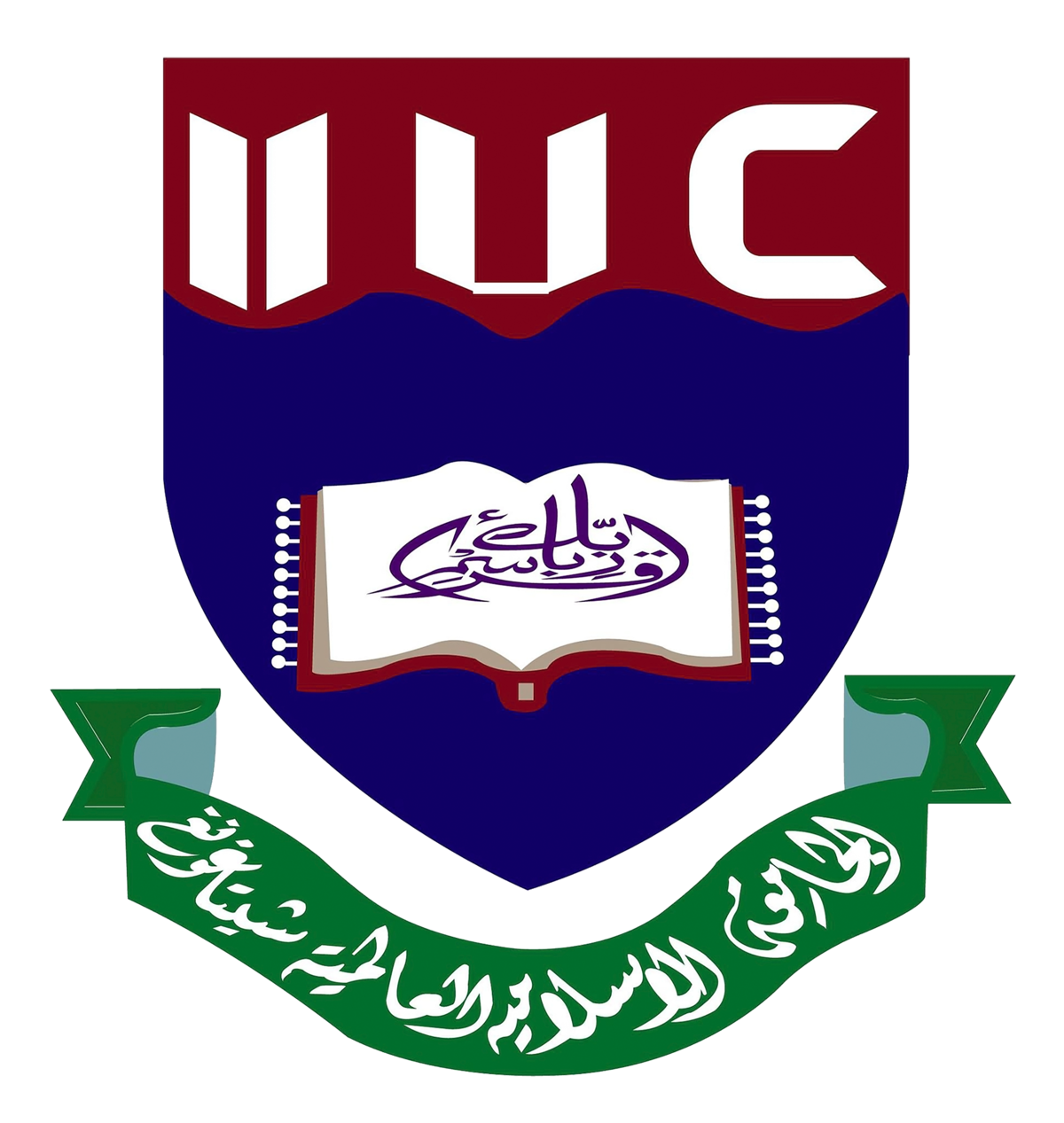
**** 

**PROJECT**

**আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম**

**وفجامعة الوسومية العالمية شيت فونغ**

**International Islamic University Chittagong**

**COURSE CODE : CSE – 2322**

**COURSE TITLE : DATA STRUCTURES LAB**

**SUBMITTED TO : Ashifatul Ferdousi**

**Assistant Lecturer,Dept. of CSE, IIUC.**

**SUBMITTED BY :**

**TEAM NAME : THE HASHTAG**

**MEMBERS :**

**1. AFIA IBNAT­­\_C231516**

**2. SABIHA AHMED TASNIM TINNI\_C231542**

**SEMESTER : 3rd**

**SECTION : 3DF**

**ENGINEERING**

**DEPARTMENT : COMPUTER SCIENCE AND**

**REMARK:**

**DATE OF SUBMISSION : 27.06.2024**

**​** ​

**INTERNATIONAL ISLAMIC UNIVERSITY CHATTOGRAM**



**DEPARTMENT OF COMPUTER SCIeNCE AND ENGINEERING**

**Project Title**

**CLASSROOM MANAGEMENT SYSTEM**

**Supervisor**

**Team Name**

**Ashifatul Ferdousi**

**Assistant Lecturer**

**Dept. of CSE, IIUC.**

**THE**​

**HASHTAG**​

**Overview**

**Introduction.**

**Objectives.**

**Research Background(Previous Research).**

**Methodology(Tools,Language,Algorithm).**

**Code Snippet & Output.**

**Result & Discussion.**

**Limitations.**

**Future Work.**

**Conclusion.**

**Introduction**

This project handles the management of a classroom . It allows users to add, display, and delete student information, manage class schedules, and conduct a quiz for assessment. The system provides a menu-driven interface for easy navigation and interaction.



**Objectives**

To make a suitable and secure system for classroom’s management.

To upgrade from a paper-based system to computerized system.

To lessen the risk of losing records.

**Research Background**

Linked Lists and Their Applications:

<https://www.programiz.com/dsa/linked-list>

<https://www.geeksforgeeks.org/introduction-to-linked-list-data-structure-and-algorithm-tutorial/>

For Project:

<https://www.researchgate.net/publication/284171115_Classroom_Management_in_Project_Work>

