BOOK STORE MANAGEMENT USING LINKED LIST

A PROJECT REPORT

Submitted by

GUNTOORI MRUNAL VARMA [RA2211026010156]

APPAJI SREE DHARMA SHASTA RAO [RA2211026010162]

MANDAPATI CHANDRA SEKHAR REDDY [RA2211026010181]

Under the Guidance of

Dr. Prithi Samuel

Assistant Professor, Department of Computational Intelligence

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

with specialization in Artificial Intelligence and Machine Learning

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

NOVEMBER 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the 21CSC201J Data Structures and Algorithms course project report titled **"BOOK STORE MANAGEMENT USING LINKED LIST"** is the bonafide work done by **GUNTOORI MRUNAL VARMA(RA2211026010156), APPAJI SREE DHARMA SHASTA RAO (RA2211026010162), MANDAPATI CHANDRA SEKHAR REDDY (RA2211026010181)** who carried out under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

Dr. Prithi S
Assistant Professor
Department of Computational Intelligence
SRMIST

Dr. R. Annie Uthra
Professor and Head
Department of Computational Intelligence
SRMIST

ABSTRACT

The Book Store Management System using linked lists is a software solution designed to streamline and optimize the operations of a book store. This system leverages the power of linked lists to efficiently manage book inventory, customer information, and sales transactions, ultimately enhancing the functionality and service quality of the book store.

This book management system employs linked list data structures to enhance the efficiency of organizing and managing a library's collection. The system utilizes singly linked lists to represent the dynamic nature of book acquisitions, allowing for seamless additions and removals. Each node in the linked list contains essential book information, such as title, author, genre, and availability status

The system incorporates features such as easy navigation through the linked list for quick access to specific books, efficient updates to accommodate new arrivals or borrowed books, and streamlined removal of outdated or damaged items. By leveraging linked lists, the book management system optimizes memory usage and provides a scalable solution for libraries with varying book quantities.

Additionally, the system ensures data integrity through robust error-checking mechanisms, contributing to a reliable and user-friendly experience for both librarians and patrons. Through the implementation of linked lists, this book management system aims to provide an effective and adaptable solution for modern library collections.

TABLE OF CONTENTS

1. INTRODUCTION

The Book Store Management System is a comprehensive software application designed to streamline the operations of a book store. This system leverages the power of linked lists, a versatile and dynamic data structure, to efficiently manage various aspects of the book store's inventory, customer interactions, and sales transactions. It serves as a pivotal tool for book store owners, managers, and staff to maintain an organized and customer-centric approach to their business.

In the realm of library management, the efficient organization and accessibility of information are paramount. This project introduces a novel approach to book management systems, utilizing linked list data structures to address the dynamic and evolving nature of library collections. Linked lists offer a flexible and scalable solution, enabling seamless updates and adjustments to the database.

1.1 MOTIVATION

The motivation behind this book management system stems from the need for a streamlined and adaptable approach to handle the constant influx and outflow of books within a library. Traditional data structures may struggle to keep pace with these changes, leading to inefficiencies and complexities in managing the catalog.

By embracing linked lists, we aim to create a system that not only optimizes memory usage but also facilitates swift and reliable operations for librarians and patrons alike. The motivation is to enhance the overall efficiency of library management, providing a robust foundation for maintaining accurate records, accommodating new acquisitions, and ensuring a smooth user experience. This project seeks to bridge the gap between traditional library systems and the demands of a dynamic, modern library environment.

