

INDEX

S. NO	TITLE	PAGE NO
1	ABSTRACT	4
2	MODULE DESCRIPTION	5
3	USECASE DIAGRAM	7
4	SOURCE CODE	8
5	SCREEN SHOT	18
6	CONCLUSION	24

ABSTRACT

The Library Management System (LMS) is a software solution designed to automate and enhance core library management tasks. It leverages data structures such as arrays, linked lists, queues, binary search trees (BST), and hash maps to manage book cataloging, user accounts, and transactions efficiently. Users can register and log in to borrow or return books, while administrators handle catalog management and user oversight.

Books are categorized into genres using hash maps for quick lookups, and binary search trees ensure optimized searches in a growing database. Borrowing and returning books are managed using queues to maintain order, while user credentials are securely verified with linked lists or hash maps. This system highlights the practical application of DSA concepts to optimize searches and transactions, demonstrating their role in building scalable and efficient library management software.

MODULE DESCRIPTION

User Login/Signup

The User Login/Signup module allows customers to securely create an account and log into the library system. Users enter their details (name, user ID, password) to create a new account. After registration, users can log in with their unique user ID and password. This ensures secure access to their account, allowing them to view and borrow books. Admin login is also available, giving the administrator access to manage book records and user activities. Upon successful login, users can access library features, marking the first layer of security in the system.

DSA Used:

- **Arrays/Structures:** For storing user details (user ID, username, password).
- **Linear Search:** To authenticate the user during login by searching through the array of registered users.
- **Linked List (optional):** For dynamically managing user sessions and login states.

Book Category Management

The Book Category Management section helps categorize books in the library, enabling users to quickly locate books based on categories such as fiction, non-fiction, science, history, etc. Admins can add new categories or modify existing ones, ensuring the library catalog remains organized. The admin also has the capability to add books to the appropriate categories.

DSA Used:

- **Structures:** For storing book details (title, author, category).
- **Array of Structures:** To store books under various categories.
- **Linked List (optional):** To dynamically manage book categories (add, delete, modify).

Book Catalog:

The Book Catalog feature allows users to view a list of books, including their titles, authors, and availability status. Users can browse through the catalog based on their interests and check the availability of books before borrowing them. Admins can also add, update, or remove books.

DSA Used:

- Array of Structures: For storing the list of books.
- Hashing: For efficient searching by book title or author (optional).
- Binary Search: To search for books in a sorted list.

Issue/Return Books

The Issue/Return Books module allows users to borrow and return books. Once logged in, users can view available books and borrow them for a specific period. The system updates the book's status to "issued" and stores the user's details along with the issue and return dates. When a book is returned, the system updates its status to "available," and removes the book from the user's account.

DSA Used:

- Arrays/Structures: For storing book details, issue date, return date, and user details.
- Queue: For managing books in the process of being borrowed or returned (FIFO).
- Linked List: For tracking issued books in a dynamic manner.

