Library Management System

Course: BTECH CSE
Submitted By: Mehak Negi
Sap id: 590026050
Submitted To: Mr. Prashant Trivedi
University: University of Petroleum and Energy Studies

Abstract

The **Library Management System** is a simple and user-friendly software developed in the C programming language to help manage day-to-day activities of a library. Instead of using paper registers or notebooks, this system stores all information digitally using file handling.

The main purpose of this project is to make library operations faster, easier, and more accurate. The system provides two types of users: **Admin** and **User**. Admins can create profiles, log in, add new books to the collection, update existing book details, delete books, and view all users and their comments. Users can create their profile, log in, search for books, view all available books, and leave comments after reading a book.

All information—such as user details, admin details, book records, and comments—is stored permanently in files, so nothing is lost even after the program is closed. The project uses C structures, functions, loops, and file handling to create a modular and efficient system.

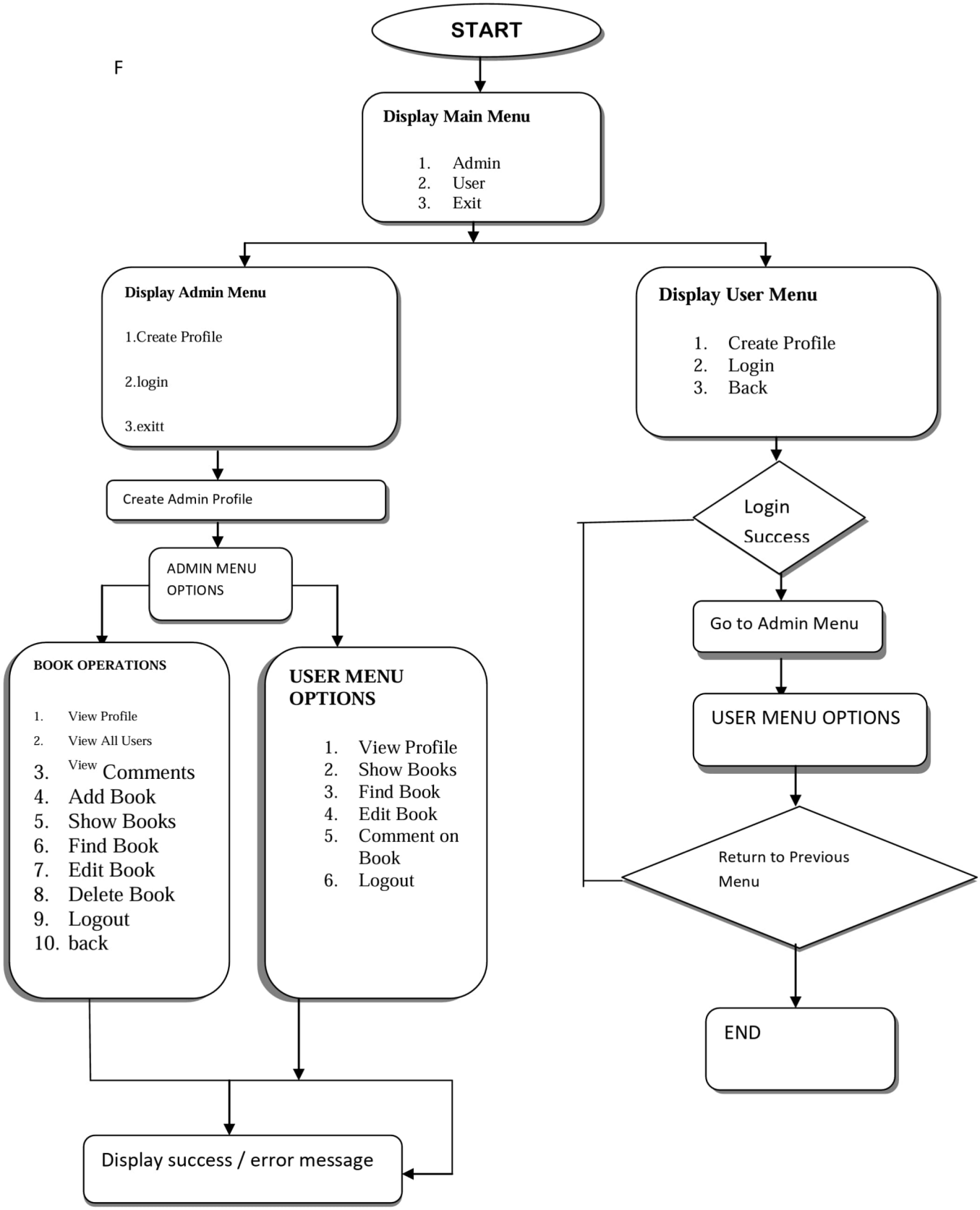
Overall, this project demonstrates how a simple console-based program can replace manual work and improve the management of a library. It reduces human error, saves time, and provides a clear example of how programming can solve real-life problems.

