We came up with the system design that is explained as follows:

4.2.1 Input

In the very first step the program asks for the name of the student to the user. The user inputs the student's first and last name. Then the user is asked whether they have to add another student's data or not with [y/n] at the end of the question. Pressing 'y' leads to addition of another student's information while 'n' leads to the termination of the name input session. Then the program tells the user to input marks of the student with their roll number (generated from the sorting module discussed below). After the marks of all students are retrieved by the program, the input session ends.

4.2.2 Processing

Now, all the names, received from the input module, are stored in a .txt file. The file supplies these values to the next module which is the sorting module. This module is responsible for sorting the students name and assigning the roll number beginning from 1 to the last number of students. After the user inputs the marks of the students, the marks are stored in the file too. The students are now ranked by the ranking module based on the total marks obtained. The GPA of passed students is also calculated. Only the students with at least 40% marks in each subject are ranked. Others are labelled failed by the system. Then the processing session ends.

4.2.3 Output

Now all the necessary processing has been done by our program. All the retrieved data is processed and stored. This data is now ready to be displayed to the user. The path to the .txt file with all the processed data is shown at the end of the program with a success message.

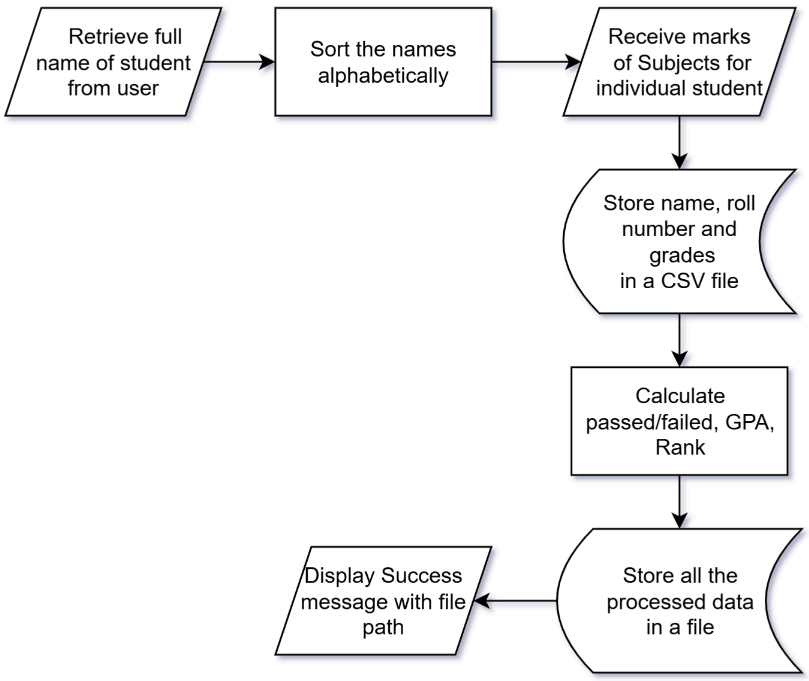
Retrieve full name of student from user
Sort the names alphabetically
Receive marks of Subjects for individual student
Store name, roll number and grades in a CSV file
Calculate passed/failed, GPA, Rank
Store all the processed data in a file
Display Success message with file path

Figure 2: flow chart of the program during runtime

4.3 Testing and Debugging

We will run this program for different volumes of students and observe the errors that occur (if any). We will have to check for duplicate names and discard the redundant ones. We will use printf() function on different steps to check if the flow of the program is as intended. Other topics may also be discussed as the advancement of writing code begins. Proper comments are to be strictly added in the program for better readability and understandability.

4.4 Future Enhancements

We really had a lot more plans that could have made this a bit better than where it is intended to be at. The plan is, to improve the UI for better user friendly interaction, to enhance the professionalism of this project by adding a secured database in the cloud, to make better with the advancement of grade management by adding different features including editing student details, searchability of the student record, advancing to grade sheet generation. Such things are possible for a larger time frame and a bit more knowledge in C with spices of any other programming language.

