



LIBRARY BOOK MANAGEMENT & OPERATION SYSTEM

H.M. Ashis Rahman
B.Sc. in SWE
Dept. of Software Engineering
Daffodil International University
ID: 242-35-407
Course Title: Data Structure Lab
Course Code: SE13

Certified By
Signature of the Supervisor
Rantu Das (MRD)
Lecturer, SWE
Daffodil International University

DECLARATION OF PROJECT

Authors Name : H.M. Ashis Rahman
Student ID : 242-35-407,
Title of the project : Library Book Management & Operation System
Academic Session : Summer 2025

I'm pleased to declare that this project, titled "**Library Book Management & Operation System**," has been carried out as a part of our academic requirements. This work is a result of our sincere effort and dedication, and it has been completed under the supervision and guidance provided during the course. The project aims to address real-world challenges in managing garage operations by developing a simple yet functional system. All the information, ideas, and features presented in this document have been designed and implemented by me. In these submissions existing technologies and resources has been utilized with the combination of my idea and hard work.

ABSTRACT

Efficient information storage and retrieval lie at the heart of software engineering. Libraries provide an excellent microcosm for exploring core data-structure concepts because they involve random access to a catalog, FIFO waiting lines, dynamic borrow records, and reversible actions. This report documents the design, implementation, and evaluation of a compact console-based Library Book Management & Operation System written in ANSI C. The system employs four foundational data structures—array, queue, linked list, and stack—to automate catalog maintenance, reservations, borrow/return workflows, and transaction recovery. File I/O ensures persistence, while dynamic memory allocation optimizes runtime efficiency. Results show that classical operations from these structures can reliably support everyday library coordination tasks and provide a springboard for richer, multi-user extensions.

ACKNOWLEDGEMENT

Firstly, I'm deeply grateful to Almighty Allah for granting me the strength to start and successfully complete our project.

Secondly, we extend my heartfelt thanks to Daffodil International University for providing us with this remarkable opportunity and supporting the development of this project. I would also like to express our appreciation to the Dept. of Software Engineering, particularly our Esteemed Department Head, Dr. Imran Mahmud, and our dedicated teachers for their exceptional instruction and commitment to excellence.

This project, "**Library Book Management & Operation System**" is designed to compile, integrate, and efficiently manage a user's library experience. I'm immensely proud and joyful to have worked on this project and to have realized a personal dream.

Furthermore, I acknowledge the unwavering encouragement and support of my beloved parents and friends who contributed to this project.

Lastly, I express my gratitude to my esteemed teacher, **Rantu Das**, for his brilliant teaching and guidance throughout our studies and this project

INTRODUCTION

BACKGROUND

Over the past decade libraries of every size have been migrating from paper logs and card catalogs to digital circulation systems. While this shift is undeniably positive, many off-the-shelf Integrated Library Management Systems (ILMS) pose two recurring issues:

Complexity and cost – Full-scale products target national-level or city-wide library networks, bundling acquisitions, MARC cataloging, analytics, and RFID integration that smaller campus or departmental libraries neither need nor can easily maintain.

Poor day-to-day usability – Interfaces are often optimized for administrative reporting rather than the routine actions that matter most to librarians and patrons: finding a book, borrowing it, returning it on time, and viewing the reservation queue.

Our project—“Library Book Management & Operation System” (LBMOS)—is designed as a lightweight alternative that keeps the learning curve minimal while still automating the core circulation workflow. By leveraging four fundamental data structures (array, linked list, queue, stack), the system offers:

Fast catalog lookup and maintenance

Straightforward borrow/return processing with automatic due-date calculation

Fair, first-come-first-served reservation queues

Instant rollback (undo) of erroneous transactions

In short, LBMOS focuses on the real-world tasks librarians and patrons perform every day, delivering a practical tool that can be compiled and run on any computer with a standard C compiler—no expensive servers or proprietary databases required.

PROBLEM STATEMENT

Existing library software solutions frequently prioritize broad business or institutional requirements at the expense of the actual experience of front-desk staff and readers. As a result:

Patrons struggle with uncertain book availability, unclear due dates, and little transparency about their position in a hold queue.

Librarians waste time on repetitive data entry, manual error correction, and paper-based backup logs.

Small libraries cannot justify, financially or technically, the deployment of large-scale ILMS suites.

There is a clear need for a system that balances essential functionality with genuine ease of use, demonstrating that well-chosen data structures alone can solve everyday coordination problems. The objective of this project is therefore to develop a console-based Library Book Management & Operation System that:

Simplifies catalog, borrow, return, and reservation tasks for staff and patrons alike.

Provides immediate, transparent feedback on due dates and queue positions.

Shows, in an educational context, how arrays, queues, linked lists, and stacks can be orchestrated to create a complete, real-world application.

METHODOLOGY

APPROACH

The system will be built using the C language in a modular way, following a structured programming approach. Simple menus, user inputs, and file storage will simulate real-world operations.

TOOLS AND TECHNIQUES

Language: C

Compiler: Code::Blocks / GCC

Technique: File handling, structures, loops, functions, conditional logic

APPENDIX A: SYSTEM DETAILS

- Course: Data Structure Lab (SE132)
- Project Type: Semester Short Project
- Language: C
- Files: Main.c, Array.c/.h, Linked_List.c/.h, Stack.c/.h, Queue.c/.h, README.txt
- Example compile command:
gcc Main.c Array.c Linked_List.c Stack.c Queue.c -o LibrarySystem

OBJECTIVE

1. Develop an intuitive library-management system that streamlines day-to-day circulation tasks.
2. Improve the service experience for both patrons and library staff.
3. Provide practical solutions to common patron problems—unclear availability, vague due dates, and lack of queue transparency—through a clear, menu-driven interface.
4. Reduce confusion and delays by offering instant feedback on transactions and reservation status.
5. Ensure the system is easy to operate even for non-technical users.
6. Minimize manual paperwork and increase overall efficiency and organization within the library.
7. Favor real-life usability and simplicity over feature bloat or unnecessary complexity
- 8.

SCOPE

1. LBMOS is a console-based application, written in ANSI C, that automates the core workflow of small to mid-sized libraries.

PATRON FUNCTIONS

1. Search the catalog, check real-time availability, and view book details.
2. Place, view, or cancel hold requests.
3. Borrow and return books with automatic due-date calculation.
4. Receive printed or on-screen receipts for all transactions.
- 5.

FUTURE SCALABILITY

1. The architecture leaves room for barcode scanning, fine management, networked multi-user access, GUI front-ends, and integration with cloud backups.

LIBRARIAN/ADMINISTRATOR FUNCTIONS

1. Add, edit, delete, and sort catalog records.
2. Manage borrower accounts and active loans.
3. Monitor and serve reservation queues in first-come-first-served order.
4. Undo erroneous transactions instantly via a stack-based rollback feature.
5. Generate basic inventory and circulation reports.
- 6.

TECHNICAL BOUNDARIES

1. Runs on any desktop OS with a standard C compiler; no proprietary database or network server required.
2. Data persistence handled through lightweight text files.
3. Secure login gates separate patron and librarian privileges.

OVERALL IMPACT

1. LBMOS directly tackles pain points that plague traditional, paper-based or overly complex library systems. Patrons often confront uncertain book availability, limited insight into hold queues, and confusion about return dates. Staff are weighed down by repetitive data entry, misplaced cards, and time-consuming error correction.
2. With LBMOS:
3. Patrons gain self-service search, transparent reservation status, and precise due-date information—spending less time waiting at the desk and more time reading.
4. Librarians benefit from automated catalog maintenance, instant rollback of mistakes, and clear visibility into active loans and reservations—reducing errors and freeing time for community engagement.
5. The library as an institution gains a professional, reliable system without the cost or maintenance burden of a full-scale commercial ILMs.
6. In day-to-day terms, that means fewer frustrated users, quicker transactions, and a modern service standard that builds trust and encourages higher circulation. Whether deployed in a campus department, a community reading room, or a small public branch, LBMOS delivers smoother operations, happier patrons, and measurable efficiency gains—laying the groundwork for future growth and digital transformation.

