Data Structures & Algorithms

Group Project Assignment

Project Design Documentation

Submitted By: 4777- sohaib faheem, 4832-Talal Asad, abdullah ghulam

Table of Contents

1. Project Introduction & Objectives

This project showcases the practical application of core data structures and algorithmic concepts using the C++ programming language. The main objective is to develop a fully functional application that efficiently manages library management data using object-oriented principles and data structures. This system integrates operations such as insertion, deletion, searching, and sorting. It is designed to help us understand the performance trade-offs between different data structures and algorithms in real-world scenarios.

2. Comparison & Optimization

This section evaluates the implemented data structures and algorithms, focusing on their efficiency and suitability for the library management system.

2.1 Data Structure Comparison

Data Structure	Time Complexity	Space Complexity	Use Case
BookList (Linked List)	Insert: O(1), Search/Delete: O(n)	O (n)	Dynamic book storage
StudyRoom (Queue)	Enqueue/Dequeue: O(1)	O (n)	FIFO room reservations
Vector (Temp Sorting)	Random Access: O(1)	O (n)	Efficient sorting/searching

Key Observations:

- BookList prioritizes fast insertions (O(1)) over searches/deletions (O(n)), ideal for frequent book additions.
- StudyRoomQueue ensures fair room allocation via FIFO operations.
- Vector Conversion enables efficient sorting/searching (O(1) access) but adds overhead during linked list ↔ vector transitions.

2.2 Algorithm Comparison

Sorting Algorithms

Algorithm	Time Complexity	Space Complexity	Use Case
Quick Sort	Avg/Best: O(n logn)	O (log n)	Large Dataset
Bubble Sort	Avg/Worst: O(n²)	O (n)	Small Dataset

Trade-offs:

- Quick Sort is faster for large book inventories but risks *O(n²)* worst-case performance.
- Bubble Sort is simple but impractical for large *n*.

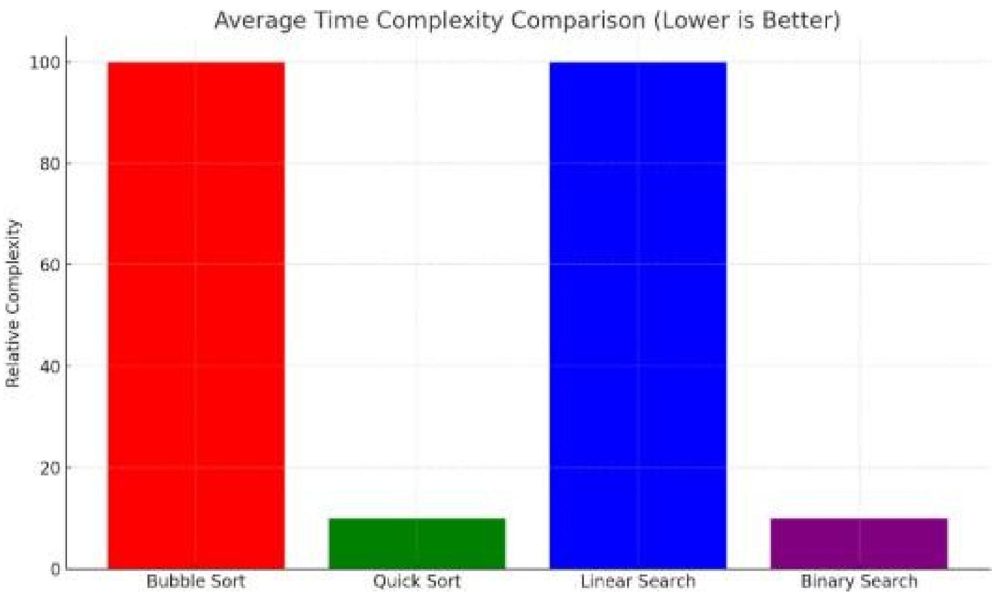
Searching Algorithms

Algorithm	Time Complexity	Space Complexity	Use Case
Linear Search	Avg: O (n)	Unsorted Data	Large Dataset
Binary Search	Avg: O (log n)	Sorted Data	Small Dataset

Trade-offs:

- Binary Search requires sorted data (via Quick Sort), adding *O(n log n)* preprocessing.
- Linear Search avoids sorting but scales poorly.

Graphical Representation:



2.3 Design Justification

1. **Data Structure Choices**

- **BookList (Linked List):**
    - Pros: Efficient insertions (O(1)), dynamic resizing.
    - Cons: Slow searches (O(n)). Mitigated by periodic sorting into vectors for Binary Search.
- **StudyRoomQueue:**
    - Ensures fair first-come-first-served room bookings.

2. **Algorithm Choices**

- **Quick Sort + Binary Search:**
    - Optimal for large book inventories (O(n log n) + O(log n)).
- **Linear Search:**
    - Reserved for small datasets or unsorted searches.

3. **Trade-offs**

    - **Memory Overhead:** Linked list pointers increase memory usage vs. arrays.

- **Sorting Overhead:** Converting linked lists to vectors for sorting adds temporary O(n) s