5 SCOPE AND APPLICATIONS

5.1 Scope of the Project:

SGMS is an efficient and simple tool designed in the C programming language to help schools and colleges manage student grades effectively. The key scope of this project includes:

5.1.1. Grade Entry:

The user can easily enter student marks from a command line. Input scores are measured against preset grading thresholds to confirm that they are accurate and correct. The teachers or system operators can mark different subjects for a student such that the system tracks the academic performance of every student.

5.1.2. Student Management:

It serves its users with capabilities that allow for the efficient handling of student records. Users can add, modify, or remove any student records as needed. This includes changing the personal details, courses being taken, and the educational level reached. It is designed to avoid replication of data which would negatively impact data integrity as well as orderliness of the system.

5.1.3. Automatic GPA Calculation:

The SGMS system is designed to calculate the GPA of the students without exceeding the limits set by the grading system in Nepal. The teacher is relieved of the burden of calculating average marks and assigning GPA values since the system automatically takes care of that.

5.1.4. File Storage:

SGMS keeps all student records in file formats that applications like Microsoft Excel and Google Sheets can easily access and use. This feature enables data portability, thus enabling teachers to export or analyze records outside of the SGMS environment. File storage simplifies data backup and recovery, ensuring that important academic records are not lost.

5.2. Application of the Project:

The program can be used in various educational institutions having a certain number of students in certain faculties.

- Schools & Colleges – Simplifies managing and tracking student progress in academic career on the basis of obtained marks of either individual students or collaboratively of the class efficiently.
- Training Institutes – Suitable for tracking any student progress for short-term courses.
- Online Education Platforms – Provides a structured path on tracking performance in online assessments and assignments evaluations.

6 TIME ESTIMATION

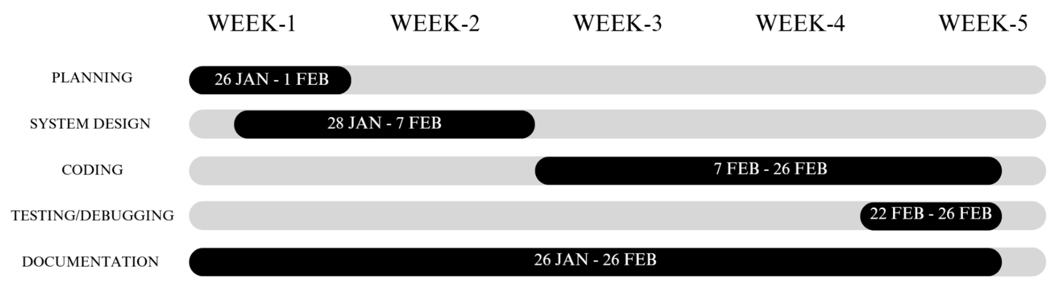

Figure 3: gantt chart for expected time estimation of the project

7 FEASIBILITY ANALYSIS

A feasibility study is done to assess the viability of the SGMS project from different perspectives, including technical, economic, and operational aspects.

7.1 Technical Feasibility

7.1.1 Minimum Hardware Requirement

- Processor: Intel Pentium or higher
- Input Devices: Keyboard and Mouse

7.1.2 Software Requirements

- Programming Language: C
- Compiler: GCC (MinGW for Windows), Turbo C++, or Dev-C++
- Operating System: Windows, Linux, or macOS
- IDE (Optional): Code::Blocks, Visual Studio Code, or Dev-C++

7.2 Economical Feasibility

This addresses whether the project is cost-effective and worth investing in. This is a pretty simple project developed in C, so the development cost is minimal. If the students or in-house develop it, the cost is negligible. If outsourced, the cost depends on the developer's rate.

7.3 Operational Feasibility

This considers whether the system meets user requirements and is also easy to use. Simple interface for teachers/admins to input and update results, Speedy and accurate retrieval of student records. secured access, Ease of Use, Error Handling, Input validation, data integrity, and Maintenance are the key factors.

References

[1] https://github.com/ishan-cse/Student-Management-System-in-C-programmin
g

[2] https://en.wikipedia.org/wiki/Academic_grading_in_Nepal

[3] https://ieeexplore.ieee.org/document/10430341

[4] https://www.geeksforgeeks.org/student-information-management-system/

[5] https://www.scaler.com/topics/student-record-management-system-c-project/