**Air university**



**Complex Engineering Activity**

**(Teacher Portal using C++)**

* **Alishba Waris(242047)**

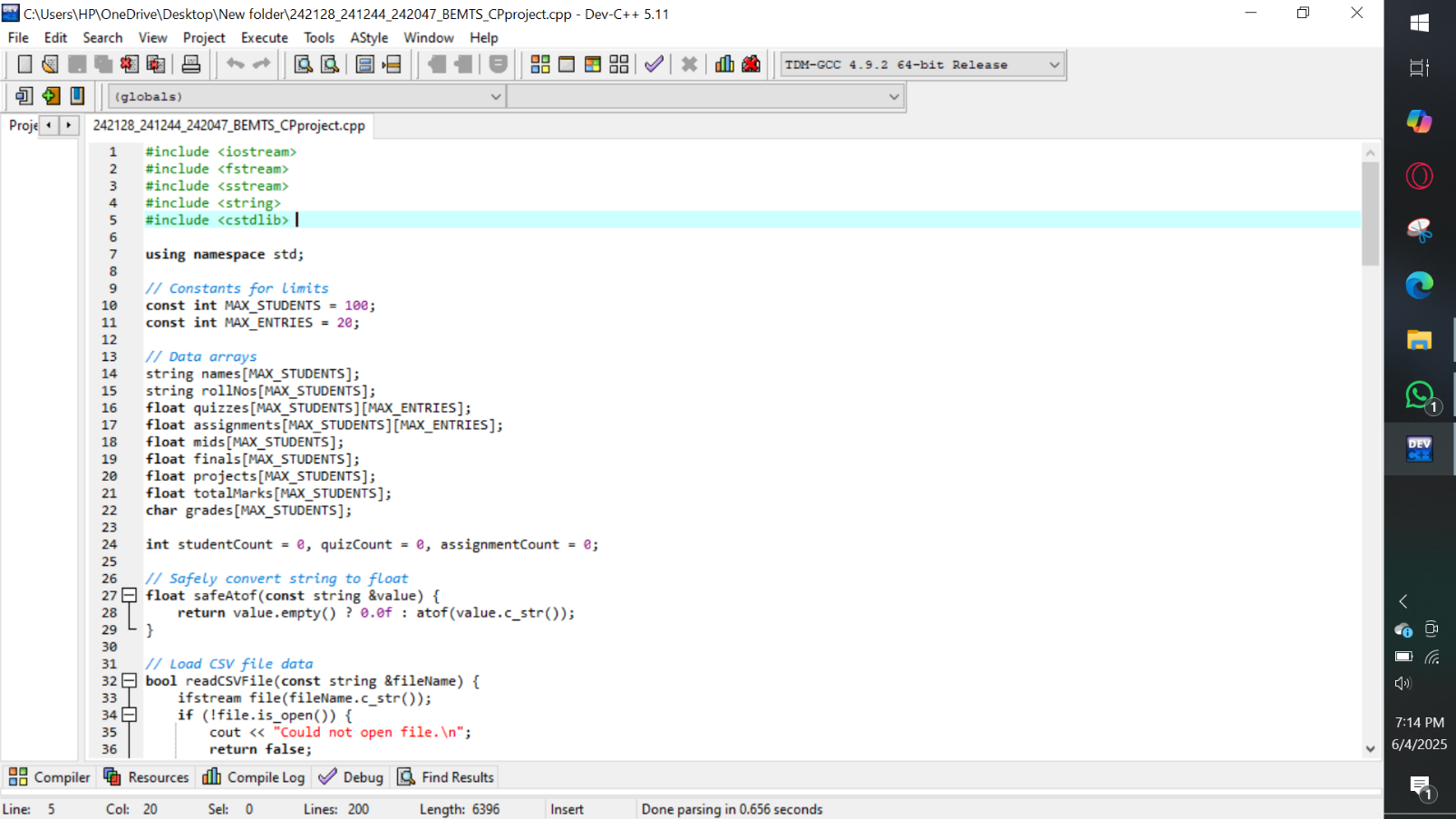
**Introduction:**

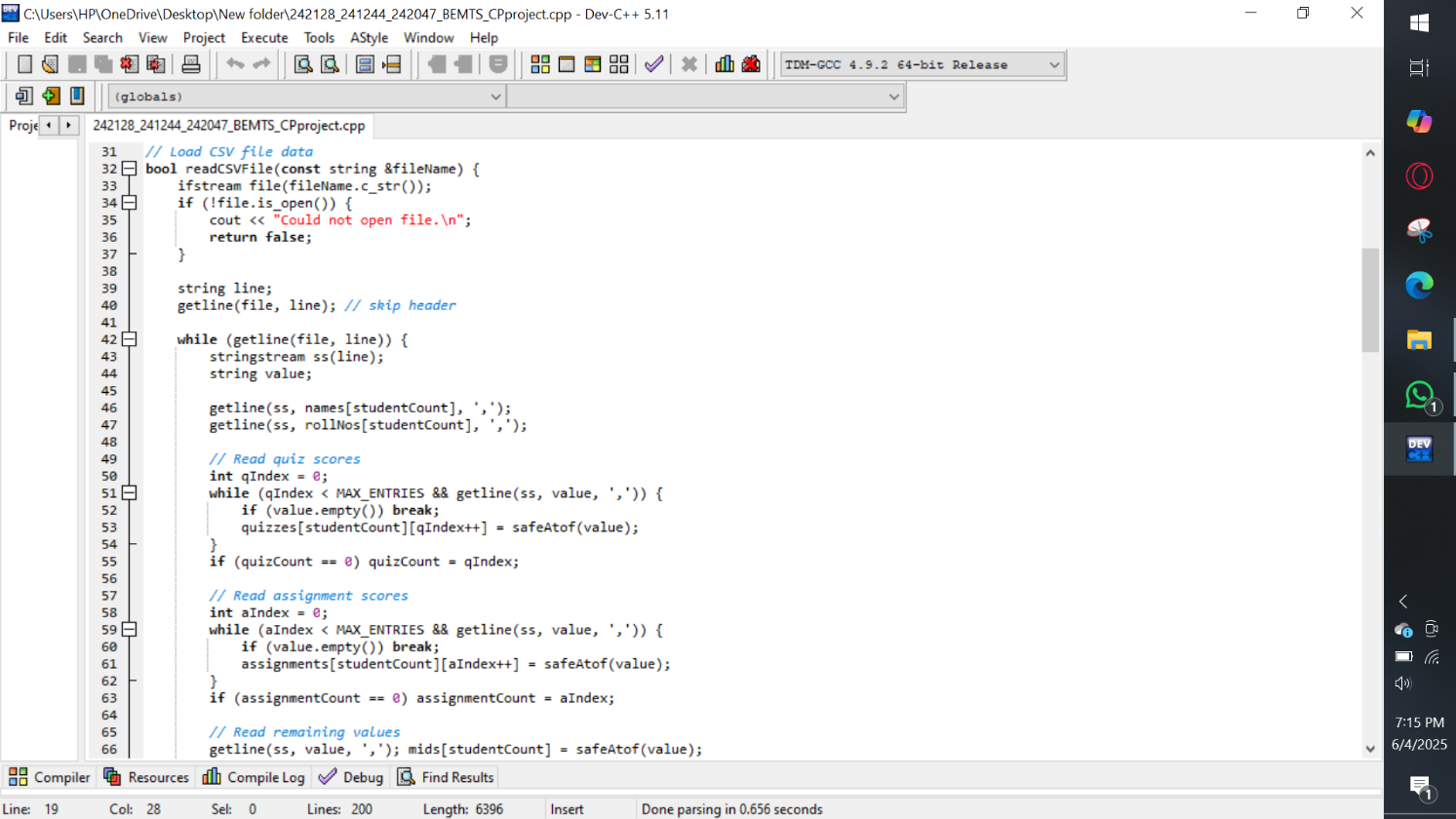
In the era of the digital age, the task of managing student records and assessing the performance of students efficiently and precisely has become an essentiality. This project—Teacher Portal using C++—has the purpose of automating the computation of student grades from raw data files with quiz, assignment, exam, and project marks. With the provision for dynamic user entry of weightages, the program guarantees flexibility and can easily adjust to varying subject frameworks.

