Project Report

Library Management System

Only for course Teacher						
		Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	25
Understanding	3					
Analysis	4					
Implementation	8					
Report Writing	10					
					Total obtained mark	
Comments						

Semester: Fall 2024

Group No.- 9

Name: Sourove Datta Souro

ID: 232-35-397

Batch: 41 **Section: A2**

Course Code: SE133 **Course Name: Software Development Capstone Project**

Course Teacher Name: Ms. Nusrat Tasnim

Designation: Lecturer, Department of Software Engineering

Submission Date: 01/12/2024

Library Management System

Table of Contents

Abstract

The Library Management System will be beneficial for Daffodil International University in several ways. It focuses on improving time efficiency, promoting good management practices, and providing easy access to library resources.

With this system, students and library managers can easily search for and borrow books without waiting in long lines. They will also be able to manage their accounts and track their borrowed books digitally, making the process quick and efficient. The system will allow for online book reservations, minimizing the need for physical visits to the library.

For students, who often have tight schedules, this system will offer time-saving features like automatic renewal of borrowed books for due dates. The system will also help staff keep track of library inventory and reduce manual labor.

Furthermore, the system includes a catalog of available books, allowing users to explore different genres and topics without physically browsing the shelves. This will improve accessibility and encourage more people to use the library.

In addition to benefiting students and library managers, the system will help manage books and maintain records more effectively. It will generate reports on books borrowed, overdue items, and inventory, enabling better decision-making and resource management.

The overall aim is to enhance the user experience, save time, and improve the library's operational efficiency.

Chapter-1

Introduction

1.1 About the System

The **Library Management System** is a software application designed to streamline and automate the day-to-day operations of a library. The system helps manage library resources, such as books, journals, and other reading materials, in a more efficient and organized manner. It aims to facilitate easy tracking of books, member information, and borrowing history while also simplifying administrative tasks.

The System will have several key features such as:

- Book Management: Add, update, delete, and search for books based on various criteria such as title, author, and publication year.
- User Management: Manage library members, including registration, authentication, and updating personal details.
- Transaction Tracking: Record and monitor books borrowed by users, along with due dates, penalties, and history.
- Search Functionality: Allow users and library staff to search for books by different parameters such as title, author, or ID.
- Admin Panel: For system administrators to oversee user activities, manage books, and generate reports.

The system is designed to be used by both librarians (who will manage books and users) and users (who can search and borrow books). This system is expected to reduce manual paperwork and the complexities of managing a library manually.

1.2 Purpose of the System

The primary purpose of the Library Management System is to provide an automated platform that makes the management of library resources more efficient, reducing manual effort, time, and errors. The specific purposes of the system are as follows:

- Efficient Book Management: Allows easy and quick addition, modification, and deletion of books, helping librarians manage inventory effectively.
- Automated Record-Keeping: Automatically tracks all transactions, such as book borrowings and returns, ensuring accuracy and up-to-date records.
- Time and Cost Efficiency: Reduces the time spent by librarians on administrative tasks such as maintaining records, generating reports, and managing overdue books.
- Accessibility: Provides easy access to books and library data through a digital system, allowing users to search and manage their accounts anytime.

Overall, the system aims to improve the functionality, management, and user interaction of the library, making it more user-friendly and less dependent on manual operations.

1.3 Necessity of the System

A traditional library operates manually, often using paper-based records for tracking books, users, and transactions. The necessity of implementing an automated Library Management System arises from the need to overcome these challenges, including:

- Manual Record Keeping: A manual system requires physical records, which are time-consuming, error-prone, and hard to manage. An automated system saves time and reduces errors.
- Easy Search and Retrieval: Finding books or tracking borrowing history is quicker and easier in a digital system.
- IInventory Management: Tracking book availability and inventory is complex manually but automatic in a digital system.
- Efficient User Management: An automated system simplifies member registration, renewals, and reminders for overdue books.
- Scalability: As the library grows, the system scales effortlessly without manual effort.

Thus, the Library Management System is necessary to address the shortcomings of a manual approach, improving both the administrative efficiency and user experience. The system will make the library more organized, accessible, and scalable, ensuring smooth operation as the library grows.

