

**North South University**

Intermediate Project Report

Project Name: Campus Event Management System using Linked List

Course Name: CSE225

Section: 10

Date: 28/04/2024

**Submitted To:**

Mohammad Rezwanul Huq (MRH1)

**Submitted By:**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Abdur Rahman Galib | 2231208642 |
| Wahidul Islam Ziad | 2231985642 |
| Tanzidur Rahman | 2232075642 |
| Omor Faruq | 2231568642 |

Table of Content

1. Introduction ---------------------------------------------------------------------------3
   1. Project Overview -------------------------------------------------------------3
   2. Objectives ---------------------------------------------------------------------3
2. Methodology --------------------------------------------------------------------------3
   1. Event operation ---------------------------------------------------------------4
   2. Attendee Management -------------------------------------------------------4
   3. Database management -------------------------------------------------------4
3. Runtime complexity ---------------------------------------------------------- (5 – 8)
4. Screenshot of the execution -------------------------------------------------- (9-15)
5. Future Scope ------------------------------------------------------------------------ 16

1. Introduction

1. a. Project Overview:

In this project, we created an event management system using C++ with Linked List structure. The objective is to let the users manage their events easily. Our project allows users to create, update, delete and search for events. It also facilitates organizing attendees and save event details in one place.

1. b. Project Objective:

Our goal of this project is:

* Allow users to create events
* Allow users to update their particular event
* Allow users to search for an event
* Allow users to see if their event conflicts with any existing event
* Allow users to delete events
* Allow users to add, delete and view attendees for their events
* Allow user to save event information

2. Methodology

This program has the following components: Event.h and EventLinkedList.h, associated with Event.cpp and EventLinkedList.cpp respective implementation files. Further, there is a main.cpp file containing the main function. The command line interface (CLI) approach is used for building the user interface.

2. a. Event operation:

For this operation, the following methods are:

* void EventLinkedList::insertEvent(Event &e)
* void EventLinkedList::deleteEvent(int eventId)
* void EventLinkedList::searchEventByTitle(string &title)
* void EventLinkedList::updateEvent(int eventId)
* void EventLinkedList::displayAllEvents()
* void Event::displayEventDetails()
* bool EventLinkedList::hasScheduleConflict(Event &newEvent)

2. b. Attendee Management

For managing Attendee, the following methods are:

* void Event::addAttendee(string attendee)
* void Event::removeAttendee(string &attendee)
* bool Event::hasAttendee(string &attendee)
* void Event::displayAttendees()
* void EventLinkedList::updateEvent(int eventId)
* void EventLinkedList::manageAttendees(int eventId)

2. c. Database management

In this program, we saved all the event information into the “events.txt” file. For this operation, the following methods are:

* void EventLinkedList::saveEventsToFile(string &filename)
* void EventLinkedList::loadEventsFromFile(string &filename)

3. Runtime Complexity

**1. Creating Event**

void insertEvent(Event &e);

**Description**: This function inserts a new event into the linked list.

**Runtime Complexity**:

* Average Case: O(1) - Constant time insertion at the beginning of the linked list.
* Worst Case: O(n) - Linear time if conflicts prevent insertion.

**2. Updating Event**

void updateEvent(int eventId);

**Description**: Allows users to update details of an existing event.

**Runtime Complexity**:

* Average Case:O(n) - Linear time, as it may need to traverse the linked list to find the event.
* Best Case:O(1) - Constant time if the event to update is the first in the list.

**3. Deleting Event**

   void deleteEvent(int eventId);`

**Description**: Deletes an event from the linked list based on its ID.

**Runtime Complexity**:

* Average Case: O(n) - Linear time, as it may need to traverse the list to find the event.
* Best Case: O(1) - Constant time if the event to delete is the first in the list.

**4. Displaying All Events**

void displayAllEvents();

**Description**: Displays details of all events in the linked list.

**Runtime Complexity**: O(n) - Linear time, as it traverses the entire list to display events.

**5. Searching Event by Title**

void searchEventByTitle(string &title);

**Description**: Searches for an event by its title.

**Runtime Complexity**: O(n) - Linear time, as it may need to traverse the list to find the event.

**6. Managing Attendees of an Event**

void manageAttendees(int eventId);

**Description**: Allows users to add, remove, and display attendees of an event.

**Runtime Complexity**: O(n) - Linear time, as it may need to traverse the list to find the event.

**7. Saving Events to File**

void saveEventsToFile(string &filename);

**Description**: Saves all events to a file.

**Runtime Complexity**: O(n) - Linear time, as it traverses the list to save all events.

**8. Constructor Event::Event:**

Event(int id, string &t, string &desc, string &d, string &startT, string &endT, string &loc)

**Description**: Initializes an Event object with provided details.

**Runtime Complexity**: O(1) - Constant time complexity as it performs a fixed number of operations regardless of input size.