Problem Definition

- Libraries that use **manual registers** face difficulty in managing large numbers of books and users.
- **Searching for a book** in paper records takes too much time.
- High chances of **human errors** such as wrong entry, missing data, or duplicate information.
- Updating book details (edit/delete) becomes slow and complicated when done manually.
- Users cannot easily check whether a book is available without asking the librarian.
- No proper system to store and track **user feedback or comments**.
- Manual methods make the library **less efficient and more time-consuming**.
- There is no quick way to **validate user login** or store passwords securely in a notebook.
- As the number of books grows, maintaining accurate records becomes **difficult**.
- Libraries need a **simple digital system** to store, search, update, and manage information automatically.

Flowchart

F


START
Display Main Menu
1. Admin
2. User
3. Exit
Display Admin Menu
1.Create Profile
2.login
3.exitt
Create Admin Profile
ADMIN MENU OPTIONS
BOOK OPERATIONS
1. View Profile
2. View All Users
3. View Comments
4. Add Book
5. Show Books
6. Find Book
7. Edit Book
8. Delete Book
9. Logout
10. back
USER MENU OPTIONS
1. View Profile
2. Show Books
3. Find Book
4. Edit Book
5. Comment on Book
6. Logout
Display success / error message
Display User Menu
1. Create Profile
2. Login
3. Back
Login Success
Go to Admin Menu
USER MENU OPTIONS
Return to Previous Menu
END

Algorithm

1. **Start the program** and show the main menu with options for Admin, User, or Exit.
2. If the **Admin** chooses their menu, they can create a profile or log in.
3. Once the Admin logs in, they can manage the library by adding books, editing them, deleting them, viewing all users, or checking comments.
4. After finishing any task, the Admin is taken back to the Admin menu.
5. If the **User** chooses their menu, they can create a profile or log in.
6. After a User logs in, they can see their profile, search for books, view all books, or leave comments on books they have read.
7. After completing any action, the User returns to the User menu.
8. If the user selects **Exit** from the main menu, the program closes.
9. **End of the program.**

Testing & Results

Creating User Profile

```
loginF = fopen("long.txt", "a");
printf("Enter ID: ");
scanf("%d", &la.id);
fflush(stdin);
gets(la.name);
gets(la.date);
scanf("%s", la.password);
fwrite(&la, sizeof(la), 1, loginF);
```

Adding a Book

```
file = fopen("books.dat", "ab+");
scanf("%d", &a.id);
scanf("%s", a.name);
scanf("%s", a.author);
scanf("%d", &a.quantity);
scanf("%d", &a.rack);
fwrite(&a, sizeof(a), 1, file);
```

Searching for a Book

```
file = fopen("books.dat", "rb");
while (fread(&a, sizeof(a), 1, file) == 1) {
    if (d == a.id) {
        printf("Book Found!");
    }
}
```

Deleting a Book

Deleting a Book

```
file = fopen("books.dat", "rb");

file2 = fopen("temp.dat", "wb");

while (fread(&a, sizeof(a), 1, file) == 1) {

   if (a.id != d) {

      fwrite(&a, sizeof(a), 1, file2);

   }

}

remove("books.dat");

rename("temp.dat", "books.dat");
```

Testing & Results

1. Admin Profile Creation Test

Input:
ID = 1, Name = "mehak", Date = "15-11-2025", Password = "Mehak1"

Result:
Admin profile should be saved successfully.