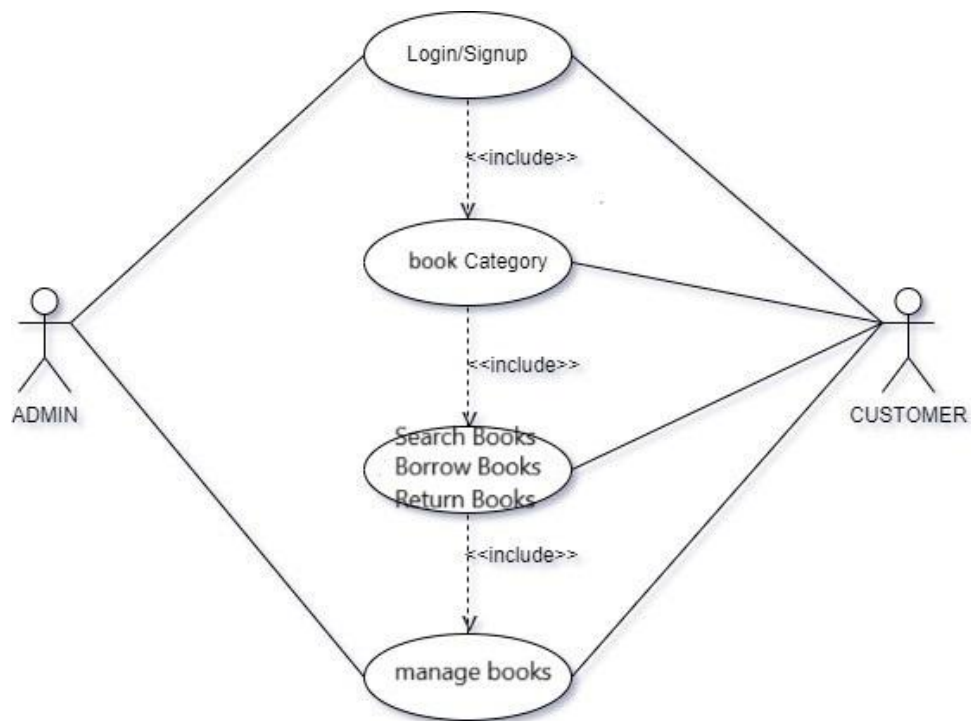
Admin Dashboard

The Admin Dashboard provides a special interface for the library administrator to manage the entire library system. Admins can view user activity, manage book records, and oversee the issue and return of books. Admins can also generate reports about the most borrowed books and user activity.

DSA Used:

- Structures: For organizing and storing user and book records.
- Hash Tables: For efficient searching and categorization of user activity and books.
- Stacks (optional): For handling undo operations or maintaining logs of actions taken by the admin.

USE CASE DIAGRAM



The use case diagram for the Library Management System depicts the interaction between two key actors: Admin and Customer (User). The Admin is responsible for managing the library by logging in or signing up, organizing book categories, and performing operations like adding, updating, or removing books in the system. The Customer, on the other hand, can log in or sign up to search for books, borrow books, and return borrowed books. Core functionalities like Login/Signup act as a prerequisite for all other actions, ensuring secure access and smooth operations. The diagram highlights how Search Books is a critical use case linked to borrowing and returning activities, while book categories and management are controlled by the Admin for better organization. The use of <<include>> relationships emphasizes dependencies between actions, ensuring logical flow and efficient use of the system.

SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_BOOKS 100
#define MAX_USERS 100
#define MAX_CATEGORIES 10
#define MAX_BORROW_DAYS 30

// Structure to store book information
struct Book {
    int id;
    char title[50];
    char author[50];
    char category[30];
    int available; // 1 for available, 0 for issued
    int issuedByUserId; // User ID who borrowed the book
    int daysLeft; // Days left for returning the book
};

// Structure to store user information
struct User {
    int id;
    char username[50];
    char password[50];
    char phoneNumber[20]; // User's phone number
};

// Structure for managing book categories
```

```

struct Category {
    int id;
    char name[30];
};

// Global arrays for storing books, users, and categories
struct Book books[MAX_BOOKS];
struct User users[MAX_USERS];
struct Category categories[MAX_CATEGORIES];
int bookCount = 0, userCount = 0, categoryCount = 0;

// Function prototypes
void addCategory();
void viewBooks();
void addBook();
void issueBook(int userId);
void returnBook(int userId);
void userLogin();
void adminLogin();
void createUser();
int loginUser(int isAdmin);
int searchBookByTitle(char title[]);
int searchBookByAuthor(char author[]);
void adminViewIssuedBooks();
void viewIssuedBooksByUser(int userId);

// Main function
int main() {
    int choice;
    printf("Welcome to the Library Management System!\n");

```

```

while(1) {
    printf("\n1. User Login\n2. Admin Login\n3. Register\n4. Exit\nEnter your choice: ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            userLogin();
            break;
        case 2:
            adminLogin();
            break;
        case 3:
            createUser();
            break;
        case 4:
            exit(0);
        default:
            printf("Invalid choice! Try again.\n");
    }
}
return 0;
}

```

// Function to handle user login

```

void userLogin() {
    if (userCount == 0) {
        printf("No users registered. Please register first.\n");
        return;
    }
}

```

```

printf("User Login\n");

```



```

int userId = loginUser(0);

if (userId != -1) {
    printf("Welcome to the library, %s!\n", users[userId - 1].username);
    int choice;
    while(1) {
        printf("\n1. View Books\n2. Issue Book\n3. Return Book\n4. View Issued Books\n5.
Logout\nEnter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                viewBooks();
                break;
            case 2:
                issueBook(userId);
                break;
            case 3:
                returnBook(userId);
                break;
            case 4:
                viewIssuedBooksByUser(userId);
                break;
            case 5:
                return;
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
}
}

```