3. Implementation Overview

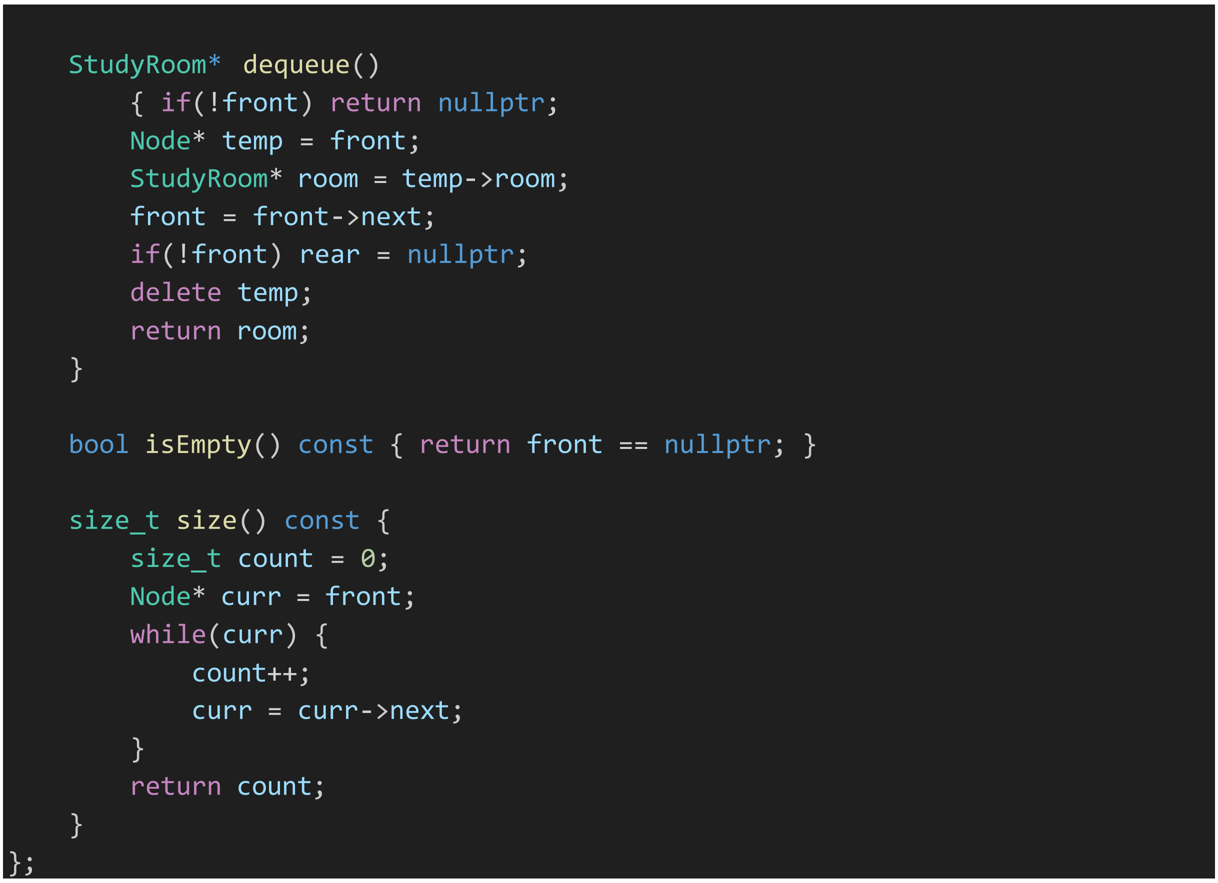
3.1 Code Structure

1. BookList Class (Linked List Implementation):

```
class BookList
{ private:
    struct BookNode
        { Book data;
        BookNode* next;
        BookNode(const Book& b) : data(b), next(nullptr) {}
    };

    BookNode* head;
    int count;

public:
    BookList() : head(nullptr), count(0) {}

    void insert(const Book& book) {
        BookNode* newNode = new BookNode(book);
        newNode->next = head;
        head = newNode;
        count++;
    }

    bool remove(const string& ISBN)
        { BookNode* curr = head;
        BookNode* prev = nullptr;

        while(curr) {
            if(curr->data.getISBN() == ISBN)
                { if(prev) prev->next = curr->next;
                else head = curr->next;
                delete curr;
                count--;
                return true;
            }
            prev = curr;
            curr = curr->next;
        }
        return false;
    }

    vector<Book> getAllBooks() const
        { vector<Book>  result;
        BookNode* curr = head;
        while(curr) {
            result.push_back(curr->data);
            curr = curr->next;
        }
        return result;
    }

    int size() const { return count; }

    void clear() {
        while(head) {
            BookNode* temp = head;
            head = head->next;
            delete temp;
        }
        count = 0;
    }
};
```

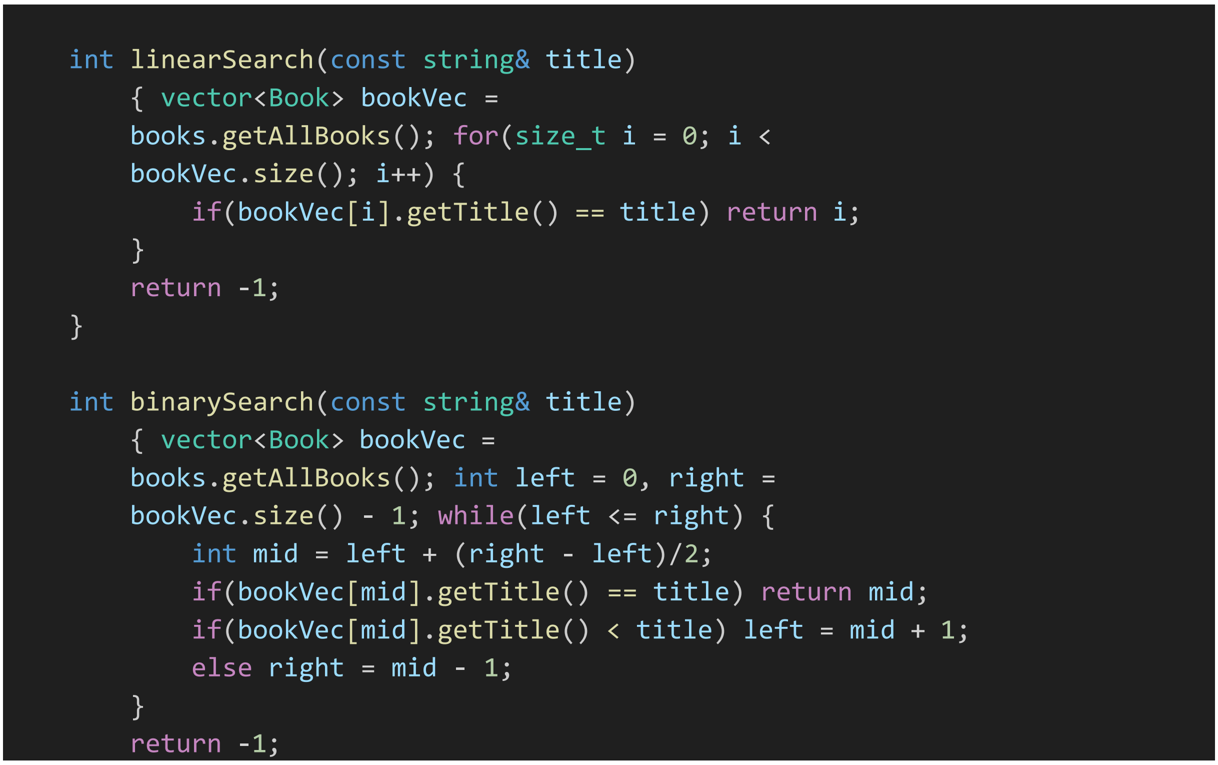
2. StudyRoomQueue (Custom Queue):

```
class StudyRoomQueue
{ private:
    struct Node {
        StudyRoom* room;
        Node* next;
        Node(StudyRoom* r) : room(r), next(nullptr) {}
    };

    Node* front;
    Node* rear;

public:
    StudyRoomQueue() : front(nullptr), rear(nullptr) {}

    void enqueue(StudyRoom* room)
        { Node* newNode = new
        Node(room); if(rear) rear->next
        = newNode; rear = newNode;
        if(!front) front = rear;
    }
```



```
    StudyRoom*  dequeue()
        { if(!front) return nullptr;
        Node* temp = front;
        StudyRoom* room = temp->room;
        front = front->next;
        if(!front) rear = nullptr;
        delete temp;
        return room;
    }

    bool isEmpty() const { return front == nullptr; }

    size_t size() const {
        size_t count = 0;
        Node* curr = front;
        while(curr) {
            count++;
            curr = curr->next;
        }
        return count;
    }
};
```

3. Search Algorithms:



```
    int linearSearch(const string& title)
        { vector<Book> bookVec =
        books.getAllBooks(); for(size_t i = 0; i <
        bookVec.size(); i++) {
            if(bookVec[i].getTitle() == title) return i;
        }
        return -1;
    }

    int binarySearch(const string& title)
        { vector<Book> bookVec =
        books.getAllBooks(); int left = 0, right =
        bookVec.size() - 1; while(left <= right) {
            int mid = left + (right - left)/2;
            if(bookVec[mid].getTitle() == title) return mid;
            if(bookVec[mid].getTitle() < title) left = mid + 1;
            else right = mid - 1;
        }
        return -1;
```

