**Project Title: Library Management System.**

**( Using Trees )**

**Developed by**

**Roll No : 22881A05N1**

**Name : K. Shiva Balaji Patel**

**Class : II B.Tech. CSE-D**

**Regulation : R22**

**Under the Guidance of**

**Dr. Vasantha SV, Associate Professor, CSE Dept.**

**Github Link: https://github.com/shiva150/ads\_project.git**

**Problem Statement:**

Design and implement a Library Management System that utilizes a binary tree for storing and managing information about books in a library. The system should provide functionalities for adding books, displaying the book inventory, searching for books, borrowing books, and returning books. The binary tree structure should be used to efficiently organize books based on their titles.

**Algorithm/ Explanation of Logic:**

* **Introduction:**

The primary objective is to design a system that efficiently handles core library operations, including adding books, displaying the book inventory, searching for books, borrowing books, and returning books. The binary tree structure will be implemented to optimize the arrangement and retrieval of books based on their titles, contributing to improved system performance and user experience.

* **Problem Description:**

Traditional library management systems often face challenges in efficiently organizing and managing a large inventory of books. To address these challenges, a modern Library Management System is proposed that leverages a binary tree data structure to enhance the organization and retrieval of book information. The proposed problem involves designing and implementing an innovative Library Management System that utilizes a binary tree data structure to enhance the organization and accessibility of book information.

* **Proposed Solution:**

To address these challenges, the proposed Library Management System aims to leverage a binary tree data structure for efficient organization and retrieval of book information. By implementing this structure, the system intends to optimize core library operations and enhance the overall user experience.

* **Key Features:**

1. **Efficient Book Organization:**

Implement a binary tree to organize books based on their titles, ensuring a balanced structure for efficient search and insertion operations.

1. **User-Friendly Interface:**

Develop a console-based interface that guides users through essential library operations, making it easy for both library staff and patrons to interact with the system.

1. **Optimized Search and Retrieval:**

Leverage the binary tree structure to facilitate quick and efficient searches for books based on their titles, minimizing the time required for book retrieval.

1. **Automated Borrowing and Returning:**

Simulate automated processes for borrowing and returning books, updating the book inventory seamlessly to reflect real-time changes.

1. **Robust Error Handling:**

Implement error-handling mechanisms to address common issues, ensuring the system provides clear feedback and guidance to users.

1. **Optional Data Persistence:**

Optionally incorporate file handling to persist book information, allowing the system to retain data across multiple sessions and safeguard against data loss.

* **User Interface:**

1. The user interface of the Library Management System is designed to be intuitive, user-friendly, and accessible through a console-based interaction. The interface aims to guide users, including both library staff and patrons, through various operations seamlessly.
2. The user interface of the Library Management System is structured to make library operations efficient and straightforward, contributing to an enhanced experience for both library staff and patrons.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define a structure for a book

struct Book {

    int book\_id;

    char title[100];

    char author[50];

    int quantity;

};

// Define a structure for a transaction

struct Transaction {

    char username[50];

    char date[20];

    int quantity;

    struct Transaction\* next;

};

// Define a structure for a book node in the binary tree

struct BookNode {

    int book\_id;

    char title[100];

    char author[50];

    int quantity;

    struct Transaction\* transactions;

    struct BookNode\* left;

    struct BookNode\* right;

};

// Function prototypes

struct BookNode\* createBookNode(struct Book newBook);

struct BookNode\* insertBook(struct BookNode\* root, struct Book newBook);

void displayBooksInOrder(struct BookNode\* root);

struct BookNode\* findBook(struct BookNode\* root, char title[]);

void borrowBook(struct BookNode\* root, char title[], char username[], char date[], int quantity);

void returnBook(struct BookNode\* root, char title[], char username[], char date[], int quantity);

void addTransaction(struct Transaction\*\* transactions, char username[], char date[], int quantity);

void displayTransactions(struct Transaction\* transactions);

void addNewBook(struct BookNode\*\* root);

void clearInputBuffer() {

    int c;

    while ((c = getchar()) != '\n' && c != EOF);