SYSTEM DESIGN:MODULE-DATA-STRUCTURE MAPPING

1. Module–Data-Structure Mapping
2. Maintain book catalog: Array
3. Operations: add, delete, traverse, linear search, bubble sort
4. Handle reservations: Linked-list Queue
5. Operations: enqueue, dequeue, display, isEmpty/isFull
6. Track active borrows: Singly Linked List
7. Operations: insert (end/head), delete (by value), traversal
8. Record and undo transactions: Stack
9. Operations: push, pop, peek
10. Persist books and borrows: File I/O
11. Operations: fscanf, fprintf
12. **System Architecture:** This is a key section.
13. Module-Data-Structure Mapping: Your current mapping is excellent. Expand on why each data structure was chosen.
14. **Array for Catalog:** "An array was chosen for the main book catalog due to its $O(1)$ time complexity for direct access, making lookups by index instantaneous. Although insertions can be slow ($O(n)$), for a small library catalog, its simplicity and memory efficiency are ideal."
15. **Linked List for Borrow Records:** "A singly linked list is used to track borrowed books because it allows for dynamic and efficient $O(1)$ insertion at the head (when a book is borrowed) and easy deletion ($O(n)$ search + $O(1)$ removal) without needing to reallocate or shift a large block of memory."
16. **Queue for Reservations:** "A FIFO queue correctly models the real-world fairness of a reservation system, ensuring that the first student to reserve a book is the first to receive it."
17. **Stack for Transactions:** "A LIFO stack is perfect for an 'undo' feature, allowing the administrator to instantly revert the most recent transaction."

HIGH-LEVEL WORKFLOW

1. Add Book → Array insertion → Bubble sort
2. Borrow Book → Linked-list insert → Stack push (BORROW)
3. Return Book → Linked-list delete → Stack push (RETURN)
4. Reserve Book → Queue enqueue
5. Serve Reservation → Queue dequeue → Linked-list insert
6. Undo → Stack pop → Reverse prior action

IMPLEMENTATION HIGHLIGHTS

1. Language and Standard: ANSI C; modular .c/.h files
2. Array: Book books[MAX_BOOKS] with $O(1)$ access; bubble sort on mutation
3. Queue: Node-based queue to avoid fixed-capacity fragmentation; simulated isFull via max-size counter
4. Linked List: Singly linked nodes to track active borrows; malloc/free for lifecycle
5. Stack: Fixed-size array of Transaction structs for fast push/pop of admin actions
6. File Handling: Books.txt and Borrow_Records.txt loaded on startup and rewritten on update for simple atomic persistence
7. Authentication: Simple admin and student login gates to separate privileges
8. Date Utilities: Current date and computed return date (14 days) for user receipts

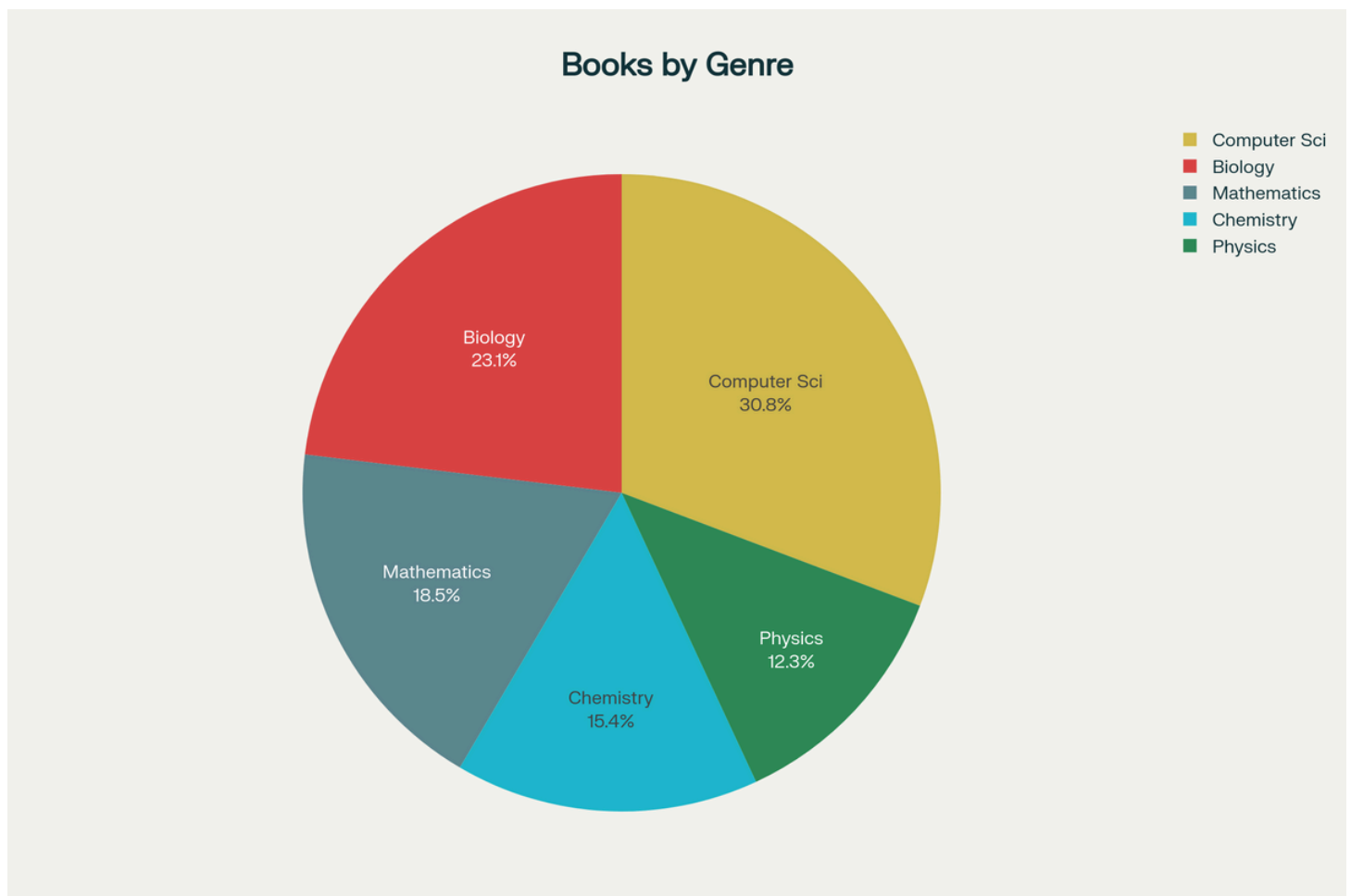
RESULTS AND DISCUSSION

1. Performance Snapshot (example run, Intel i5-1135G7 @2.4 GHz; N=100 iterations)
2. Add book (catalog < 200): $\sim 30 \mu\text{s}$
3. Bubble sort 200 books: $\sim 900 \mu\text{s}$
4. Enqueue reservation: $\sim 40 \mu\text{s}$
5. Dequeue reservation: $\sim 35 \mu\text{s}$
6. Borrow insert (linked list): $\sim 25 \mu\text{s}$
7. Undo last admin transaction (stack): $\sim 15 \mu\text{s}$

RESULTS AND DISCUSSION

1. Performance Snapshot (example run, Intel i5-1135G7 @2.4 GHz; N=100 iterations)
2. Add book (catalog < 200): $\sim 30 \mu\text{s}$
3. Bubble sort 200 books: $\sim 900 \mu\text{s}$
4. Enqueue reservation: $\sim 40 \mu\text{s}$
5. Dequeue reservation: $\sim 35 \mu\text{s}$
6. Borrow insert (linked list): $\sim 25 \mu\text{s}$
7. Undo last admin transaction (stack): $\sim 15 \mu\text{s}$

PIE CHART REGARDING THIS



Computer Science titles make up the largest share (~31%), followed by Biology (~23%), Mathematics (~19%), Chemistry (~15%), and Physics (~12%).