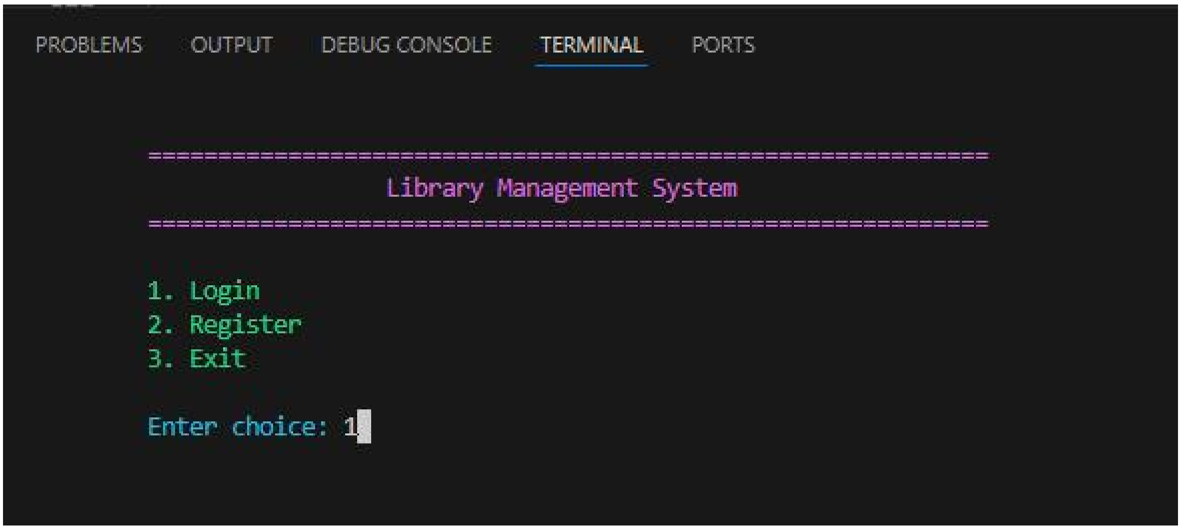
4. Sorting Algorithms:

```
void quickSort(vector<Book>& arr, int low, int high)
       { if (low < high) {
           int pi = partition(arr, low, high);
           quickSort(arr, low, pi - 1);
           quickSort(arr, pi + 1, high);
       }
   }

   int partition(vector<Book>& arr, int low, int high)
       { Book pivot = arr[high];
       int i = low - 1;

       for (int j = low; j <= high - 1; j++) {
           if (arr[j].getTitle() < pivot.getTitle())
               { i++;
               swap(arr[i], arr[j]);
           }
       }
       swap(arr[i + 1], arr[high]);
       return i + 1;
   }

   bool isSorted(const vector<Book>& arr)
       { for(size_t i = 1; i < arr.size(); i++) {
           if(arr[i-1].getTitle() > arr[i].getTitle()) return false;
       }
       return true;
   }

   void bubbleSort(vector<Book>& arr)
       { bool swapped;
       for(size_t i = 0; i < arr.size()-1; i++)
           { swapped = false;
           for(size_t j = 0; j < arr.size()-i-1; j++)
               { if(arr[j].getTitle() > arr[j+1].getTitle()) {
                   swap(arr[j], arr[j+1]);
                   swapped = true;
               }
           }
           if(!swapped) break;
       }
   }
```

3.2 Key Features and Overview

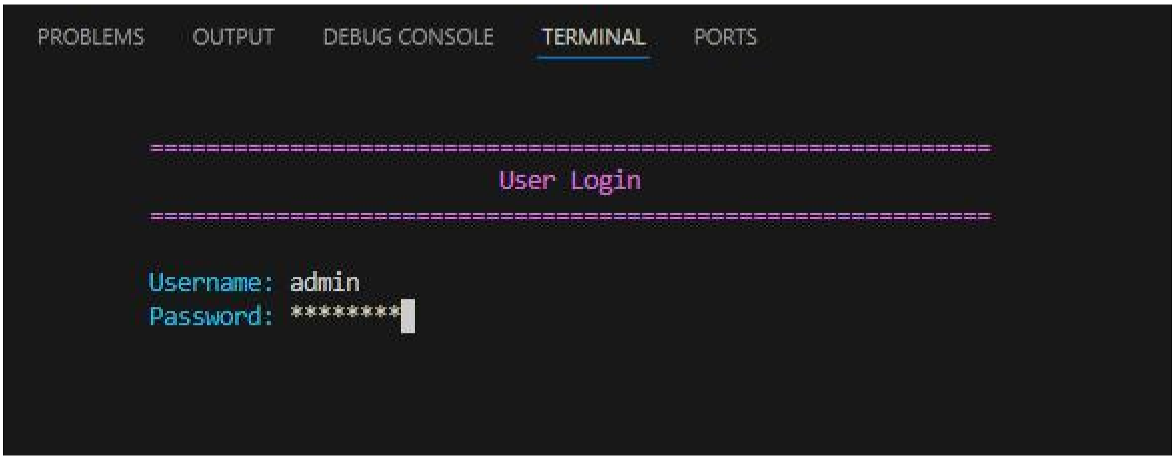
The system overview looks like this:

1. First select either login or register, if you’re a new user then register first otherwise the already registered in users are “admin” and “student” so login with their credentials



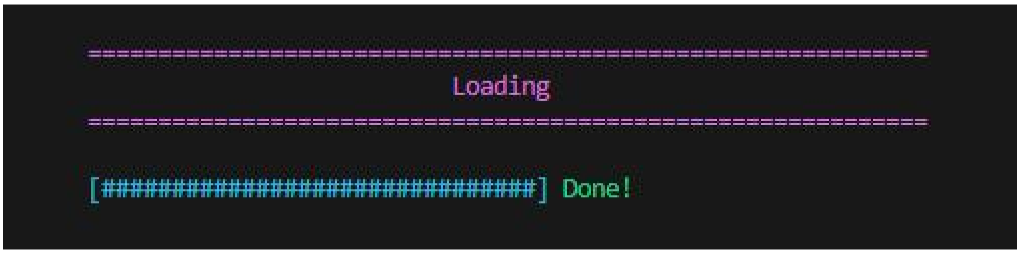
```
============================================================
                 Library Management System
============================================================

1. Login
2. Register
3. Exit

Enter choice: 1
```

2: Logging in as admin by inputting the already defined admin credentials which are “username = admin” and “password = admin123”



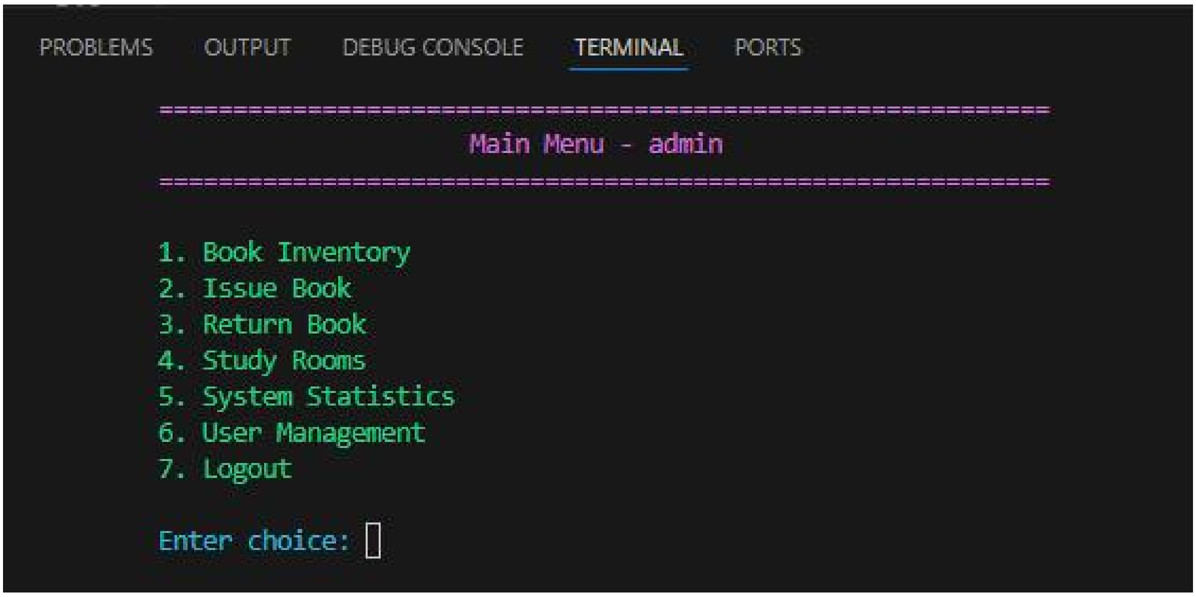
```
============================================================
                         User Login
============================================================

Username: admin
Password: ********
```