```
// Function to handle admin login
void adminLogin() {
    printf("Admin Login\n");
    int adminId = loginUser(1);

    if (adminId != -1) {
        printf("Welcome Admin!\n");
        int choice;
        while(1) {
            printf("\n1. Add Category\n2. Add Book\n3. View Books\n4. View Issued Books\n5.
Logout\nEnter your choice: ");
            scanf("%d", &choice);

            switch(choice) {
                case 1:
                    addCategory();
                    break;
                case 2:
                    addBook();
                    break;
                case 3:
                    viewBooks();
                    break;
                case 4:
                    adminViewIssuedBooks();
                    break;
                case 5:
                    return;
                default:
                    printf("Invalid choice! Try again.\n");
            }
        }
    }
}
```

```

        }
    }
}
}

```

// Function to register a new user

```

void createUser() {
    if(userCount < MAX_USERS) {
        printf("Enter username: ");
        scanf("%s", users[userCount].username);
        printf("Enter password: ");
        scanf("%s", users[userCount].password);
        printf("Enter phone number: ");
        scanf("%s", users[userCount].phoneNumber);
        users[userCount].id = userCount + 1;
        userCount++;
        printf("Thank you for registering! You can now login.\n");
    } else {
        printf("User limit reached! Unable to register.\n");
    }
}

```

// Function to login either user or admin

```

int loginUser(int isAdmin) {
    char username[50], password[50];
    int i;

    printf("Enter username: ");
    scanf("%s", username);
    printf("Enter password: ");
    scanf("%s", password);

```

```

for(i = 0; i < (isAdmin ? 1 : userCount); i++) {
    if (strcmp(isAdmin ? "admin" : users[i].username, username) == 0 &&
        strcmp(isAdmin ? "admin" : users[i].password, password) == 0) {
        return isAdmin ? 0 : users[i].id;
    }
}
printf("Invalid credentials! Try again.\n");
return -1;
}

```

// Function to add a book category

```

void addCategory() {
    if(categoryCount < MAX_CATEGORIES) {
        printf("Enter category name: ");
        scanf("%s", categories[categoryCount].name);
        categories[categoryCount].id = categoryCount + 1;
        categoryCount++;
        printf("Category added successfully!\n");
    } else {
        printf("Category limit reached!\n");
    }
}

```

// Function to view all books

```

void viewBooks() {
    printf("\nBook Catalog:\n");
    if (bookCount == 0) {
        printf("No books available in the catalog.\n");
    } else {
        for(int i = 0; i < bookCount; i++) {

```

```

printf("%d. Title: %s | Author: %s | Category: %s | %s\n",
    books[i].id, books[i].title, books[i].author, books[i].category,
    books[i].available ? "Available" : "Issued");
if (!books[i].available) {
    printf("    Issued to User ID: %d, Days Left: %d, Phone: %s\n",
        books[i].issuedByUserId, books[i].daysLeft, users[books[i].issuedByUserId -
1].phoneNumber);
    }
}
}
}

```

// Function to add a new book

```

void addBook() {
    if(bookCount < MAX_BOOKS) {
        printf("Enter book title: ");
        scanf(" %[^\\n]*c", books[bookCount].title);
        printf("Enter book author: ");
        scanf(" %[^\\n]*c", books[bookCount].author);
        printf("Enter book category: ");
        scanf(" %[^\\n]*c", books[bookCount].category);

        books[bookCount].id = bookCount + 1;
        books[bookCount].available = 1; // Available by default
        books[bookCount].issuedByUserId = 0; // Not issued
        books[bookCount].daysLeft = 0; // No borrowing yet
        bookCount++;
        printf("Book added successfully!\\n");
    } else {
        printf("Book limit reached!\\n");
    }
}

```

```
}
```

```
// Function to issue a book
```

```
void issueBook(int userId) {
```

```
    int bookId;
```

```
    printf("Enter book ID to issue: ");
```

```
    scanf("%d", &bookId);
```

```
    if(bookId > 0 && bookId <= bookCount && books[bookId-1].available) {
```

```
        books[bookId-1].available = 0; // Mark book as issued
```

```
        books[bookId-1].issuedByUserId = userId; // Mark user who borrowed
```

```
        books[bookId-1].daysLeft = MAX_BORROW_DAYS; // Set the borrow period
```

```
        printf("Thank you for issuing the book! Please return it within %d days.\n",
```

```
MAX_BORROW_DAYS);
```

```
    } else {
```

```
        printf("Book is not available for issue.\n");
```

```
    }
```

```
}
```

```
// Function to return a book
```

```
void returnBook(int userId) {
```

```
    int bookId;
```

```
    printf("Enter book ID to return: ");
```

```
    scanf("%d", &bookId);
```

```
    if(bookId > 0 && bookId <= bookCount && !books[bookId-1].available && books[bookId-1].issuedByUserId == userId) {
```