1.2 OBJECTIVE

The main objective is to create an efficient and organized inventory management system for a retail store. This system is designed to maintain accurate records of product names, quantities, and brands. It enables the addition, updating, and removal of products, categorizes products by brand, and provides a user-friendly interface for interaction. The system also aims to prevent data errors and enhance customer satisfaction by ensuring products are readily available. It supports data-driven decisions, contributes to cost efficiency, and aids in preventing stock losses. Overall, the objective is to streamline inventory management, ultimately supporting the store's operational success and customer satisfaction. In pursuit of the primary goal of establishing an efficient and organized inventory management system for the retail store, the emphasis is on leveraging cutting-edge technology and user-friendly interfaces. The system not only meticulously records product names, quantities, and brands but also facilitates seamless additions, updates, and removals of products, ensuring the database remains current and reflective of real-time stock levels. A key feature is the categorization of products by brand, fostering a structured approach to inventory organization. The user-friendly interface enhances accessibility, enabling staff to interact with the system effortlessly, thereby minimizing the likelihood of data errors. By prioritizing accuracy, the system contributes to customer satisfaction by ensuring products are consistently available, reducing instances of stockouts or discrepancies between digital records and actual stock. This commitment to accuracy not only supports the store's day-to-day operations but also underpins data-driven decision-making processes, promoting informed choices in purchasing, pricing, and overall inventory management. The system's role extends beyond mere record-keeping; it actively contributes to cost efficiency by optimizing stock levels and preventing unnecessary losses. Ultimately, the overarching objective is to streamline every facet of inventory management, from product entry to customer availability, thereby bolstering the store's operational success and reinforcing customer satisfaction as a cornerstone of its business ethos.

1.3 PROBLEM STATEMENT

Traditional book management systems often face challenges in adapting to the dynamic nature of library collections. Conventional data structures may struggle to efficiently handle frequent additions, removals, and updates to the catalog, leading to increased complexities for librarians. As libraries evolve to meet the diverse needs of their patrons, there is a pressing need for a system that can seamlessly manage these changes while maintaining data accuracy and accessibility.

1.4 CHALLENGES

1.Dynamic Book Collections: Libraries experience a constant influx of new books and the removal of outdated or damaged ones. Conventional systems may struggle to handle these dynamic changes effectively.

2. Memory Efficiency: Efficient memory management is crucial for systems dealing with large datasets. Traditional structures might not provide optimal memory usage, leading to potential performance issues.

3.User Accessibility: Ensuring a user-friendly interface for librarians and patrons is essential. The challenge lies in designing a system that is intuitive, easy to navigate, and capable of providing quick access to relevant book information.

4. Data Integrity: Maintaining accurate and up-to-date records is a challenge, especially when dealing with frequent updates. Ensuring data integrity becomes more complex as the volume of transactions increases.

5. Scalability: Libraries vary widely in size, and the system should be scalable to accommodate both small community libraries and large academic institutions. Designing a solution that can scale seamlessly presents a significant challenge.

2. REQUIREMENTS ANALYSIS

2.1 REQUIREMENT

From the given scenario, we draw the following requirements:
1. User Interface:
- Intuitive interface for librarians and patrons.
- Easy navigation to search, add, update, and remove books.
- User authentication for librarians with different access levels.

2. Book Information:
- Storage for essential book details: title, author, genre, ISBN, availability status, etc.
- Support for additional metadata such as publication date, description, and cover images.

3. Linked List Implementation:
- Implementation of singly linked lists to represent the book catalog.
- Efficient algorithms for insertion, deletion, and traversal of linked lists.

4. Dynamic Updates:
- Ability to add new books seamlessly.
- Efficient removal of books, updating availability status.
- Handling of book status changes, such as borrowed or returned.

5. Memory Management:
- Optimized memory usage through efficient data structures.
- Prevention of memory leaks and efficient garbage collection.

6. User Interaction:
- User-friendly commands for librarians to perform common tasks.
- Search functionality based on various criteria (title, author, genre).

7. Data Integrity:
- Robust error-checking mechanisms to ensure data accuracy.
- Logging and alerting for any inconsistencies in the book catalog.

8. Scalability:
- Design that accommodates varying library sizes and book quantities.
- Efficient scaling with minimal impact on performance.

9. Security:
- Secure user authentication mechanisms to protect sensitive librarian functions.
- Encryption for stored data to safeguard patron privacy.

10. Reporting and Analytics:
- Generation of reports on book availability, popularity, and trends.
- Analytics tools for librarians to make informed collection management decisions.