3: Loading page:



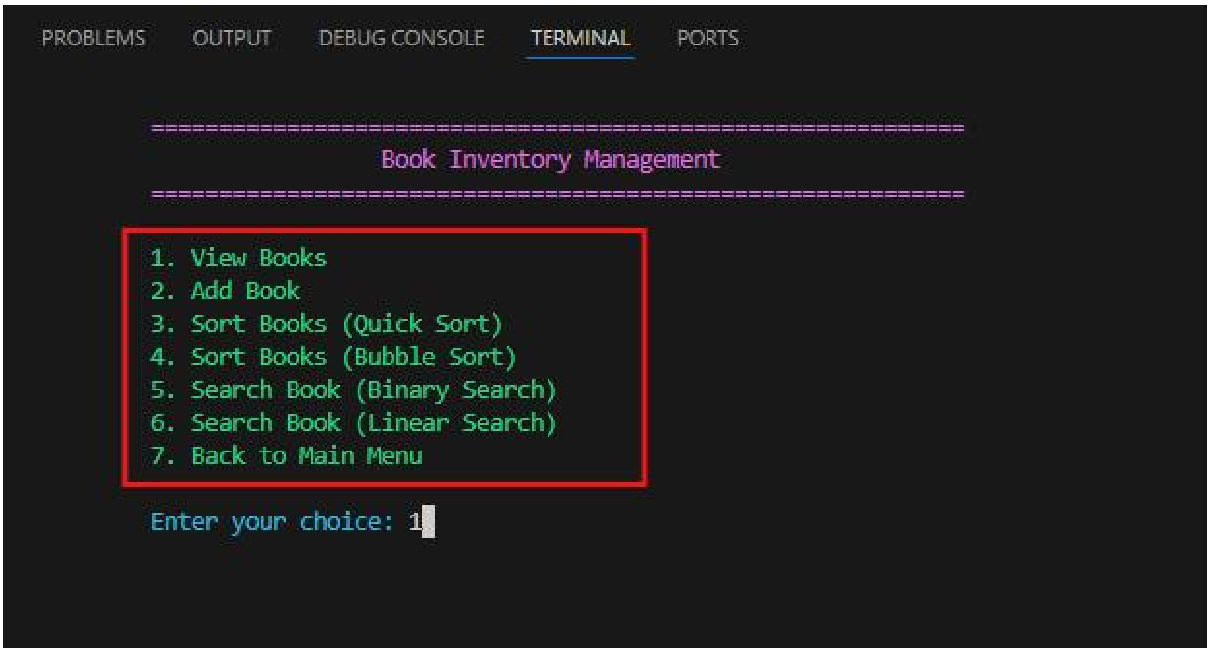
```
============================================================
                          Loading
============================================================

[################################] Done!
```

4. The following screen with the available features appear:



```
==============================================================
                     Main Menu - admin
==============================================================

1. Book Inventory
2. Issue Book
3. Return Book
4. Study Rooms
5. System Statistics
6. User Management
7. Logout

Enter choice:
```

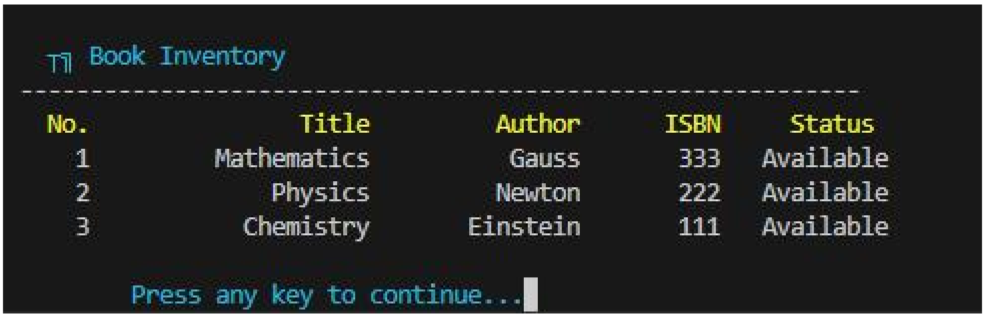
5. Choosing "Book Inventory" we have the following features in it:



```
============================================================
                Book Inventory Management
============================================================

1. View Books
2. Add Book
3. Sort Books (Quick Sort)
4. Sort Books (Bubble Sort)
5. Search Book (Binary Search)
6. Search Book (Linear Search)
7. Back to Main Menu

Enter your choice: 1
```

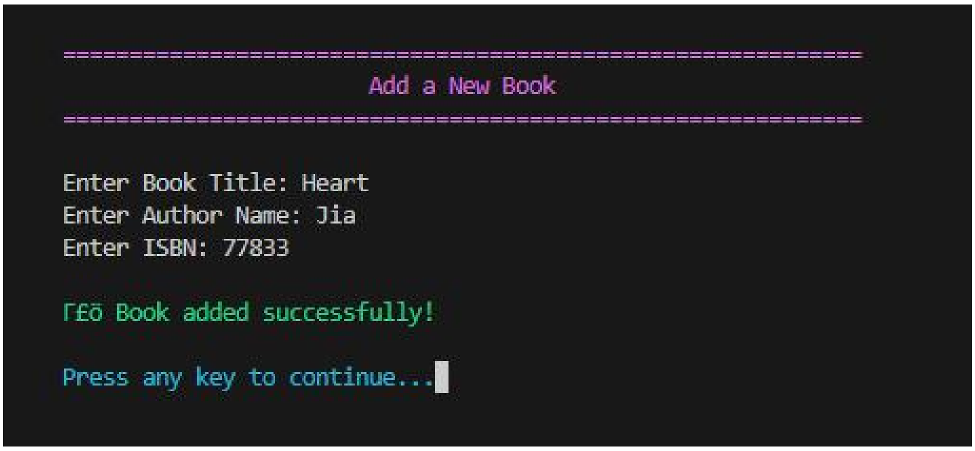
Displaying only view and add book in this report:

View Books:



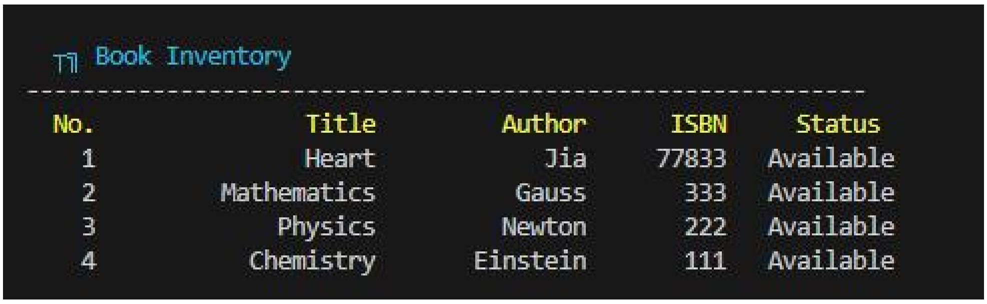
```
 ╥╗ Book Inventory
------------------------------------------------------------
 No.                Title          Author       ISBN     Status
   1          Mathematics           Gauss        333  Available
   2              Physics          Newton        222  Available
   3            Chemistry        Einstein        111  Available

       Press any key to continue...
```

Add Books:



```
============================================================
                      Add a New Book
============================================================

Enter Book Title: Heart
Enter Author Name: Jia
Enter ISBN: 77833

Γ£ö Book added successfully!

Press any key to continue...
```