```
        books[bookId-1].available = 1; // Mark book as available
```

```
        books[bookId-1].issuedByUserId = 0; // Clear user who borrowed it
```

```
        books[bookId-1].daysLeft = 0; // Clear days left
```

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

```

    } else {
        printf("Invalid book ID or the book was not issued to you.\n");
    }
}

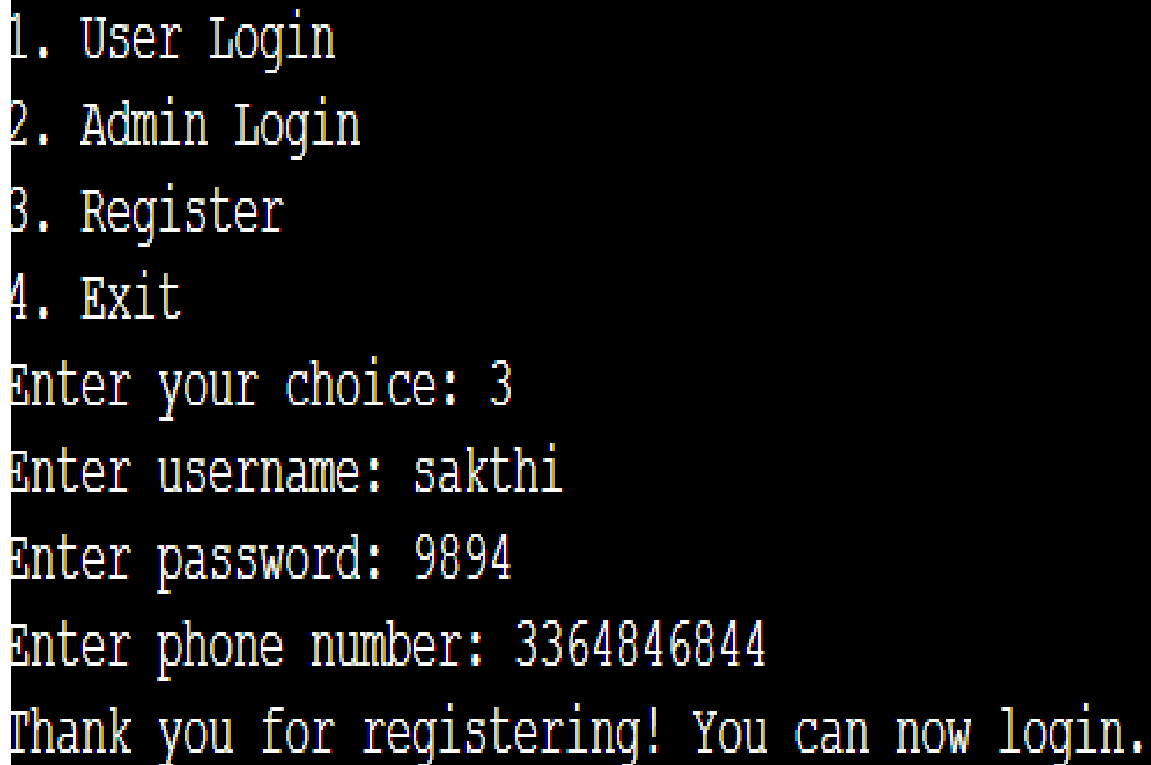
// Function for admin to view all issued books
void adminViewIssuedBooks() {
    printf("\nIssued Books Information:\n");
    for(int i = 0; i < bookCount; i++) {
        if (!books[i].available) {
            printf("Book ID: %d | Title: %s | Issued to User ID: %d | Days Left: %d | User Phone: %s\n",
                books[i].id, books[i].title, books[i].issuedByUserId, books[i].daysLeft,
                users[books[i].issuedByUserId - 1].phoneNumber);
        }
    }
}

// Function to view books issued by a user
void viewIssuedBooksByUser(int userId) {
    printf("\nIssued Books by You:\n");
    int found = 0;
    for(int i = 0; i < bookCount; i++) {
        if (!books[i].available && books[i].issuedByUserId == userId) {
            printf("Book ID: %d | Title: %s | Days Left: %d\n", books[i].id, books[i].title,
books[i].daysLeft);
            found = 1;
        }
    }
    if (!found) {
        printf("No books issued yet.\n");
    }
}

```

SCREENSHOT

Signup:



```
1. User Login
2. Admin Login
3. Register
4. Exit
Enter your choice: 3
Enter username: sakthi
Enter password: 9894
Enter phone number: 3364846844
Thank you for registering! You can now login.
```

Registration process completed successfully. The user account has been created and stored in the system's database. You can now access the Library Management System through the login interface. The registered user has the ability to query the catalog, issue and return books, and track book borrowings. User credentials are securely stored for authentication, allowing access to personalized features. Proceed with logging in to fully interact with the system's functionalities.

Login:

Sign-in process successfully initiated. The system has validated your credentials against the stored user data. Upon successful authentication, access to the Library Management System is granted. You can now perform actions like viewing the book catalog, issuing and returning books, and tracking your borrowed books. User session is securely established, and all actions will be logged for audit purposes. If you wish to end your session, please use the logout functionality to ensure secure access management.

```
1. User Login
2. Admin Login
3. Register
4. Exit
Enter your choice: 1
User Login
Enter username: sakthi
Enter password: 9894
Welcome to the library, sakthi!
```

BOOK CATALOG:

```
1. View Books
2. Issue Book
3. Return Book
4. View Issued Books
5. Logout
Enter your choice: 1

Book Catalog:
1. Title: author of my own destiny | Author: blazei | Category: romance | Available
2. Title: revenge | Author: sakthi | Category: action | Available
3. Title: atomic habits | Author: karthik | Category: novel | Available
```

The Book Catalog displays a comprehensive list of all available books, including details such as the title, author, category, and current availability. Users can view both available and issued books. If a book is issued, details about the user who borrowed it and the number of days left for return will also be displayed. This feature helps users to easily find books they are interested in.

ADD BOOK, ADD CATEGORY:

```
1. Add Category
2. Add Book
3. View Books
4. View Issued Books
5. Logout
Enter your choice: 2
Enter book title: author of my own destiny
Enter book author: blazei
Enter book category: romance
Book added successfully!

1. Add Category
2. Add Book
3. View Books
4. View Issued Books
5. Logout
Enter your choice: 2
Enter book title: revenge
Enter book author: sakthi
Enter book category: action
Book added successfully!
```

In the Admin section, administrators have the capability to add new books to the catalog and manage book categories. The 'Add Book' functionality allows admins to input book details such as title, author, and category. Additionally, new book categories can be added to the system, helping to organize the catalog efficiently for both users and administrators.

ADMIN VIEW ISSUED BOOKS:

```
1. Add Category
2. Add Book
3. View Books
4. View Issued Books
5. Logout
Enter your choice: 4

Issued Books Information:
Book ID: 3 | Title: atomic habits | Issued to User ID: 1 | Days Left: 30 | User Phone: 3364846844
```

The 'View Issued Books' feature in the Admin panel allows administrators to monitor and manage all books that have been borrowed. Admins can view detailed information such as the book ID, title, the user who borrowed the book, their contact details, and the number of days left for each book's return. This ensures effective tracking of issued books, helping admins stay on top of the library's book circulation and manage overdue items more efficiently.

RETURNING A BOOK

```
1. View Books
2. Issue Book
3. Return Book
4. View Issued Books
5. Logout
Enter your choice: 3
Enter book ID to return: 1
Book returned successfully!
```

Returning a borrowed book is a simple process. Users can select the book they wish to return by entering the book ID. Upon confirmation, the system updates the book's status back to 'available,' clears the user details, and resets the due date. This feature ensures proper book tracking and facilitates efficient inventory management.

CONCLUSION

The Library Management System is an efficient solution designed to streamline library operations and resource management. It provides tailored features for users and administrators, digitizing processes like book issuance, returns, and catalog management to reduce manual effort and enhance efficiency.

For users, the system simplifies browsing the book catalog, borrowing items, and tracking borrowed books. Personalized messages such as "Welcome to the library" and "Thank you for issuing the book" create a user-friendly and engaging experience. Its intuitive design ensures accessibility for all users, regardless of technical expertise.

Administrators benefit from comprehensive tools to manage book categories, track issued items, and oversee the library catalog. Features such as adding books and monitoring transactions ensure operational efficiency, while clear messages like "Welcome, Admin" enhance the overall user experience.

In summary, the Library Management System bridges traditional library operations with modern technology, promoting transparency, accountability, and productivity. It serves as a valuable asset for both users and administrators, addressing the evolving needs of modern libraries.