<https://www.researchgate.net/publication/351819204_Classroom_management>

<https://youtu.be/u8kUwpO3ucw>

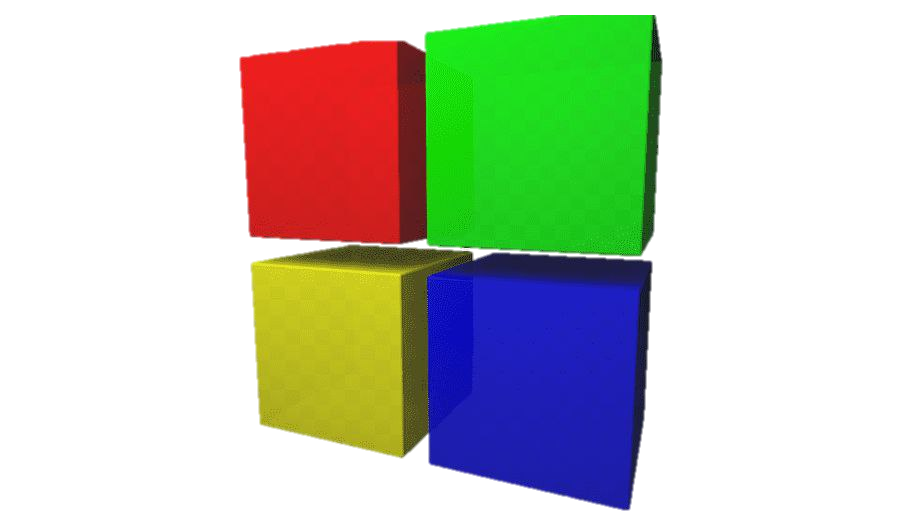
**Methodology**

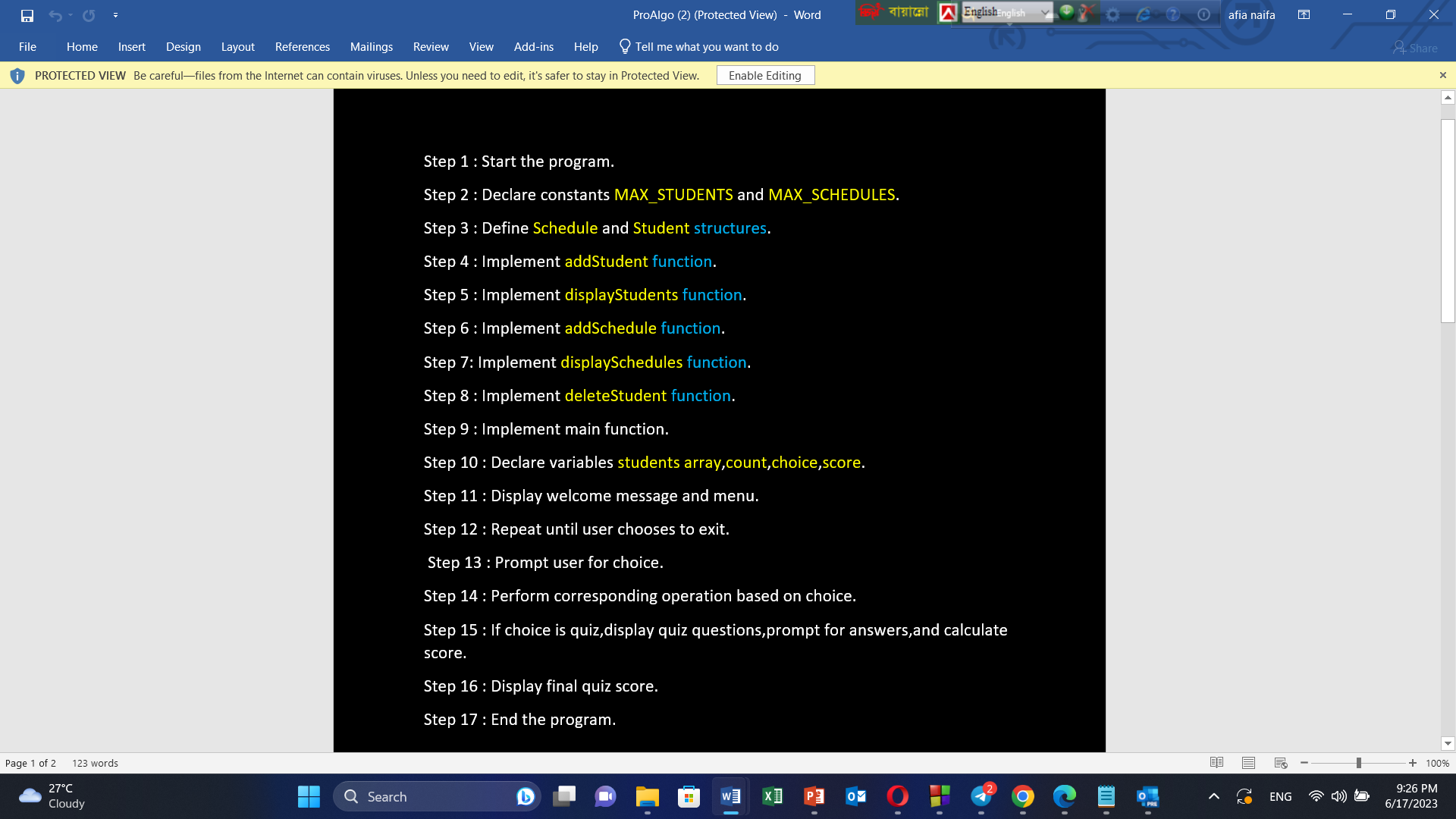
**ALGORITHM**

**LANGUAGE :**

Here we use C++ programming language.