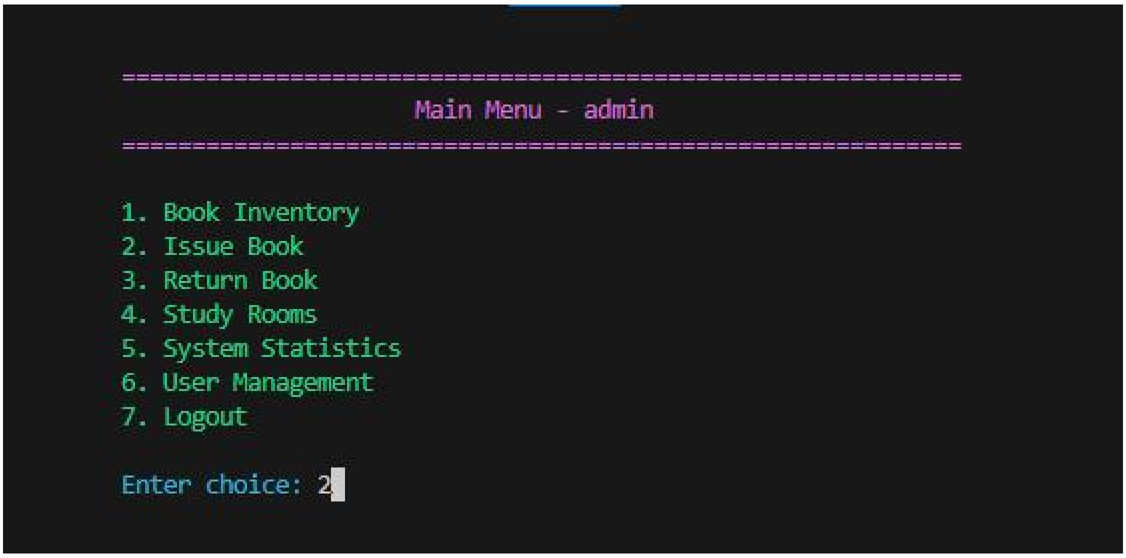
View Books after adding book:



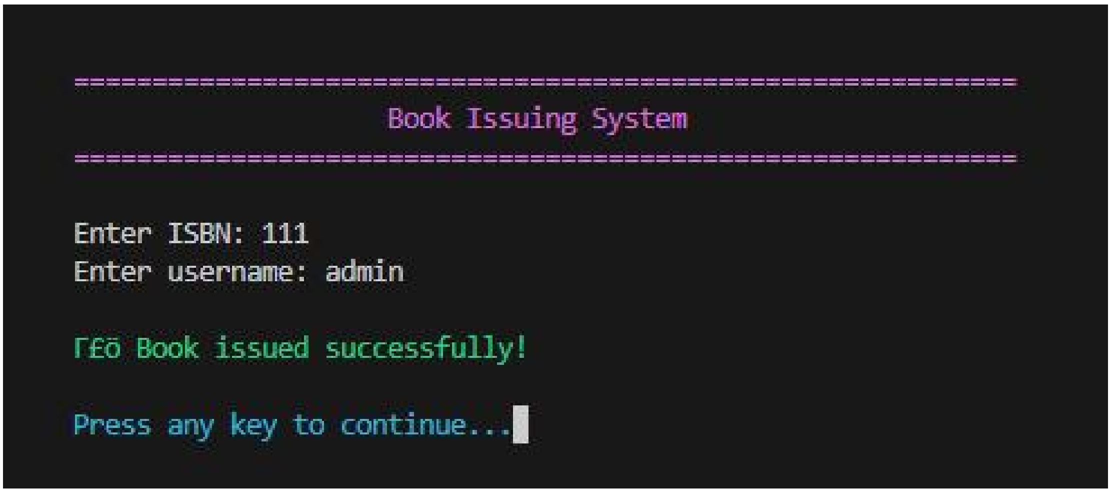
```
 ╥╗ Book Inventory
------------------------------------------------------------
 No.                Title          Author       ISBN     Status
   1                Heart             Jia      77833  Available
   2          Mathematics           Gauss        333  Available
   3              Physics          Newton        222  Available
   4            Chemistry        Einstein        111  Available
```

Other sorting and searching functions work effectively and efficiently as well as these so now going back to the main page:

6. Choosing “Issue Book” we have following output flow:

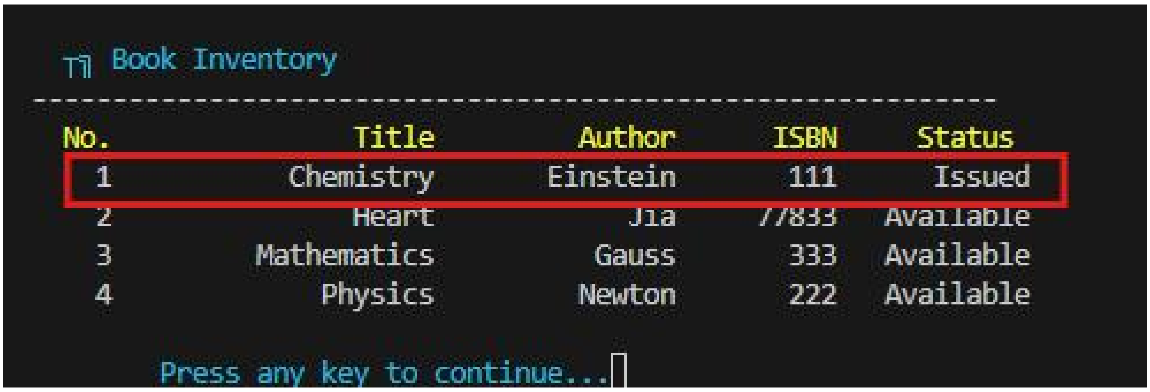


```
============================================================
                    Main Menu - admin
============================================================

1. Book Inventory
2. Issue Book
3. Return Book
4. Study Rooms
5. System Statistics
6. User Management
7. Logout

Enter choice: 2
```



```
============================================================
                    Book Issuing System
============================================================

Enter ISBN: 111
Enter username: admin

Γ£ö Book issued successfully!

Press any key to continue...
```

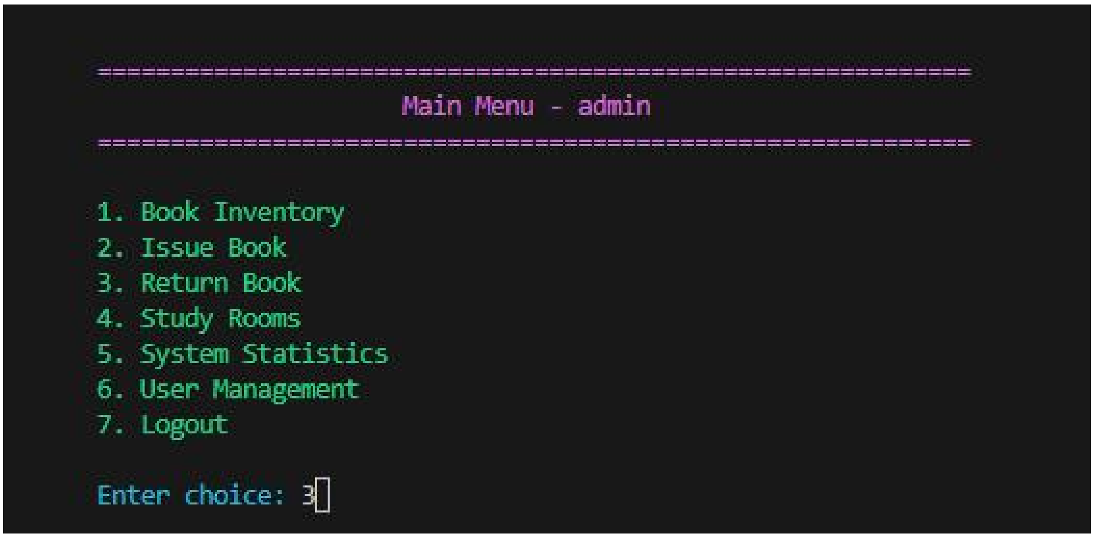
Now if we go and view the books in inventory then the one with this ISBN should show that its been issued as follows:



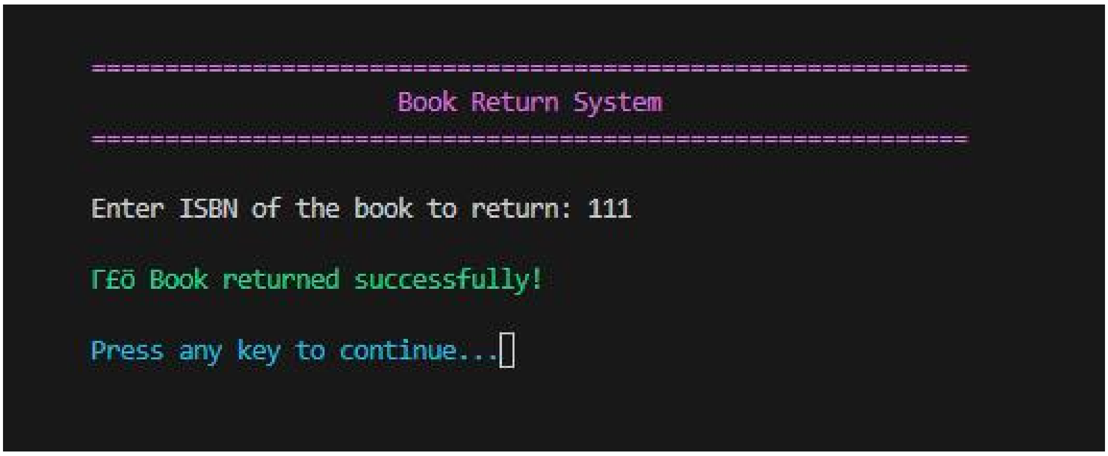
```
  Tï Book Inventory
------------------------------------------------------------
  No.              Title        Author      ISBN     Status
   1           Chemistry      Einstein       111     Issued
   2               Heart           Jia     77833  Available
   3         Mathematics         Gauss       333  Available
   4             Physics        Newton       222  Available

        Press any key to continue...
```

Now again going back to main page.

7. Choosing “Return Book” to return the book which we issued:

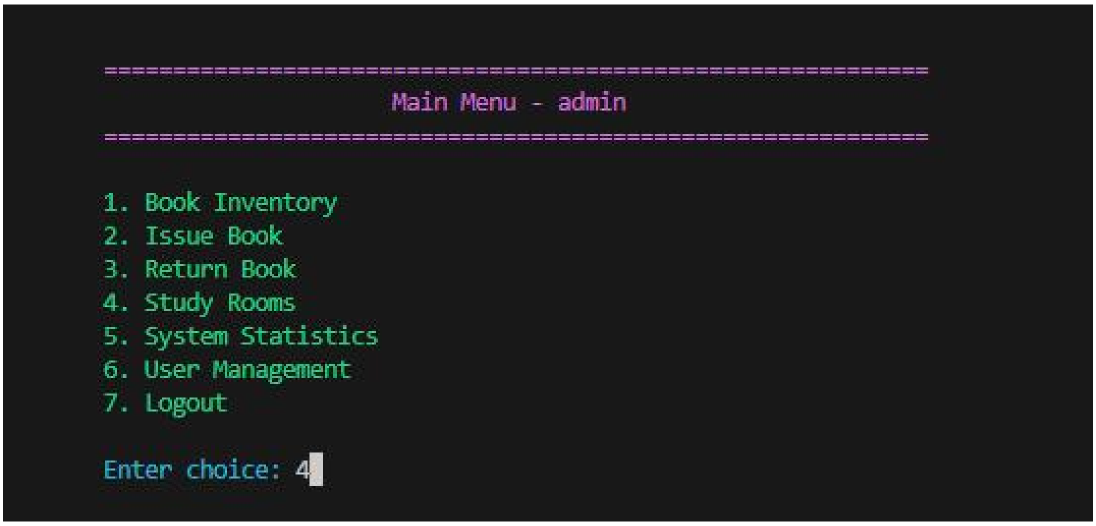


```
============================================================
                    Main Menu - admin
============================================================

1. Book Inventory
2. Issue Book
3. Return Book
4. Study Rooms
5. System Statistics
6. User Management
7. Logout

Enter choice: 3
```



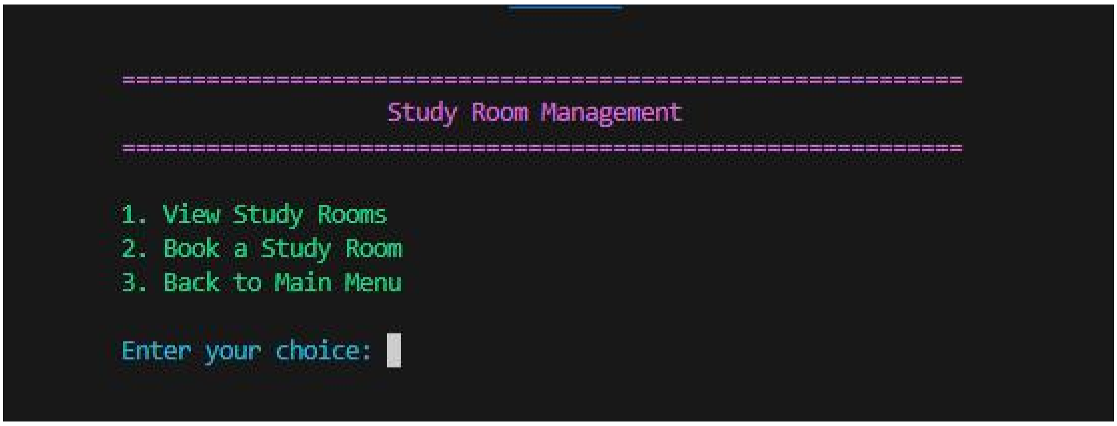
```
============================================================
                   Book Return System
============================================================

Enter ISBN of the book to return: 111

Γ£ö Book returned successfully!

Press any key to continue...
```

8. Choosing “Study Rooms” we have the following options in it:

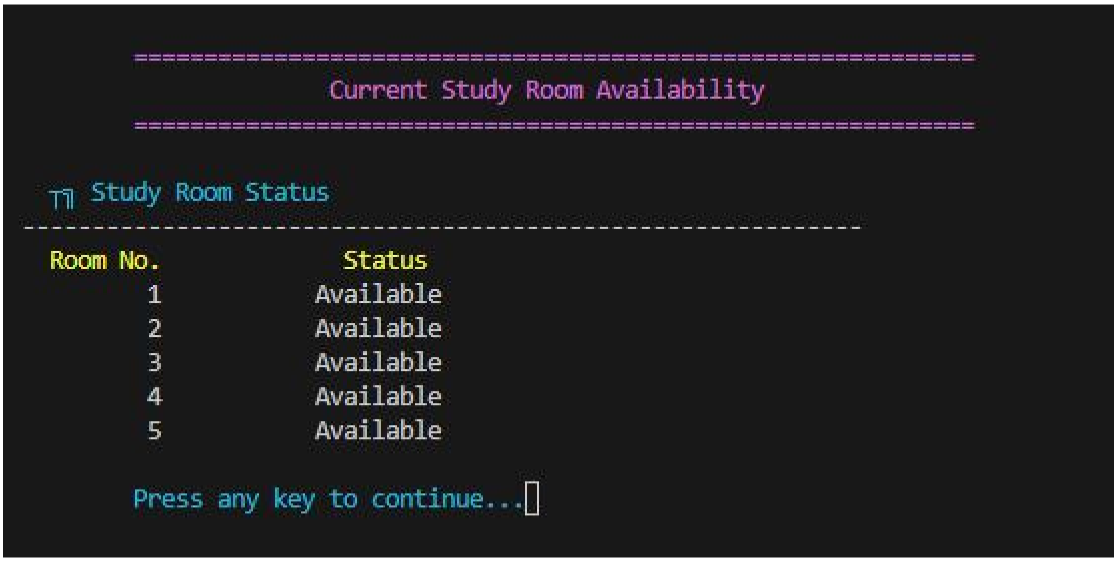


```
============================================================
                    Main Menu - admin
============================================================

1. Book Inventory
2. Issue Book
3. Return Book
4. Study Rooms
5. System Statistics
6. User Management
7. Logout

Enter choice: 4
```