Chapter-2

Structure

2.1 System Architecture Overview

The Library Management System follows a modular architecture. The system is divided into several components that each handle different aspects of the library's operations. These components interact with each other through functions and the use of external text files to store the data. The system architecture can be visualized as follows:

- User Interface: This is the point where the librarian and users interact with the system. It is a text-based menu system where users can choose actions such as adding books, searching for books, managing users, and more.
- Book Management: This module is responsible for managing books in the library, including adding, updating, deleting, and searching books. The books' data is stored in the `books.txt` file.
- Student Management: This module is responsible for managing student (or user) records. It handles user registration, updating information, and managing their borrowing records. The student data is stored in the `students.txt` file.
- Manager Management: This module handles the administration side, allowing the manager to oversee library operations, including adding new users, managing the library's inventory, and generating reports. Manager credentials are stored in the `managers.txt` file.
- File Handling: The system uses three main text files to store and manage data:
    - students.txt: Stores information about library users (students), such as their ID, name, contact details, and borrowed books.
    - manager.txt: Stores the login credentials (username and password) of library managers for authentication purposes.
    - books.txt: Stores information about books in the library, including the book's ID, title, author, and publication date.

Each module works independently, but they are integrated into a unified system through function calls and shared data files.

2.2 Module Descriptions

Each module of the system serves a distinct purpose and is responsible for specific tasks. Below are the descriptions of the key modules implemented:

- **Student Management Module:**
  - Registration: This module allows new students to register by providing necessary details, which are then saved in the `students.txt` file.
  - Login: Validates user credentials (username and password) against the `students.txt` file to grant access to the system for borrowing books, searching, etc.
  - Borrowing Books: Allows the student to borrow books, and the system updates the records for borrowed books in the `books.txt` file.
  - Search and View Books: Students can search for books based on various criteria (ID, name, author, etc.) and view details about books available in the library.
- **Manager Management Module:**
  - Login: Authenticates the manager's credentials from the `managers.txt` file to grant access to the admin interface.
  - Book Management: Allows managers to add new books, update existing books, or delete books from the library. These operations modify the `books.txt` file.
  - Search and Display Books: Managers can view all the books, search by book details, and print reports regarding book inventory.
- **Book Management Module:**
  - Add New Book: Allows managers to add books to the library's collection. Each book has an ID, name, author, and publication date.
  - Update Book: Managers can update details of a specific book, like its title, author, or publication date.
  - Delete Book: Enables managers to remove a book from the system, and it updates the `books.txt` file.
  - Search Book: Allows students and managers to search for books based on different parameters such as book ID, name, or author.
- **Data Storage:**
  - students.txt: This file stores student user credentials, such as the student ID, name, and password.
  - manager.txt: This file holds login details like the manager's username and password for system access.
  - books.txt: Contains records of all books in the library, with each book's ID, title, author, and publication date stored as separate lines.

2.3 Data Flow and Interactions

In this section, we describe how the system processes data and interacts with the text files (`students.txt`, `managers.txt`, and `books.txt`), which serve as the primary means of persistent storage.

- **Login Process:**
  - When a student or manager attempts to log in, the system compares the entered credentials against the data stored in either `students.txt` (for students) or `managers.txt` (for the manager).
  - For students, the system checks if the student's ID and password match the records in `students.txt`.
  - For the manager, the system checks if the username and password match the details in `managers.txt`.
  - If the credentials are valid, the user is granted access to the system. Otherwise, an error message is displayed.
- **Book Operations:**
  - When a user (student or manager) adds a book, the system assigns a unique ID to the book (calculated by the `getNextID()` function) and writes the book details to the `books.txt` file.
  - Book deletion: When a manager deletes a book, the system removes the book's record from `books.txt`, and the remaining books' IDs are adjusted sequentially to maintain integrity.
  - Book update: If a book's details are updated, the system overwrites the old details in the `books.txt` file with the new ones.
- **Search Operations:**
  - For students and managers, when a search operation is requested (e.g., search by book name, author, or ID), the system reads through the `books.txt` file and displays matching results.
  - Search by book ID: It searches for a specific book by its unique ID.
  - Search by author or name: It performs a partial string match for books by author or name using `strstr()`.
- **Flow of Data:**
  - Student Registration and Login: When a new student registers, their details are appended to `students.txt`. When a student logs in, the system reads from this file to authenticate the user.
  - Book Management: Books are added, deleted, or updated in `books.txt`. Any modification of the library inventory is performed by reading and rewriting this file.