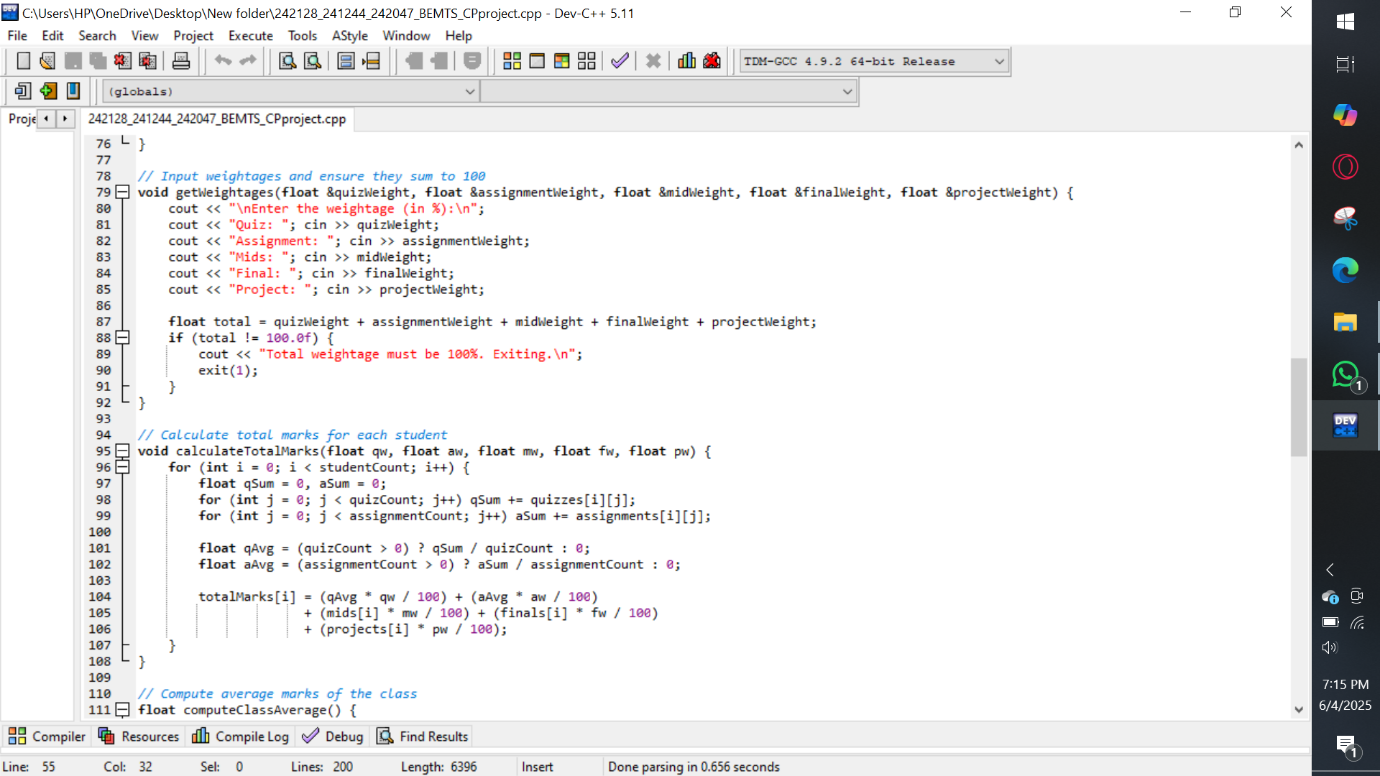
Developed as a part of a Complex Engineering Activity for the Computer Programming Laboratory, this project promotes abstract thinking and real-world problem-solving. It applies fundamental programming constructs like file handling, loops, and conditionals, but with the results being scalable, generic, and grade-aware according to computed class averages. The system provides functionality like class-wide result viewing, student-wise searching, and result exporting in nca.csv and cp.csv format—replicating real administrative portals.