In "View study rooms" we have the following:

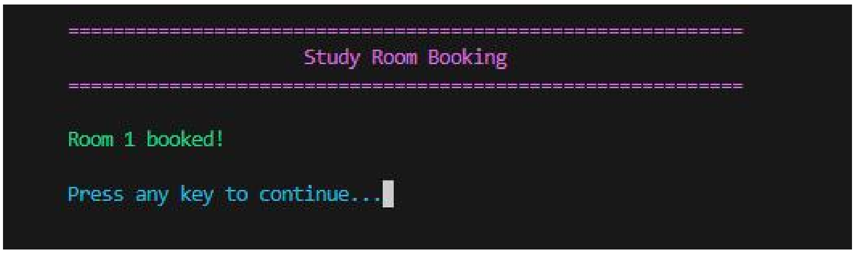


```
============================================================
                  Study Room Management
============================================================

1. View Study Rooms
2. Book a Study Room
3. Back to Main Menu

Enter your choice:
```



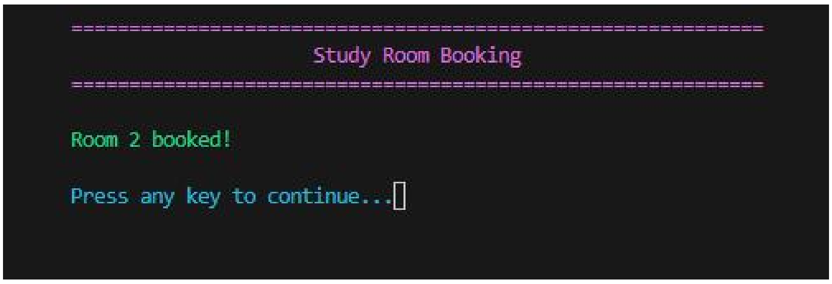
```
============================================================
              Current Study Room Availability
============================================================

   Study Room Status
------------------------------------------------------------
  Room No.              Status
         1            Available
         2            Available
         3            Available
         4            Available
         5            Available

         Press any key to continue...
```

And the booking process is just by choosing the 'book a study room' option like this:

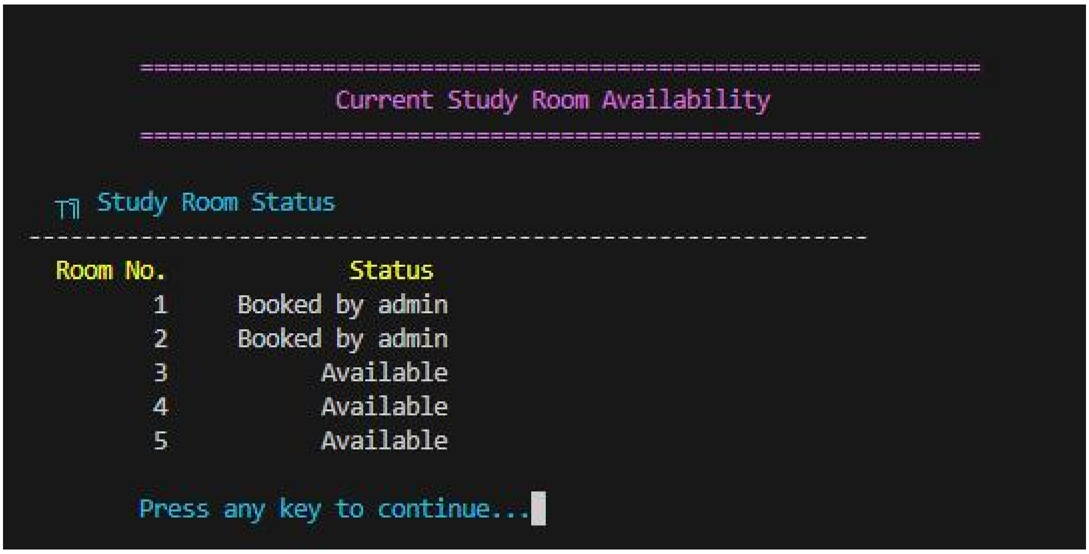


```
============================================================
                    Study Room Booking
============================================================

Room 1 booked!

Press any key to continue...
```

If want to book more rooms then select the option again

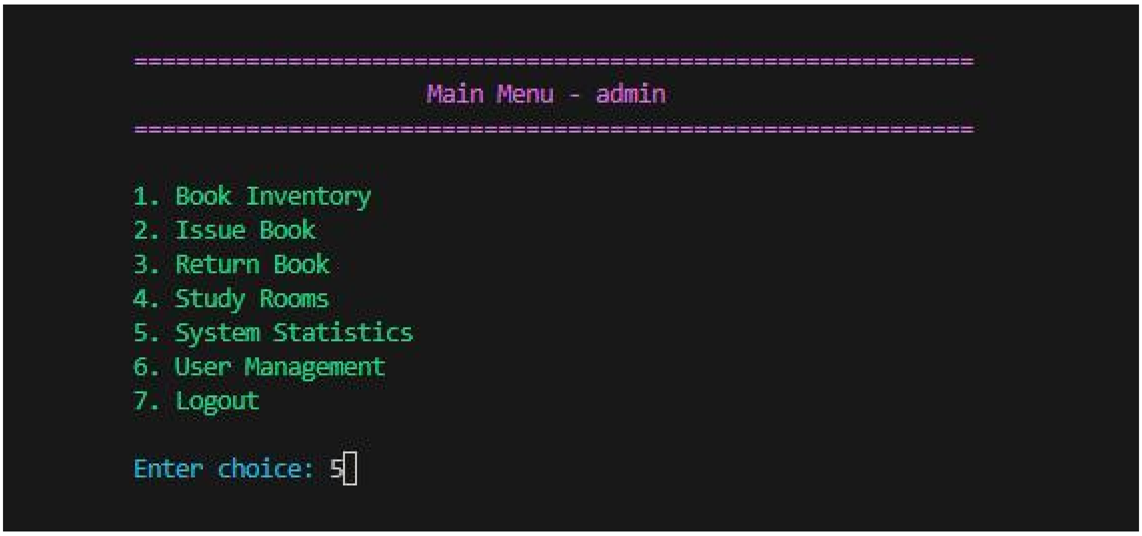


```
============================================================
                    Study Room Booking
============================================================

Room 2 booked!

Press any key to continue...
```

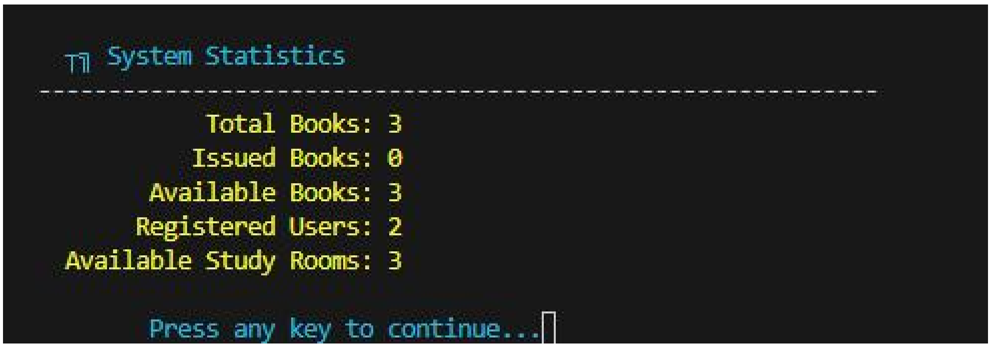


```
      ============================================================
                    Current Study Room Availability
      ============================================================

 ╤╗ Study Room Status
------------------------------------------------------------
 Room No.             Status
        1      Booked by admin
        2      Booked by admin
        3            Available
        4            Available
        5            Available

      Press any key to continue...
```