**PLATFORM :**

Codeblocks



**CODE**

**case 5:**

**deleteStudentLinkedList(headLinkedList);**

**break;**

**case 6:**

**deleteScheduleLinkedList(headLinkedList);**

**break;**

**case 7:**

**cout << "Exiting the program." << endl;**

**break;**

**default:**

**cout << "Invalid choice!" << endl; }**

**} while (choice != 7);**

**Node\* current = headLinkedList;**

**Node\* temp;**

**while (current != nullptr) {**

**temp = current;**

**current = current->next;**

**delete temp; }**

**return 0; }**

**void addStudentLinkedList(Node\*& head, int& count){**

**Node\* newNode = new Node;**

**if (newNode == nullptr) {**

**cout << "Memory allocation failed." << endl;**

**return; }**

**cout << "Enter student name: ";**

**cin.ignore();**

**cin.getline(newNode->data.name, 50);**

**cout << "Enter student email: ";**

**cin >> newNode->data.email;**

**cout << "Enter student mobile: ";**

**cin >> newNode->data.mobile;**

**cout << "Enter student id: ";**

**cin >> newNode->data.id;**

**newNode->data.scheduleCount = 0;**

**newNode->next = nullptr;**

**if (head == nullptr) {**

**head = newNode;**

**} else {**

**Node\* temp = head;**

**while (temp->next != nullptr) {**

**temp = temp->next; }**

**temp->next = newNode; }**

**count++;**

**cout << "Student added successfully to linked list." << endl; }**

**void displayStudentsLinkedList(Node\* head) {**

**// }**

**void addScheduleLinkedList(Node\* head) {**

**// }**

**void displaySchedulesLinkedList(Node\* head) {**

**// }**

**void deleteStudentLinkedList(Node\*& head) {**

**// }**

**void deleteScheduleLinkedList(Node\* head) {**

**// }**

**#include<iostream>**

**#include<cstring>**

**using namespace std;**

**#define MAX\_STUDENTS 100**

**#define MAX\_SCHEDULES 5**

**struct Schedule {**

**char day[20];**

**char time[20];**

**char course[50];**

**char instructor[50]; };**

**struct Student {**

**char name[50];**

**char email[20];**

**int mobile;**

**char id[10];**

**Schedule schedules[MAX\_SCHEDULES];**

**int scheduleCount; };**

**struct Node {**

**Student data;**

**Node\* next; };**

**void addStudentLinkedList(Node\*& head, int& count);**

**void displayStudentsLinkedList(Node\* head);**

**void addScheduleLinkedList(Node\* head);**

**void displaySchedulesLinkedList(Node\* head);**

**void deleteStudentLinkedList(Node\*& head);**

**void deleteScheduleLinkedList(Node\* head);**

**int main() {**

**Node\* headLinkedList = nullptr;**

**int countLinkedList = 0, choice;**

**cout << "...........INTERNATIONAL ISLAMIC UNIVERSITY CHATTOGRAM............\n\n";**

**cout << "\n\t\_\_\_\_Faculty of Science And Engineering\_\_\_\_\n\n";**

**cout << "\n\t\t\_\_\_\_Department of CSE\_\_\_\_\n\n";**

**cout << "\n\tHOME PAGE";**

**cout << "\n=========================\n\n";**

**do {**

**cout << "\nClassroom Management System" << endl;**

**cout << "1. Add Student" << endl;**

**cout << "2. Display Students" << endl;**

**cout << "3. Add Schedule" << endl;**

**cout << "4. Display Schedules" << endl;**

**cout << "5. Delete Student" << endl;**

**cout << "6. Delete Schedule" << endl;**

**cout << "7. Exit" << endl;**

**cout << "Enter your choice: ";**

**cin >> choice;**

**switch (choice) {**

**case 1:**

**addStudentLinkedList(headLinkedList, countLinkedList);**

**break;**

**case 2:**

**displayStudentsLinkedList(headLinkedList);**

**break;**

**case 3:**

**addScheduleLinkedList(headLinkedList);**

**break;**

**case 4:**

**displaySchedulesLinkedList(headLinkedList);**

**break;**

deleteStudentLinkedList(headLinkedList);

break;

case 7:

cout << "Exiting the program." << endl;

break;

default:

cout << "Invalid choice!" << endl;

}

} while (choice != 7);

// Clean up allocated memory for linked list nodes

Node\* current = headLinkedList;

Node\* temp;

while (current != nullptr) {

temp = current;

current = current->next;

delete temp;

}

return 0;

}

// Linked list functions

void addStudentLinkedList(Node\*& head, int& count) {

Node\* newNode = new Node;

if (newNode == nullptr) {

cout << "Memory allocation failed." << endl;

return;

}

cout << "Enter student name: ";

cin.ignore(); // Ignore newline character in buffer

cin.getline(newNode->data.name, 50);

cout << "Enter student email: ";

cin >> newNode->data.email;

cout << "Enter student mobile: ";

cin >> newNode->data.mobile;

cout << "Enter student id: ";

cin >> newNode->data.id;

newNode->data.scheduleCount = 0; // Initialize schedule count

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

} else {

Node\* temp = head;

while (temp->next != nullptr) {

temp = temp->next;

}

temp->next = newNode;

}

count++;

cout << "Student added successfully to linked list." << endl;