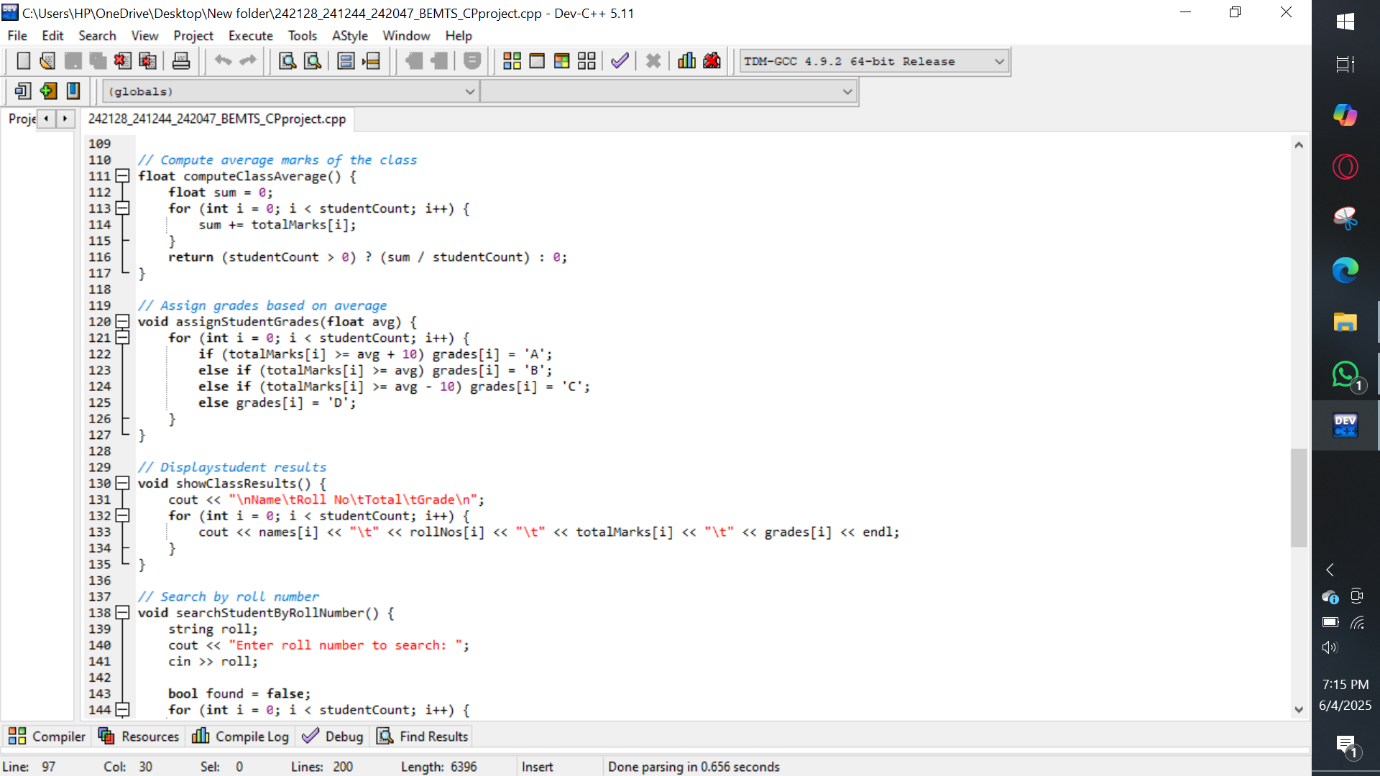
**Implementation:**

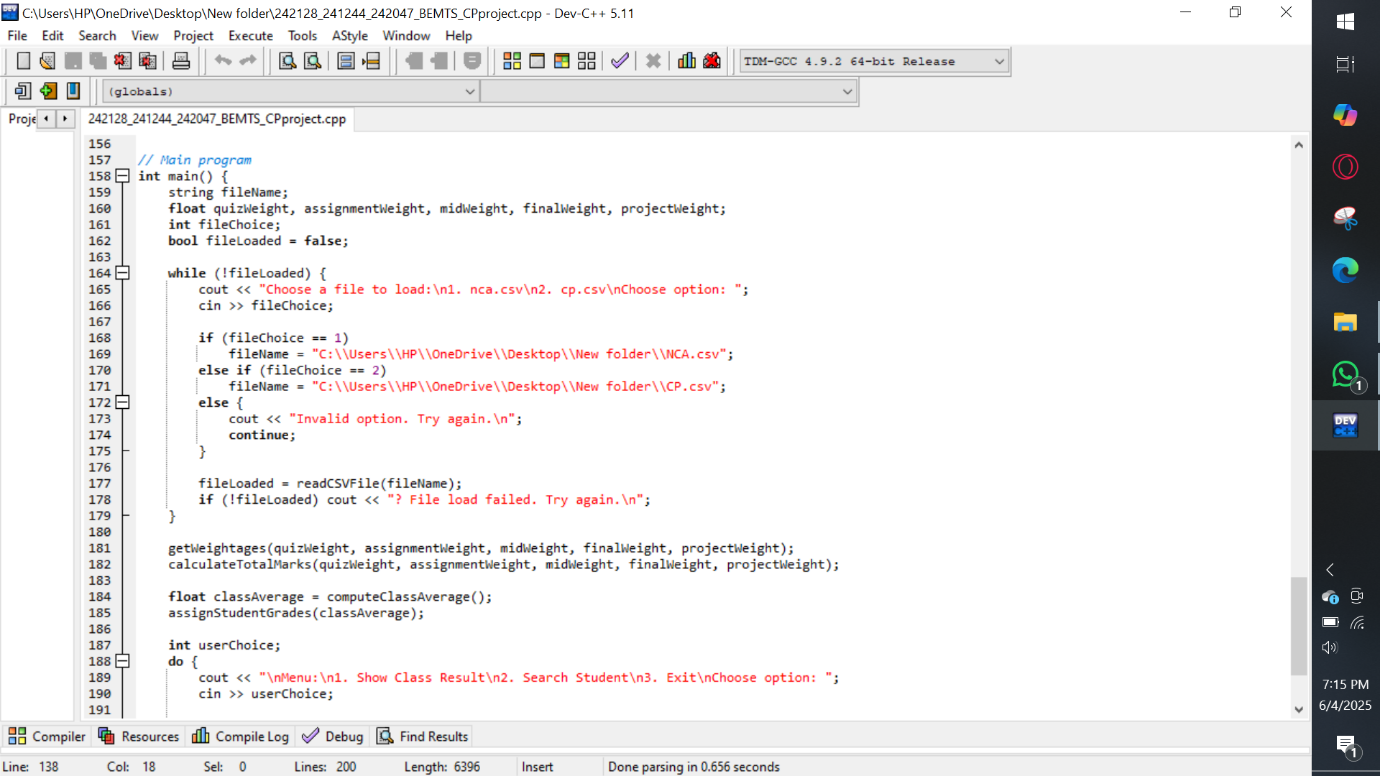
**Screenshots:**

****

****

****

****

****

**Code explanation(working):**

The program works as follows:

1. **File Input:**
   * Prompts the user to choose a CSV file containing student marks.
   * Dynamically detects how many quizzes and assignments are present based on file headers.
2. **User Input for Weightage:**
   * Asks the user to enter percentage weights for quizzes, assignments, midterm, final, and project.
   * Validates that the total weight is 100%.
3. **Data Processing:**
   * Reads marks for each student.
   * Calculates averages for quizzes and assignments.
   * Uses the given formula to compute total weighted marks for each student:

Total Marks=

(Quiz Total/Max Quiz Marks)×Quiz Weight+(Assignment Total/Max Assignment Marks) ×Assignment Weight.

