DEPARTMENT OF MECHATRONICS ENGINEERING

COMPLEX ENGINEERING PROBLEM

GROUP MEMBERS:

Fazeel Jahan 210736.

Abdullah Javed 210760.

PROJECT: Teacher Portal using C++.

COURSE: Computer Programming.

SUBMITTED TO: Sir Umer Farooq.

DATE OF SUBMISION: 6-6-2025

Table of Contents

List of Figures

Chapter 1
Introduction

In educational institutes, calculating and managing student results manually can be a time-consuming and error-prone task, especially when dealing with many students and multiple assessments like quizzes, assignments, exams, and projects. With the increasing use of digital data and online submissions, there is a need for an automated system that can process student marks quickly and accurately. This project is a Teacher Portal developed using the C++ programming language, designed to read student data from a CSV file and calculate total marks based on the weightage provided by the teacher for each component. The program then assigns grades based on the class average, displays results on the screen, allows result searching by roll number, and generates a new result file. By using this system, teachers can save time, reduce calculation mistakes, and manage results more efficiently.

1.1 Background Knowledge

In educational institution, student evaluation is a necessary component of learning and assessment. Instructors frequently give several tests within a course of study, such as quizzes, homework, midterm exams, final exams, and projects. Historically, the scores from these tests have been done manually by hand or with spreadsheets, which can be time-taking, exhausting, and more exposed to errors. With increasing class sizes and the reliance on electronic files, student data becomes harder to manage for the instructor. It is for this reason that most establishments are shifting towards computer systems that are able to read data from files, calculate, and provide results effectively. Programming languages such as C++ are commonly employed to develop such programs since they are efficient, stable, and highly taught in computer science and engineering courses. This project employs C++ to create a teacher portal that automates the calculation of results and assists teachers in saving time while enhancing accuracy by reading the csv file that contains the data of the students.

1.2 Problem Statement

To minimize the teacher's effort to manually calculate student marks and grades, there is a need for an automated system that can handle result processing efficiently.Teachers often spend a lot of time calculating final results for students by manually adding marks. This process becomes even more difficult when the number of students is large or when the marks are stored in CSV or Excel files. Manual calculations can lead to errors, inconsistencies. There is also no easy way to search for individual student results or generate final result reports. To solve these problems, there is a need for an automated system that can quickly read student data, apply custom weight-age, calculate total marks, assign grades based on class average, and generate result reports in a simple and user-friendly way.

1.3 Objectives

The objective of this complex engineering activity is to carry out research, analysis, design, investigation, and implementation of a real-world complex programming project that has the following attributes: 1. To apply abstract thinking, originality in analysis to formulate suitable programming models of the activity. 2. To apply the creative use of programming principles and research-based knowledge in novel ways. 3. The activity can extend beyond previous experiences by applying principles-based approaches. 4.To apply core programming concepts of file handling, functions, and structures in C++. 5.To automate student result processing using real data from a .csv file. 6.To calculate total scores using dynamic weightages for quizzes, assignments, midterms, finals, and project. 7.To assign grades based on class average with a relative grading system. 8.To allow users to search, display, and export individual or class results.

1.4 Scope of the project

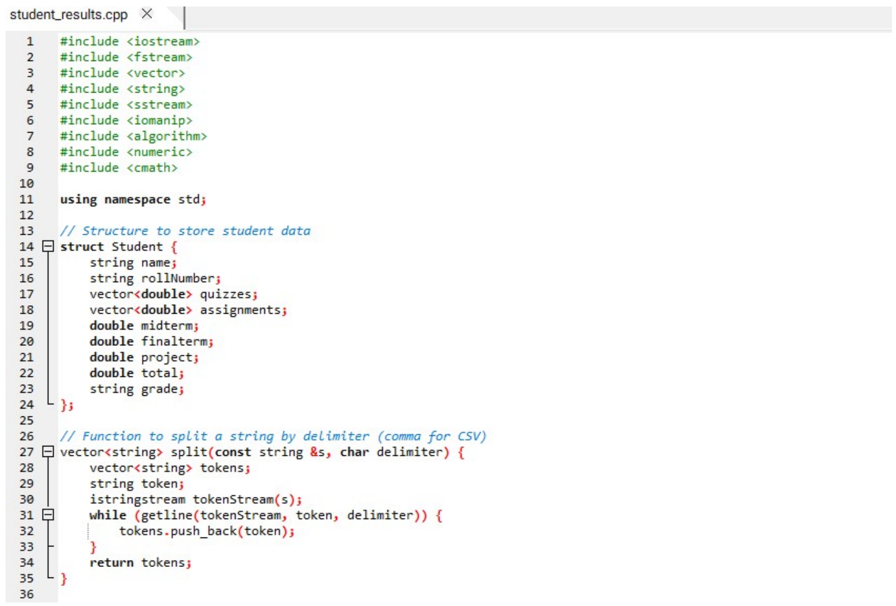
By reading information from CSV files, the Teacher Portal project, which is created in C++, aims to automate the process of calculating student grades. With the help of this console-based program, instructors may quickly process student grades that are kept in organized files and determine final grades using c++. The final grade is produced by the system once it has read each student's scores from the CSV and completed any necessary computations, such as weighted totals or averages. This minimizes errors, saves time, and offers a quick and easy way to manage grades.

Chapter 2
Implementation

2.1 Introduction:

This chapter describes how the C++ programming language is used used to create the Teacher Portal system. It walks through the entire program development process, from reading student data from a CSV file to figuring out final grades and total marks. The implementation emphasizes the usage of both fundamental and basics C++ ideas, including loops, conditionals, file handling, structures, vectors, and user-defined functions. The program's flexibility allows it to be at generic level to various weightages entered by the teacher and work with any number of quizzes or assignments.

2.2 Code Explanation:

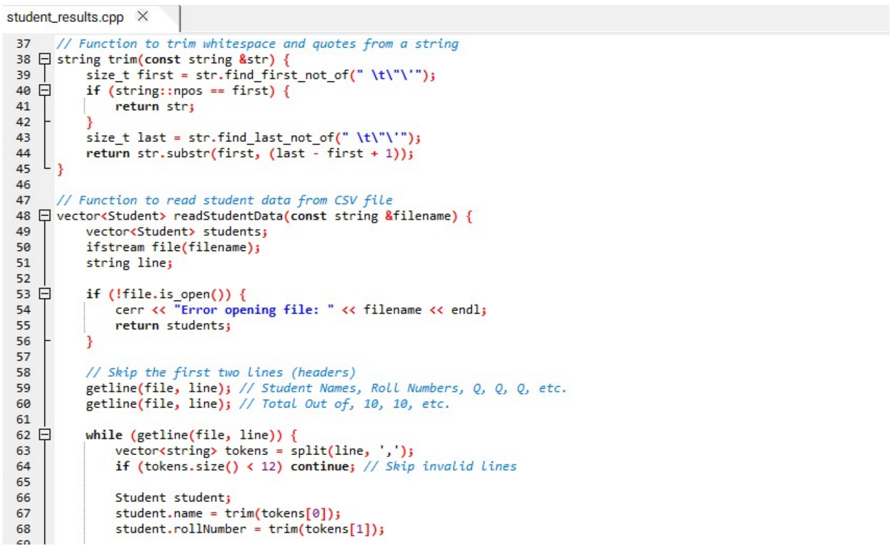


student_results.cpp

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <sstream>
#include <iomanip>
#include <algorithm>
#include <numeric>
#include <cmath>

using namespace std;

// Structure to store student data
struct Student {
    string name;
    string rollNumber;
    vector<double> quizzes;
    vector<double> assignments;
    double midterm;
    double finalterm;
    double project;
    double total;
    string grade;
};