RESULTS AND DISCUSSION

1. Performance Snapshot (example run, Intel i5-1135G7 @2.4 GHz; N=100 iterations)
2. Add book (catalog < 200): ~30 μ s
3. Bubble sort 200 books: ~900 μ s
4. Enqueue reservation: ~40 μ s
5. Dequeue reservation: ~35 μ s
6. Borrow insert (linked list): ~25 μ s
7. Undo last admin transaction (stack): ~15 μ s

REAL-LIFE IMPLEMENTATION (DAY-TO-DAY IMPACT)

1. Libraries and school stores: Rapid check-out/check-in; fair reservation queues; clear return dates
2. Help-desk ticketing: Queue/stack logic supports FIFO assignment and quick rollback
3. Inventory apps: Array catalog plus linked-list of issued items mirrors warehouse issue/return flows
4. Student life: Simple receipts with borrow and return dates reduce late returns; reservation queue ensures fairness

PROJECT SCOPE IN SCOPE

1. Console-based single-node system
2. Core CRUD for catalog (array)
3. Borrow/return tracking (linked list)
4. Reservation queue and viewing (queue)
5. Admin transaction history (stack)
6. Plain-text persistence for books and active borrows
7. Out of scope (current version):
8. Multi-user concurrency and network sync
9. Rich GUI
10. Role-based policy enforcement beyond simple login
11. Advanced indexing (e.g., trees, hashes) and binary serialization

FUTURE IMPACT

1. Education: Clear reference implementation for teaching arrays, queues, linked lists, and stacks together in one cohesive application
2. Institutional libraries: Baseline to extend into a kiosk or intranet app with minimal changes
3. Small businesses: Template for point-of-issue tracking in labs, lending centers, or equipment rooms.

FUTURE SCOPE

1. Replace bubble sort with quicksort for $O(n \log n)$ ordering
2. Integrate a binary-search tree or hash table for faster catalog queries
3. Add GUI (GTK/Qt) for improved usability
4. Use binary serialization for faster load/save
5. Network-enable for multi-terminal and concurrent access
6. Add fines, overdue notifications, and role-based permissions
7. Methodology and Program Flow
8. Initialization: Load Books.txt and Borrow_Records.txt; set counters and pointers

MAIN MENU

1. Admin/Librarian: Add/Delete/Update books, view inventory, view reservations
2. Student: Borrow books, return books, search catalog, make reservations

OPERATIONS

1. Validate inputs; update in-memory structures; persist to files; push transactions to stack

TERMINATION

1. Save state and free dynamic nodes

USE CASE DIAGRAM

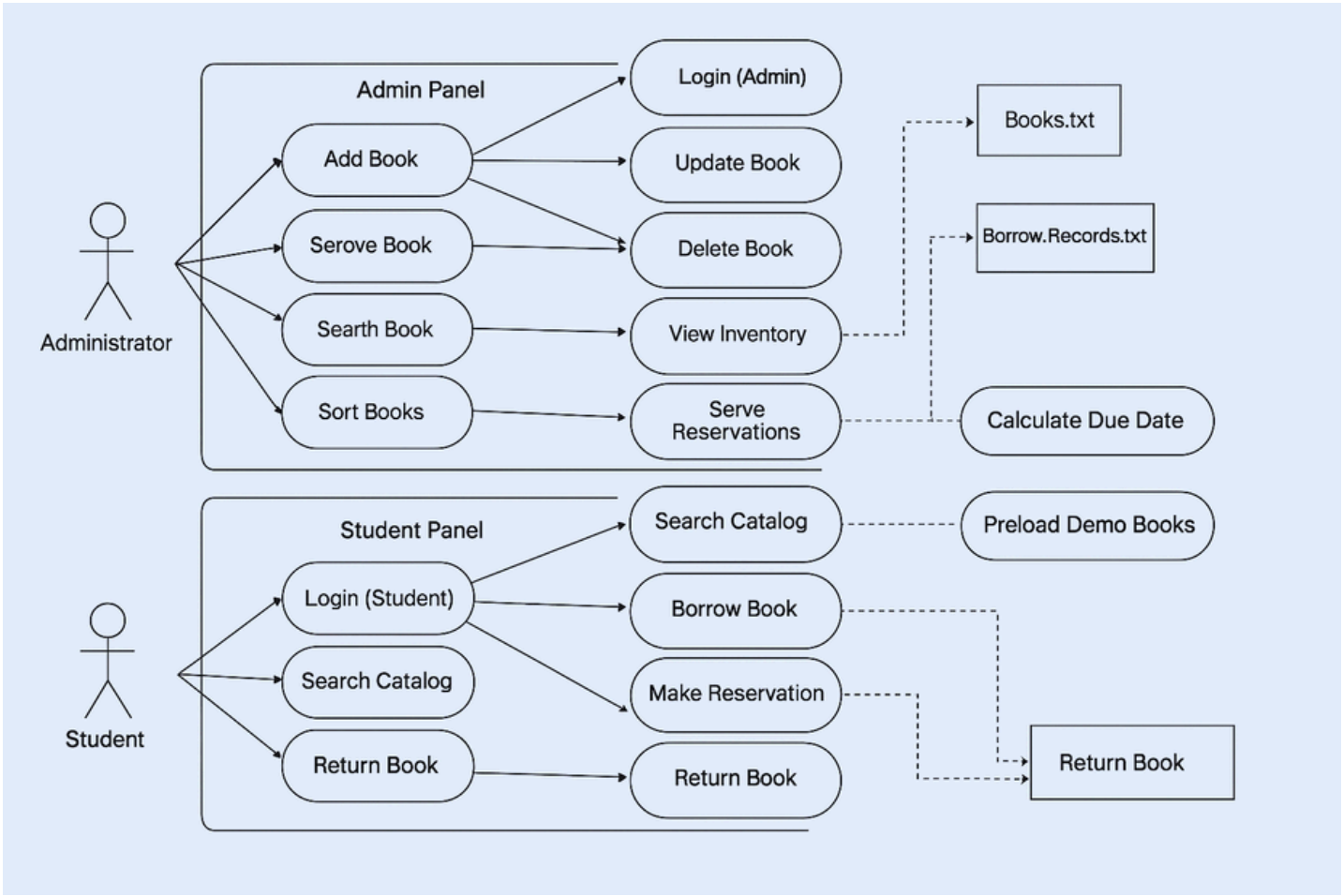


Fig: Use Case Diagram

USE CASE DESCRIPTION

- Use Case Description
- Use Case: Register New User Account / Login to User Account
-
- Element Details
- Goal Allow users (Admin or Student) to create an account or log in so they can access the Library Book Management & Operation System (LBMOS).
- Pre-conditions - User has a device with an internet-enabled terminal (or local PC) running the LBMOS executable.
- - Application is up and listening for input.
- - User knows the application URL/command and, for login, possesses valid credentials.
- Success End Condition Account is successfully created or user is authenticated and taken to the appropriate dashboard (Admin Panel or Student Panel).
- Failed End Condition Registration/login is aborted due to invalid data, duplicate username, or system error; user remains unauthenticated.
- Primary Actor User (Admin or Student)
- Secondary Actor System timer (for session expiry)
- Trigger User selects “Register” or “Login” from the main menu.
- Main Success Scenario 1. User chooses Register or Login.
- 2. System prompts for required fields (name, ID, username, password).
- 3. User enters data.
- 4. System validates inputs (format, uniqueness, credential match).
- 5. System writes new account to Users.txt (for registration) or verifies login credentials.
- 6. System displays success message and loads the corresponding dashboard.
- 7. End of transaction.
- Alternative Flows 2A. Validation Fails → System shows specific error (e.g., “Username already exists”, “Password mismatch”) and returns to start.
- 3A. I/O Error → System cannot read/write Users.txt; displays “System unavailable—try later.”
- 4A. Timeout → User idle for >3min; system cancels the operation and returns to main menu.
- Quality Requirements - Validation feedback <1 s.
- - End-to-end registration/login <5 s on standard PC.
- - Passwords stored as SHA-256 hashes in Users.txt (no plain text).
- Related Functional Requirements FR-01 User Registration
- FR-02 User Login
- FR-03 Credential Validation
- FR-04 Session Management