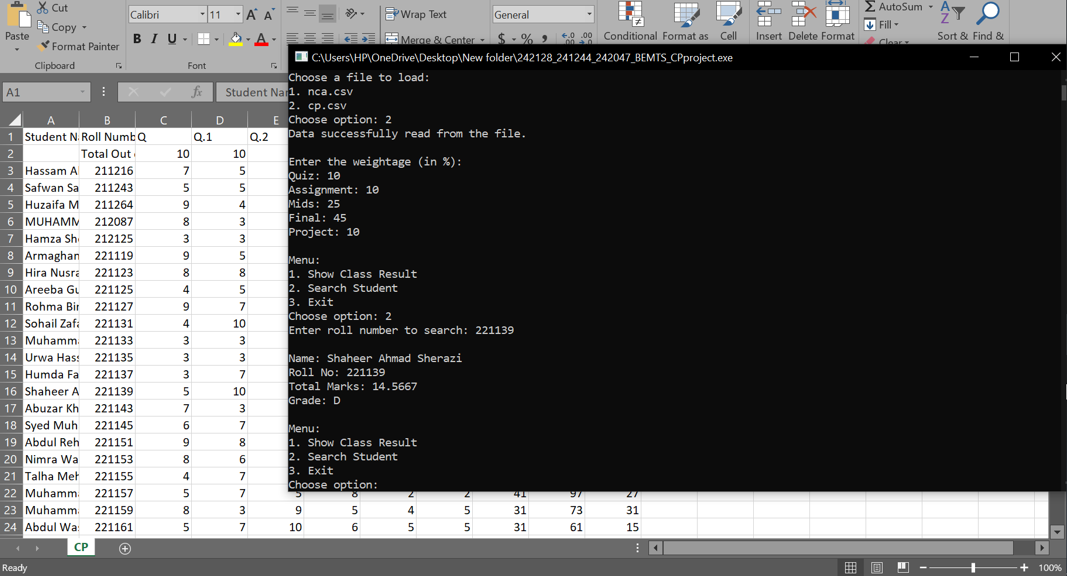
**4.Grading:**

* + Computes the class average.
  + Dynamically defines grading bands with the class average as the centre of a B- range.
  + Assigns grades accordingly (e.g., 59-63 = B-, 64-68 = B, etc.).

**5.User Menu:**

* + Option 1: Show entire class results on the console.
  + Option 2: Search for a student using roll number and show their grade.
  + Option 3: Export the result to a CSV file.
  + Option 4: Exit the program.

**Program outputs:**



**Learning outcomes:**

Through the completion of this project, we learned how to:

* Read and parse data from CSV files with C++.
* Organize our program into manageable sections using functions.
* Develop error checking, like ensuring that the sum of weight we input equals 100%.
* Learn to apply grading rules based on class performance and average scores.
* Enhance our ability to handle user input/output and store results in CSV format.
* Enhance our problem-solving and thinking capabilities, and learn how to write neat and well-documented code.

**References:**

* Bjarne Stroustrup, *The C++ Programming Language*, Addison-Wesley.
* Herbert Schildt, *C++: The Complete Reference*, McGraw-Hill.
* Online C++ documentation: <https://cplusplus.com>
* Lecture notes and lab manuals provided in CP Lab sessions.
* Stack Overflow discussions and community coding solutions for CSV parsing.

**C++ code:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <cstdlib>

using namespace std;

// Constants for limits

const int MAX\_STUDENTS = 100;

const int MAX\_ENTRIES = 20;

// Data arrays

string names[MAX\_STUDENTS];

string rollNos[MAX\_STUDENTS];

float quizzes[MAX\_STUDENTS][MAX\_ENTRIES];

float assignments[MAX\_STUDENTS][MAX\_ENTRIES];

float mids[MAX\_STUDENTS];

float finals[MAX\_STUDENTS];

float projects[MAX\_STUDENTS];

float totalMarks[MAX\_STUDENTS];

char grades[MAX\_STUDENTS];

int studentCount = 0, quizCount = 0, assignmentCount = 0;

// Safely convert string to float

float safeAtof(const string &value) {

return value.empty() ? 0.0f : atof(value.c\_str());

}

// Load CSV file data

bool readCSVFile(const string &fileName) {

ifstream file(fileName.c\_str());

if (!file.is\_open()) {

cout << "Could not open file.\n";

return false;

}

string line;

getline(file, line); // skip header

while (getline(file, line)) {

stringstream ss(line);

string value;

getline(ss, names[studentCount], ',');

getline(ss, rollNos[studentCount], ',');

// Read quiz scores

int qIndex = 0;

while (qIndex < MAX\_ENTRIES && getline(ss, value, ',')) {

if (value.empty()) break;

quizzes[studentCount][qIndex++] = safeAtof(value);

}

if (quizCount == 0) quizCount = qIndex;

// Read assignment scores

int aIndex = 0;

while (aIndex < MAX\_ENTRIES && getline(ss, value, ',')) {

if (value.empty()) break;

assignments[studentCount][aIndex++] = safeAtof(value);

}

if (assignmentCount == 0) assignmentCount = aIndex;

// Read remaining values

getline(ss, value, ','); mids[studentCount] = safeAtof(value);

getline(ss, value, ','); finals[studentCount] = safeAtof(value);

getline(ss, value, ','); projects[studentCount] = safeAtof(value);

studentCount++;

}

file.close();

cout << "Data successfully read from the file.\n";

return true;

}

// Input weightages and ensure they sum to 100

void getWeightages(float &quizWeight, float &assignmentWeight, float &midWeight, float &finalWeight, float &projectWeight) {

cout << "\nEnter the weightage (in %):\n";

cout << "Quiz: "; cin >> quizWeight;

cout << "Assignment: "; cin >> assignmentWeight;

cout << "Mids: "; cin >> midWeight;

cout << "Final: "; cin >> finalWeight;

cout << "Project: "; cin >> projectWeight;

float total = quizWeight + assignmentWeight + midWeight + finalWeight + projectWeight;

if (total != 100.0f) {

cout << "Total weightage must be 100%. Exiting.\n";

exit(1);