// Function to split a string by delimiter (comma for CSV)
vector<string> split(const string &s, char delimiter) {
    vector<string> tokens;
    string token;
    istringstream tokenStream(s);
    while (getline(tokenStream, token, delimiter)) {
        tokens.push_back(token);
    }
    return tokens;
}

```

Figure 2.1: Code Snippets 1

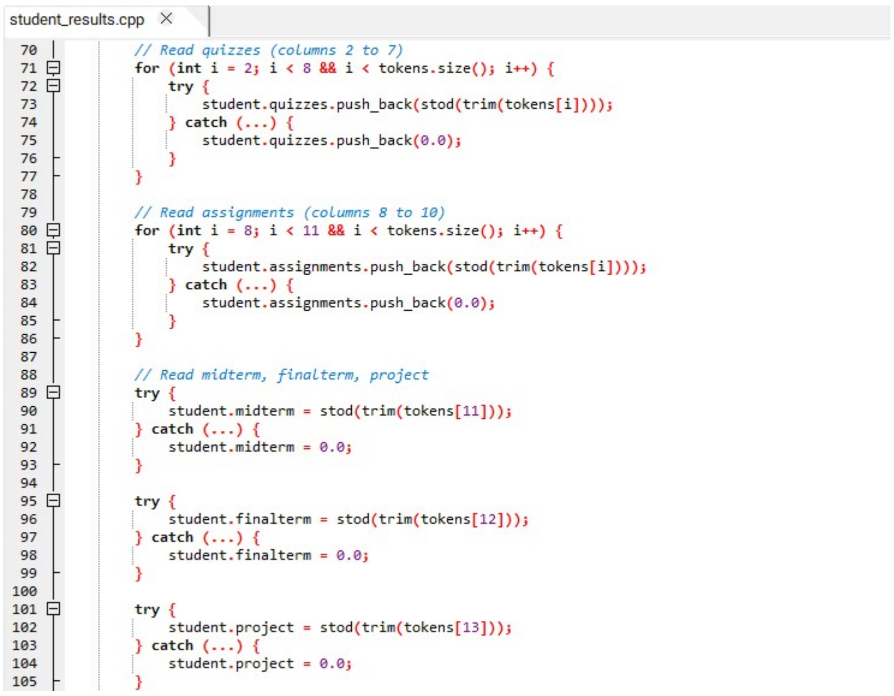
The program starts by including standard C++ libraries like ¡iostream¿, ¡fstream¿, ¡vector¿, ¡string¿, ¡sstream¿, ¡iomanip¿, and others. These are used for input/output, file handling, working with strings, formatting output, and performing mathematical operations. A structure named Student is defined to store all the data related to each student, including their name, roll number, quizzes, assignments, midterm, final, project marks, total score, and final grade.


student_results.cpp

```cpp
// Function to trim whitespace and quotes from a string
string trim(const string &str) {
    size_t first = str.find_first_not_of(" \t\"\'");
    if (string::npos == first) {
        return str;
    }
    size_t last = str.find_last_not_of(" \t\"\'");
    return str.substr(first, (last - first + 1));
}

// Function to read student data from CSV file
vector<Student> readStudentData(const string &filename) {
    vector<Student> students;
    ifstream file(filename);
    string line;

    if (!file.is_open()) {
        cerr << "Error opening file: " << filename << endl;
        return students;
    }

    // Skip the first two lines (headers)
    getline(file, line); // Student Names, Roll Numbers, Q, Q, Q, etc.
    getline(file, line); // Total Out of, 10, 10, etc.

    while (getline(file, line)) {
        vector<string> tokens = split(line, ',');
        if (tokens.size() < 12) continue; // Skip invalid lines

        Student student;
        student.name = trim(tokens[0]);
        student.rollNumber = trim(tokens[1]);
```

Figure 2.2: Code Snippet 2

Two helper functions are created: split() and trim(). The split() function breaks a line of CSV text into smaller parts (tokens) using commas as separators. The trim() function is used to remove any extra spaces or unwanted characters like quotation marks from each data item. These functions ensure that the data extracted from the CSV file is clean and ready for processing.

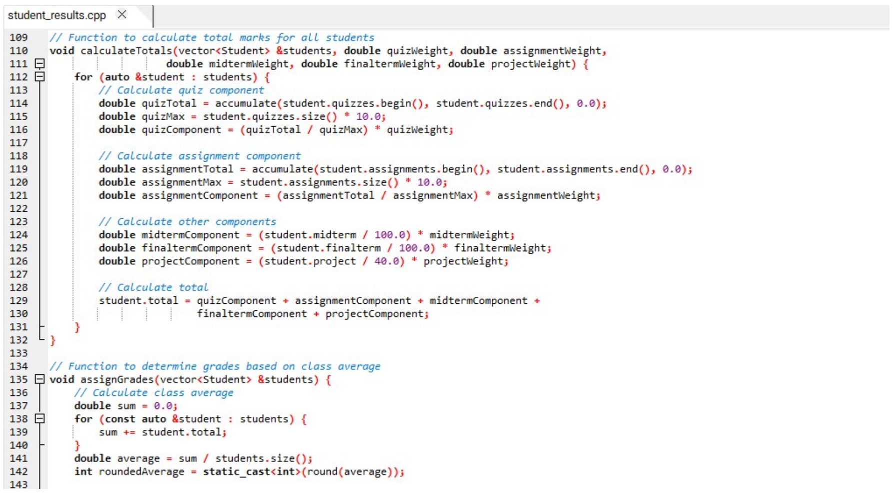


student_results.cpp

```cpp
        // Read quizzes (columns 2 to 7)
        for (int i = 2; i < 8 && i < tokens.size(); i++) {
            try {
                student.quizzes.push_back(stod(trim(tokens[i])));
            } catch (...) {
                student.quizzes.push_back(0.0);
            }
        }

        // Read assignments (columns 8 to 10)
        for (int i = 8; i < 11 && i < tokens.size(); i++) {
            try {
                student.assignments.push_back(stod(trim(tokens[i])));
            } catch (...) {
                student.assignments.push_back(0.0);
            }
        }

        // Read midterm, finalterm, project
        try {
            student.midterm = stod(trim(tokens[11]));
        } catch (...) {
            student.midterm = 0.0;
        }

        try {
            student.finalterm = stod(trim(tokens[12]));
        } catch (...) {
            student.finalterm = 0.0;
        }

        try {
            student.project = stod(trim(tokens[13]));
        } catch (...) {
            student.project = 0.0;
        }
```

Figure 2.3: Code Snippet 3

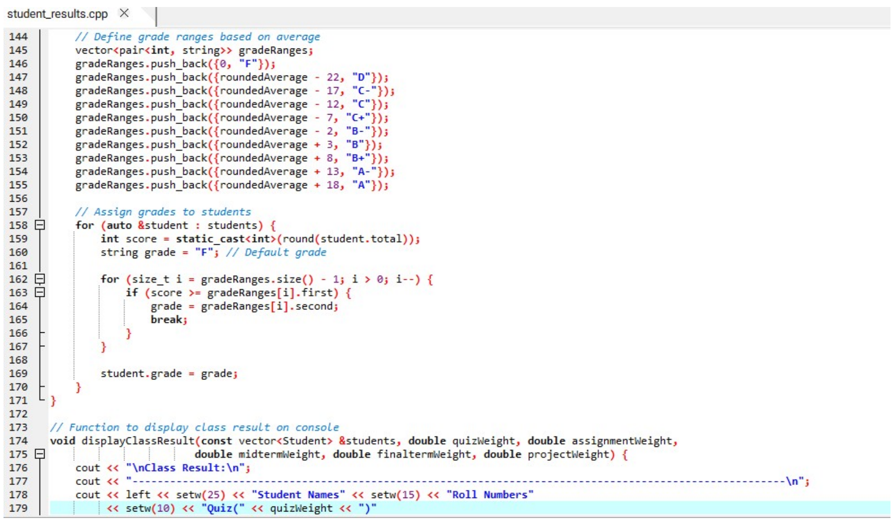
The readStudentData() function reads student records from a CSV file provided by the user. It skips the first two lines (headers) and then reads each student's name, roll number, quiz marks, assignment marks, midterm, final, and project marks. Quizzes and assignments are stored in vectors to handle multiple entries. If any value is missing or invalid, the program safely replaces it with zero using a try-catch block.


student_results.cpp

```cpp
// Function to calculate total marks for all students
void calculateTotals(vector<Student> &students, double quizWeight, double assignmentWeight,
                     double midtermWeight, double finaltermWeight, double projectWeight) {
    for (auto &student : students) {
        // Calculate quiz component
        double quizTotal = accumulate(student.quizzes.begin(), student.quizzes.end(), 0.0);
        double quizMax = student.quizzes.size() * 10.0;
        double quizComponent = (quizTotal / quizMax) * quizWeight;

