

STUDENT GRADING SYSTEM

DONE BY:

- G.NENA(AP22110010163)
- G.OSHITHA(AP22110010174)
- N.POOJITHA(AP22110010187)
- B.PRANATHI(AP22110010138)

CONTENT



*OBJECTIVE OF THIS PROJECT

*CODE SNIPPET

*CODE

*OUTPUT


*CONCLUSION

OBJECTIVE OF THIS PROJECT



Assessing Academic performance

Grading systems are used to evaluate students' academic performance and provide feedback on their understanding of the subject matter.




Providing Clear Feedback

Grades communicate objective feedback to students and parents regarding their individual strengths and challenges, allowing them to understand their progress and set goals for improvement.



Motivating Students

Grading systems can motivate students to study and perform well in their courses, as they provide a clear understanding of their performance and the importance of achieving high marks..



Facilitating Academic Transitions

Grades are essential for students to apply for scholarships, enroll in desired universities, and demonstrate their eligibility for various academic programs.

CODE SNIPPET

✦ Add Student Function:

- Purpose: Adds a new student to the vector of students.
- Input: User inputs the student's name and assignment grades.
- Processing : Validates grades, calculates and stores the final grade.
- Output: Updates the vector with the new student.

✦ Calculate Final Grade Function:

- Purpose: Calculates the final grade based on assignment grades.
- Input: Vector of assignment grades.
- Processing: Computes the average of assignment grades.
- Output: Returns the calculated final grade.

✦ Display Statistics Function:

- Purpose: Displays statistics for all students.
- Input: Vector of students.
- Processing: Calculates and displays total students, average, minimum, and maximum grades.
- Output: Outputs the statistics.

CODE SNIPPET

✦ Export Report Function:

- Purpose: Exports student grades to a specified file.
- Input: Vector of students, filename.
- Processing: Writes student names and final grades to the specified file.
- Output: Outputs success or an error message.

✦ Main Function:

- Purpose: Implements the main program flow and menu.
- Input: User choices for adding a student, displaying statistics, exporting a report, or exiting.
- Processing: Calls corresponding functions based on user choice.
- Output: Continues until the user chooses to exit, providing a menu-driven interface for the program.

CODE

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <iomanip>
#include <limits>
using namespace std;
struct Student {
    string name;
    vector<double> assignmentGrades;
    double finalGrade;
    // Added to store the final grade
};
double calculateFinalGrade(const vector<double>& assignmentGrades); // Function declaration
void addStudent(vector<Student>& students)
{
```

```
Student student;
cout << "Enter student name: ";
cin.ignore();
getline(cin, student.name);
cout << "Enter assignment grades (enter -1 to finish):\n";
double grade;
while (true) {
cout << "Enter grade (-1 to finish): ";
cin >> grade;
if (grade == -1) break;
else if (grade < 0 || grade > 100) {
cout << "Invalid grade. Grades should be between 0 and 100." << endl;
continue; // Ask for the grade again
}
student.assignmentGrades.push_back(grade);
}
// Calculate and store the final grade
student.finalGrade = calculateFinalGrade(student.assignmentGrades);
students.push_back(student);
}
```



```
double calculateFinalGrade(const vector<double>& assignmentGrades)
{
    double total = 0.0;
    for (const double& grade : assignmentGrades) {
        total += grade;
    }
    return (total / assignmentGrades.size());
}

void displayStatistics(const vector<Student>& students)
{
    if (students.empty())
    {
        cout << "No students entered yet." << endl;
        return;
    }
    double totalFinalGrade = 0.0;
    double minGrade = students[0].finalGrade;
    double maxGrade = students[0].finalGrade;
```

```
for (const Student& student : students)
{
double finalGrade = student.finalGrade;
totalFinalGrade += finalGrade;
if (finalGrade < minGrade) minGrade = finalGrade;
if (finalGrade > maxGrade) maxGrade = finalGrade;
}
double averageGrade = totalFinalGrade / students.size();
cout << "Statistics:\n";  cout << "Total Students: " << students.size() << endl;
cout << "Average Grade: " << fixed << setprecision(2) << averageGrade << endl;
cout << "Minimum Grade: " << minGrade << endl;
cout << "Maximum Grade: " << maxGrade << endl;
}
void exportReport(const vector<Student>& students, const string& filename)
{
ofstream outFile(filename);
if (!outFile.is_open())
{
cerr << "Error: Unable to open the file for exporting." << endl;
return;
}
```

```
outFile << "Student Name\tFinal Grade" << endl;
for (const Student& student : students)
{
outFile << student.name << "\t" << student.finalGrade << endl;
}
cout << "Exported student grades to " << filename << endl;
outFile.close();
}
int main()
{
vector<Student> students;
while (true)
{
cout << "Options:\n";
cout << "1. Add Student\n";
cout << "2. Display Statistics\n";
cout << "3. Export Report\n";
cout << "4. Exit\n";
cout << "Enter your choice: ";
```

```
int choice;
cin >> choice;
switch (choice) {
case 1:
addStudent(students);
break;
case 2:
displayStatistics(students);
break;
case 3: {
    string filename;
    cout << "Enter the filename to export: ";
    cin >> filename;
    exportReport(students, filename);
    break;      }
case 4:
return 0;
default:
cout << "Invalid input. Please enter a valid choice." << endl;
cin.clear();
cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
} return 0;
}
```

output

Options:

- 1.Add Student
- 2.Display Statistics
- 3.Export Report
- 4.Exit

Enter your choice:1

Enter student name:rishi

Enter assignment grades (enter -1 to finish):

Enter grade (-1 to finish):7

Enter grade (-1 to finish):-1

Options:

- 1.Add Student
- 2.Display Statistics
- 3.Export Report
- 4.Exit

Enter your choice: 2

Statistics:

Total Students: 1

Average Grade: 7.00

Minimum Grade: 7.00

Maximum Grade: 7.00

CONCLUSION

A well-designed student grading system is essential for fair assessment and academic progress tracking. By providing transparent criteria, timely feedback, and a comprehensive overview of student performance, such a system contributes to a positive learning environment and supports educational growth. Continuous evaluation and adaptation ensure its effectiveness in meeting the evolving needs of students and educators alike.

THANK YOU