}

void displayStudentsLinkedList(Node\* head) {

if (head == nullptr) {

cout << "No students in the linked list." << endl;

return;

}

cout << "\n";

cout << "|==============================|" << endl;

cout << "| CLASSROOM STUDENTS |" << endl;

cout << "|==============================|" << endl;

cout << "\n";

Node\* temp = head;

while (temp != nullptr) {

cout << "Name: " << temp->data.name << endl;

cout << "Email: " << temp->data.email << endl;

cout << "Mobile: " << temp->data.mobile << endl;

cout << "ID: " << temp->data.id << endl;

cout << "==============================" << endl;

temp = temp->next;

}

cout << "\n";

}

void addScheduleLinkedList(Node\* head) {

char studentId[10];

cout << "Enter student ID: ";

cin >> studentId;

Node\* current = head;

while (current != nullptr) {

if (strcmp(current->data.id, studentId) == 0) {

if (current->data.scheduleCount >= MAX\_SCHEDULES) {

cout << "Maximum number of schedules reached for this student." << endl;

return;

}

Schedule newSchedule;

cout << "Enter class day: ";

cin >> newSchedule.day;

cout << "Enter class time: ";

cin >> newSchedule.time;

cout << "Enter course name: ";

cin >> newSchedule.course;

cout << "Enter instructor name: ";

cin >> newSchedule.instructor;

current->data.schedules[current->data.scheduleCount] = newSchedule;

current->data.scheduleCount++;

cout << "Schedule added successfully." << endl;

return;

}

current = current->next;

}

cout << "Student with ID " << studentId << " not found." << endl;

}

void displaySchedulesLinkedList(Node\* head) {

char studentId[10];

cout << "Enter student ID: ";

cin >> studentId;

Node\* current = head;

while (current != nullptr) {

if (strcmp(current->data.id, studentId) == 0) {

if (current->data.scheduleCount == 0) {

cout << "No schedules found for this student." << endl;

return;

}

cout << "\nSchedules for student " << current->data.name << ":" << endl;

for (int i = 0; i < current->data.scheduleCount; i++) {

cout << "Day: " << current->data.schedules[i].day << endl;

cout << "Time: " << current->data.schedules[i].time << endl;

cout << "Course: " << current->data.schedules[i].course << endl;

cout << "Instructor: " << current->data.schedules[i].instructor << endl;

cout << "==============================" << endl;

}

return;

}

current = current->next;

}

cout << "Student with ID " << studentId << " not found." << endl;

}

void deleteStudentLinkedList(Node\*& head) {

char studentId[10];

cout << "Enter student ID to delete: ";

cin >> studentId;

Node\* current = head;

Node\* prev = nullptr;

while (current != nullptr) {

if (strcmp(current->data.id, studentId) == 0) {

if (prev == nullptr) {

// If the student to be deleted is the head node

head = current->next;

} else {

prev->next = current->next;

}

delete current;

cout << "Student with ID " << studentId << " deleted successfully." << endl;

return;

}

prev = current;

current = current->next;

}

cout << "Student with ID " << studentId << " not found." << endl;

}

void deleteScheduleLinkedList(Node\* head) {

char studentId[10];

cout << "Enter student ID: ";

cin >> studentId;

Node\* current = head;

while (current != nullptr) {

if (strcmp(current->data.id, studentId) == 0) {

if (current->data.scheduleCount == 0) {

cout << "No schedules found for this student." << endl;

return;

}

int scheduleIndex;

cout << "Enter the index of the schedule to delete (0 to " << current->data.scheduleCount - 1 << "): ";

cin >> scheduleIndex;

if (scheduleIndex < 0 || scheduleIndex >= current->data.scheduleCount) {

cout << "Invalid schedule index." << endl;

return;

}

for (int i = scheduleIndex; i < current->data.scheduleCount - 1; i++) {

current->data.schedules[i] = current->data.schedules[i + 1];

}

current->data.scheduleCount--;

cout << "Schedule at index " << scheduleIndex << " deleted for student with ID " << studentId << endl;

return;