        // Calculate assignment component
        double assignmentTotal = accumulate(student.assignments.begin(), student.assignments.end(), 0.0);
        double assignmentMax = student.assignments.size() * 10.0;
        double assignmentComponent = (assignmentTotal / assignmentMax) * assignmentWeight;

        // Calculate other components
        double midtermComponent = (student.midterm / 100.0) * midtermWeight;
        double finaltermComponent = (student.finalterm / 100.0) * finaltermWeight;
        double projectComponent = (student.project / 40.0) * projectWeight;

        // Calculate total
        student.total = quizComponent + assignmentComponent + midtermComponent +
                        finaltermComponent + projectComponent;
    }
}

// Function to determine grades based on class average
void assignGrades(vector<Student> &students) {
    // Calculate class average
    double sum = 0.0;
    for (const auto &student : students) {
        sum += student.total;
    }
    double average = sum / students.size();
    int roundedAverage = static_cast<int>(round(average));

```

Figure 2.4: Code Snippet 4

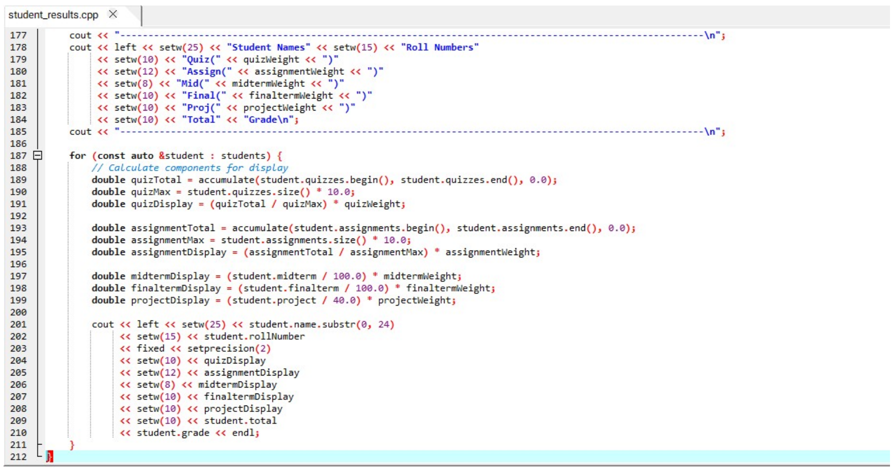
The calculateTotals() function takes the weightage values given by the user and calculates each student's total score. It computes individual components like quizzes, assignments, midterm, final, and project based on their maximum possible scores and the given weightage. All parts are then combined to get a total score out of 100. This section uses accumulate() from the numeric library to simplify summing up quiz and assignment values.


student_results.cpp

```cpp
    // Define grade ranges based on average
    vector<pair<int, string>> gradeRanges;
    gradeRanges.push_back({0, "F"});
    gradeRanges.push_back({roundedAverage - 22, "D"});
    gradeRanges.push_back({roundedAverage - 17, "C-"});
    gradeRanges.push_back({roundedAverage - 12, "C"});
    gradeRanges.push_back({roundedAverage - 7, "C+"});
    gradeRanges.push_back({roundedAverage - 2, "B-"});
    gradeRanges.push_back({roundedAverage + 3, "B"});
    gradeRanges.push_back({roundedAverage + 8, "B+"});
    gradeRanges.push_back({roundedAverage + 13, "A-"});
    gradeRanges.push_back({roundedAverage + 18, "A"});

    // Assign grades to students
    for (auto &student : students) {
        int score = static_cast<int>(round(student.total));
        string grade = "F"; // Default grade

        for (size_t i = gradeRanges.size() - 1; i > 0; i--) {
            if (score >= gradeRanges[i].first) {
                grade = gradeRanges[i].second;
                break;
            }
        }

        student.grade = grade;
    }
}

// Function to display class result on console
void displayClassResult(const vector<Student> &students, double quizWeight, double assignmentWeight,
                       double midtermWeight, double finaltermWeight, double projectWeight) {
    cout << "\nClass Result:\n";
    cout << "-------------------------------------------------------------------------------------------------\n";
    cout << left << setw(25) << "Student Names" << setw(15) << "Roll Numbers"
         << setw(10) << "Quiz(" << quizWeight << ")"
```

Figure 2.5: Code Snippet 5

In the assignGrades() function, the program first calculates the class average from all students' total marks. Based on this average, it defines grade boundaries using a flexible range (e.g., B- centered on average, A being above it, F being much lower). It then assigns grades to students depending on their total marks using a descending loop through the defined grade ranges.

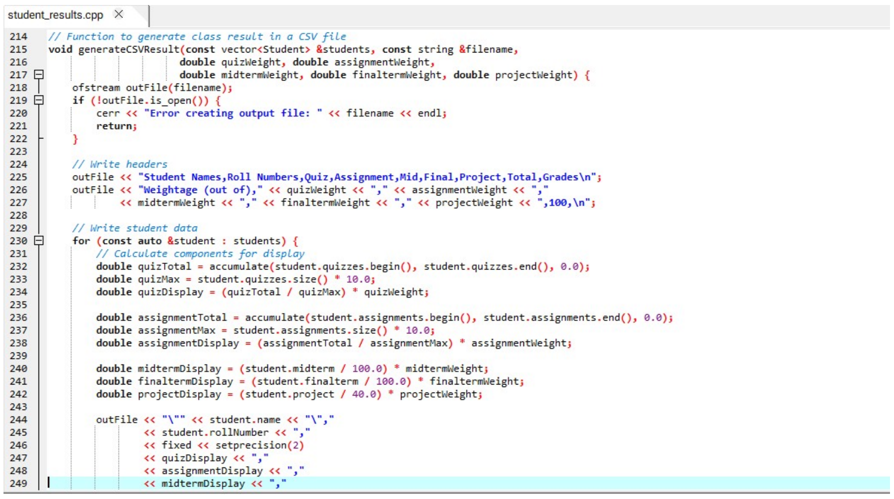


student_results.cpp

```cpp
    cout << "----------------------------------------------------------------------------------------------------\n";
    cout << left << setw(25) << "Student Names" << setw(15) << "Roll Numbers"
         << setw(10) << "Quiz(" << quizWeight << ")"
         << setw(12) << "Assign(" << assignmentWeight << ")"
         << setw(8) << "Mid(" << midtermWeight << ")"
         << setw(10) << "Final(" << finaltermWeight << ")"
         << setw(10) << "Proj(" << projectWeight << ")"
         << setw(10) << "Total" << "Grade\n";
    cout << "----------------------------------------------------------------------------------------------------\n";

    for (const auto &student : students) {
        // Calculate components for display
        double quizTotal = accumulate(student.quizzes.begin(), student.quizzes.end(), 0.0);
        double quizMax = student.quizzes.size() * 10.0;
        double quizDisplay = (quizTotal / quizMax) * quizWeight;

        double assignmentTotal = accumulate(student.assignments.begin(), student.assignments.end(), 0.0);
        double assignmentMax = student.assignments.size() * 10.0;
        double assignmentDisplay = (assignmentTotal / assignmentMax) * assignmentWeight;

        double midtermDisplay = (student.midterm / 100.0) * midtermWeight;
        double finaltermDisplay = (student.finalterm / 100.0) * finaltermWeight;
        double projectDisplay = (student.project / 40.0) * projectWeight;