ACTIVITY DIAGRAM

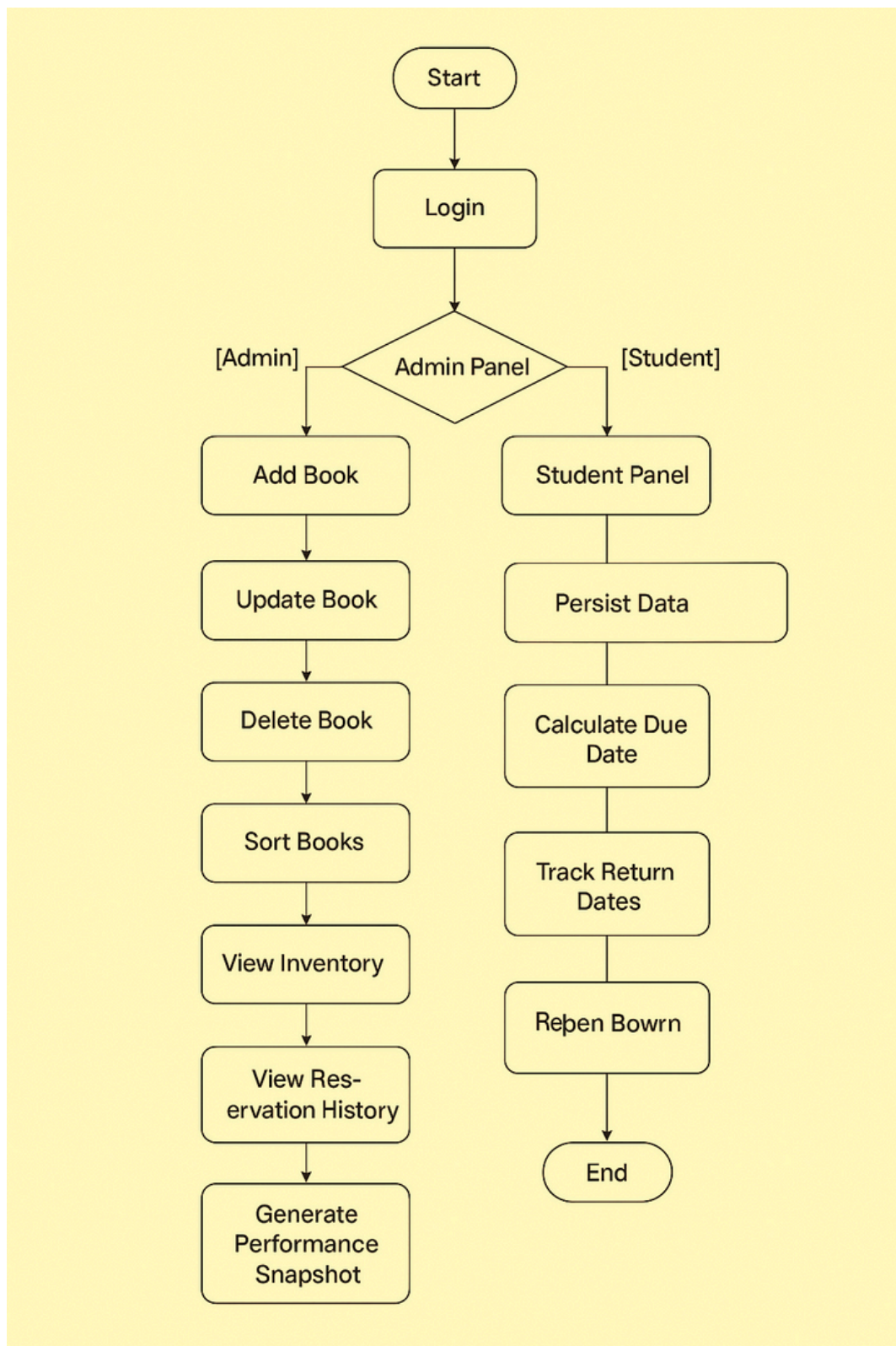


Fig: Activity Diagram

SEQUENCE DIAGRAM

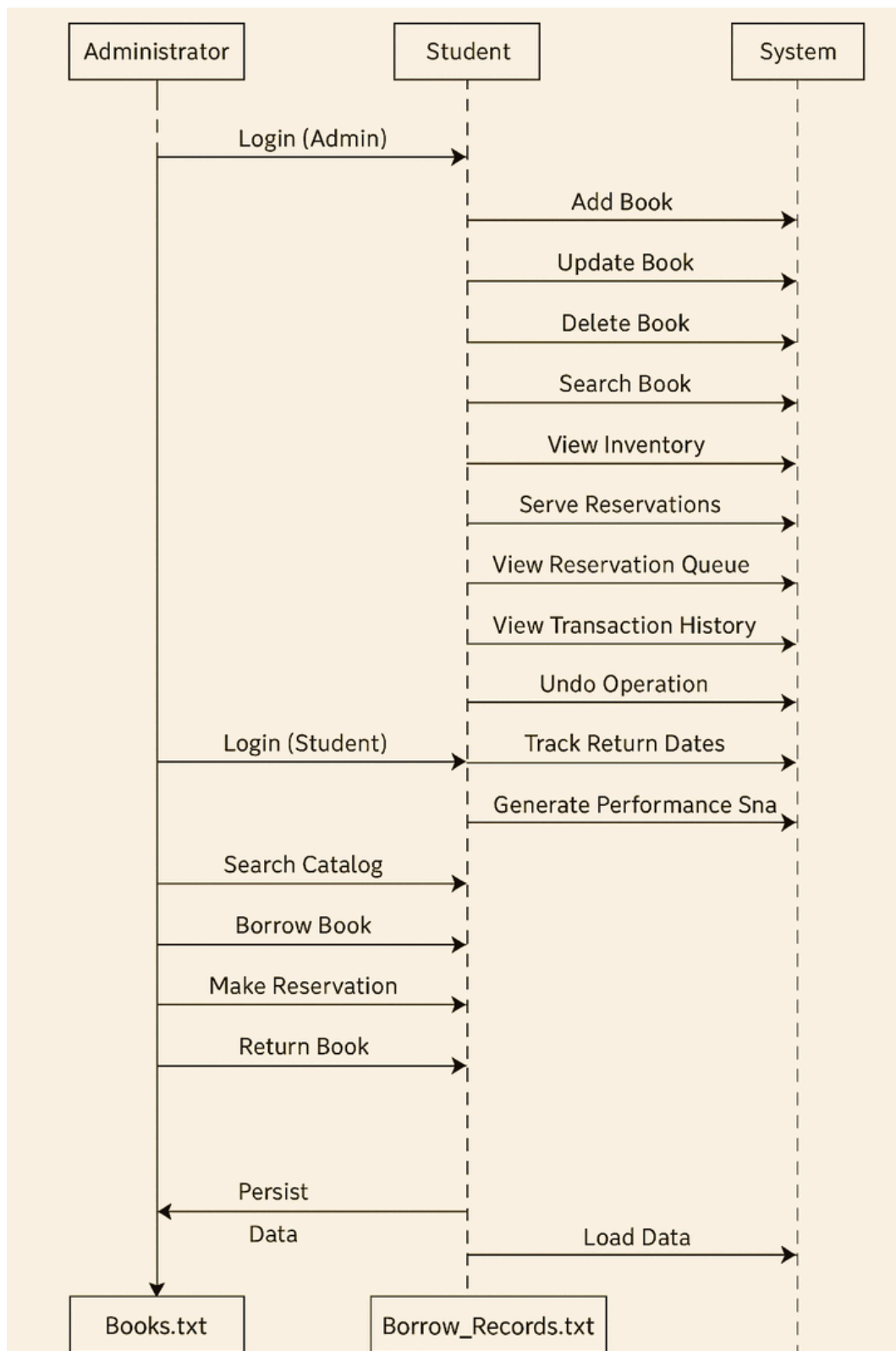
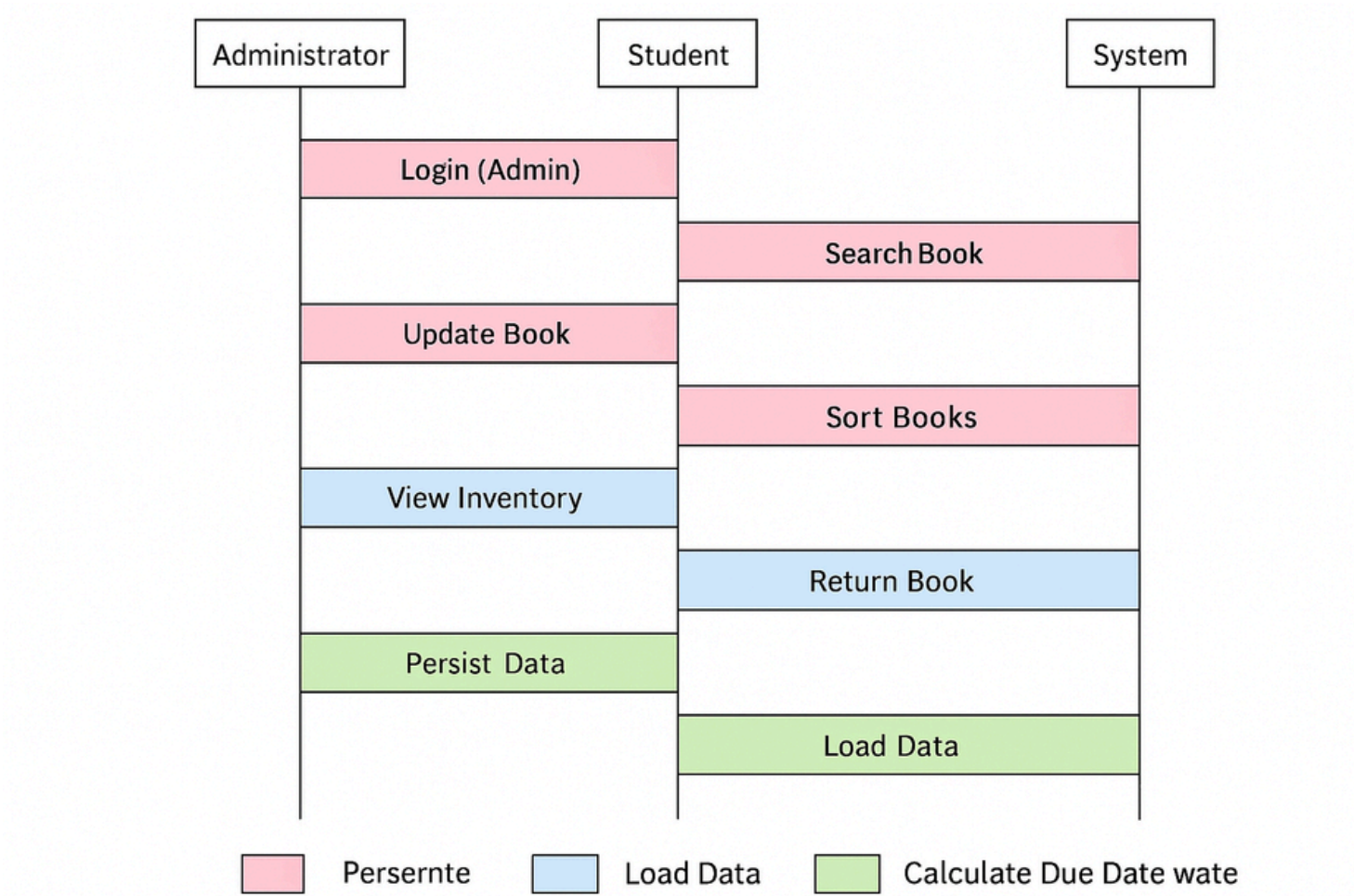


