**Report:**

1. **Problem Statement:**

A computer-based system is required to be used in a typical university for subject registration. Users of this system are students and administrative personnel. The administrative personnel may add the list of subjects to be offered by faculty for the coming trimester, with the details of subject code, subject name, credit hour, day, time, venue and etc.

A list of students with valid ID and password must be created by system. Every student must enter valid ID and password for being entered to the system. System then checks the validity of ID and password and then allows the authorized user to continue.

Each student must be able to register for a minimum of 10 credit hours and maximum of 20 credit hours’ subjects; therefore, system must show a subject list from which students can select the subjects. The list of subjects can be sorted by subject code, subject name or credit hour.

Student then must select the subjects from a list of available subjects based on the information provided by the system about the subject. Subjects must be appeared in the subject list for each student according to his/her year of study. For example, for students in first year, only the subjects, which are offered for the first year, must be appeared in the subject list.

Students must be able to change their selected objects. They must be able to delete and add subjects. Finally, students must confirm their selected subjects. Upon confirmation, system must check whether the number of selected subjects is within the range of 10 to 20 credit hours. In addition, system must check there is no clashing. If not a message must be shown to student and request student to make the changes.

1. **Data Structures and Algorithms used**

Following data structures are used in developing the application: **Linked List, Stack and Queue**. Since no addition and deletion of students is required, **stack** is used for storing student information. For saving subjects, **queue** is used because while displaying the subjects in sorted order it will be easier to perform sorting on queue. To save the registered subjects by each student, **linked list** is used. Since, subjects can be added and deleted based on credit hours.

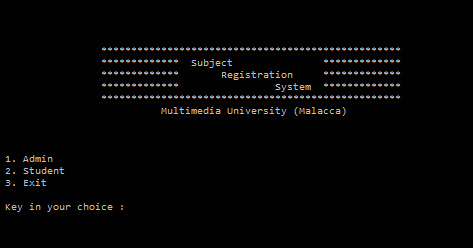
**Sequential search** is used while validating a student from a list of students. **Bubble sort** is used for displaying the subjects available to a student sorted by credit hours.

1. **Performance and comparison of Data Structures and algorithms**

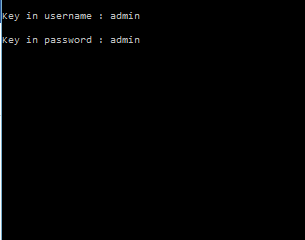
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Access | Search | Insertion | Deletion | Space |
| Stack | **O(n)** | **O(n)** | **O(1)** | **O(1)** | **O(n)** |
| Queue | **O(n)** | **O(n)** | **O(1)** | **O(1)** | **O(n)** |
| Linked List | **O(n)** | **O(n)** | **O(1)** | **O(1)** | **O(n)** |
| Sequential Search | **O(n)** | | | | **O(1)** |
| Bubble Sort | **O(n^2)** | | | | **O(1)** |
|  |  | | | |  |

Stack which is and abstract data type, features LIFO operation. The element which is placed (inserted or added) last, is accessed first. Stack ADT allows all data operations at one end only. Insertion operation is called PUSH and removal operation is POP. Meanwhile, queue features the FIFO operation, In queue, the object that is inserted first is removed first. Both the stack and queue offers only sequential search. In that case, any decent implementation of a stack or queue will offer 0(1) access for both insert and remove operations. Moreover, linked list used to locate the first element of the list and also the end. It can find successor and predecessor using link it contains. Mostly, insertion and deletion node operations are easily implemented in a linked list. Stacks and queues can be implemented in a linked list.  
Finding the point of insertion/deletion is O(n) while performing the insertion/deletion is 0(1) for linked list. Apart from that, sequential search, a method for finding target value within a list. It sequentially checks each element for the target value until a match is found or until all the elements have been searched. The efficiency at worst case and average case is O(n) while for best case is O(1) . Linear search which is also known as sequential search, can be used for small arrays instead of using binary search. Finally, bubble sort has worst case and average both O(n^2), where n is the number of items being sorted. For best case , it has O(n). Bubble sort requires n-1 passes through an array and pass 1 requires n-1 comparison and at most n-1 swaps, pass 2 requires n-2, etc. Bubble sort has an advantage of being simple to write , and it only takes few lines to code compared to other sorting algorithm. Bubble sort only requires O(1) space.

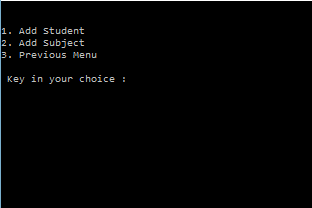
**4. Screenshots**



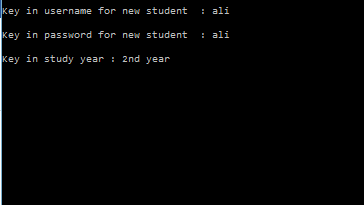
( Main Page )



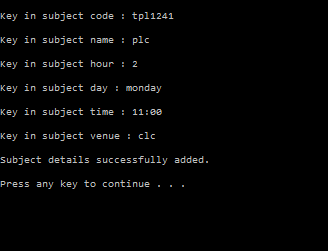
( Admin Login )



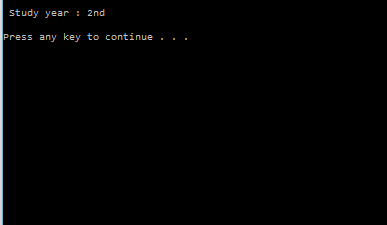
(Admin : Adding student and subjects)



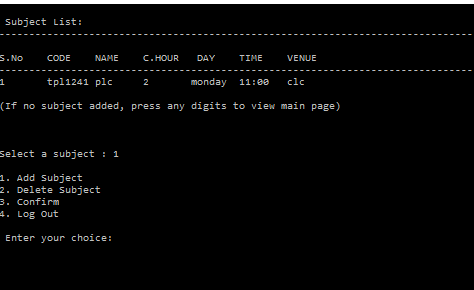
(Adding student)



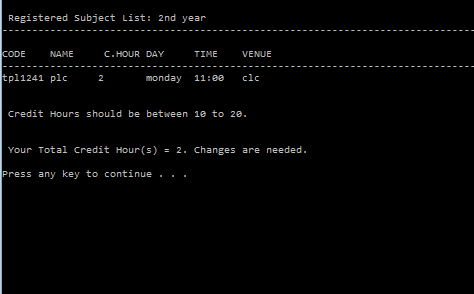
(Adding subject)



(After student successfully login , year of study screened)



(Student can add, confirm and then delete subject from subject list which is sorted according to credit hour)



( View registered subject list and if credit hours between 10-20 the system allows to register the subject)