        cout << left << setw(25) << student.name.substr(0, 24)
             << setw(15) << student.rollNumber
             << fixed << setprecision(2)
             << setw(10) << quizDisplay
             << setw(12) << assignmentDisplay
             << setw(8) << midtermDisplay
             << setw(10) << finaltermDisplay
             << setw(10) << projectDisplay
             << setw(10) << student.total
             << student.grade << endl;
    }
}
```

Figure 2.6: Code Snippet 6

The displayClassResult() function prints the complete result of the class on the screen in a formatted table. It shows each student's name, roll number, score breakdown (quizzes, assignments, exams, project), total marks, and assigned grade. The output is well aligned using the setw() and setprecision() functions for clean formatting.

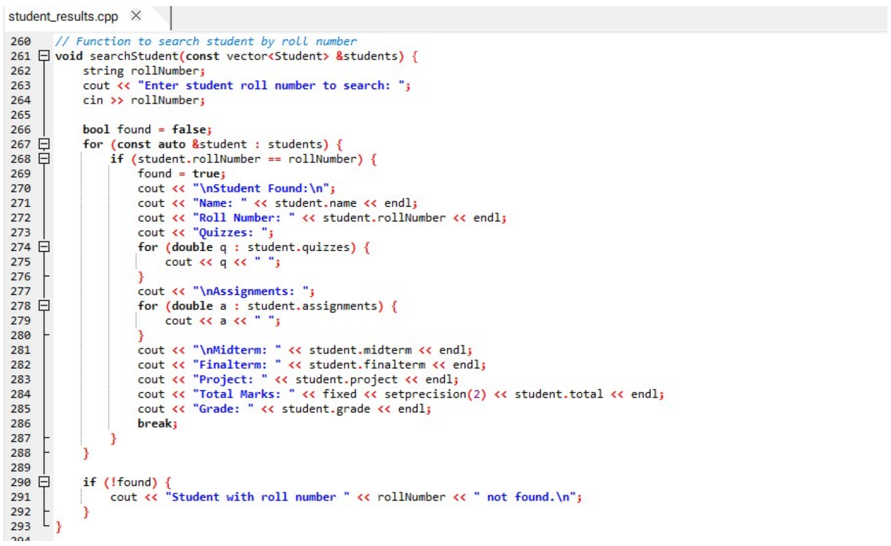


student_results.cpp

```cpp
// Function to generate class result in a CSV file
void generateCSVResult(const vector<Student> &students, const string &filename,
                       double quizWeight, double assignmentWeight,
                       double midtermWeight, double finaltermWeight, double projectWeight) {
    ofstream outFile(filename);
    if (!outFile.is_open()) {
        cerr << "Error creating output file: " << filename << endl;
        return;
    }

    // Write headers
    outFile << "Student Names,Roll Numbers,Quiz,Assignment,Mid,Final,Project,Total,Grades\n";
    outFile << "Weightage (out of)," << quizWeight << "," << assignmentWeight << ","
            << midtermWeight << "," << finaltermWeight << "," << projectWeight << ",100,\n";

    // Write student data
    for (const auto &student : students) {
        // Calculate components for display
        double quizTotal = accumulate(student.quizzes.begin(), student.quizzes.end(), 0.0);
        double quizMax = student.quizzes.size() * 10.0;
        double quizDisplay = (quizTotal / quizMax) * quizWeight;

        double assignmentTotal = accumulate(student.assignments.begin(), student.assignments.end(), 0.0);
        double assignmentMax = student.assignments.size() * 10.0;
        double assignmentDisplay = (assignmentTotal / assignmentMax) * assignmentWeight;

        double midtermDisplay = (student.midterm / 100.0) * midtermWeight;
        double finaltermDisplay = (student.finalterm / 100.0) * finaltermWeight;
        double projectDisplay = (student.project / 40.0) * projectWeight;

        outFile << "\"" << student.name << "\","
                << student.rollNumber << ","
                << fixed << setprecision(2)
                << quizDisplay << ","
                << assignmentDisplay << ","
                << midtermDisplay << ","
```

Figure 2.7: Code Snippet 7

The generateCSVResult() function writes the complete class result into a new CSV file. This file includes headers, weightage information, and detailed marks and grades of each student. This helps teachers save results in a standard format that can be opened in Excel or shared with others. This function makes the system practical and useful in real-life scenarios.

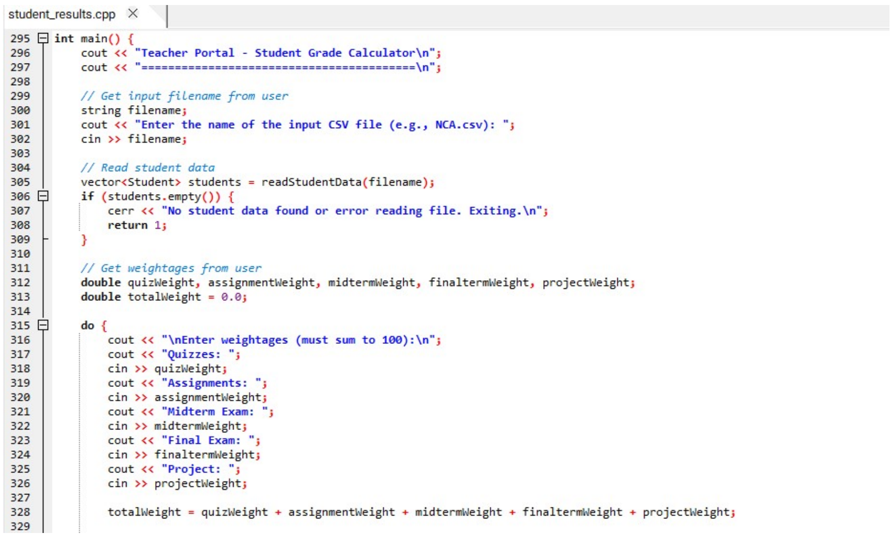


student_results.cpp

```cpp
// Function to search student by roll number
void searchStudent(const vector<Student> &students) {
    string rollNumber;
    cout << "Enter student roll number to search: ";
    cin >> rollNumber;

    bool found = false;
    for (const auto &student : students) {
        if (student.rollNumber == rollNumber) {
            found = true;
            cout << "\nStudent Found:\n";
            cout << "Name: " << student.name << endl;
            cout << "Roll Number: " << student.rollNumber << endl;
            cout << "Quizzes: ";
            for (double q : student.quizzes) {
                cout << q << " ";
            }
            cout << "\nAssignments: ";
            for (double a : student.assignments) {
                cout << a << " ";
            }
            cout << "\nMidterm: " << student.midterm << endl;
            cout << "Finalterm: " << student.finalterm << endl;
            cout << "Project: " << student.project << endl;
            cout << "Total Marks: " << fixed << setprecision(2) << student.total << endl;
            cout << "Grade: " << student.grade << endl;
            break;
        }
    }

    if (!found) {
        cout << "Student with roll number " << rollNumber << " not found.\n";
    }
}
```

Figure 2.8: Code Snippet 8

The searchStudent() function allows the teacher to enter a roll number and find the result of a specific student. It searches the student list and, if found, displays all details including individual scores and grade. If the roll number is not found, it shows an appropriate message. This improves usability by allowing quick access to individual performance.

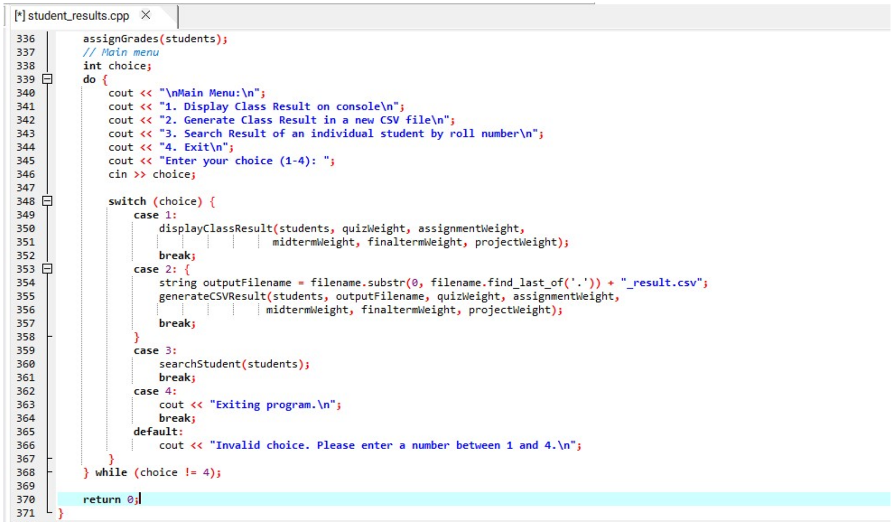


student_results.cpp