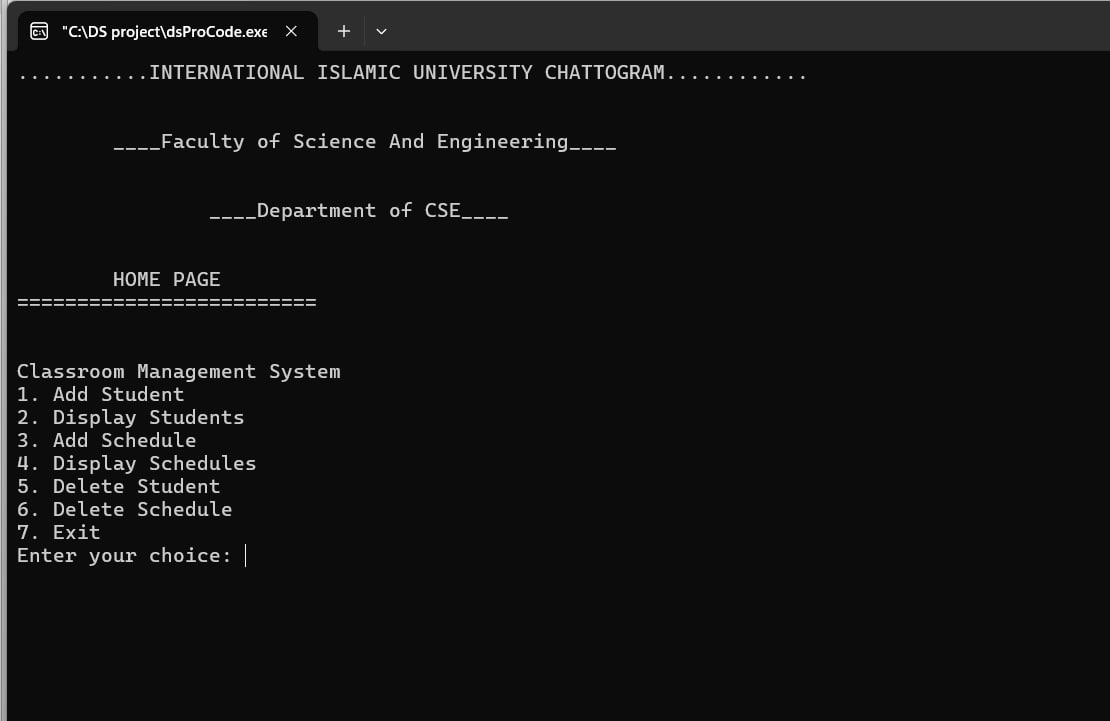
}

current = current->next;