Chapter-3

Features

3.1 Overview of Features

The Library Management System provides a set of features aimed at simplifying and streamlining the management of library operations, including student and manager interactions, book management, and data persistence. The system is designed to facilitate the efficient management of a library by enabling tasks such as searching for books, adding new books, updating book information, deleting books, and handling student registrations and logins.

Key Features of the System:

1. **Student Features:**
    - Student registration and login
    - View available books in the library
    - Search for books by various criteria (ID, name, author)
    - Borrow books (take), Return borrowed books
    - View borrowing history
2. **Manager Features:**
    - Manager login for system access
    - Add, update, and delete books
    - View all books in the library
    - Search books based on various attributes
    - Show students taken books history
    - Data persistence with text files (books.txt, students.txt, manager.txt)
3. **Book Management:**
    - Book management by the library manager (Add, Update, Delete)
    - Display book list in a tabular format
    - Book search by name, author, or ID
4. **User Management:**
    - User (Student) registration and management
    - Manager authentication and access control

These features are implemented using C, with the data stored in plain text files (`books.txt`, `students.txt`, `student_books`, `temp_user`, `managers.txt`) for simplicity and ease of use. Below, we provide a detailed description of each feature.

3.2 Detailed Description of System Features

Student Features:

- **Student Registration and Login:**
  - Registration: Students can create an account by providing their details (ID, name, password). This information is saved in the `students.txt` file, allowing the system to authenticate students in future sessions.
  - Login: Students authenticate by entering their credentials (student ID and password). The system verifies the details against the `students.txt` file. If the details are correct, the student is granted access to the system.
- **Book Search:**
  - Search by ID: Students can search for a book using its unique ID. The system reads the `books.txt` file and displays the book's details (name, author, publication date) if a match is found.
  - Search by Name: Students can search for books by their title (name). The system performs a substring search using `strstr()` to match the provided name with the book titles.
  - Search by Author: Students can also search for books by the author's name. Similar to the name search, this feature performs a substring match to find books by the author.
- **View Book List:**
  - Students can view a list of all available books in the library. The system displays a table-like format showing the book ID, name, author, and publication date for each book in the `books.txt` file.
- **Borrow (Take) Books:**
  - Borrow Book: Students can borrow a book by providing its ID. The system checks if the book is available (not currently borrowed by another student). Once borrowed, the student's record is updated with the book's ID, and the status of the book in the `books.txt` file is updated to indicate it is currently taken.
  - The borrowing details are stored in the `students.txt` file, where each student's borrowed books are tracked by their IDs.
- **Return Books:**
  - Return Book: Students can return a borrowed book by providing its ID. When a book is returned, the system updates the student's record to remove the book's ID from their borrowed list. The book's status in the `books.txt` file is updated to indicate that it is available again.
  - If the student attempts to return a book that is not listed as borrowed by them, the system notifies the student that the book was not borrowed.
- **View Borrowing History:**

    - Borrowing History: Each student can view the history of books they have borrowed. This feature lists all the books a student has previously borrowed, along with the dates they were borrowed and returned. The history is stored in the student's record in the `students.txt` file, ensuring that past borrowing actions are tracked.

Manager Features:

- **Manager Login:**
    - Managers authenticate with a unique username and password. These credentials are stored in the `managers.txt` file. If the details are correct, the manager is granted access to the administrative features of the system.
- **Add New Book:**
    - Managers can add new books to the library's inventory by providing details such as the book name, author, and publication date. The system automatically generates a unique ID for the new book and appends it to the `books.txt` file.
- **Update Existing Book:**
    - Managers can update the details of an existing book. This includes modifying the book's name, author, or publication date. The system reads from the `books.txt` file, finds the book by its ID, and allows the manager to modify the details.
- **Delete Book:**
    - Managers can delete a book from the library system by providing its ID. The system reads the `books.txt` file, removes the book with the specified ID, and renumbers the remaining books to ensure the IDs remain sequential.
- **Search Books:**
    - Managers can search for books by their ID, name, or author. The system provides search options similar to those available to students but with additional managerial capabilities (such as the ability to modify or delete the found books).
- **Student Books Taken History:**
    - The system allows managers to view all students' books history from the library.
- **Display All Books:**
    - The system allows managers to view all books in the library in a formatted table. This feature provides a clear, easy-to-read list of all books, which is particularly useful for managers when reviewing the library's inventory.