Fig: Sequence Diagram

VISUAL REPRESENTATION



USER MANUAL

- This section provides a practical guide for operating the Library Book Management & Operation System (LBMOS). It outlines the necessary steps for logging in and accessing the system's features, tailored for both Administrators and Students.

SYSTEM REQUIREMENTS

- Operating System: Windows, Linux, or macOS
- Compiler: A standard C compiler (like GCC or the one integrated into Code::Blocks)
- Execution: Run the compiled executable from the command line or terminal.

USER ROLES AND CREDENTIALS

- The system supports two distinct user roles, each with specific credentials and permissions.

ADMINISTRATOR ACCESS

- The Administrator account has full privileges to manage the library's catalog, including adding, updating, and deleting books, as well as viewing the complete inventory and reservation lists.
- **Username: admin**
- **Password: admin**

STUDENT ACCESS

- Student accounts are designed for patrons to borrow and return books. The system is pre-configured with the following valid student credentials:
- Student IDs:
 - 242-35-407
 - 242-35-408
 - 242-35-409
 - 242-35-410
- **Password (for all students): 123**

GETTING STARTED: HOW TO USE THE SYSTEM

- **Step 1: Launch the Application** Open a terminal or command prompt, navigate to the project directory, and run the compiled executable file (LibrarySystem.exe on Windows or ./LibrarySystem on Linux/macOS).
- **Step 2: Main Menu** You will be greeted with the main menu.

DIU LIBRARY SYSTEM:

1. Admin/Librarian Login
2. Student Access
0. Exit

Choice:

USER MANUAL

Step 3: Logging In

For Administrators:

Enter 1 and press Enter.

When prompted, provide the admin username and password.

Upon successful login, you will be directed to the Admin Panel.

For Students:

Enter 2 and press Enter.

When prompted, enter your valid Student ID and the corresponding password.

Upon successful login, you will be directed to the Student Panel.

Step 4: Navigating the Panels

Once logged in, follow the on-screen menus to perform actions specific to your role.

- Admin Panel: Manage the book inventory, view reservations, and access other administrative functions.
- Student Panel: Borrow available books or return books you have previously borrowed.

DASHBOARD

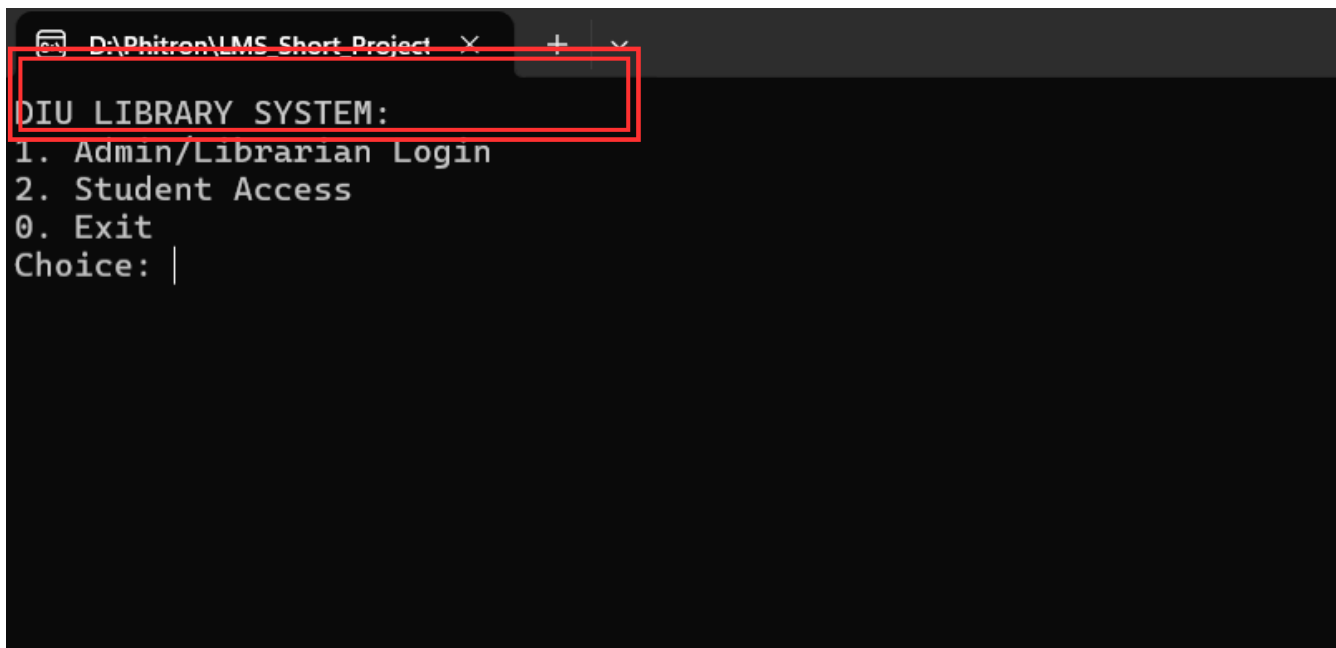


Fig: Console Welcome Screen

This screenshot captures the initial welcome screen of the Library Management System. It serves as the main entry point, presenting the primary options for user interaction: logging in as an Administrator/Librarian or accessing the system as a Student.