}

int main() {

    struct BookNode\* root = NULL;

    // Example books

    struct Book books[] = {

        {1, "The Catcher in the Rye", "J.D. Salinger", 5},

        {2, "To Kill a Mockingbird", "Harper Lee", 3},

        {3, "1984", "George Orwell", 7}

        // Add more books as needed

    };

    // Insert books into the binary tree

    for (int i = 0; i < sizeof(books) / sizeof(books[0]); i++) {

        root = insertBook(root, books[i]);

    }

    int choice;

    char searchTitle[100];

    char username[50];

    char date[20];

    do {

        // Display menu

        printf("\nLibrary Management System Menu\n");

        printf("1. Display Books\n");

        printf("2. Search Book\n");

        printf("3. Borrow Book\n");

        printf("4. Return Book\n");

        printf("5. Display Transactions\n");

        printf("6. Add New Book\n");

        printf("0. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

         clearInputBuffer();

        switch (choice) {

            case 1:

                printf("Books in the library:\n");

                displayBooksInOrder(root);

                break;

            case 2:

                printf("Enter the title of the book to search: ");

                scanf("%s", searchTitle);

                struct BookNode\* foundBook = findBook(root, searchTitle);

                if (foundBook != NULL) {

                    printf("Book found!\n");

                    printf("Title: %s\nAuthor: %s\nQuantity: %d\n", foundBook->title, foundBook->author, foundBook->quantity);

                } else {

                    printf("Book not found.\n");

                }

                break;

            case 3:

                printf("Enter the title of the book to borrow: ");

                scanf("%s", searchTitle);

                printf("Enter your username: ");

                scanf("%s", username);

                printf("Enter the date (YYYY-MM-DD): ");

                scanf("%s", date);

                int quantityToBorrow;

                printf("Enter the quantity to borrow: ");

                scanf("%d", &quantityToBorrow);

                borrowBook(root, searchTitle, username, date, quantityToBorrow);

                break;

            case 4:

                printf("Enter the title of the book to return: ");

                scanf("%s", searchTitle);

                printf("Enter your username: ");

                scanf("%s", username);

                printf("Enter the date (YYYY-MM-DD): ");

                scanf("%s", date);

                int quantityToReturn;

                printf("Enter the quantity to return: ");

                scanf("%d", &quantityToReturn);

                returnBook(root, searchTitle, username, date, quantityToReturn);

                break;

            case 5:

                printf("Enter the title of the book to display transactions: ");

                scanf("%s", searchTitle);

                struct BookNode\* bookWithTransactions = findBook(root, searchTitle);

                if (bookWithTransactions != NULL) {

                    displayTransactions(bookWithTransactions->transactions);

                } else {

                    printf("Book not found.\n");

                }

                break;

            case 6:

                addNewBook(&root);

                break;

            case 0:

                printf("Exiting the program. Thank you!\n");

                break;

            default:

                printf("Invalid choice. Please try again.\n");

        }

    } while (choice != 0);

    return 0;

}

struct BookNode\* createBookNode(struct Book newBook) {

    // Helper function to create a new book node

    struct BookNode\* newNode = (struct BookNode\*)malloc(sizeof(struct BookNode));

    newNode->book\_id = newBook.book\_id;

    strcpy(newNode->title, newBook.title);

    strcpy(newNode->author, newBook.author);

    newNode->quantity = newBook.quantity;

    newNode->transactions = NULL;

    newNode->left = newNode->right = NULL;

    return newNode;

}

struct BookNode\* insertBook(struct BookNode\* root, struct Book newBook) {

    // Function to insert a book into the binary tree

    if (root == NULL) {

        return createBookNode(newBook);

    }

    // Compare book titles to determine the insertion point

    int compareResult = strcmp(newBook.title, root->title);

    if (compareResult < 0) {

        root->left = insertBook(root->left, newBook);

    } else if (compareResult > 0) {

        root->right = insertBook(root->right, newBook);