**9. Getters and Setters:**

getTitle, getDescription, getDate, getStartTime, getEndTime, getLocation, setTItle, setDescription, setDate, setStartTime, setEndTime, setLocation

**Description**: Accessors and mutators for event attributes.

**Runtime Complexity**: O(1) - Constant time complexity as these functions access or modify a single attribute.

**10. Adding Attendee (addAttendee):**

void addAttendee(string attendee)

**Description**: Adds an attendee to the event's attendee list.

**Runtime Complexity**: O(1) - Constant time complexity as it appends to the end of the vector.

**11. Removing Attendee (removeAttendee):**

void removeAttendee(string &attendee)

**Description**: Removes an attendee from the event's attendee list.

**Runtime Complexity**: O(n) - Linear time complexity in the worst case as it may need to search for the attendee to remove.

**12. Checking Attendee Existence (hasAttendee):**

bool hasAttendee(string &attendee)`

**Description**: Checks if a given attendee exists in the event's attendee list.

**Runtime Complexity**: O(n) - Linear time complexity as it may need to search for the attendee.

**13. Displaying Attendees (displayAttendees):**

void displayAttendees()

**Description**: Displays the list of attendees for the event.

**Runtime Complexity**: O(n), where n is the number of attendees - Linear time complexity as iterates through the list of attendees.

**14. Displaying Event Details (displayEventDetails):**

void displayEventDetails()

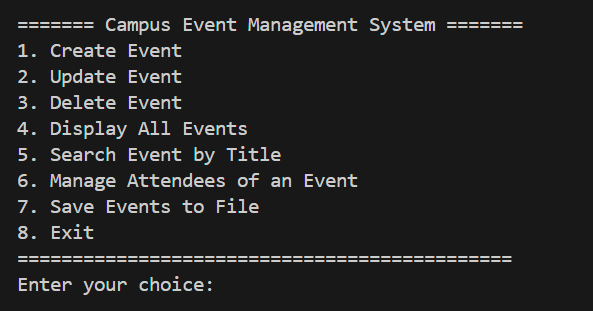
**Description**: Displays details of the event, including attendees.

**Runtime Complexity**: O(n), where n is the number of attendees - Linear time complexity as it iterates through the list of attendees to display them.

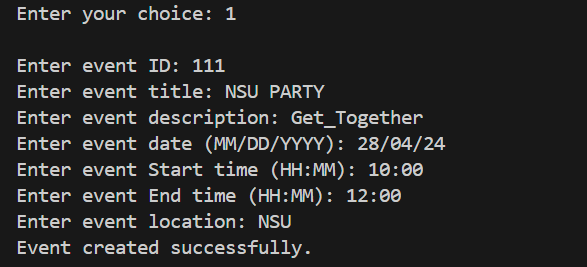
4. Execution Screenshots

Github Repo Code: <https://github.com/faruq05/event_management_project>

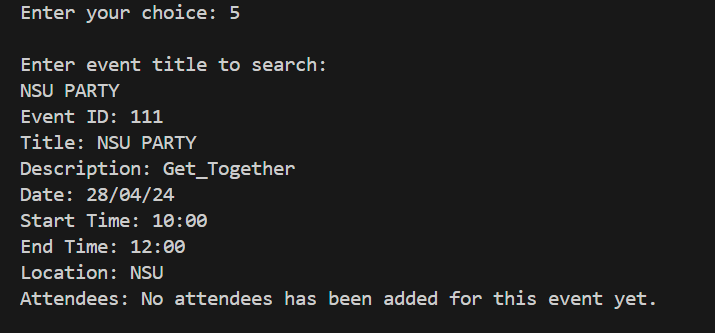
Main menu:



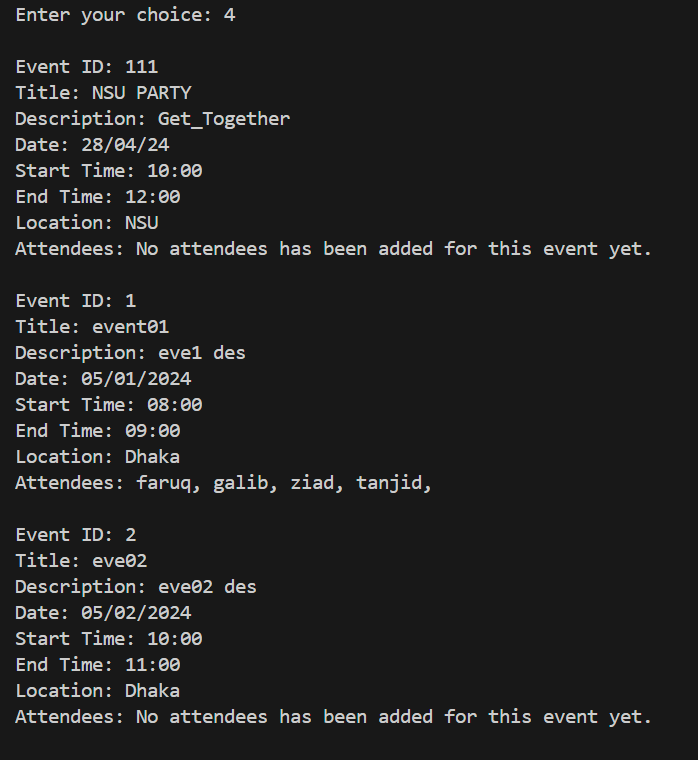
Creating new event :



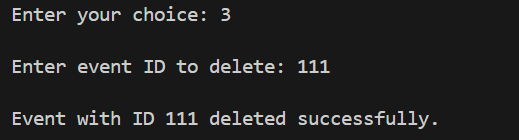
Searching events:



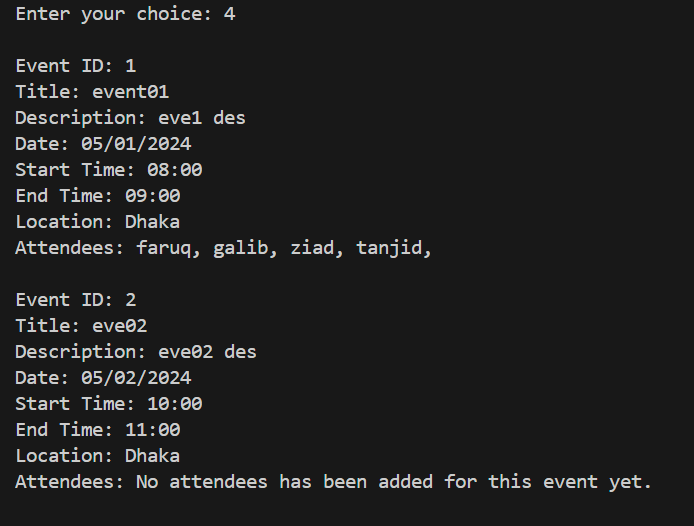
Displaying events:



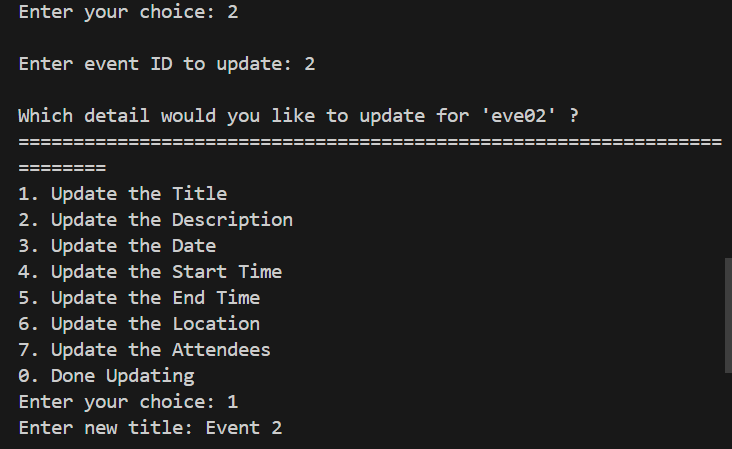
Deleting an event:

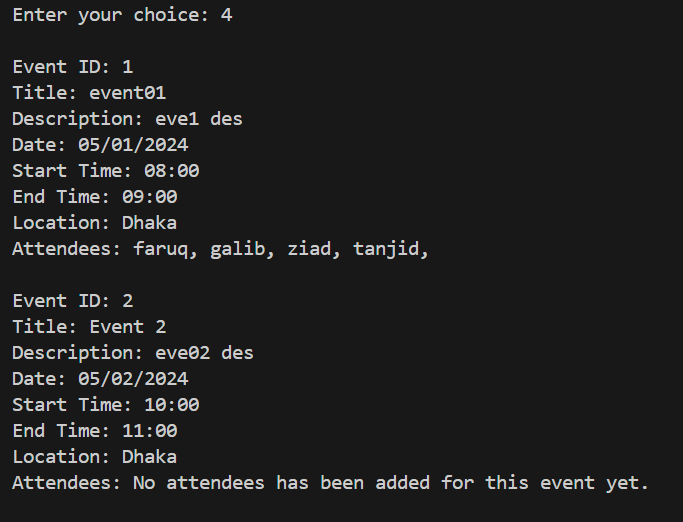


After deleting an event:

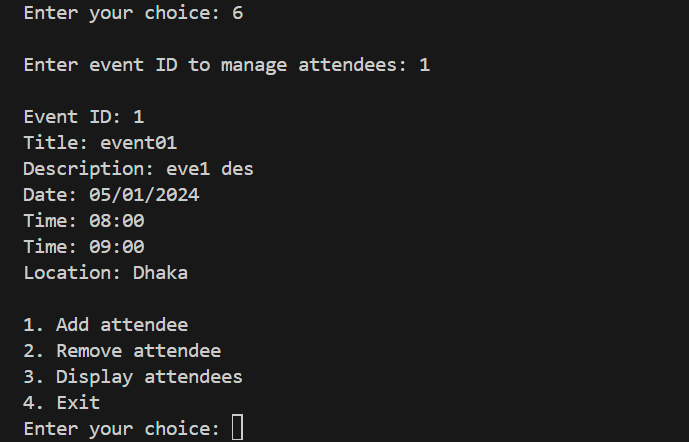


Update Event:

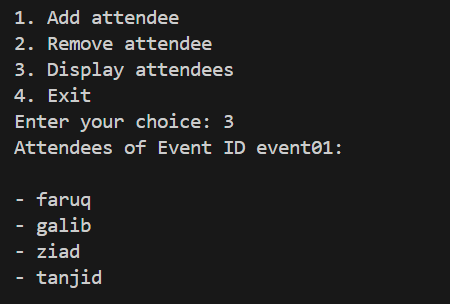


After Updating the Title of an Event : 

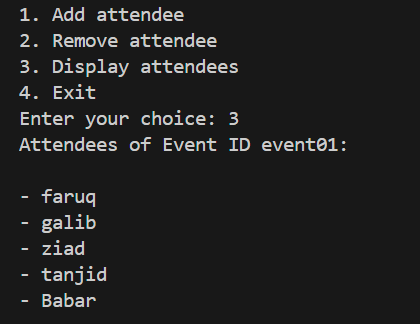
Manage Attendees:



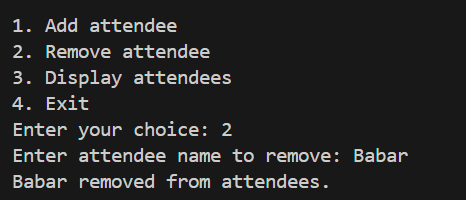
Display Attendees:



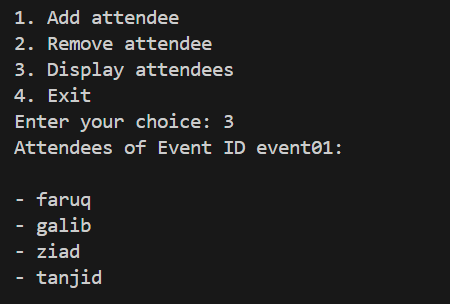
Display after adding attendees in an event:



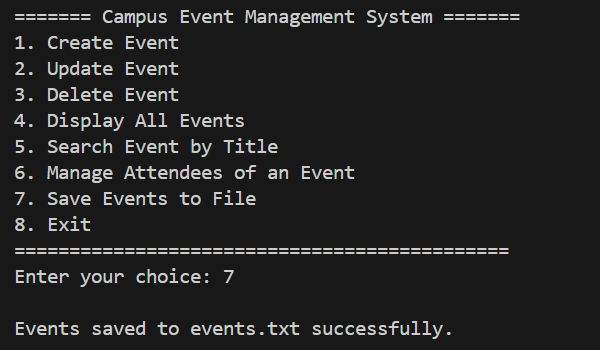
Removing an attendee:



After Removing an Attendee:



Saving the program in the file:



5. Future Scope

In our current version, we've been using a Command Line Interface (CLI). Moving forward, we want to create a Graphical User Interface (GUI) that's more like the apps you use every day. This change will make our program easier and more fun to use.

As we keep working on this program, we'll add more features to make it work better and faster. Sometimes, we might need to reorganize parts of the program to make everything run smoothly and make it easier to understand.

Here's what we're planning for the future:

* Switching from a text-based interface to a colorful and interactive interface.
* Adding new functions and options to make the program more powerful and optimized.
* Rearranging parts of the program to keep everything neat.