9. Choosing ‘System statistics’ option from the main menu:

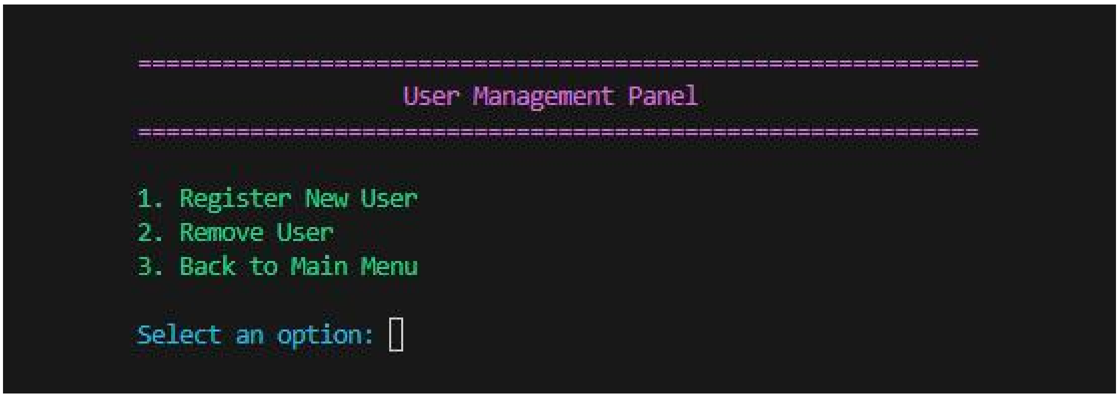


```
      ============================================================
                          Main Menu - admin
      ============================================================

      1. Book Inventory
      2. Issue Book
      3. Return Book
      4. Study Rooms
      5. System Statistics
      6. User Management
      7. Logout

      Enter choice: 5
```



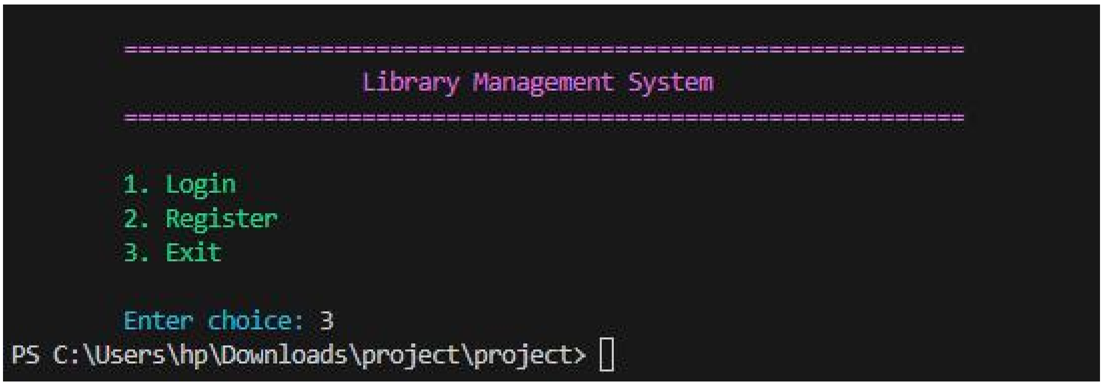
```
 ╤╗ System Statistics
------------------------------------------------------------
           Total Books: 3
          Issued Books: 0
       Available Books: 3
      Registered Users: 2
 Available Study Rooms: 3

      Press any key to continue...
```

10. And finally in the “User Management” from main menu we have the following options:



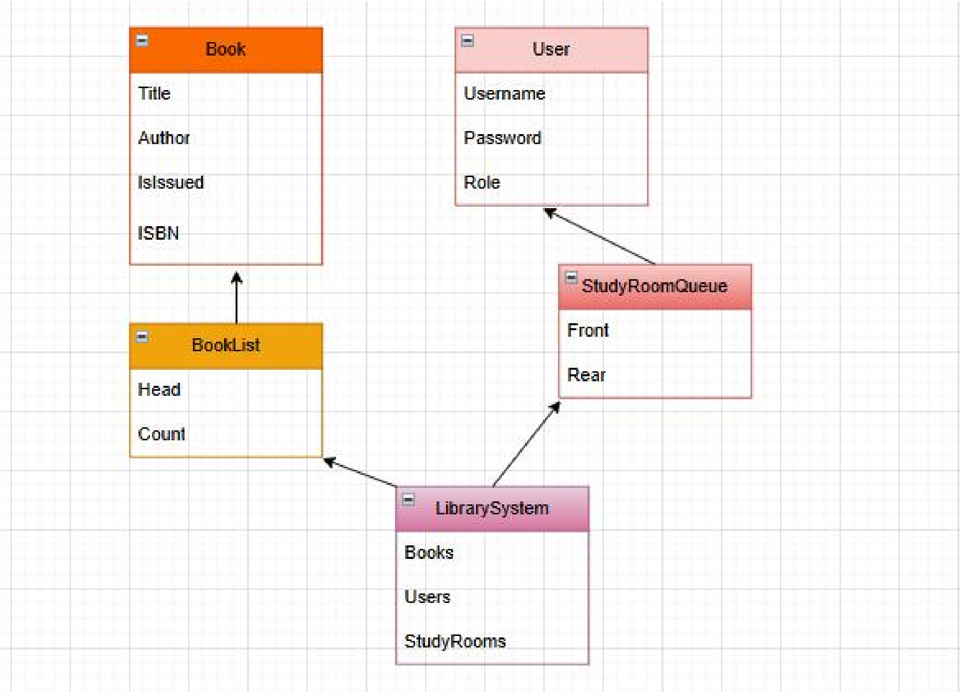
```
============================================================
                  User Management Panel
============================================================

1. Register New User
2. Remove User
3. Back to Main Menu

Select an option: 
```

Tada~ that’s it for the system overview.



```
        ============================================================
                        Library Management System
        ============================================================

        1. Login
        2. Register
        3. Exit

        Enter choice: 3
PS C:\Users\hp\Downloads\project\project> 
```

4. UML Class Diagram



5. Optimization Summary

5.1 Current Optimizations

- Vector Conversion: Linked list → vector for faster sorting.
- Binary Search: Pre-sorting reduces search time to O(log n).

5.2 Future Improvements

1. Balanced BST: Replace linked list with AVL tree for O(log n) insertions/searches.
2. Caching: Store sorted vectors to avoid repeated conversions.
3. Hash Table: O(1) ISBN-based lookups.

6. Conclusion

This Library Management System project successfully demonstrates the practical application of object-oriented programming principles and efficient data structure implementation in C++. By integrating custom data structures (linked lists for book management and queues for study room reservations) with optimized algorithms (Quick Sort and Binary Search), the system provides a robust solution for managing library resources. The challenges faced during memory management and pointer validation were resolved through careful vector allocation and input handling, ensuring stable performance. While the current implementation effectively serves small to medium-sized libraries, future enhancements like balanced BSTs for faster searches and hash tables for instant ISBN lookups could further elevate scalability. Overall, the project highlights the importance of algorithm selection and memory safety in building reliable real-world systems.