Status: PASS

---

2. Admin Login Test

Input:
Correct ID = 1, Correct Password = mehak1

Expected: Successful login
Incorrect Login Test:
ID = 101, Wrong Password = 4321
✔ Shows error message
✔ Attempts reduce
✔ After 3 wrong attempts → program exits

Status: PASS

---

3. Add Book Test

Input:
Book ID = 1001, Name = “no_longer_human”, Author = “osmau_dazai”, Qty = 2, Rack = 1

Expected: Book saved successfully
Duplicate Book Test:
Adding same ID again
✔ System detects duplicate
✔ Shows “Book Already Exists”

Status: PASS

---

4. View All Books Test

Input: User selects “Show Books”
Expected: Display all books in file
Status: PASS

---

5. Find Book Test

Input: Search for Book ID = 1001
Expected: Display book details
Not Found Test:
Search for ID = 999
✔ System shows “Book Not Found”

Status: PASS

---

6. Edit Book Test

Input: Edit Book ID = 1002 and update info
Expected: New details updated

Status: PASS

---

7. Delete Book Test

Input: Delete Book ID = 1006
Expected: Book removed
Status: PASS

---

8. User Profile Creation Test

Input:
ID = 3, Name = “oro”, Date = “20-11-2025”, Password = “oro3”

Expected: User saved
Status: PASS

---

9. User Login Test

Correct Login:
✔ Successful
✔ User menu displayed

Wrong Login:
✔ Shows error message
✔ 3 attempts allowed

Status: PASS

---

10. Comment on Book Test

Input: User comments on Book ID = 3
Comment = "Very helpful book!"

Expected: Comment saved
Status: PASS

Conclusion & Future Work

Conclusion

The Library Management System successfully demonstrates how a simple C program can automate common tasks in a library. By using file handling, the system stores book records, user profiles, admin data, and comments in an organized way without relying on manual registers.

The project reduces human errors, speeds up searching, and makes the process of adding or updating books more efficient. It also ensures that users have access to important features such as viewing books, checking details quickly, and sharing their feedback.

This project clearly shows the importance of computerization in daily operations and highlights how even a basic console application can improve accuracy, save time, and make the library easier to manage. Overall, the system meets its goals of providing a simple, reliable, and easy-to-use digital solution for library management.

Future Work

Although the current system works well as a basic library manager, there are several improvements that can make it more powerful and user-friendly in the future:

1. **Issue and Return System**
   Add a complete borrowing and returning module where users can issue books, check due dates, and track borrowed books.
2. **Fine Calculation**
   Introduce automatic fine calculation for overdue books to make the system more realistic and useful.
3. **Database Integration**
   Instead of using files, integrate a database like MySQL or SQLite for faster, safer, and more scalable data management.
4. **Online/Cloud-Based System**
   Convert the system into a web or cloud-based application so users can check books from anywhere.
5. **User Notifications**
   Add email/SMS notifications for issued books, due reminders, new books, and announcements.
6. **Search Filters**
   Provide advanced search options like search by author, category, year, or keywords.

7. **Admin Dashboard**
   Add statistics like number of users, total books, most-read books, and user activity reports.
8. **Multi-user Access**
   Make the system capable of handling multiple users at the same time.

References

- Online C Documentation (https://www.w3schools.com/, https://www.geeksforgeeks.org/)
- Course Material Provided by teacher