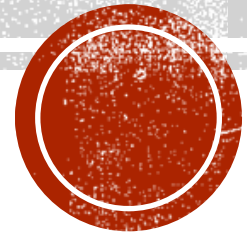


AUTOMATED STUDENT REPORT MANAGEMENT SYSTEM

Presented by: Aryan Kumar

Guide: Ms. Naina Devi (IBM SME)



AGENDA

- Objective
- Topic Outline
- Introduction
- Tools and Technologies
- Project Working
- Applications & Advantages
- Future Scope & GitHub Link



INTRODUCTION

- This project is a C-based application designed to automate academic grading.
- Solves manual calculation errors.
- Processes multi-subject marks for any number of students.
- Provides deep insights into subject-wise performance and grade distribution.



TOOLS AND TECHNOLOGIES USED

- Programming Language: C Programming
- IDE: Visual Studio Code (VS Code)
- Compiler: MinGW (Minimalist GNU for Windows)



PROJECT WORKING

■ Code :

```
1 #include <stdio.h>
2 #include <string.h>
3
4 char getGrade(float avg) {
5     if (avg >= 90) return 'A';
6     else if (avg >= 80) return 'B';
7     else if (avg >= 70) return 'C';
8     else if (avg >= 60) return 'D';
9     else return 'F';
10 }
11
12 int main() {
13     int students, subjects;
14     int i, j;
15
16     char studentName[50];
17     char subjectName[20][50];
18
19     float marks;
20     float subjectSum, studentAvg;
21     float classTotal = 0;
22
23     float highestAvg, lowestAvg;
24     char highestName[50], lowestName[50];
25
26     /* Overall topper */
27     float overallTopTotal = 0;
28     float overallTopAvg = 0;
29     char overallTopName[50];
30
31     int passCount = 0, failCount = 0;
32     int countA = 0, countB = 0, countC = 0, countD = 0, countF = 0;
33 }
```

```
33
34 /* Subject-wise topper */
35 float subjectTopMarks[20];
36 char subjectTopper[20][50];
37
38 /* Subject-wise fail record */
39 char failStudent[100][50]; // store student name
40 char failSubject[100][50]; // store subject name
41 int failIndex = 0;
42
43 /* ----- Input ----- */
44 printf("Enter number of students in class: ");
45 scanf("%d", &students);
46
47 printf("Enter number of subjects: ");
48 scanf("%d", &subjects);
49
50 printf("\nEnter Subject Names:\n");
51 for (i = 0; i < subjects; i++) {
52     printf("Subject %d name: ", i + 1);
53     scanf("%s", subjectName[i]);
54     subjectTopMarks[i] = -1;
55 }
56
57 /* ----- Students Loop ----- */
58 for (i = 1; i <= students; i++) {
59     printf("\nEnter Student %d Name: ", i);
60     scanf("%s", studentName);
61
62     subjectSum = 0;
63
64     for (j = 0; j < subjects; j++) {
65         printf("Enter marks of %s in %s: ", studentName, subjectName[j]);
66         scanf("%f", &marks);
67
68         subjectSum += marks;
69
70         /* Subject-wise topper */
71         if (marks > subjectTopMarks[j]) {
72             subjectTopMarks[j] = marks;
73             strcpy(subjectTopper[j], studentName);
74         }
75 }
```



```

75
76     /* Subject-wise fail check */
77     if (marks < 40) {
78         strcpy(failStudent[failIndex], studentName);
79         strcpy(failSubject[failIndex], subjectName[j]);
80         failIndex++;
81     }
82 }
83
84 studentAvg = subjectSum / subjects;
85 printf("Average of %s: %.2f\n", studentName, studentAvg);
86
87 classTotal += studentAvg;
88
89 /* Pass / Fail */
90 if (studentAvg >= 40)
91     passCount++;
92 else
93     failCount++;
94
95 /* Overall Topper */
96 if (i == 1 || subjectSum > overallTopTotal) {
97     overallTopTotal = subjectSum;
98     overallTopAvg = studentAvg;
99     strcpy(overallTopName, studentName);
100 }
101
102 /* Highest & Lowest Average */
103 if (i == 1) {
104     highestAvg = lowestAvg = studentAvg;
105     strcpy(highestName, studentName);
106     strcpy(lowestName, studentName);
107 } else {
108     if (studentAvg > highestAvg) {
109         highestAvg = studentAvg;
110         strcpy(highestName, studentName);
111     }
112     if (studentAvg < lowestAvg) {
113         lowestAvg = studentAvg;
114         strcpy(lowestName, studentName);
115     }
116 }
117