    } else {

        // Book with the same title already exists, update quantity or handle accordingly

        root->quantity += newBook.quantity;

    }

    return root;

}

void displayBooksInOrder(struct BookNode\* root) {

    // Function to display books in the binary tree in ascending order of title

    if (root != NULL) {

        displayBooksInOrder(root->left);

        printf("Title: %s\tAuthor: %s\tQuantity: %d\n", root->title, root->author, root->quantity);

        displayBooksInOrder(root->right);

    }

}

struct BookNode\* findBook(struct BookNode\* root, char title[]) {

    // Function to search for a book in the binary tree by title

    if (root == NULL || strcmp(title, root->title) == 0) {

        return root;

    }

    if (strcmp(title, root->title) < 0) {

        return findBook(root->left, title);

    } else {

        return findBook(root->right, title);

    }

}

void borrowBook(struct BookNode\* root, char title[], char username[], char date[], int quantity) {

    // Function to simulate borrowing a book

    // Update the book quantity in the binary tree

    struct BookNode\* book = findBook(root, title);

    if (book != NULL && book->quantity >= quantity) {

        book->quantity -= quantity;

        addTransaction(&(book->transactions), username, date, quantity);

        printf("Book borrowed successfully!\n");

    } else {

        printf("Book not available for borrowing or insufficient quantity.\n");

    }

}

void returnBook(struct BookNode\* root, char title[], char username[], char date[], int quantity) {

    // Function to simulate returning a book

    // Update the book quantity in the binary tree

    struct BookNode\* book = findBook(root, title);

    if (book != NULL) {

        book->quantity += quantity;

        addTransaction(&(book->transactions), username, date, -quantity);

        printf("Book returned successfully!\n");

    } else {

        printf("Book not found.\n");

    }

}

void addTransaction(struct Transaction\*\* transactions, char username[], char date[], int quantity) {

    // Function to add a transaction to the transaction list of a book

    struct Transaction\* newTransaction = (struct Transaction\*)malloc(sizeof(struct Transaction));

    strcpy(newTransaction->username, username);

    strcpy(newTransaction->date, date);

    newTransaction->quantity = quantity;

    newTransaction->next = \*transactions;

    \*transactions = newTransaction;

}

void displayTransactions(struct Transaction\* transactions) {

    // Function to display transactions for a book

    printf("Transaction History:\n");

    while (transactions != NULL) {

        printf("Username: %s\tDate: %s\tQuantity: %d\n", transactions->username, transactions->date, transactions->quantity);

        transactions = transactions->next;

    }

}

void addNewBook(struct BookNode\*\* root) {

    // Function to add a new book to the library

    struct Book newBook;

    printf("Enter the book details:\n");

    printf("Book ID: ");

    scanf("%d", &newBook.book\_id);

    printf("Title: ");

    scanf("%s", newBook.title);

    printf("Author: ");

    scanf("%s", newBook.author);

    printf("Quantity: ");

    scanf("%d", &newBook.quantity);

    \*root = insertBook(\*root, newBook);

    printf("New book added successfully!\n");

}

**Output :**

Library Management System Menu

1. Display Books

2. Search Book

3. Borrow Book

4. Return Book

5. Display Transactions

6. Add New Book

0. Exit

Enter your choice: 1

Books in the library:

Title: 1984 Author: George Orwell Quantity: 7

Title: The Catcher in the Rye Author: J.D. Salinger Quantity: 5

Title: To Kill a Mockingbird Author: Harper Lee Quantity: 3

Library Management System Menu

1. Display Books

2. Search Book

3. Borrow Book

4. Return Book

5. Display Transactions

6. Add New Book

0. Exit

Enter your choice: 2

Enter the title of the book to search: 1984

Book found!

Title: 1984

Author: George Orwell

Quantity: 7

Library Management System Menu

1. Display Books

2. Search Book

3. Borrow Book

4. Return Book

5. Display Transactions

6. Add New Book

0. Exit

Enter your choice: 6

Enter the book details:

Book ID: 1004

Title: shiv

Author: shiva

Quantity: 7

New book added successfully!