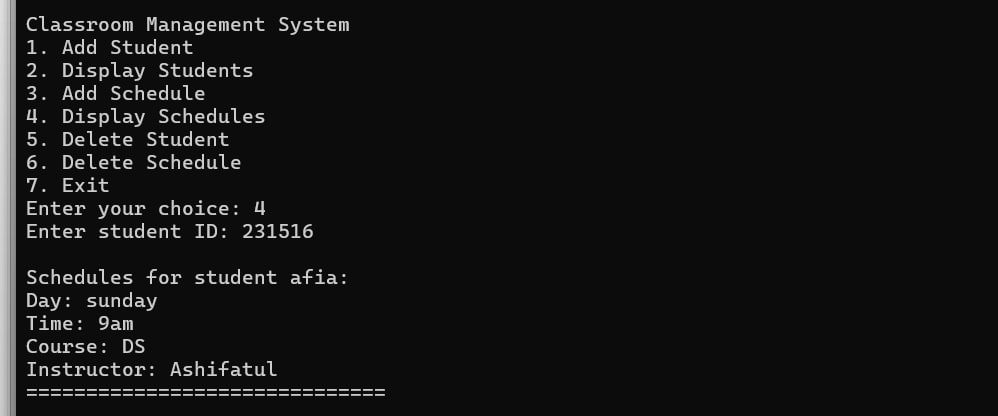
}

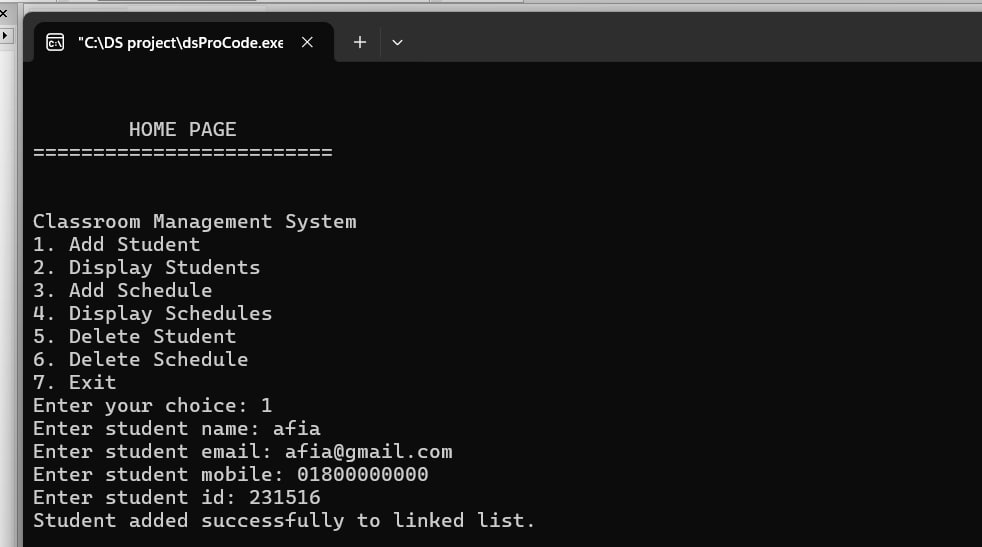
cout << "Student with ID " << studentId << " not found." << endl;