```

```

118     /* Grade Count */
119     char grade = getGrade(studentAvg);
120     if (grade == 'A') countA++;
121     else if (grade == 'B') countB++;
122     else if (grade == 'C') countC++;
123     else if (grade == 'D') countD++;
124     else countF++;
125 }
126
127 /* ----- Output ----- */
128 printf("\n----- Class Report ----- \n");
129 printf("Class Average: %.2f\n", classTotal / students);
130
131 printf("\nOverall Topper: \n");
132 printf("Name : %s\n", overallTopName);
133 printf("Total Marks : %.2f\n", overallTopTotal);
134 printf("Average Marks : %.2f\n", overallTopAvg);
135
136 printf("\nHighest Average: %.2f (%s)\n", highestAvg, highestName);
137 printf("Lowest Average : %.2f (%s)\n", lowestAvg, lowestName);
138
139 printf("\nPass Students: %d\n", passCount);
140 printf("Fail Students: %d\n", failCount);
141
142 printf("\nSubject-wise Topper: \n");
143 for (i = 0; i < subjects; i++) {
144     printf("%s -> %s (%.2f)\n",
145         subjectName[i],
146         subjectTopper[i],
147         subjectTopMarks[i]);
148 }
149
150 /* Subject wise Fail Report */
151 printf("\nSubject-wise Fail Report: \n");
152 if (failIndex == 0)
153     printf("No student failed in any subject.\n");
154 else {
155     for (i = 0; i < failIndex; i++) {
156         printf("Student: %s failed in %s\n", failStudent[i], failSubject[i]);
157     }
158 }
159
160 printf("\nGrade Distribution: \n");
161 printf("A: %d\n", countA);
162 printf("B: %d\n", countB);
163 printf("C: %d\n", countC);
164 printf("D: %d\n", countD);
165 printf("F: %d\n", countF);
166
167 return 0;
168 }

```



■ Output :

```
Enter Student 1 Name: aryan
Enter marks of aryan in maths: 90
Enter marks of aryan in c: 80
Average of aryan: 85.00

Enter Student 2 Name: divyanshu
Enter marks of divyanshu in maths: 91
Enter marks of divyanshu in c: 80
Average of divyanshu: 85.50

----- Class Report -----
Class Average: 85.25

Overall Topper:
Name      : divyanshu
Total Marks : 171.00
Average Marks : 85.50

Highest Average: 85.50 (divyanshu)
Lowest Average : 85.00 (aryan)

Pass Students: 2
Fail Students: 0

Subject-wise Topper:
maths -> divyanshu (91.00)
c -> aryan (80.00)

Subject-wise Fail Report:
No student failed in any subject.

Grade Distribution:
A: 0
B: 2
C: 0
D: 0
F: 0
```



APPLICATIONS

- Educational Institutions (Schools/Colleges)
- Coaching & Training Centers
- Corporate Training Evaluations
- Automated Performance Tracking Systems



ADVANTAGES

- **High Accuracy:** Eliminates manual math errors.
- **Efficiency:** Processes large datasets instantly.
- **Subject-wise Analysis:** Identifies specific toppers and failures.
- **Comprehensive Reports:** Generates class averages and grade counts.



FUTURE SCOPE

- File Handling: Save data permanently to .txt or .csv files.
- GUI Development: Transition from Console to a Graphic Interface.
- Database Integration: Use SQL to store historical student records.



GITHUB LINK

- Link : <https://github.com/AryanKumar128/Programming-Fundamentals-OF-C>

Thank You