```cpp
int main() {
    cout << "Teacher Portal - Student Grade Calculator\n";
    cout << "========================================\n";

    // Get input filename from user
    string filename;
    cout << "Enter the name of the input CSV file (e.g., NCA.csv): ";
    cin >> filename;

    // Read student data
    vector<Student> students = readStudentData(filename);
    if (students.empty()) {
        cerr << "No student data found or error reading file. Exiting.\n";
        return 1;
    }

    // Get weightages from user
    double quizWeight, assignmentWeight, midtermWeight, finaltermWeight, projectWeight;
    double totalWeight = 0.0;

    do {
        cout << "\nEnter weightages (must sum to 100):\n";
        cout << "Quizzes: ";
        cin >> quizWeight;
        cout << "Assignments: ";
        cin >> assignmentWeight;
        cout << "Midterm Exam: ";
        cin >> midtermWeight;
        cout << "Final Exam: ";
        cin >> finaltermWeight;
        cout << "Project: ";
        cin >> projectWeight;

        totalWeight = quizWeight + assignmentWeight + midtermWeight + finaltermWeight + projectWeight;

```

Figure 2.9: Code Snippet 9

In the main() function, after displaying a welcome message, the program asks the user for the input CSV filename and reads the student data. It then asks the teacher to enter the weightage for each section (quiz, assignment, midterm, etc.). The input is validated to ensure the total equals 100; otherwise, it asks the user to enter again. This step ensures accurate grading and avoids errors caused by incorrect weight distribution.



[*] student_results.cpp

```cpp
    assignGrades(students);
    // Main menu
    int choice;
    do {
        cout << "\nMain Menu:\n";
        cout << "1. Display Class Result on console\n";
        cout << "2. Generate Class Result in a new CSV file\n";
        cout << "3. Search Result of an individual student by roll number\n";
        cout << "4. Exit\n";
        cout << "Enter your choice (1-4): ";
        cin >> choice;

        switch (choice) {
            case 1:
                displayClassResult(students, quizWeight, assignmentWeight,
                                   midtermWeight, finaltermWeight, projectWeight);
                break;
            case 2: {
                string outputFilename = filename.substr(0, filename.find_last_of('.')) + "_result.csv";
                generateCSVResult(students, outputFilename, quizWeight, assignmentWeight,
                                  midtermWeight, finaltermWeight, projectWeight);
                break;
            }
            case 3:
                searchStudent(students);
                break;
            case 4:
                cout << "Exiting program.\n";
                break;
            default:
                cout << "Invalid choice. Please enter a number between 1 and 4.\n";
        }
    } while (choice != 4);

    return 0;
}
```

Figure 2.10: Code Snippet 10

Finally, the main menu is shown with four options: display the result on screen, generate the result in a file, search for a student, or exit. Based on the teacher's input, the corresponding function is called. The menu runs in a loop until the user chooses to exit. This structure makes the system user-friendly and interactive, suitable for practical use in academic environments.

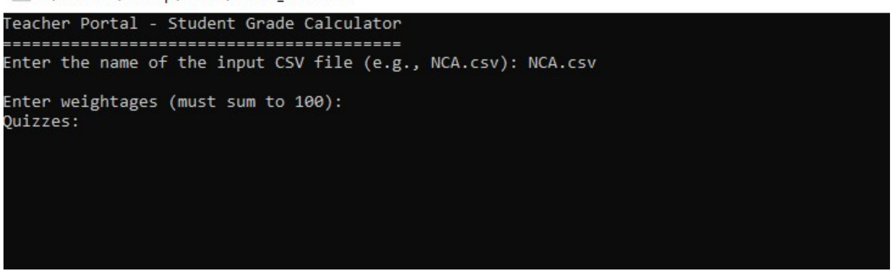
Chapter 3
Program Outputs

3.1 Introduction:

The results of running the Teacher Portal application with actual or sample student data are shown in this chapter. The outputs include the creation of a CSV result file, the display of the entire class result on the console, and the search result for a specific student by roll number. With the aid of these outputs, the user may confirm that the system is operating appropriately and producing precise computations based on the supplied data and assigned weights. The teacher can more easily assess and supervise student performance because the results clearly display each student's total marks and related grade. Screenshots and explanations of all significant outputs produced during program testing and usage are included in this chapter.

3.2 Loading csv File:

This section includes the loading of the csv file where students data is stored it is done by calling the name of the csv file in the console we have to make sure that the csv file should be kept in the same folder as the c++ file of the code is stored.

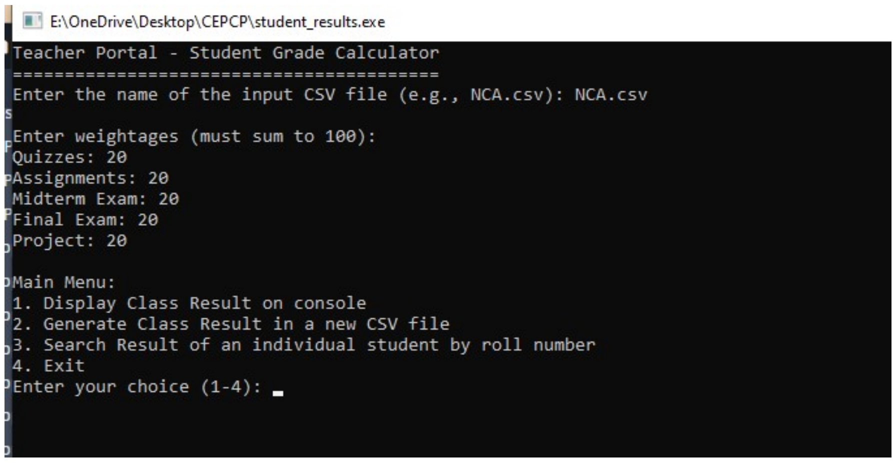


```
E:\OneDrive\Desktop\CEPCP\student_results.exe
Teacher Portal - Student Grade Calculator
========================================
Enter the name of the input CSV file (e.g., NCA.csv): NCA.csv

Enter weightages (must sum to 100):
Quizzes:
```

Figure 3.1: Output 1

3.3 Declearing Weightages:

This Program is made generic for the users or we can say instructor they can by choice enter their own size of weightages for quizzes, assignments, mid terms, final terms and project.The sum of all the weightage should be kept equal to 100 sum less than or grater than 100 would generate an error on the console. After entering the weightages of all five parameters we would be given a menu section displaying whether we want the result on the console, whether we want to generate a new csv file, or want to find the data of a specific student.The last option is exit it is used to end the program.

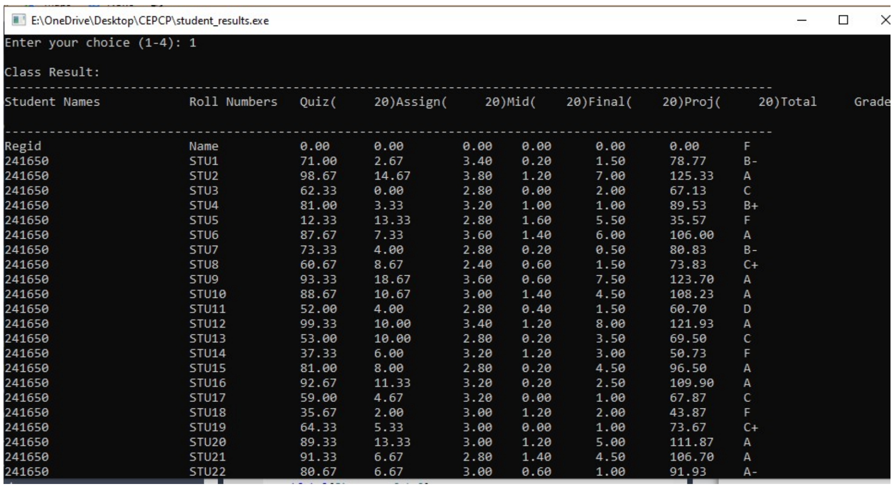


```
E:\OneDrive\Desktop\CEPCP\student_results.exe
Teacher Portal - Student Grade Calculator
========================================
Enter the name of the input CSV file (e.g., NCA.csv): NCA.csv