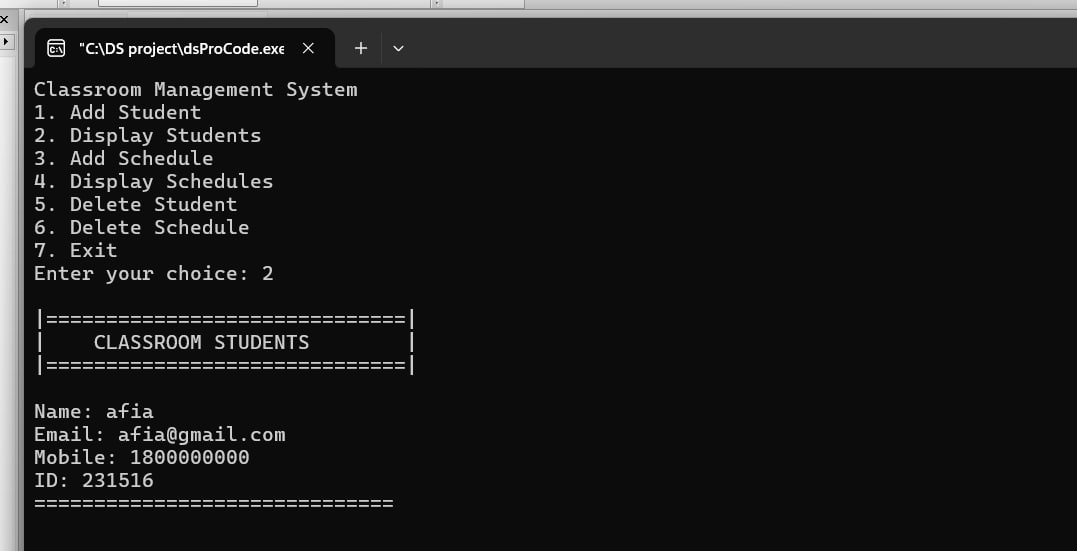
}

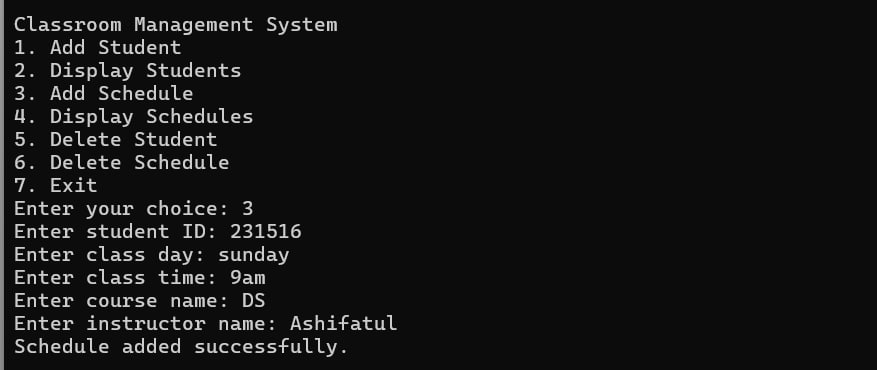


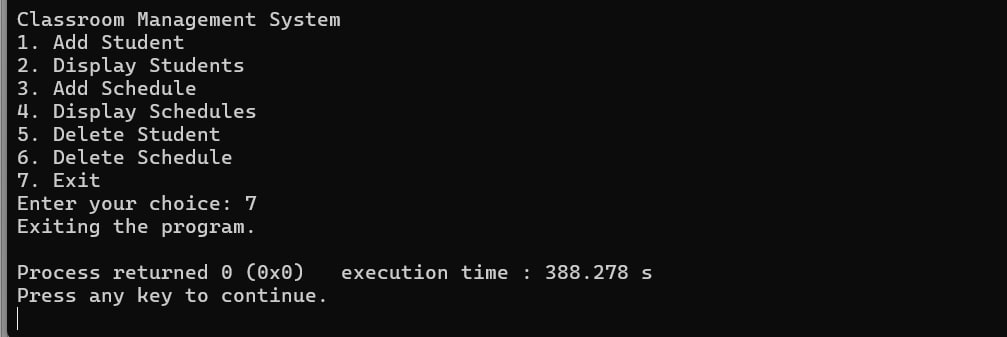
**OUTPUT**

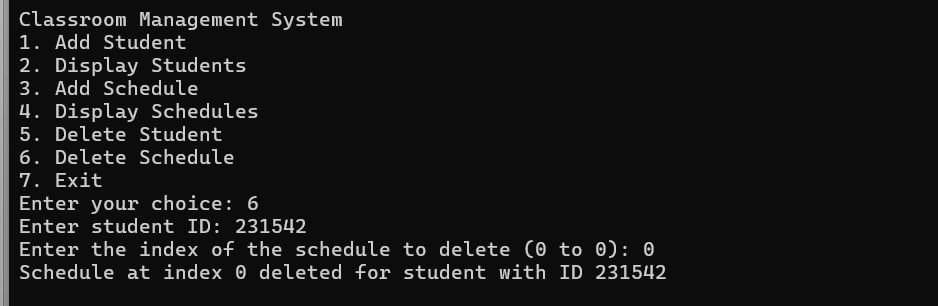


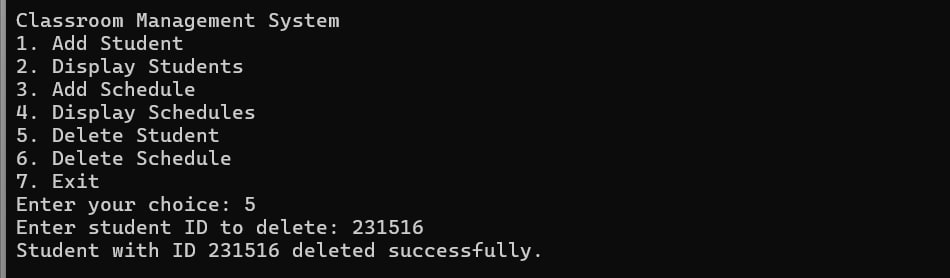












**RESULT & DISCUSSION**

**Result:**

The system provides an interactive way to manage student records and their schedules through a console interface, allowing for adding, displaying, and deleting both student information and their associated schedules.

**Discussion:**

This program provides a basic classroom management system with functionalities to add and manage students and their schedules using a linked list. Each student can have up to a defined maximum number of schedules. The program allows for dynamic addition, display, and deletion of both students and their schedules.

**LIMITATIONS**

**Scalability: May struggle with large student populations, limiting suitability for bigger institutions.**

**Limited Customization: May not adapt well to unique educational needs, reducing workflows or preferences.**



Enhanced Data Analytics : Leveraging advanced analytics to gain valuable insights from student performance data for informed decision-making and personalized learning experiences.

Seamless Integration : Integrating with existing educational systems for seamless data exchange, streamlined workflows, and enhanced efficiency in classroom management.

Mobile Accessibility : Developing a mobile application to provide on-the-go access to information, schedules, assignments, and notifications, fostering better communication and engagement among teachers, students, and parents.

THE END

**This project simplifies classroom management with a computerized system, enabling easy access to information and efficient student handling. However, it has scalability and customization limitations. Overall, it lays the groundwork for improving classroom operations.**