Book Management:

- **Book Information Storage:**

    - All books are stored in the `books.txt` file, which maintains the following information for each book:
        - ID: A unique identifier for each book (automatically generated).
        - Name: The title of the book.
        - Author: The author of the book.
        - Published Date: The publication date of the book.
        - Status: Indicates if the book is available or currently borrowed.
    - When a new book is added or an existing book is updated, the information is saved in this text file.
- **Book Operations:**
    - Add: New books are added by appending their details (ID, name, author, published date) to the `books.txt` file.
    - Update: When a book's details are updated, the file is rewritten, and the book's entry is replaced with the updated information.
    - Delete: When a book is deleted, the system writes the remaining books to a temporary file (`temp_books.txt`), and then the original file is replaced with the updated one.

User Management:

- **Student Management:**
    - Student records (username, password, borrowed books) are stored in the `students.txt` file. New students are added when they register, and their details are saved in this file. The borrowed book information is stored with each student's record, allowing tracking of books they have taken.
- **Manager Authentication:**
    - The manager's credentials are stored in the `managers.txt` file. These credentials are used for system authentication before access to the managerial functions (add, update, delete books) is granted.

Data Persistence:

- **Text Files for Data Storage:**
    - students.txt: Stores student details (ID, name, password) and their borrowed books.
    - manager.txt: Stores manager credentials (username, password).
    - books.txt: Stores all books in the library, including their unique ID, name, author, publication date, and status (available/taken).
- **File Operations:**
    - The system performs file operations to ensure data is updated correctly. Temporary files are used when performing operations like deleting or updating books, ensuring data integrity.

Chapter-4

Implementation

4.1 C Concepts Used in the Project

The Library Management System is implemented in C using a combination of essential programming concepts and techniques to ensure efficient handling of book and student records. The following C concepts have been extensively used to implement various features of the system:

1. **File Handling**:
    - **File I/O** is crucial for storing and retrieving data such as student information, book records, and borrowing history. In this project, text files (`students.txt`, `managers.txt`, and `books.txt`) are used to persist the data between program executions.
        - Functions like `fopen()`, `fclose()`, `fscanf()`, `fprintf()`, `fgets()`, and `fputs()` are used to read from and write to these files.
2. **Structs**:
    - **Structures** are used to define the key entities in the system: books, students, and managers. These structures provide a way to organize data efficiently and make it easier to manipulate and store information in arrays or files.
        - Example structures:
            - **Book**: Contains fields like `book_id`, `name`, `author`, `published_date`, and `status` (available/taken).
            - **Student**: Contains fields like `student_id`, `name`, `password`, and a list of borrowed book IDs.
            - **Manager**: Contains fields like `manager_id`, `username`, and `password`.
3. **Dynamic Memory Allocation**:
    - Memory is allocated dynamically using **`malloc()`** and **`free()`** to handle varying amounts of data, such as dynamically allocated arrays for storing borrowed book IDs or student records. This ensures that memory is efficiently managed, especially when dealing with large sets of data.
4. **Pointers**:
    - **Pointers** are used to manage memory locations and manipulate the data efficiently. Pointers to structures are used when passing large structures (like student and book records) to functions, as passing by reference avoids unnecessary copying of data.

5. **Functions**:
    - The program is divided into functions to modularize the implementation, making it more readable and maintainable. These functions handle specific tasks such as adding a new book, searching for a book, registering a student , updating student records and so on.
        - Examples:
            - `addBook()`: Adds a new book to the system.
            - `borrowBook()`: Allows students to borrow a book.
            - `returnBook()`: Allows students to return a borrowed book.
            - `viewBooks()`: Displays all the books in the system.
6. **Control Structures**:
    - **If-else** statements and **switch-case** constructs are used to handle conditional logic in user input, such as checking whether a book is available for borrowing or if a student has already borrowed the maximum allowed books.
    - **Loops** (while, for) are used to iterate through lists of books, students, or other data stored in arrays or files. For example, `while` loops are used to read through the list of students in the `students.txt` file until the required student is found.
7. **String Manipulation**:
    - The **string.h** library is heavily used to manipulate strings (e.g., for student names, book titles, author names). Functions like `strcmp()`, `strcpy()`, and `strcat()` are used for comparing, copying, and concatenating strings.
8. **Error Handling**:
    - Basic error handling is implemented throughout the system. For example, when opening files using `fopen()`, the program checks if the file is opened successfully before proceeding with file operations. Similarly, input validation is performed to ensure that the user enters correct data (e.g., checking if a book ID is valid or a student exists).
9. **Basic Authentication**:
    - The system uses basic **authentication** for managing student and manager logins. Passwords are compared to stored values in the `students.txt` and `managers.txt` files. Simple encryption (e.g., using a basic hash or just comparing plain text) could be added for extra security.