Enter weightages (must sum to 100):
Quizzes: 20
Assignments: 20
Midterm Exam: 20
Final Exam: 20
Project: 20

Main Menu:
1. Display Class Result on console
2. Generate Class Result in a new CSV file
3. Search Result of an individual student by roll number
4. Exit
Enter your choice (1-4): _
```

Figure 3.2: Output 2

3.4 Main Menu Outputs:

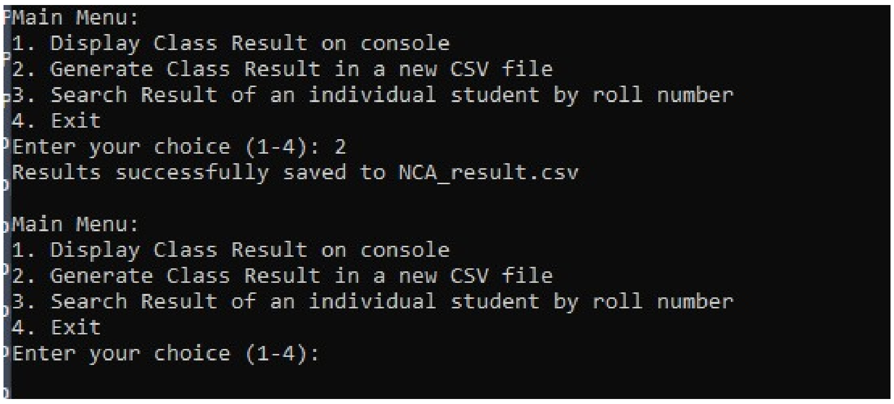
Four Choices are given after decleration of weightages the choice user can make. I.Show Results On Console: The application shows the complete class result in a tidy tabular format on the console when the user chooses Option 1. The function displayClassResult() handles this in the backend and is invoked within the switch statement of the main menu. This function uses setw() to align the columns and setprecision() to round the output after receiving the list of students and their computed scores. Using accumulate(), it recalculates each student's component scores (quiz, assignment, midterm, final, and project) and adds them up to a total, which is then printed with the grade that was given. With this option, the console pane displays a comprehensive summary of the class performance.



```
E:\OneDrive\Desktop\CEPCP\student_results.exe
Enter your choice (1-4): 1

Class Result:
--------------------------------------------------------------------------------------------------------
Student Names            Roll Numbers   Quiz(     20)Assign(     20)Mid(   20)Final(    20)Proj(     20)Total     Grade

--------------------------------------------------------------------------------------------------------
Regid                    Name           0.00      0.00        0.00    0.00      0.00      0.00      F
241650                   STU1           71.00     2.67        3.40    0.20      1.50      78.77     B-
241650                   STU2           98.67     14.67       3.80    1.20      7.00      125.33    A
241650                   STU3           62.33     0.00        2.80    0.00      2.00      67.13     C
241650                   STU4           81.00     3.33        3.20    1.00      1.00      89.53     B+
241650                   STU5           12.33     13.33       2.80    1.60      5.50      35.57     F
241650                   STU6           87.67     7.33        3.60    1.40      6.00      106.00    A
241650                   STU7           73.33     4.00        2.80    0.20      0.50      80.83     B-
241650                   STU8           60.67     8.67        2.40    0.60      1.50      73.83     C+
241650                   STU9           93.33     18.67       3.60    0.60      7.50      123.70    A
241650                   STU10          88.67     10.67       3.00    1.40      4.50      108.23    A
241650                   STU11          52.00     4.00        2.80    0.40      1.50      60.70     D
241650                   STU12          99.33     10.00       3.40    1.20      8.00      121.93    A
241650                   STU13          53.00     10.00       2.80    0.20      3.50      69.50     C
241650                   STU14          37.33     6.00        3.20    1.20      3.00      50.73     F
241650                   STU15          81.00     8.00        2.80    0.20      4.50      96.50     A
241650                   STU16          92.67     11.33       3.20    0.20      2.50      109.90    A
241650                   STU17          59.00     4.67        3.20    0.00      1.00      67.87     C
241650                   STU18          35.67     2.00        3.00    1.20      2.00      43.87     F
241650                   STU19          64.33     5.33        3.00    0.00      1.00      73.67     C+
241650                   STU20          89.33     13.33       3.00    1.20      5.00      111.87    A
241650                   STU21          91.33     6.67        2.80    1.40      4.50      106.70    A
241650                   STU22          80.67     6.67        3.00    0.60      1.00      91.93     A-
```

Figure 3.3: Output 3

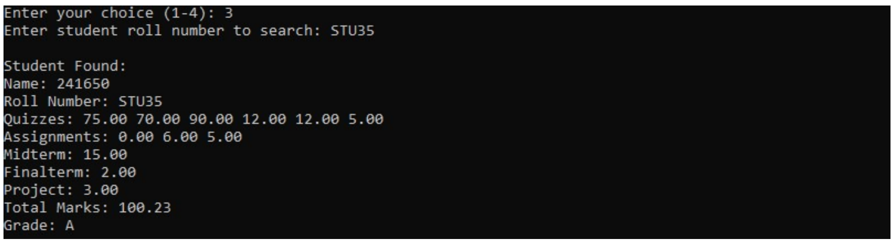
II.Make a new result csv File:The user can save the entire result to a new CSV file by choosing Option 2. This is handled in the backend by the generateCSVResult() method, which is likewise activated by a switch case. The application uses ofstream to open the file and automatically creates a new filename by attaching $_result.$
csv to the original file name. All student information, including individual components and overall grades, is written into the file together with the headers and weights. The same reasoning used in the display function is applied here, except calculations are written to the file rather than printed to the console. With this choice, the instructor can maintain an enduring, publicly accessible record of the outcomes.



```
Main Menu:
1. Display Class Result on console
2. Generate Class Result in a new CSV file
3. Search Result of an individual student by roll number
4. Exit
Enter your choice (1-4): 2
Results successfully saved to NCA_result.csv

Main Menu:
1. Display Class Result on console
2. Generate Class Result in a new CSV file
3. Search Result of an individual student by roll number
4. Exit
Enter your choice (1-4):
```

Figure 3.4: Output 4

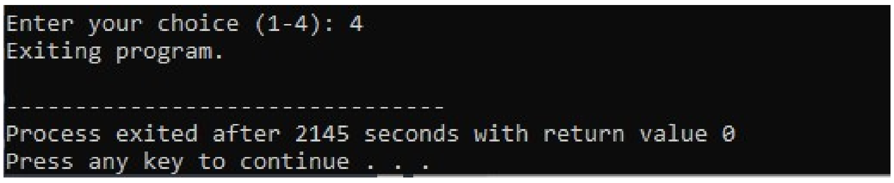
III.Search by Roll Number:Selecting Option 3 enables the user to utilize their roll number to search for a particular student's results. The searchStudent() method in the backend manages this capability. After requesting a roll number, the application employs a loop to look through the student's vector. The student's name, quiz and assignment scores, midterm, final, project marks, total, and grade are printed if a match is discovered. If the roll number cannot be located, a message stating that it does not exist is displayed. By using this feature, teachers may easily view and confirm each student's performance without having to go through the entire result.



```
Enter your choice (1-4): 3
Enter student roll number to search: STU35

Student Found:
Name: 241650
Roll Number: STU35
Quizzes: 75.00 70.00 90.00 12.00 12.00 5.00
Assignments: 0.00 6.00 5.00
Midterm: 15.00
Finalterm: 2.00
Project: 3.00
Total Marks: 100.23
Grade: A
```

Figure 3.5: Output 5

IV. Exit: Selecting Option 4 causes the software to terminate. This is managed via a loop in the code that continues to display the menu until the user inputs 4. The program merely writes an exit message and breaks the loop when the switch block detects this input. This is handled simply in the main() function using a do-while loop; no special function is called. After completing all necessary activities, this choice is crucial for a clean application shutdown.



```
Enter your choice (1-4): 4
Exiting program.