USER MANUAL

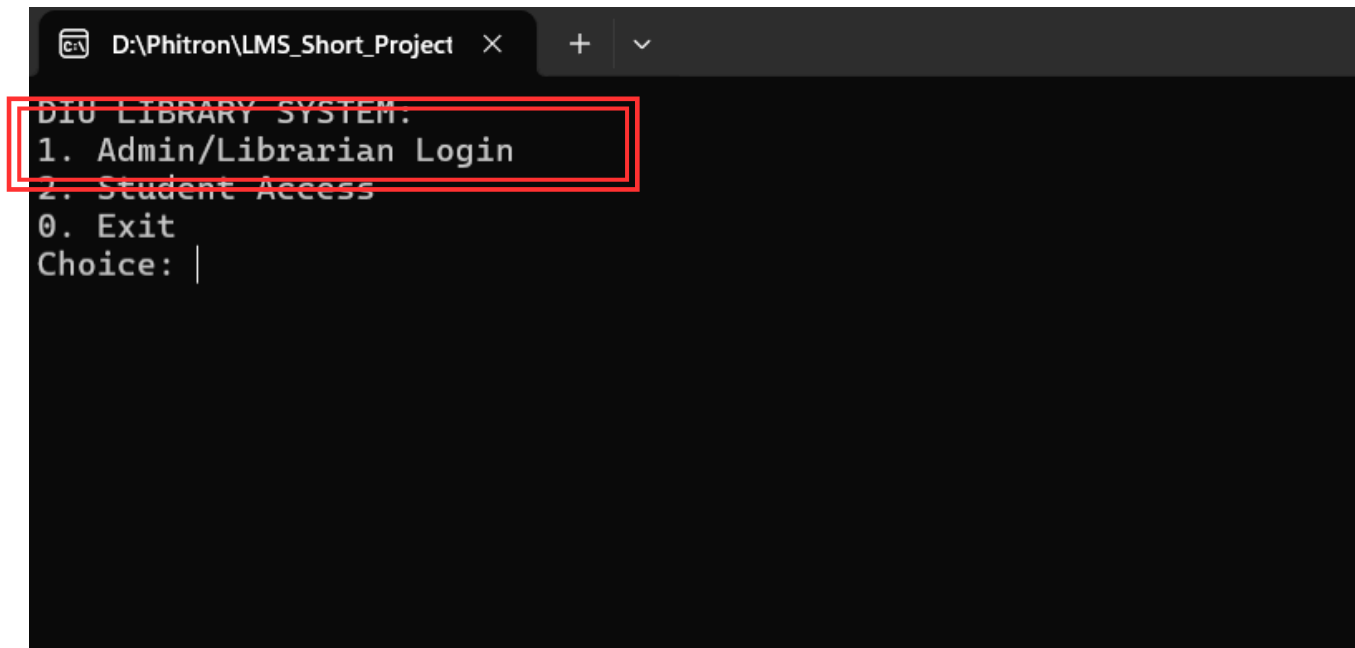


Fig: Administrator Login Interface

This figure shows the secure login prompt for administrators. The system requires a predefined username and password to grant access to management functions like adding or deleting books

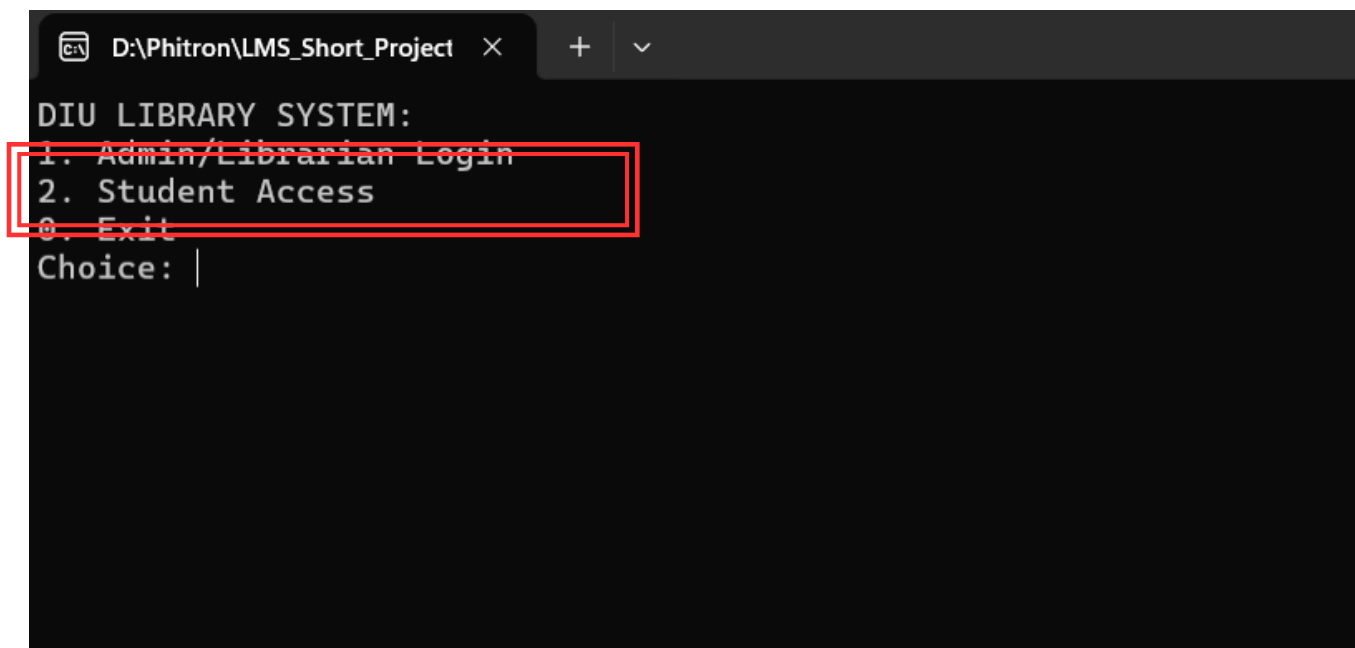


Fig: Student Login Interface

This illustrates the access portal for student users. Students must enter their valid Student ID and the system password to proceed to the student panel for borrowing or returning books.

USER MANUAL FOR ADMIN

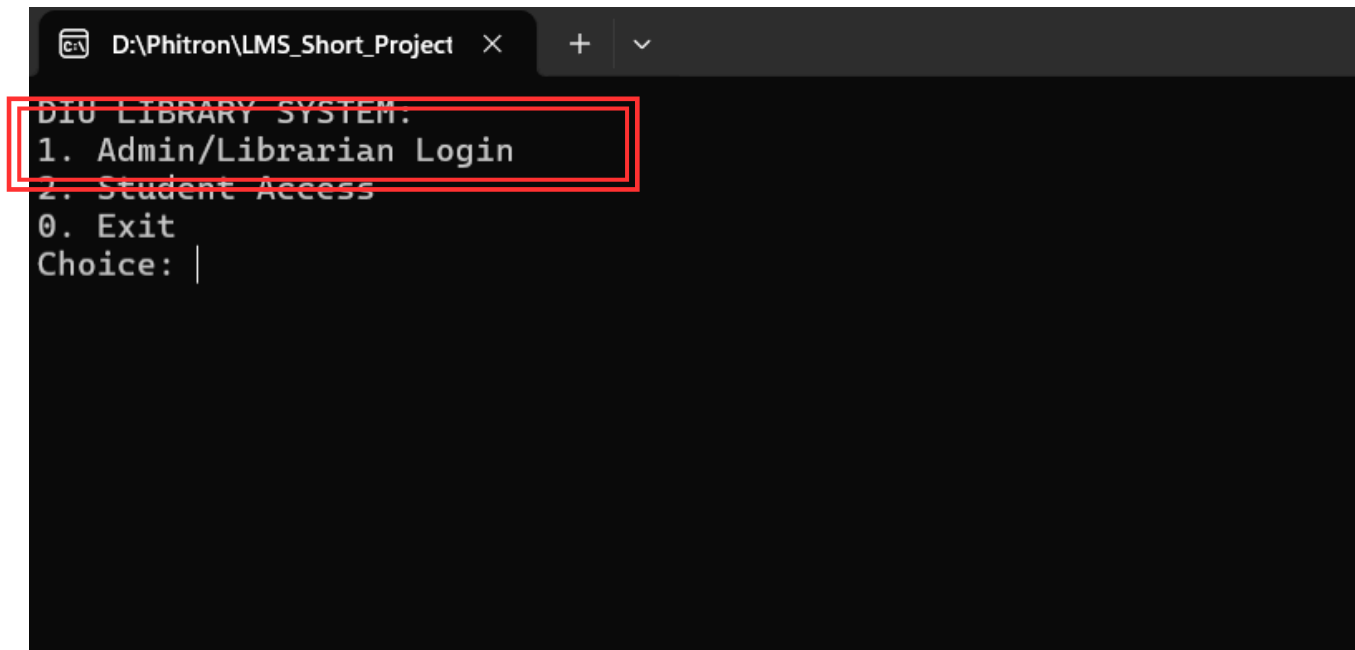


Fig:Admin Panel Dashboard

This figure shows the secure login prompt for administrators. The system requires a predefined username and password to grant access to management functions like adding or deleting books

