**Short description:**

Our main goal in this project is to develop a library management system where students may check out books and the librarian or administration. Can modify or remove the library's record of its books. As a result, the system is divided into two sections: one for user perception and the other for administrators.

To manage accounts where the username and password have already been configured, the admin must first log in. After successfully logging in, he can add, remove, and update the books. Any new book may be added to the current list of books. He can also erase any book that is already published. The administrator can update both the book's title and its quantity under the update option. A binary search tree will be generated as the admin adds the books, with the nodes containing the book names and arranged in sorted order.

A student must now authenticate into the system using a valid university ID if they want to issue or return any books. Only if the student's ID fits the list of student university IDs will they be permitted to issue or return. When a student enters the name of a book to be issued, the already established binary search tree will be searched using that book's name. A message stating "Book is not available in the library" will be printed if the book is not discovered in the tree.

Also, if the book is out of stock, the following message will be displayed: "This book is now unavailable. Please try it again after a few days". Moreover, the student is only allowed to release two books at once. The librarian logs the date and time the student issued the book. And if the student forgets to return the book before the deadline, he will be charged for that certain penalty.

**List of Data Structures Used in Logic Design:**

1) Binary Search Tree: Binary search uses linked lists to generate a binary search tree, which is used to search for a specific subject. The binary search tree is traversed in order during the search operation to get the elements in sorted order.

2) Hash-map: In general, hash-maps map keys to values in a way that prevents the generation of duplicate keys. So, in our project, distinct keys are created using a hash map and allocated to each book so that we may store and retrieve records of books in a 2-D array using these keys.

3) 2D- array: In this case, we have utilised a 2-D array to hold the records for each book. The columns show the overall number of books and the number of books that are currently available, and the rows show the unique value for each book that a hash-map returns.

**Operations Should be carried out on each data structure:**

**Insertion:** We are adding books to the binary search tree using the insertion operation. And the names of the new books will be listed in the tree.

**Deletion:** To remove the node for that specific book from the binary search tree, we utilise the deletion operation. The remaining items in the tree are once more reorganised once the node has been deleted.

**Update:** The librarian is able to increase the number of books in a collection that is already stored in a 2D array.

**Print a list of all the values:** There is also the choice of printing all the book information. The title of the book, the number of copies that can be provided, and the total number of books the library has.

**Print book in order:** When the binary search tree is balanced, printing the contents of the tree, or the titles of the books, in ascending order, is performed. Processing requirements for data structures in a balanced Binary Search tree, elements are maintained in ascending order.

**Input data and output generated:**

1) Add book

* **Input data**

Enter name of the book:

Enter quantity of book:

* **Output generated**

Book added successfully...

2) Delete book

* **Input data**

Enter name of the book:

Enter quantity of book:

* **Output generated**

Book deleted successfully...

2) Update book

* **Input data**

Enter name of book:

Enter quantity of book to add more:

* **Output generated**

Successfully updated...

4) Print book details

* **Output generated**

All the details of book will be printed

5) Print books in-order

* **Output generated**

The list of all books will be printed in ascending order of their names as it is balanced binary search tree (Balanced BST).

6) Print tree

* **Output generated**

Balanced binary search tree is printed

7) Issue book by user

* **Input data**

Enter your university ID:

Enter name of book:

* **Output generated**

Book issued successfully!!

Current date time:

Due date and time:

8) Return book:

* **Input data**

Enter your university ID:

Enter name of book:

* **Output generated**

Book returned successfully!!

Current date time:

Return date and time:

**References:**

1. How to get Key from Value in Hashtable, HashMap or Map in <https://stackoverflow.com/questions/590327/method-to-find-key-in-hashtable>

2. Number of days between dates (Beginning Java forum at Coderanch) <https://coderanch.com/t/543495/java/Number-days-dates>

3. Masking password input from the console : Java - Stack Overflow [https://stackoverflow.com/questions/8138411/masking-password-input-from- the-console-java](https://stackoverflow.com/questions/8138411/masking-password-input-from-%20the-console-java)

4. Binary Search Tree | Set 1 (Search and Insertion) – GeeksforGeeks <https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/>

5. How to Get Current Date and Time in Java – Javatpoint <https://www.javatpoint.com/java-get-current-date>

6. How to compare dates in Java? - Stack Overflow <https://stackoverflow.com/questions/2592501/how-to-compare-dates-in-java>