11. Documentation:
- Comprehensive documentation for system installation, configuration, and usage.
- Code documentation for maintainability and future development.

2.2 HARDWARE REQUIREMENT

1. Processor: Dual-core or higher processor.

2. RAM: 2 GB of RAM or more.

3. Storage: At least 8 GB of internal storage.

4. Display: A screen with a resolution of 720x1280 pixels or higher.

5. Battery: A battery with sufficient capacity to run the device for a reasonable amount of time, typically around 2000 mAh or more.

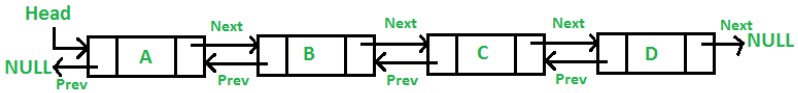
6. Operating System: Android 6.0 (Marshmallow) or higher.

7. Stable internet connection for cloud-based systems.

8. Local area network (LAN) for on-premises installations.

3. DATA STRUCTURE USED

Doubly Linked List (Product Struct): The code defines a custom data structure named 'Product' to represent individual products in the inventory. Each Product object contains data fields for the product name, quantity, and two pointers ('next' and 'Prev') that create a doubly linked list structure. This data structure enables the efficient organization of products in a linked list fashion, facilitating easy traversal and manipulation.



Multimap (product Brand Map): The code uses 'multimap' data structure to map product brands (categorized by the first word of the product name) to their respective 'Product' objects. This multimap data structure allows for the efficient organization and retrieval of products by brand. It supports categorization and quick access to all products belonging to a particular brand, enhancing data organization and retrieval.

Dynamic Memory Allocation (new Product): Dynamic memory allocation is used to create new 'Product' objects, allowing products to be added and removed from the inventory. This dynamic allocation ensures that memory is allocated as needed, reducing wastage and enabling the efficient use of resources.

Traversal and Search (Product Linked List): The code utilizes traversal of the linked list to search for specific products based on their names. It checks whether a product with a given name already exists in the inventory and updates its quantity if found. This traversal involves iterating through the linked list of 'Product objects.

Data Access and Display (Product Brand Map and Display Functions): Data structures are used to access and display the inventory data. The multimap 'productBrandMap 'allows for easy retrieval of products by brand, facilitating the display of products by brand. The code leverages data structures to efficiently present the inventory to the user based on their input.

4. ARCHITECTURE AND DESIGN

The architecture is as follows:

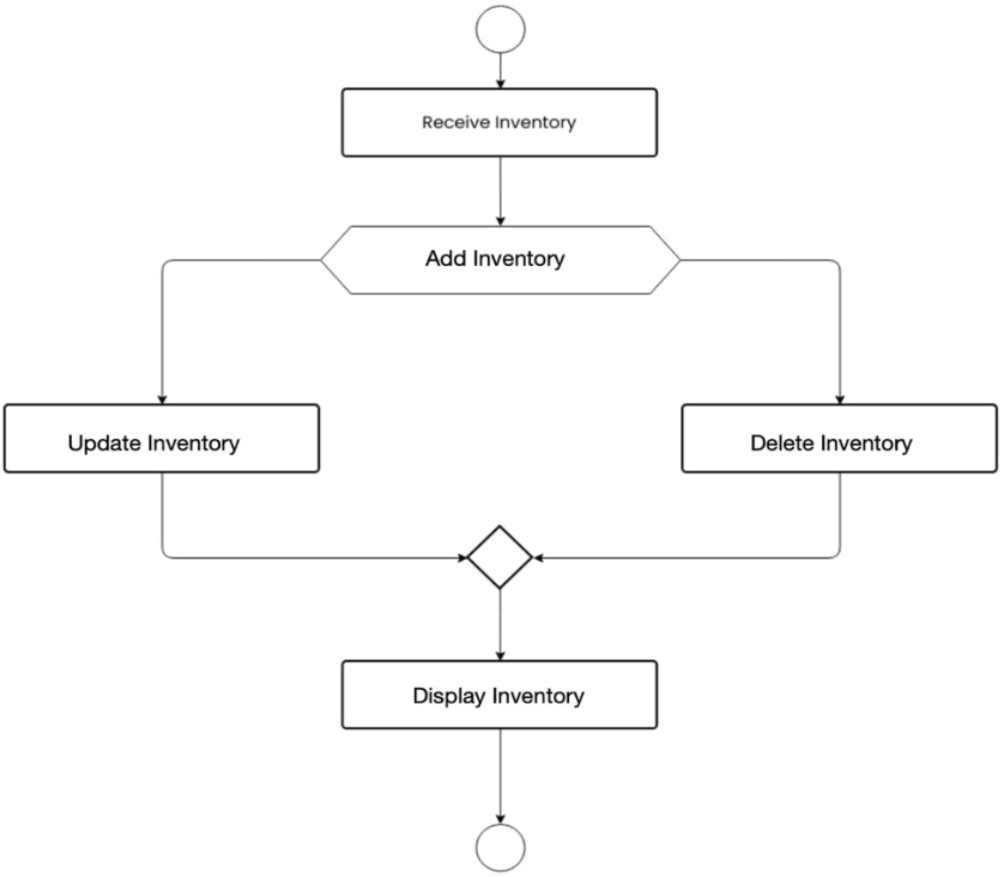


Fig No. 1

The architecture in Fig No. 1 consists of the following management modules:

- User Module
- Inventory Display Module
- Add Inventory Module

These modules are interconnected with each other.