4.2 Key Algorithms and Data Structures

The Library Management System employs several key algorithms and data structures to handle various tasks like book borrowing, returning, and searching for books efficiently. The following sections describe these algorithms and data structures in detail:

4.2.1 Data Structures Used

1. **Structures**:
    - Structures are used to define entities like **Books**, **Students**, and **Managers**. These structures hold the essential information for each entity and are crucial for organizing the data.
      Example Structure for Book:

```c
#define MAX_NAME_LEN 40
#define MAX_AUTHOR_LEN 30
#define MAX_DATE_LEN 20

typedef struct {
    int id;
    char name[MAX_NAME_LEN];
    char author[MAX_AUTHOR_LEN];
    char publishedDate[MAX_DATE_LEN];
} Book;
```

2. **Arrays:**
    - **Arrays** are used to store multiple records of books and students. The number of records (books or students) is generally managed using a counter variable that tracks how many entries are currently stored in each array.
    - Example: An array of `Book` structures to store all the books in the library.

```c
struct Book books[MAX_BOOKS];
```

4.2.2 Key Algorithms

- **Search Algorithm:**
  - **Linear Search**: The system uses **linear search** to find books or students in the array or file. This search checks each entry sequentially, comparing the search query (book ID, student ID, etc.) with each record until a match is found.
  - **Book Search by ID**:

```
for (int i = 0; i < bookCount; i++) {
    if (books[i].book_id == search_id) {
        // Book found, display details
        break;
    }
}
```

- **Book Return Algorithm:**
  - Return Book: When a student returns a book, the system checks if the book ID is in the student's borrowed list. If found, it updates the student's record to remove the book ID, and updates the book's status.

```
for (int i = 0; i < student.borrowed_count; i++) {
    if (student.borrowed_books[i] == book_id) {
        // Remove the book from the borrowed list
        for (int j = i; j < student.borrowed_count - 1; j++) {
            student.borrowed_books[j] = student.borrowed_books[j + 1];
        }
        student.borrowed_count--;
        bookStatus = 'available';
        break;
    }
}
```

- **File Handling Algorithm:**
  - The system uses sequential file processing to read and write data to text files (students.txt, manager.txt, books.txt). Each time a book or student record is modified, the updated data is written back to the file.

```
FILE *file = fopen("books.txt", "a"); // Open file in append mode

fprintf(file, "%d %s %s %s %s\n", book_id, name, author, published_date, status);
fclose(file);
```

