

***SCHOOL OF COMPUTER AND INFORMATION ENGINEERING***

***Data Structure (SOC 2010)***

***Student Activity***

***Library management system***

*Team members 003:*

*U1910255Jakhongirov Jasurbek (Leader)*

*U1910242Khasanov Muhammad Ali*

*U1910239Akhmedov Shamsiddin*

*U1910241Kholdorov Tursunali*

*U1910221Yangiboev Komiljon*

Page 2

**Index**

* Problem Overview 3
* Operations using data structure 4
* Deeping in efficient Data structure 5
* Code implementation 6
* Limitations and maintenance 8
* Conclusion 9

Page 3

**Library management issues**

In the world where as much as Books are one of the necessities for all human beings, getting effortless access to them is needed too. Most of the huge libraries possess the enormous amount of database of books, journals, magazines that should be maintained and updated regularly on a daily basis. Unfortunately, the system that provides service of processing books in the different classified categories are not provided in the most library managements. First of all, Human interaction should be decreased sufficiently so that computers can find and traverse the book in the most efficient way. When we go in the library to get the book, we often face difficulties finding and accessing the book that we want and also, time it takes to own the book is much more wasting the huge amount of time. Secondly, Precise time usage of books is important that most of the people overdue. With respect to their delaying of returning books into library, they have to pay particular amount of fine, which improve the discipline and eliminate the losing of books in libraries. Finally, in the modern world, it is a must to use the modern application that increase the connection between users and administrators through the high efficient ways.

Page 4

**Operation usage**

There are a bunch of specific operations that should be implemented to make it application viable for improving the library management systems. Firstly, **Insertion sort** Has been used to push the new books and update the process manually. When the new book is added to library, administrator insert the book into database system and update the condition regularly. Second, **Deletion** operation is applied when the user come and take the book from database. the book will be deleted automatically when it is taken out of database. Next, **Search** operation is implemented when administrator is searching for a particular book. We apply **Linear search** algorithm to find a book traversing through the different classified category of the books and pointing at the exact one in an efficient way. The Last one is **Bubble Sort** algorithm which helps us to sort the book chronologically.

Page 5

**Efficient Data structures**

The challenging part we faced on the implementations of the application is choosing an efficient data structure to make an user friendly application. There are the whole bunch of abstract data structures that can be used to make application more efficiently such as Array, Linked list, Tree, BST, Queue. Among the most viable ones, we choose the **Array** data structure. There are some reasons behind it. First of all, Array is linear collection of similar data types stored in database. In the different categories, there are similar amount of books in the classification of History, Technology, Language, and e.t.c For the technology part, books are arranged in linear form whereby we can implement array data structure to find books and update them and sort them through. On the other hand, **linked list** could be an option to store the data in database. But, when it comes to efficiency, Array takes less memory place and time to implement the application. Hence, the application is implemented using Array function and classes.

Page 6

**Code Implementations**

This is an implementation of code we use the functions of array to apply the necessary operation listed above. Deletion, Insertion, Sorting.

void add\_book()

{

  string book\_name;

    string book\_subject;

    int book\_date;

    ofstream book("book.txt", ios::app);

  cout<<"Please enter name of book : ";

  cin>>book\_name;

  cout<<"enter subject of the book :";

  cin>>book\_subject;

  cout<<"please enter the published date of the book :";

  cin>>book\_date;

  book<<book\_name<<"  "<<book\_subject<<"  "<<"  "<<book\_date<<endl;

}

void modify\_book()

{

cout<<"1.To change book subject"<<endl;

  cout<<"2.To change published date "<<endl;

  int number\_modify;

  cin>>number\_modify;

*if*(number\_modify==1)

{

  cout<<"please write current subject of the book"<<endl;

  string subject\_book;

  string subject\_book\_new;

  cin>>subject\_book;

  cout<<"please enter your new subject of the book ";

  cin>>subject\_book\_new;

  subject\_book=subject\_book\_new;

  }

*if*(number\_modify==2)

  {

    cout<<"please write published date of the book"<<endl;

    string book\_date;

    int book\_date\_new;

    cin>>book\_date;

    cout<<"please enter your new date";

    cin>>book\_date\_new;

    book\_date=book\_date\_new;

  }

}

void delete\_book()

{

 string book\_name1;

  cout<<"Please enter the name of the book to be deleted "<<endl;

  string book\_name;

  cin>>book\_name;

   ofstream librarian("book.txt");

  ifstream namein;

  namein.open("book.txt");

  ofstream nameout;

  nameout.open("new.txt");

  namein>>book\_name;

*while*(!namein.eof())

{

*if*(book\_name!=book\_name1)

{

  nameout<<book\_name;

  }

  namein>>book\_name1;

  }

  namein.close();

  nameout.close();

  remove("book.txt");

  rename("new1.txt","book.txt")  ;

}

void liblogin()

{

  string name\_lib;

  int id\_lib;

  string password\_lib;

Page 8

**Limitations and Maintenance**

Although we try to use the most efficient data structure for application, we have some limitations and faults of Array. If the quantity of the books in the library is enormous to the situation, Array data structure mostly fail. It obviously takes more time to store data in linear form. Deletion process cannot be performed in the anywhere of the database, the beginning and end point positions can be implemented through the traversing every element of array for finding the particular book. Apparently, it takes linear form time complexity O(n) in Big O notation. In this case, **Binary search tree** data structure is the efficient one to perform this task which is time complexity O(log n). Regarding maintenance problem, we should get good network connection regularly to update the process of renting and returning books.

Page 9

**Conclusion**

Overall, library management system project we work on will help the people to make their life more convenient and productive as well as time efficient. It helps to improve the connection between client and customer experiences bringing efficiency in their work and life.

[Source code on Github](https://github.com/Alixan777/DS_PROJECT)