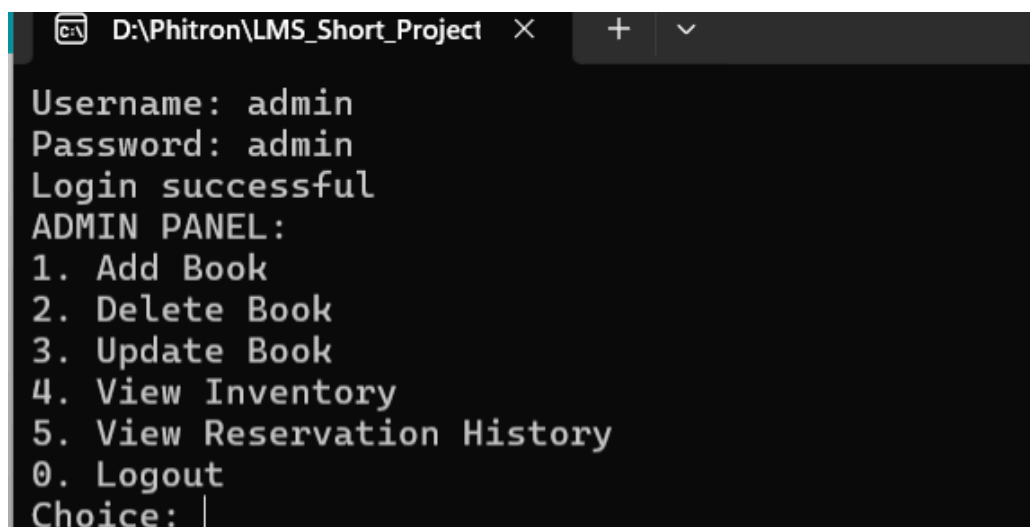
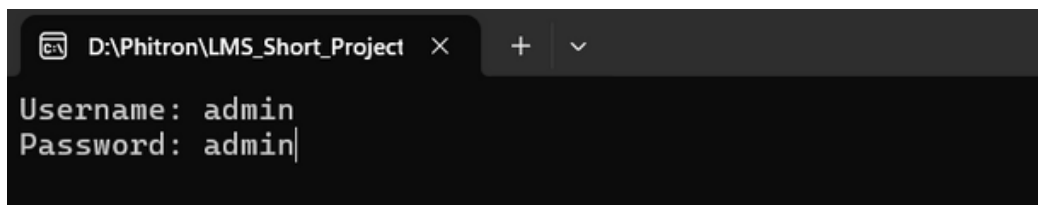
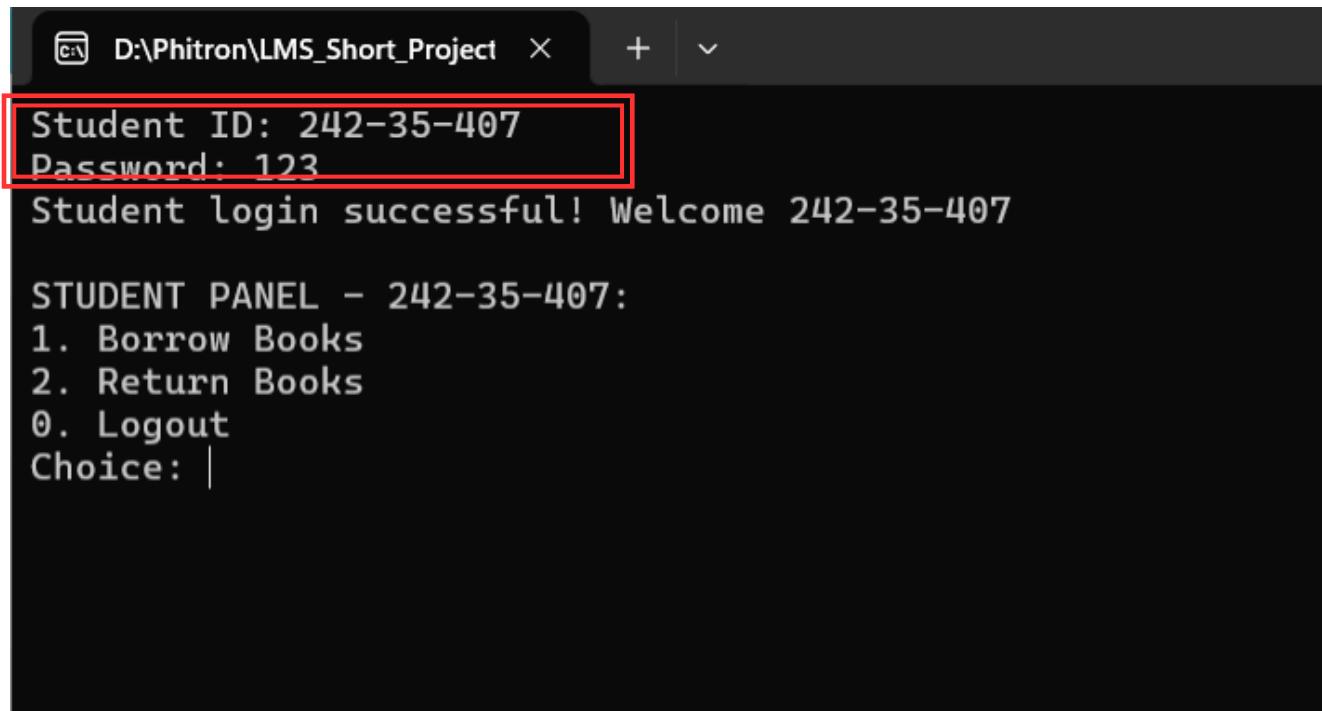


Fig: Admin Panel UI

USER MANUAL FOR STUDENTS



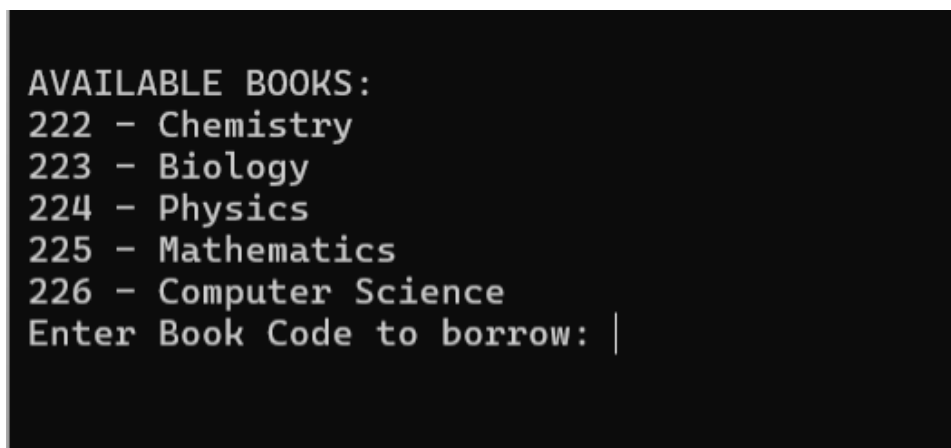
```
D:\Phitron\LMS_Short_Project x + v
Student ID: 242-35-407
Password: 123
Student login successful! Welcome 242-35-407

STUDENT PANEL - 242-35-407:
1. Borrow Books
2. Return Books
0. Logout
Choice: |
```

The screenshot shows a terminal window with a dark background. The title bar at the top reads 'D:\Phitron\LMS_Short_Project' followed by window control icons. The main content area displays the login process: the user enters 'Student ID: 242-35-407' and 'Password: 123', which are highlighted by a red rectangular box. After successful login, a welcome message 'Student login successful! Welcome 242-35-407' is shown. Below this, the 'STUDENT PANEL' is displayed with a list of options: '1. Borrow Books', '2. Return Books', and '0. Logout'. The prompt 'Choice: |' is at the bottom, indicating the user is ready to make a selection.