5. IMPLEMENTATION

SAMPLE PROGRAM:

```
#include <iostream>
using namespace std;

int failure=0;
int success=0;
void Trans(int a)                                   //counts the total failed and successful transaction
{
    if(a==0)
        failure++;
    else
        success++;
}
class book
{
private:
    int bookcode;
    string author;
    string title;
    double price;
    string publisher;
    int stock;

    void priceUpdate()                                          //to update price
    {
        cout<<"\nEnter the new price: ";
        cin>>price;
        cout<<endl;
    }
    void stockUpdate()                                           //to update stock
    {
        cout<<"\nEnter the new stock: ";
        cin>>stock;
        cout<<endl;
    }
public:
    book(int u,string x,string y,string z, double w, int v)            //constructor
    {
        bookcode=u;
        title=x;
        author=y;
        publisher=z;
        price=w;
        stock=v;
```

```
    }
    book(string x,string y)                                          //constructor
    {
        author=x;
        publisher=y;
    }
    int search(book x)                                               //searching the book in the list
    {
        if(author==x.author&&publisher==x.publisher)
            return 1;
        else
            return 0;
    }
         void noOfcopies()                              //checking no. of copies and calculating the total pric
    {
        int n;
        cout<<"\nBook Details: "<<endl;
        showdetail();
        cout<<"Enter required number of copies: ";
        cin>>n;
        if(n>stock)
        {
```

```
            cout<<"Required copies is not in stock"<<endl;
            Trans(0);
        }
        else
        {
            cout<<"Total cost of the books: "<<n*price<<endl;
            stock=stock-n;
            cout<<"\nRemaining stock: "<<stock<<endl;
            Trans(1);
        }
    }
   void showdetail()                                                  //display book details
    {
        cout<<"Book code: "<<bookcode<<endl;
        cout<<"Author: "<<author<<endl;
        cout<<"Publisher: "<<publisher<<endl;
        cout<<"Title: "<<title<<endl;
        cout<<"Price: "<<price<<endl;
        cout<<"The stocks available: "<<stock<<endl;
        cout<<endl;
    }
    void update()                                                        //update price and stock
```

```
88    {
89        int x;
90        cout<<"Select what to update\n1.Price\n2.Stock"<<endl;
91        cin>>x;
92        if(x==1)
93            priceUpdate();
94        else
95            stockUpdate();
96    }
97 ;
98
99    book p1(111,"Computer_Architecture","William_Stallings","Pearson",211.5,10);
100   book p2(112,"Operating_Systems","Abraham_Silberchatz","Wiley",250,14);
101   book p3(113,"C++","Herbert_Shildt","McGraw_Hill",312.5,14);
102   book p4(114,"Digital_Electronics","Morris_Mano","Pearson",156,7);
103   book p5(115,"Data_Structures","Ellis_Horowitz","University_Press",235,4);
104
105
106 oid seller()                                                    // seller options
107
108   int a,n;
109   while(1){
```

```
110     cout<<"Enter ur choice: \n1.Show books\n2.Update\n3.Transactions\n4.Exit"<<endl;
111     cin>>a;
112     cout<<endl;
113     switch(a)
114     {
115         case 1: cout<<"\nBook Details: "<<endl;
116                 p1.showdetail();
117                 p2.showdetail();
118                 p3.showdetail();
119                 p4.showdetail();
120                 p5.showdetail();
121                 break;
122
123         case 2: cout<<"Book Update"<<endl;
124                 cout<<"Enter the book code which has to be updated: ";
125                 cin>>n;
126                 if(n==111)
127                     p1.update();
128                 else if(n==112)
129                     p2.update();
130                 else if(n==113)
```

```
                p3.update();
            else if(n==114)
                p4.update();
            else if(n==115)
                p5.update();
            else
                cout<<"Invalid"<<endl;
                break;

        case 3: cout<<"Total failed transaction: "<<failure<<endl;
                cout<<"Total successful transaction: "<<success<<endl;
                cout<<endl;
                break;
        case 4:
                return;
    }
   }
}
void customer()                                         //customer options
{
    int x;
```

```
    while(1)
    {
        cout<<"\nEnter a option \n1.Show Books\n2.Purchase a book\n3.Exit"<<endl;
        cin>>x;
        if(x==1)
        {
            cout<<"\nBook Details: "<<endl;
            p1.showdetail();
            p2.showdetail();
            p3.showdetail();
            p4.showdetail();
            p5.showdetail();
        }

        else if(x==2)
        {
            string a;
            string b;
            cout<<"Enter author name: ";
            cin>>a;
            cout<<"Enter publisher name: ";
```

```
            cin>>b;
            book p6(a,b);
            if(p6.search(p1)==1)
                p1.noOfcopies();
            else if(p6.search(p2)==1)
                p2.noOfcopies();
            else if(p6.search(p3)==1)
                p3.noOfcopies();
            else if(p6.search(p4)==1)
                p4.noOfcopies();
            else if(p6.search(p5)==1)
                p5.noOfcopies();
            else
            {
                cout<<"This book is not available"<<endl;
                Trans(0);
            }
        }
        else
            return;
    }
}

int main()
{
    int a;
    while(1)
    {
        cout<<"\nSelect any one\n1.Seller\n2.Customer\n3.Exit"<<endl;
        cin>>a;
        cout<<endl;
        if(a==1)
            seller();
        else if(a==2)
                customer();
        else
            return 0;

    }
}
```

OUTPUT:

```
Select any one
1.Seller
2.Customer
3.Exit
1

Enter ur choice:
1.Show books
2.Update
3.Transactions
4.Exit
1


Book Details:
Book code: 111
Author: William_Stallings
Publisher: Pearson
Title: Computer_Architecture
Price: 211.5
The stocks available: 10

Book code: 112
Author: Abraham_Silberchatz
Publisher: Wiley
Title: Operating_Systems
Price: 250
The stocks available: 14
```

6. CONCLUSION

The Book Store Management System is a comprehensive software application designed to streamline the operations of a book store. This system leverages the power of linked lists, a versatile and dynamic data structure, to efficiently manage various aspects of the book store's inventory, customer interactions, and sales transactions. It serves as a pivotal tool for book store owners, managers, and staff to maintain an organized and customer-centric approach to their business.

7. REFERENCES

1.GITHUB

2.GEEKOFGODS

3.AI TOOLS