Chapter-5

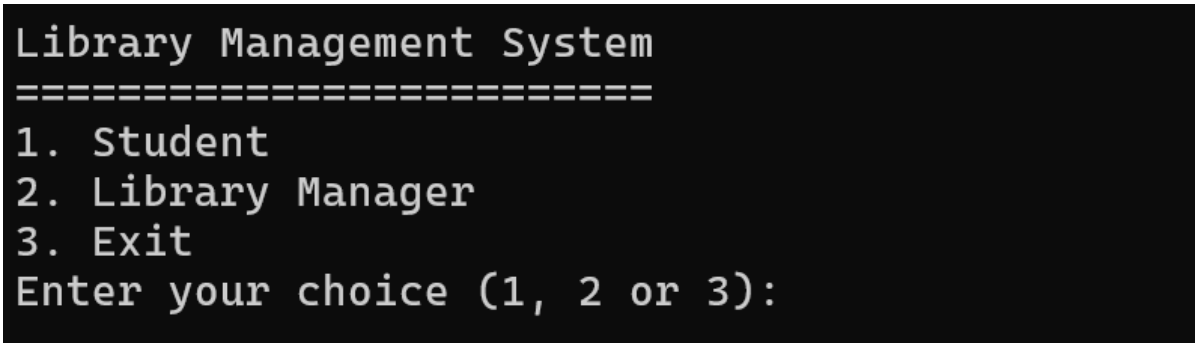
System Testing

5.1 Testing Introduction

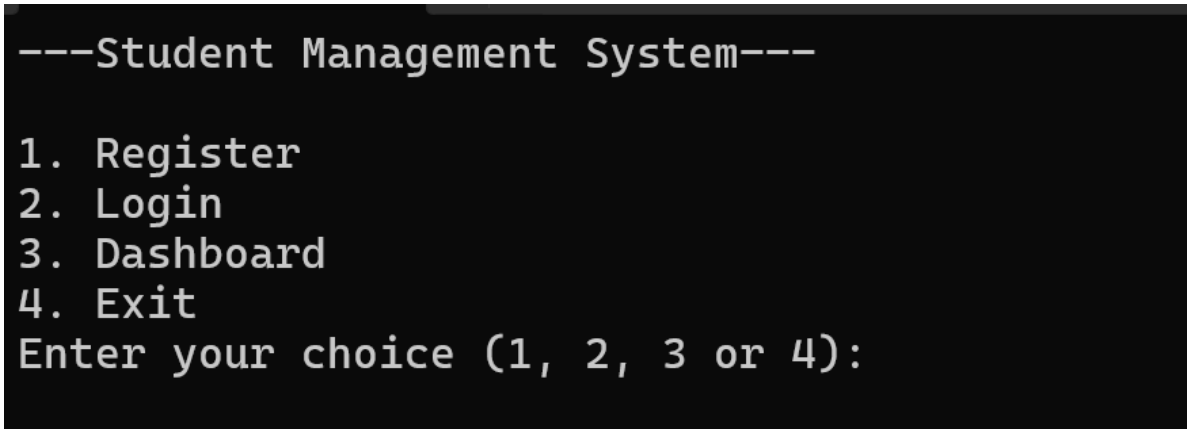
System testing is a critical phase of software development aimed at validating that the system meets its specified requirements and functions as intended. In the Library Management System, testing was performed to ensure the correct behavior of core functionalities, including adding, deleting, searching, borrowing, and returning books, as well as managing student and manager access. The testing phase focused on both functional and non-functional aspects, including usability, error handling, and data integrity.

5.2 Input and Expected Output

a. Dashboard:



a.1. Student:



a.1.1. Register:

```
Register a New Student
----------------------
Enter a new Username: souro
Enter Full Name: Sourov Datta
Enter Student ID: 232-35-397
Enter Department: SWE
Enter a new Password: asdf
```

If user already exists:

```
Register a New Student
----------------------
Enter a new Username: souro
Username already exists! Please choose a different one.
Enter a new Username:
```

After Register auto show login option

a.1.2. Login:

```
Student Login
-------------
Enter Username: souro
Enter Password: asdf
```

a.1.2.0.

```
Welcome, Sourov Datta!
Student ID: 232-35-397
Department: SWE

Login successful! Redirecting to student login...

---Library Management System---

1. Book list
2. Search Books
3. Take Book
4. Return Book
5. History
6. Clear Display
7. Exit
Enter your choice (1-7):
```

a.1.2.1. Book list:

```
Enter your choice (1-7): 1

Book List:
-------------------------------------------------------------------------------------
| ID    | Name                                      | Author                        |
-------------------------------------------------------------------------------------
|  1001 | atomic                                    | souro                         |
|  1002 | 7 habit                                   | bd                            |
|  1003 | dbms                                      | souro                         |
-------------------------------------------------------------------------------------
```

a.1.2.2. Search Books:

```
Enter your choice (1-7): 2

Search Options:
1. Search by ID
2. Search by Name
3. Search by Author
Enter your choice (1-3): 1
Enter the Book ID to search: 1001

Found Book:
ID: 1001
Name: atomic
Author: souro
Published Date: 2-3-4
```

```
Enter your choice (1-3): 2
Enter the Book Name to search: dbms

Search Results by Name:
ID: 1003
Name: dbms
Author: souro
Published Date: 5-5
```

```
Enter your choice (1-3): 3
Enter the Author Name to search: souro

Search Results by Author:
ID: 1001
Name: atomic
Author: souro
Published Date: 2-3-4

ID: 1003
Name: dbms
Author: souro
Published Date: 5-5
```

a.1.2.3. Take Book:

```
Enter your choice (1-7): 3

Enter the ID of the book to take: 1001
Book 'atomic' successfully borrowed.
```

```
Enter the ID of the book to take: 1010
Book ID not found.
```

a.1.2.4. Return Book:

```
Enter your choice (1-7): 4

Enter the ID of the book to return: 1002
Book with ID 1002 successfully returned.
```

```
Enter the ID of the book to return: 1010
Book with ID 1010 not found in your records.
```

a.1.2.5. History:

```
Borrowing History:
------------------
souro - Borrowed: 1002 7 habit
souro - Borrowed: 1001 atomic
souro - Borrowed: 1003 dbms
souro - Borrowed: 1001 atomic
```

a.1.2.6. Clear Display:

Clean terminal display and show a.1.2.0 screen

a.1.2.7. Exit:

Exit from the program

a.1.3. Dashboard:

Back on: a. Dashboard

a.1.4: Exit:

Program End

a.2. Library Manager:

```
--- Manager Management System ---

1. Login
2. Dashboard
3. Exit
Enter your choice (1-3):
```

a.2.1. Login:

Accounts are pre build on **managers.txt** file

Examples:

admin password

manager password

etc…

```
Manager Login
-------------
Enter username: admin
Enter password: password
```

a.2.1.0. Manager Dashboard:

```
--- Manager Book Management ---

1. Display Book List
2. Search Book
3. Add New Book
4. Delete Book by ID
5. Update Book by ID
6. Users taken books history
7. Clear Display
8. Exit
Enter your choice (1-7):
```

a.2.1.1. Display Book List:

Same, Student Book list option (a.1.2.1)

When Add, Delete and Update Books the table data are updated.

a.2.1.2. Search Book:

Same, Student Search Books option (a.1.2.2)

a.2.1.3. Add New Book:

```
Enter your choice (1-7): 3

Enter Book Name: Intro to python
Enter Author: alamin
Enter Published Date: 12/12/2012
Book 'Intro to python' successfully added with ID 1004.
```

When adding a new book, the book ID was automatically created.

a.2.1.4. Delete Book by ID:

```
Enter your choice (1-7): 4

Enter the ID of the book to delete: 1003
Book with ID 1003 successfully deleted and IDs adjusted.
```

When deleting a book then all books IDs are adjusted, based on the book lists sequence.

a.2.1.5. Update Book by ID:

```
Enter your choice (1-7): 5

Enter the ID of the book to update: 1003
Updating Book with ID 1003
Enter New Book Name (current: Intro to python): Python
Enter New Author (current: alamin): souro
Enter New Published Date (current: 12/12/2012): 11/11/2021
Book with ID 1003 successfully updated.
```

a.2.1.6. Students taken books history:

```
Enter your choice (1-7): 6


Student Borrowing Summary:
--------------------------
Student: souro
Total Books Borrowed: 3
Books:
  - atomic
  - dbms
  - atomic

Student: sumonto
Total Books Borrowed: 4
Books:
  - atomic
  - atomic
  - Python
  - 7 habit
```

a.2.1.7. Clear Display:

Clean terminal display and show a.2.1.0 screen

a.2.1.8. Exit:

Exit from the program

a.2.2. Dashboard:

Back on: a. Dashboard

a.2.3. Exit:

Program End

a.3. Exit:

```
Enter your choice (1, 2 or 3): 3
Exiting the program...

Process returned 0 (0x0)   execution time : 156.907 s
Press any key to continue.
```

5.3 Summary of Test Results

The tests were performed to verify that the Library Management System functions as specified. Below is a summary of the results:

Test Case	Status	Remarks
Register Account	Passed	Successfully registered the students account and added it to the students.txt file.
Login Accounts	Passed	Both accounts successfully login and can access their portal.
Display Book List	Passed	Proper formatting and listing behavior observed.
Add New Book	Passed	New book is correctly added to the list.
Search Book by Name	Passed	Accurate search results displayed.
Delete Book by ID	Passed	Book removal and ID adjustment worked correctly.
Borrow a Book	Passed	Book status updated as expected.
Return a Book	Passed	Return process and history update successful.
Invalid Book Search	Passed	Handled gracefully with an appropriate message.
Invalid Input Handling	Passed	Errors managed effectively; no crashes occurred.
Back in Dashboard	N/A	Expect this can be done multiple times but for system dependency that can do only one time.

Overall, the Library Management System passed all major test cases, indicating that it is robust and ready for deployment with minimal issues. Future testing may involve stress tests, multi-user scenarios, and integration tests for scalability improvements.