Library Management System Menu

1. Display Books

2. Search Book

3. Borrow Book

4. Return Book

5. Display Transactions

6. Add New Book

0. Exit

Enter your choice: 3

Enter the title of the book to borrow: shiv

Enter your username: ads\_shiv

Enter the date (YYYY-MM-DD): 2024-01-01

Enter the quantity to borrow: 2

Book borrowed successfully!

Library Management System Menu

1. Display Books

2. Search Book

3. Borrow Book

4. Return Book

5. Display Transactions

6. Add New Book

0. Exit

Enter your choice: 2

Enter the title of the book to search: shiv

Book found!

Title: shiv

Author: shiva

Quantity: 5

Library Management System Menu

1. Display Books

2. Search Book

3. Borrow Book

4. Return Book

5. Display Transactions

6. Add New Book

0. Exit

Enter your choice: 4

Enter the title of the book to return: shiv

Enter your username: ads\_shiv

Enter the date (YYYY-MM-DD): 2024-01-02

Enter the quantity to return: 1

Book returned successfully!

Library Management System Menu

1. Display Books

2. Search Book

3. Borrow Book

4. Return Book

5. Display Transactions

6. Add New Book

0. Exit

Enter your choice: 5

Enter the title of the book to display transactions: shiv

Transaction History:

Username: ads\_shiv Date: 2024-01-02 Quantity: -1

Username: ads\_shiv Date: 2024-01-01 Quantity: 2

Library Management System Menu

1. Display Books

2. Search Book

3. Borrow Book

4. Return Book

5. Display Transactions

6. Add New Book

0. Exit

Enter your choice: 0

Exiting the program. Thank you!

**Expected Outcome :**

The proposed Library Management System aims to revolutionize library operations by introducing a more efficient and organized approach to book management. By utilizing a binary tree data structure, the system seeks to improve search and retrieval times, automate essential processes, and provide a user-friendly interface for seamless interactions. The expected impact includes enhanced efficiency, reduced manual errors, and an overall improved experience for both library staff and patrons.

***Course-End Project Conclusion :***

*As I conclude my course-end project, we reflect on the journey of creating the Library Management System. This project has been a comprehensive exploration of software development, data structures, and user interaction. Here are the key takeaways:*

*1.* ***Design and Architecture:***

*We successfully designed a robust Library Management System using binary tree data structures to efficiently organize and manage a collection of books.*

*2.* ***User Interaction:***

*The system offers a user-friendly interface, allowing users to seamlessly navigate through various functionalities such as displaying books, searching, borrowing, returning, and viewing transaction history.*

*3.* ***Data Structures:***

*The utilization of binary trees for book organization and linked lists for transaction history showcases our understanding of fundamental data structures and their practical applications.*

*4.* ***Error Handling:***

*We implemented error-handling mechanisms to ensure a smooth user experience. Input validation and clear error messages were incorporated to guide users and prevent unexpected behavior.*

*5.* ***Extensibility:***

*The project demonstrates extensibility by including features like adding new books to the library. This ensures the system can adapt and scale as the library collection grows.*

*6.* ***Real-World Simulations:***

*The borrowing and returning functionalities simulate real-world scenarios, allowing users to interact with the system as they would in an actual library setting.*

*7.* ***Documentation:***

*Thorough code comments, function descriptions, and a proper conclusion provide clarity for both developers and end-users, promoting maintainability.*

*8.* ***Learning Experience:***

*This project has been an invaluable learning experience, allowing us to apply theoretical knowledge into practical implementation. It reinforces concepts learned throughout the course.*

*As I move forward, I carry the skills and insights gained from this project into our future endeavors. I appreciate the opportunity to have a good project like this, and I look forward to applying these skills in real-world scenarios. This project marks the culmination of our efforts, and we are proud of the outcome.*

*Best regards,*

***Shiva Balaji***

***22881A05N1***