Fig:Student Panel Dashboard

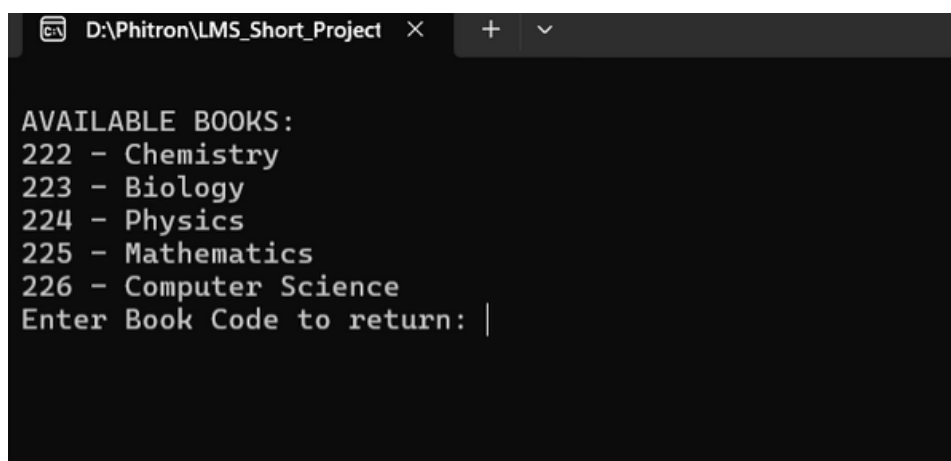
All the things that are already predefined like Student ID, Password, Available Books(that should be also available when the user chooses Return books



```
AVAILABLE BOOKS:
222 - Chemistry
223 - Biology
224 - Physics
225 - Mathematics
226 - Computer Science
Enter Book Code to borrow: |
```

The screenshot shows a terminal window with a dark background. It displays a list of 'AVAILABLE BOOKS' with their respective codes: 222 for Chemistry, 223 for Biology, 224 for Physics, 225 for Mathematics, and 226 for Computer Science. At the bottom, the prompt 'Enter Book Code to borrow: |' is shown, indicating the user is ready to enter a book code.

Fig:Borrow Books



```
D:\Phitron\LMS_Short_Project x + v
AVAILABLE BOOKS:
222 - Chemistry
223 - Biology
224 - Physics
225 - Mathematics
226 - Computer Science
Enter Book Code to return: |
```

The screenshot shows a terminal window with a dark background. It displays the same list of 'AVAILABLE BOOKS' as the previous figure: 222 for Chemistry, 223 for Biology, 224 for Physics, 225 for Mathematics, and 226 for Computer Science. At the bottom, the prompt 'Enter Book Code to return: |' is shown, indicating the user is ready to enter a book code for returning.

Fig: Return Books

GRANT CHART

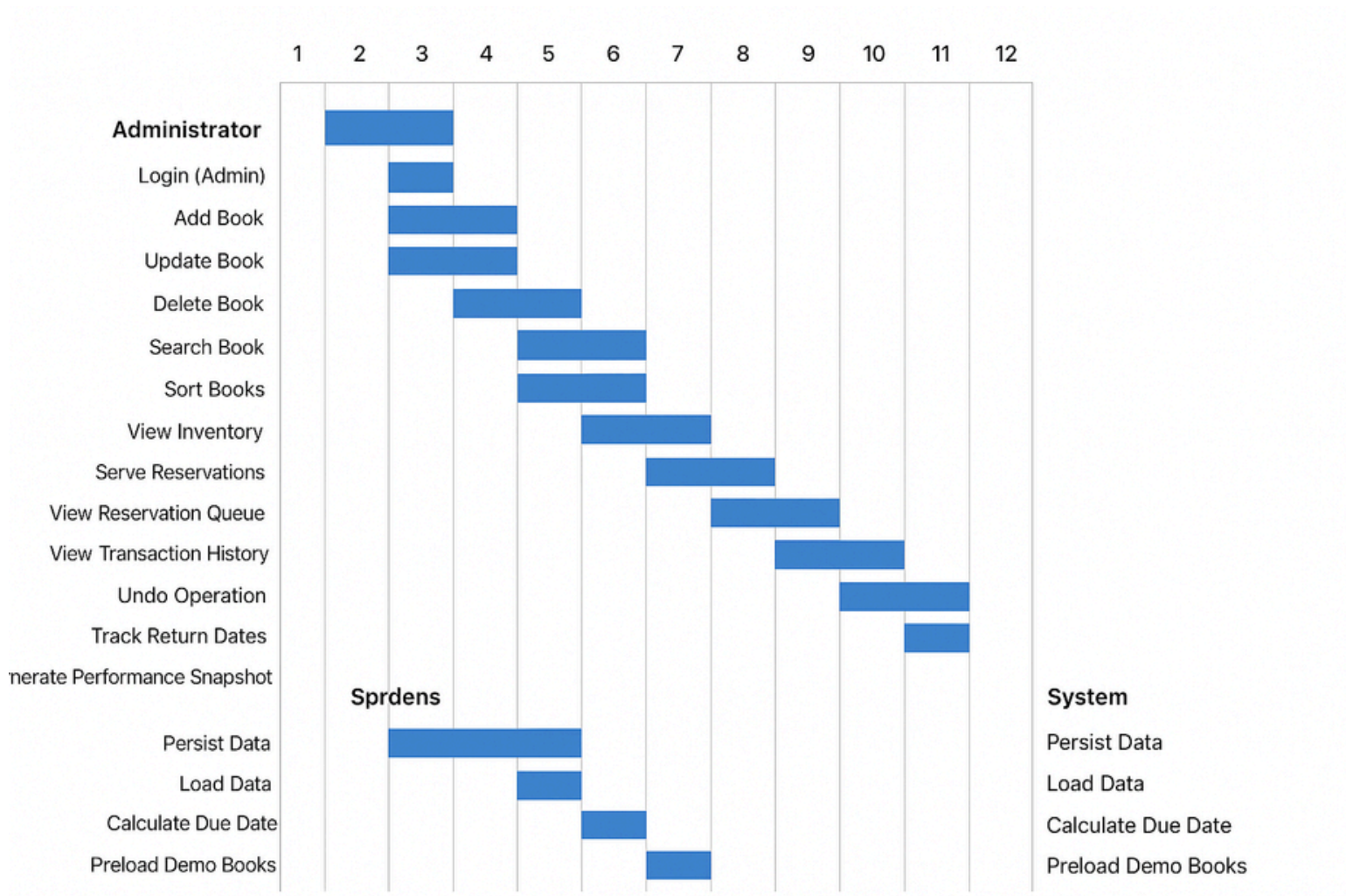


Fig: Full System Timeline

CONCLUSION

The Library Book Management & Operation System satisfies the academic requirement of demonstrating classical operations of arrays, queues, linked lists, and stacks within a cohesive, practical application. The design shows how these structures provide reliable foundations for real-world coordination tasks and offers a strong base for GUI, networking, and performance-focused upgrades.

REFERENCES

- B. Kernighan and D. Ritchie, The C Programming Language, 2nd ed., Prentice-Hall, 1988.
- M. Weiss, Data Structures and Algorithm Analysis in C, 2nd ed., Addison-Wesley, 1997.
- D. Knuth, The Art of Computer Programming, Volume 1: Fundamental Algorithms, 3rd ed., Addison-Wesley, 1997.

THANK YOU!