--------------------------------
Process exited after 2145 seconds with return value 0
Press any key to continue . . .
```

Figure 3.6: Output 6

Chapter 4
Learning Outcomes

The technical and problem-solving parts of working on this project have been quite beneficial. The creation of the Teacher Portal system in C++ allowed for the practice and practical application of various important programming topics. The ability to use C++'s file handling capabilities to read and analyze structured data from external files, such as CSVs, was one of the main results. This improved comprehension of string manipulation, input/output streams, and data parsing through the use of getline(), ifstream, and string splitting techniques. Additionally, my ability to effectively handle collections of items was enhanced by the use of vectors and structures to store and manage dynamic student data. My knowledge of control flow, loops, conditional statements, and mathematical operations was enhanced by the logic used to compute weighted scores, apply grading policies, and validate user input. In addition to honing my technical skills, this project helped me improve my ability to solve problems by forcing me to use logic and divide a challenging grading work into smaller, more manageable programming tasks. It demonstrated how crucial it is to manage exceptions and make sure the system maintains stability even in the face of inaccurate or missing data. Finally, this project demonstrated how important it is to write code that is not just useful but also flexible and user-focused. I gained an understanding of the importance of flexibility in software design by developing a platform that can accommodate any number of students and quizzes. All things considered, this project gave hands-on practice utilizing C++ to address a significant issue, which will be helpful for upcoming programming assignments and projects in both academic and professional settings.

Chapter 5
Conclusion

To sum up, this project effectively illustrates how C++ may be utilized to programmatically tackle actual academic difficulties. Reading student data, determining final grades using dynamic weights, and producing results in console and file formats are all automated by the Teacher Portal system. It guarantees correctness and equity in grading, lowers the possibility of human error, and saves teachers a great deal of time. The project promoted logical thinking and well-organized code while enabling the actual application of fundamental C++ concepts including file handling, structures, vectors, and functions. Its user-friendliness is further enhanced by its menu-driven layout and the capability to search for specific student outcomes. The system successfully accomplishes its goals and offers a practical, adaptable tool for managing student evaluation.

References

1.https://www.scribd.com/document/434628462/c-school-management

2.https://github.com/Mubeen-Channa/Student-Grade-Calculator

Chapter 8
Appendix

Appendix A:

Code of Portal:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <sstream>
#include <iomanip>
#include <algorithm>
#include <numeric>
#include <cmath>

using namespace std;

// Structure to store
student data
struct Student {
   string name;
   string rollNumber;
   vector<double>
quizzes;
   vector<double>
assignments;
   double midterm;
   double finalterm;
   double project;
   double total;
```

```
    string grade;
};

// Function to split a
string by delimiter
(comma for CSV)
vector<string>
split(const string &s,
char delimiter) {
    vector<string>
tokens;
    string token;
    istringstream
tokenStream(s);
    while
(getline(tokenStream,
token, delimiter)) {

tokens.push_back(to
ken);
    }
    return tokens;
}

// Function to trim
whitespace and
quotes from a string
string trim(const
string &str) {
    size_t first = str.find_first_not_of(" \t\"\'");
    if (string::npos == first) {
        return str;
    }
    size_t last = str.find_last_not_of(" \t\"\'");
    return str.substr(first, (last - first + 1));
}

// Function to read student data from CSV file
vector<Student> readStudentData(const string &filename) {
    vector<Student> students;
    ifstream file(filename);
    string line;

    if (!file.is_open()) {
        cerr << "Error opening file: " << filename << endl;
        return students;
    }

    // Skip the first two
lines (headers)
    getline(file, line); //
Student Names, Roll
Numbers, Q, Q, Q,
etc.
    getline(file, line); //
Total Out of, 10, 10,
etc.

    while (getline(file,
line)) {
        vector<string>
tokens = split(line, ',');
        if (tokens.size() <
12) continue; // Skip
invalid lines

        Student student;
        student.name =
trim(tokens[0]);

student.rollNumber =
trim(tokens[1]);

        // Read quizzes
(columns 2 to 7)
```

```
        for (int i = 2; i < 8
&&  i  <  tokens.size();
i++) {
            try {

student.quizzes.push
_back(stod(trim(toke
ns[i])));
            } catch (...) {

student.quizzes.push
_back(0.0);
            }
        }
        //                Read
assignments
(columns 8 to 10)
        for (int i = 8; i < 11
&&  i  <  tokens.size();
i++) {
            try {

student.assignments.
push_back(stod(trim(
tokens[i])));
            } catch (...) {

student.assignments.
push_back(0.0);
            }
        }
        // Read midterm,
finalterm, project
        try {

student.midterm      =
stod(trim(tokens[11])
);
        } catch (...) {

student.midterm      =
0.0;
        }
        try {

student.finalterm      =
stod(trim(tokens[12])
);
        } catch (...) {

student.finalterm      =
0.0;
        }
        try {

student.project        =
stod(trim(tokens[13])
);
        } catch (...) {

student.project = 0.0;
        }
```

```
students.push_back(s
tudent);
   }

   file.close();
   return students;
}

//      Function      to
calculate  total  marks
for all students
void
calculateTotals(vector
<Student> &students,
double     quizWeight,
double
assignmentWeight,
               double
midtermWeight,
double
finaltermWeight,
double
projectWeight) {
   for (auto &student :
students) {
      // Calculate  quiz
component
      double   quizTotal
=
accumulate(student.
quizzes.begin(),
student.quizzes.end()
, 0.0);
        double    quizMax
=
student.quizzes.size()
* 10.0;
        double
quizComponent        =
(quizTotal  /  quizMax)
* quizWeight;

        //          Calculate
assignment
component
        double
assignmentTotal        =
accumulate(student.a
ssignments.begin(),
student.assignments.
end(), 0.0);
        double
assignmentMax        =
student.assignments.
size() * 10.0;
        double
assignmentCompone
nt = (assignmentTotal
/   assignmentMax)   *
assignmentWeight;
```

```
        // Calculate other
components
        double
midtermComponent
= (student.midterm /
100.0)              *
midtermWeight;
        double
finaltermComponent
= (student.finalterm /
100.0)              *
finaltermWeight;
        double
projectComponent  =
(student.project    /
40.0)               *
projectWeight;

        // Calculate total
        student.total   =
quizComponent       +
assignmentCompone
nt                  +
midtermComponent
+

finaltermComponent
+ projectComponent;
    }
}
```

```
// Function to
determine grades
based on class
average
void
assignGrades(vector<
Student> &students) {
   // Calculate class
average
   double sum = 0.0;
   for (const auto
&student : students) {
      sum +=
student.total;
   }
   double average =
sum / students.size();
   int
roundedAverage =
static_cast<int>(roun
d(average));

   // Define grade
ranges based on
average
   vector<pair<int,
string>>
gradeRanges;

gradeRanges.push_b
ack({0, "F"});
```

```
gradeRanges.push_b
ack({roundedAverage
- 22, "D"});

gradeRanges.push_b
ack({roundedAverage
- 17, "C-"});

gradeRanges.push_b
ack({roundedAverage
- 12, "C"});

gradeRanges.push_b
ack({roundedAverage
- 7, "C+"});

gradeRanges.push_b
ack({roundedAverage
- 2, "B-"});

gradeRanges.push_b
ack({roundedAverage
+ 3, "B"});

gradeRanges.push_b
ack({roundedAverage
+ 8, "B+"});

gradeRanges.push_b
```

```
ack({roundedAverage
+ 13, "A-"});

gradeRanges.push_b
ack({roundedAverage
+ 18, "A"});

   // Assign grades to
students
   for (auto &student :
students) {
      int score =
static_cast<int>(roun
d(student.total));
      string grade =
"F"; // Default grade

      for (size_t i =
gradeRanges.size() -
1; i > 0; i--) {
         if (score >=
gradeRanges[i].first) {
            grade =
gradeRanges[i].secon
d;
            break;
