CS2263
SYSTEM SOFTWARE DEVELOPMENT
FINAL REPORT
Team –14

Divyanshi Kashyap (3769254)
Swati Mehta (3765212)
Muhammad Jahanzaib (3757073)

Features Implemented (Summary)

Student Functionalities: -

- **createStudentNode**():- We have created this functionality so that we can create StudentNode .It Allows dynamic memory usage and easy insertion into the list
- .**addStudent**():- We can create student objects here , basically add students to our management system.It Prevents duplicate student IDs.It calculates the GPA and adds the student to the linked list.
- **deleteStudent**():- We can basically remove students from our student management system
- **modifyStudent**() :- We can basically modify students and upgrade their grades and save them in our student management system
- **freeStudentList**():- Our program basically frees all memory used
- **searchStudent():-** We can basically search for student using student ID
- **calculateStudentEfficiency():-** Our program calculates StudentEfficiency.Based on GPA, prints qualitative efficiency levels.
- **sortStudents():-** Sorts our list bases of GPA or Name or Student ID in ascending order

Data Visualization:-

- **displayStudent():-** This function displays data in tabular form.It prints each student's ID, name, and GPA in a consistent layout, making it easy for users to view and compare multiple students at once.
- **visualizeStudent():-** This basically is sub option under bar visualization so that we can get bar chart of students on bases of their CGPA.It prints a textual "bar" next to each student's name and GPA, using color coding to differentiate GPA levels (e.g., green for high GPAs, red for low).

- **printCourseBarChart():-** This basically displays bar chart visualizing the grades of students for each course.It creates a bar chart for each course using colored bars that represent grade performance visually.

FileI/0 functionalities:-

- **ReadCSV():-** Reads student data from a CSV file and loads it into the linked list. It extracts student information such as ID, name, and grades, calculates the GPA, and creates nodes to represent each student. If the file doesn't exist, it creates an empty one.
- **WriteCSV():-** Writes the student data from the linked list back to the CSV file. Whenever a student is added, modified, or deleted, this function is called to ensure that the CSV file is updated accordingly.

GPA Calculation:-

- **calculateGPA():-** Calculates the GPA based on the student's grades. It converts each grade into a corresponding grade point value.This is used throughout the system in sorting, efficiency analysis, and display functions.

User Interface:-

- **displayTitleScreen():-** Functionality basically Displays a title screen with project information. It centers the title and author information using padding and prompts the user to press any key to continue.
- **showMenu():-** Functionality basically show menu and asks user to input a choice.

Use of Key Programming Concepts&Examples

KEY STRUCTURES USED

- **Student Struct:-** This is Structure we have created basically to hold information about students and their details. It takes each student’s ID, name, an array of grades, a calculated GPA, and the number of grades they have entered

```c
typedef struct Student {
    int id;
    char name[MAX_NAME_LENGTH];
    float grades[MAX_GRADES];
    int numGrades;
    float gpa;
} Student;
```

- **StudentNode Struct :-** This structure represents a node in a linked list.It holds a Student struct as data and pointer next to move to next node allowing dynamic storage and management of multiple students.

```
typedef struct StudentNode {
    Student data;
    struct StudentNode* next;
} StudentNode;
```

- **StudentNode** head :-** This is bascially a pointer to pointer of StudentNode, we are using it so that we can change what head is pointing towards

Example

```
void readCSV(StudentNode** head) {
```

- **const char* courseCodes[MAX_GRADES] :-** This is an array we have created to store name of courses. It is used throughout the program to label grades and display course-specific information.

```
const char* courseCodes[MAX_GRADES] = {"CS2043", "CS2263", "CS2253", "CS2613", "CS2614"};
```

- **Linked List:-** This code uses single linked list and bascially helps us to store and manage student object and head pointer basically point to first pointer in list

Memory Allocation:-

Here we are using malloc() so that we can dynamically allocate memory for StudentNode object. Dynamic allocation allows to grow the list as needed without wasting memory.
StudentNode* newNode = malloc(sizeof(StudentNode));
Similarly, while reading from the CSV file in readCSV() each student record is loaded into memory using:
StudentNode* node = (StudentNode*)malloc(sizeof(StudentNode));
Here we are using free() so that we can dynamically free the memory in freeStudentList(). This ensures that all memory dynamically allocated during the program is properly released before the program exits.

Debugging and Memory Management

Tool Basically command line were used to verify our memory leak and checks were

valgrind --leak-check=full --show-leak-kinds=all ./main

This is our output when we run Valgrind command on terminal to check StudentNode** leaks and after going through all functionalities, when we Exit out command, we get a message that there are no Errors, and all Heap are which were allocated were freed
as shown in images
we did it couple of time and we got same result

```
k3h5z@remotelabm01 ~/CS]$
k3h5z@remotelabm01 ~/CS]$ valgrind --leak-check=full --show-leak-kinds=all ./ma
n
=3466428== Memcheck, a memory error detector
=3466428== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
=3466428== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
=3466428== Command: ./main
=3466428==
            ========================================
             2263 Project: Student_Management_System
         MADE BY DIVYANSHI, Muhammad Jahanzib, Swati Mehta
            ========================================
                    Press any key to continue...
```

```
Enter your choice: 10
Exiting program. Data saved.
Wow memory Freed is successfull.
==3466428==
==3466428== HEAP SUMMARY:
==3466428==     in use at exit: 0 bytes in 0 blocks
==3466428==   total heap usage: 7 allocs, 7 frees, 19,472 bytes allocated
==3466428==
==3466428== All heap blocks were freed -- no leaks are possible
==3466428==
==3466428== For lists of detected and suppressed errors, rerun with: -s
==3466428== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[k3h5z@remotelabm01 ~/CS]$
```