}

}

// Calculate total marks for each student

void calculateTotalMarks(float qw, float aw, float mw, float fw, float pw) {

for (int i = 0; i < studentCount; i++) {

float qSum = 0, aSum = 0;

for (int j = 0; j < quizCount; j++) qSum += quizzes[i][j];

for (int j = 0; j < assignmentCount; j++) aSum += assignments[i][j];

float qAvg = (quizCount > 0) ? qSum / quizCount : 0;

float aAvg = (assignmentCount > 0) ? aSum / assignmentCount : 0;

totalMarks[i] = (qAvg \* qw / 100) + (aAvg \* aw / 100)

+ (mids[i] \* mw / 100) + (finals[i] \* fw / 100)

+ (projects[i] \* pw / 100);

}

}

// Compute average marks of the class

float computeClassAverage() {

float sum = 0;

for (int i = 0; i < studentCount; i++) {

sum += totalMarks[i];

}

return (studentCount > 0) ? (sum / studentCount) : 0;

}

// Assign grades based on average

void assignStudentGrades(float avg) {

for (int i = 0; i < studentCount; i++) {

if (totalMarks[i] >= avg + 10) grades[i] = 'A';

else if (totalMarks[i] >= avg) grades[i] = 'B';

else if (totalMarks[i] >= avg - 10) grades[i] = 'C';

else grades[i] = 'D';

}

}

// Displaystudent results

void showClassResults() {

cout << "\nName\tRoll No\tTotal\tGrade\n";

for (int i = 0; i < studentCount; i++) {

cout << names[i] << "\t" << rollNos[i] << "\t" << totalMarks[i] << "\t" << grades[i] << endl;

}

}

// Search by roll number

void searchStudentByRollNumber() {

string roll;

cout << "Enter roll number to search: ";

cin >> roll;

bool found = false;

for (int i = 0; i < studentCount; i++) {

if (rollNos[i] == roll) {

cout << "\nName: " << names[i]

<< "\nRoll No: " << rollNos[i]

<< "\nTotal Marks: " << totalMarks[i]

<< "\nGrade: " << grades[i] << endl;

found = true;

break;

}

}

if (!found) cout << "? Student not found.\n";

}

// Main program

int main() {

string fileName;

float quizWeight, assignmentWeight, midWeight, finalWeight, projectWeight;

int fileChoice;

bool fileLoaded = false;

while (!fileLoaded) {

cout << "Choose a file to load:\n1. nca.csv\n2. cp.csv\nChoose option: ";

cin >> fileChoice;

if (fileChoice == 1)

fileName = "C:\\Users\\HP\\OneDrive\\Desktop\\New folder\\NCA.csv";

else if (fileChoice == 2)

fileName = "C:\\Users\\HP\\OneDrive\\Desktop\\New folder\\CP.csv";

else {

cout << "Invalid option. Try again.\n";

continue;

}

fileLoaded = readCSVFile(fileName);

if (!fileLoaded) cout << "? File load failed. Try again.\n";

}

getWeightages(quizWeight, assignmentWeight, midWeight, finalWeight, projectWeight);

calculateTotalMarks(quizWeight, assignmentWeight, midWeight, finalWeight, projectWeight);

float classAverage = computeClassAverage();

assignStudentGrades(classAverage);

int userChoice;

do {

cout << "\nMenu:\n1. Show Class Result\n2. Search Student\n3. Exit\nChoose option: ";

cin >> userChoice;

if (userChoice == 1) showClassResults();

else if (userChoice == 2) searchStudentByRollNumber();

else if (userChoice != 3) cout << "Invalid choice. Try again.\n";

} while (userChoice != 3);

return 0;

}

**Conclusion:**

This Teacher Portal project closes the gap between programming theory and its practical implementation in school record systems. In reading input from flexible data files, accepting dynamic user-defined parameters, and generating correct outputs with context grading, the program demonstrates the complete capability of C++ in solving real-world issues. As much as technical success, this project also teaches the value of program scalability, user interaction, and automation of systems, which are key ingredients for any software engineer. Students who finish this project are exposed to analytical thinking, modular design, and automated grading systems—all of which prepare them to tackle future projects in software development and academic data management.