         }
      }

      student.grade =
grade;
```

```
    }
}

// Function to display
class result on
console
void
displayClassResult(co
nst vector<Student>
&students, double
quizWeight, double
assignmentWeight,
                double
midtermWeight,
double
finaltermWeight,
double
projectWeight) {
    cout << "\nClass
Result:\n";
    cout << "-------------
----------------------------
----------------------------
----------------------------
------\n";
    cout << left <<
setw(25) << "Student
Names" << setw(15)
<< "Roll Numbers"
```

```
        << setw(10) <<
"Quiz(" << quizWeight
<< ")"
        << setw(12) <<
"Assign(" <<
assignmentWeight <<
")"
        << setw(8) <<
"Mid(" <<
midtermWeight << ")"
        << setw(10) <<
"Final(" <<
finaltermWeight <<
")"
        << setw(10) <<
"Proj(" <<
projectWeight << ")"
        << setw(10) <<
"Total" << "Grade\n";
    cout << "-------------
---------------------------
---------------------------
---------------------------
------\n";

    for (const auto
&student : students) {
        // Calculate
components for
display
```

```
    double quizTotal
=
accumulate(student.
quizzes.begin(),
student.quizzes.end()
, 0.0);
    double quizMax
=
student.quizzes.size()
* 10.0;
    double
quizDisplay =
(quizTotal / quizMax)
* quizWeight;

    double
assignmentTotal =
accumulate(student.a
ssignments.begin(),
student.assignments.
end(), 0.0);
    double
assignmentMax =
student.assignments.
size() * 10.0;
    double
assignmentDisplay =
(assignmentTotal /
assignmentMax) *
assignmentWeight;
```

```
        double
midtermDisplay        =
(student.midterm     /
100.0)                *
midtermWeight;
        double
finaltermDisplay      =
(student.finalterm   /
100.0)                *
finaltermWeight;
        double
projectDisplay        =
(student.project     /
40.0)                 *
projectWeight;

        cout  <<  left  <<
setw(25)             <<
student.name.substr(
0, 24)
             << setw(15) <<
student.rollNumber
             <<   fixed   <<
setprecision(2)
             << setw(10) <<
quizDisplay
             << setw(12) <<
assignmentDisplay
             <<  setw(8)  <<
midtermDisplay
```

```
          << setw(10) <<
finaltermDisplay
          << setw(10) <<
projectDisplay
          << setw(10) <<
student.total
          <<
student.grade       <<
endl;
   }
}

//      Function      to
generate class result
in a CSV file
void
generateCSVResult(c
onst vector<Student>
&students,       const
string &filename,
                 double
quizWeight,    double
assignmentWeight,
                 double
midtermWeight,
double
finaltermWeight,
double
projectWeight) {
   ofstream
outFile(filename);
```

```
    if
(!outFile.is_open()) {
        cerr    <<    "Error
creating output file: "
<< filename << endl;
        return;
    }

    // Write headers
    outFile << "Student
Names,Roll
Numbers,Quiz,Assign
ment,Mid,Final,Proje
ct,Total,Grades\n";
    outFile               <<
"Weightage (out of),"
<< quizWeight << ","
<< assignmentWeight
<< ","
            <<
midtermWeight << ","
<<    finaltermWeight
<<          ","          <<
projectWeight        <<
",100,\n";

    //    Write    student
data
    for    (const    auto
&student : students) {
        // Calculate components for display
        double quizTotal = accumulate(student.quizzes.begin(), student.quizzes.end(), 0.0);
        double quizMax = student.quizzes.size() * 10.0;
        double quizDisplay = (quizTotal / quizMax) * quizWeight;

        double assignmentTotal = accumulate(student.assignments.begin(), student.assignments.end(), 0.0);
        double assignmentMax = student.assignments.size() * 10.0;
        double assignmentDisplay = (assignmentTotal /
assignmentMax)      *
assignmentWeight;

        double
midtermDisplay      =
(student.midterm    /
100.0)              *
midtermWeight;
        double
finaltermDisplay    =
(student.finalterm  /
100.0)              *
finaltermWeight;
        double
projectDisplay      =
(student.project    /
40.0)               *
projectWeight;

        outFile << "\"" <<
student.name << "\","
                <<
student.rollNumber
<< ","
                << fixed <<
setprecision(2)
                <<
quizDisplay << ","
                <<
assignmentDisplay <<
","
```

```
            <<
midtermDisplay << ","
            <<
finaltermDisplay     <<
","
            <<
projectDisplay << ","
            <<
student.total << ","
            <<
student.grade        <<
"\n";
    }

    outFile.close();
    cout    <<    "Results
successfully saved to "
<< filename << endl;
}

// Function to search
student    by    roll
number
void
searchStudent(const
vector<Student>
&students) {
    string rollNumber;
    cout    <<    "Enter
student roll number
to search: ";
```

```
    cin >> rollNumber;

    bool found = false;
    for (const auto
&student : students) {
        if
(student.rollNumber
== rollNumber) {
            found = true;
            cout <<
"\nStudent
Found:\n";
            cout <<
"Name: " <<
student.name <<
endl;
            cout << "Roll
Number: " <<
student.rollNumber
<< endl;
            cout <<
"Quizzes: ";
            for (double q :
student.quizzes) {
                cout << q <<
" ";
            }
            cout <<
"\nAssignments: ";
```

```
        for (double a :
student.assignments)
{
            cout << a <<
" ";
        }
        cout            <<
"\nMidterm:    "    <<
student.midterm    <<
endl;
        cout            <<
"Finalterm:     "    <<
student.finalterm   <<
endl;
        cout            <<
"Project:        "      <<
student.project      <<
endl;
        cout  <<  "Total
Marks:  "  <<  fixed  <<
setprecision(2)       <<
student.total << endl;
        cout            <<
"Grade:        "       <<
student.grade        <<
endl;
        break;
      }
   }

   if (!found) {
        cout << "Student
with roll number " <<
rollNumber << " not
found.\n";
    }
}

int main() {
    cout << "Teacher
Portal - Student
Grade Calculator\n";
    cout <<
"=================
=================
=======\n";

    // Get input
filename from user
    string filename;
    cout << "Enter the
name of the input CSV
file (e.g., NCA.csv): ";
    cin >> filename;

    // Read student
data
    vector<Student>
students =
readStudentData(file
name);
```

```
    if
(students.empty()) {
        cerr << "No
student data found or
error reading file.
Exiting.\n";
        return 1;
    }

    // Get weightages
from user
    double quizWeight,
assignmentWeight,
midtermWeight,
finaltermWeight,
projectWeight;
    double totalWeight
= 0.0;

    do {
        cout << "\nEnter
weightages (must
sum to 100):\n";
        cout << "Quizzes:
";
        cin >>
quizWeight;
        cout <<
"Assignments: ";
        cin >>
assignmentWeight;
```

```
    cout <<
"Midterm Exam: ";
    cin >>
midtermWeight;
    cout << "Final
Exam: ";
    cin >>
finaltermWeight;
    cout << "Project:
";
    cin >>
projectWeight;

    totalWeight =
quizWeight +
assignmentWeight +
midtermWeight +
finaltermWeight +
projectWeight;

    if
(abs(totalWeight -
100.0) > 0.001) {
        cout << "Error:
Weightages must sum
to 100 (current sum: "
<< totalWeight << ").
Please try again.\n";
    }
    } while (abs(totalWeight - 100.0) > 0.001);
    // Calculate totals and grades

calculateTotals(students, quizWeight, assignmentWeight, midtermWeight, finaltermWeight, projectWeight);

assignGrades(students);
    // Main menu
    int choice;
    do {
        cout << "\nMain Menu:\n";
        cout << "1. Display Class Result on console\n";
        cout << "2. Generate Class Result in a new CSV file\n";
        cout << "3. Search Result of an individual student by roll number\n";
    cout << "4.
Exit\n";
    cout << "Enter
your choice (1-4): ";
    cin >> choice;

    switch (choice) {
      case 1:

displayClassResult(st
udents, quizWeight,
assignmentWeight,

midtermWeight,
finaltermWeight,
projectWeight);
        break;
      case 2: {
        string
outputFilename =
filename.substr(0,
filename.find_last_of
('.')) + "_result.csv";

generateCSVResult(st
udents,
outputFilename,
quizWeight,
assignmentWeight,

midtermWeight,
finaltermWeight,
projectWeight);
            break;
         }
         case 3:

searchStudent(stude
nts);
            break;
         case 4:
            cout        <<
"Exiting program.\n";
            break;
         default:
            cout        <<
"Invalid        choice.
Please      enter      a
number  between  1
and 4.\n";
      }
   }  while  (choice  !=
4);

   return 0;
}
```