Testing and Validation

For testing and Validation for our project we have created a new test_main.c class and got an output test main
We have tested all our functionalities in test_main.c and here is breakdown for the same

For example:- this is our test_addStudent() functionality

```c
void test_addStudent() {
    StudentNode* head = NULL;
    Student newStudent;
    newStudent.id = 1;
    strncpy(newStudent.name, "Test Student", MAX_NAME_LENGTH - 1);
    newStudent.name[MAX_NAME_LENGTH - 1] = '\0';
    float grades[MAX_GRADES] = {90, 80, 70, 60, 50};
    for (int i = 0; i < MAX_GRADES; i++) {
        newStudent.grades[i] = grades[i];
    }
    newStudent.numGrades = MAX_GRADES;
    newStudent.gpa = calculateGPA(grades, MAX_GRADES);

    StudentNode* newNode = malloc(sizeof(StudentNode));
    if (newNode == NULL) {
        perror("Memory allocation failed");
        return;
    }

    newNode->data = newStudent;
    newNode->next = head;
    head = newNode;

    if (head != NULL && head->data.id == 1 && strcmp(head->data.name, "Test Student") == 0) {
        printf("Test Case Passed: Add student successfully\n");
    } else {
        printf("Test Case Failed: Add student successfully\n");
    }

    freeStudentList(&head);
}
```

We basically add a student here and if student was added, our terminal would return us Test

Case Passed message

```
Suspended
[k3h5z@gd120m06 ~/CS]$ gcc test_main.c student.c -o test_main
[k3h5z@gd120m06 ~/CS]$ ./test_main
Test Case Passed: Add student successfully
Wow memory Freed is successfull.
Student with ID 1 deleted successfully.
Test Case Passed: Delete student successfully
```

Exactly same progress goes with other functionalities and here is output for same

```
[k3h5z@gd120m06 ~/CS]$ gcc test_main.c student.c -o test_main
[k3h5z@gd120m06 ~/CS]$ ./test_main
Test Case Passed: Add student successfully
Wow memory Freed is successfull.
Student with ID 1 deleted successfully.
Test Case Passed: Delete student successfully
Wow memory Freed is successfull.
Test Case Passed: Search student successfully
Test Case Passed: Search student not found
Wow memory Freed is successfull.
Testing GPA Calculation:
GPA1 Got: 3.86 (Expected: Varies based on grading scale)
GPA2 Got: 4.30 (Expected: Varies based on grading scale)
GPA3 Got: 3.06 (Expected: Varies based on grading scale)
Testing displayTable:

+-------+----------------------+-------+
|  ID   |         Name         |  GPA  |
+-------+----------------------+-------+
|     1 | Kareena khan         |  3.86 |
+-------+----------------------+-------+
Wow memory Freed is successfull.
Testing calculateStudentEfficiency:

--- Student Efficiency Based on GPA ---
Student: Beyonce (ID: 1)
GPA: 3.86
Efficiency: Very Good

Wow memory Freed is successfull.
Testing sortStudents (by Name):
  Test Passed: Name sorting
Wow memory Freed is successfull.
Testing BarChartVisualization:
Student: Test Student (ID: 1)
CS2043 | ################## | (95)
CS2263 | ################   | (85)
CS2253 | ##############     | (75)
CS2613 | ############       | (65)
CS2614 | ##########         | (55)


ALL TESTS COMPLETED
[k3h5z@gd120m06 ~/CS]$
```

We did all major functionalities for our code .All tests were passed

Challenges Faced and Lessons Learned

Our Group did face alot of challenges here are few listed

Memory leaks:- During first phase of our Project, whenever we used to run valgrind we used to encounter with lot of memory leaks and erros but eventually we made sure that all are functions get memory free and eventually we were able to run successful Valgrind test and when we Exit our choice, we get a Error Message:- Denoting that we have 0 error messages

ASCII-Bar Chart:- We didnt have clue, excactly how to get colors on our terminal and get our code executed so, we refered form Internet regarding that and defined colors on Top of class as shown in image

```
#define RED "\x1B[31m"
#define GRN "\x1B[32m"
#define YEL "\x1B[33m"
#define BLU "\x1B[34m"
#define MAG "\x1B[35m"
#define CYN "\x1B[36m"
#define WHT "\x1B[37m"
#define RESET "\x1B[0m"
```

This is in our Student.c

We didn’t want our management system to AddStudents with same StudentID so we created AddStudent Method keeping that in mind, code ensures that we don’t input same Student ID

```
// we have created this method to check for duplicate ID and prompt an error message on invaild input
for (StudentNode* temp = *head; temp != NULL; temp = temp->next) {
    if (temp->data.id == newStudent.id) {
        printf("Error: Student with ID %d already exists.\n", newStudent.id);
        return;
    }
}
```