Chapter-6

Conclusion

6.1 Positive Aspects of the System

The Library Management System developed using C has demonstrated several positive aspects, making it a valuable solution for the intended users:

- **Improved Organization**: The system provides an efficient way to manage books and student records, offering faster access and better organization compared to manual library management methods.
- **User-Friendly Interface**: The system is designed with a simple and clear interface, making it easy for users (students, managers) to understand and operate.
- **Effective Search Mechanism**: Users can quickly search for books by title, author, or ID, improving their experience and saving time.
- **Automated Borrowing and Returning**: The system automates the borrowing and returning of books, reducing manual labor and minimizing human errors.
- **Data Persistence**: By using file handling techniques (`students.txt`, `manager.txt`, `books.txt`), the system ensures data persistence, allowing records to be saved even after the program is closed.
- **Security and Authentication**: Basic authentication for students and managers ensures a level of security, preventing unauthorized access to sensitive features.

6.2 Limitations of the System

Despite its strengths, the system has certain limitations that could be addressed to make it more robust and scalable:

- **Limited Security Measures**: The current authentication system uses basic password storage and lacks advanced security features such as encryption or user role-based access control.
- **File-Based Data Storage**: The system relies on text files for storing data, which limits scalability and can lead to slower performance with larger datasets. Using a database management system (DBMS) would be a more efficient solution.
- **Single-User Access**: The system is designed for single-user access at a time, making it less effective for environments where multiple users need simultaneous access.
- **Limited Data Validation**: While basic input validation is performed, the system could be more robust in handling incorrect or malicious input from users.
- **No Graphical User Interface (GUI)**: The system uses a console-based interface, which might not be as appealing or user-friendly as a graphical user interface (GUI).
- **Lack of Advanced Features**: The system lacks some features typically found in modern library management systems, such as notifications for due dates, book recommendations, and detailed reports.

6.3 Potential Future Enhancements

To address the current limitations and enhance the functionality of the Library Management System, several potential future improvements could be made:

- **Database Integration**: Migrating from file-based data storage to a relational database (e.g., MySQL, SQLite) would enhance data retrieval speed, scalability, and security.
- **Advanced Security Features**: Implementing robust security mechanisms such as password hashing, encryption, and role-based access control would improve data security and user privacy.
- **Graphical User Interface (GUI)**: Developing a GUI using a framework such as GTK or integrating with web technologies would make the system more user-friendly and visually appealing.
- **Multi-User Support**: Enabling multi-user access with concurrent data management capabilities would allow multiple users to interact with the system simultaneously.
- **Enhanced Data Validation and Error Handling**: More rigorous data validation mechanisms could be implemented to ensure data integrity and better handling of user errors or invalid inputs.
- **Notification System**: Adding features like email or SMS notifications for due dates, new arrivals, or other important events would improve the user experience.
- **Detailed Reporting**: Generating detailed reports on book availability, borrowing history, student activity, etc., would provide useful insights to library managers and administrators.
- **Recommendation System**: Implementing a recommendation engine that suggests books based on borrowing history or preferences would add value to the system.
- **Online Access**: Extending the system to support online access would allow users to search for and reserve books remotely, making it more flexible and convenient.

Chapter-7

User Manual

7.1 User Guide

This section outlines how to use the **Library Management System** effectively for different types of users, such as students and managers. It covers the functionalities available through the system's interface and guides the user step-by-step.

- **Accessing the System:**
  - Run the system executable from your terminal or command prompt.
  - Upon startup, you will see the Library Management System menu.
- **Library Management System Dashboard**:
  - Show see option
    - Student (1)
    - Library Manager (2)
    - Exit (3)
- **Available Manager Options:**
  - **Display Book List:** View a list of all books with their details (ID, Name, Author).
    - Select this option by entering 1.
  - **Search Book:** Search for a specific book by ID, name, or author.
    - Select this option by entering 2.
  - **Add New Book:** Add a new book to the collection.
    - Select this option by entering 3 and provide the necessary details such as Name, Author, and Published Date.
  - **Delete Book by ID:** Remove a book using its unique ID.
    - Select this option by entering 4 and enter the book ID to delete.
  - **Update Book by ID:** Modify the details of an existing book by ID.
    - Select this option by entering 5 and provide new values as prompted.
  - **Student Books Taken History:** Show students taken books
    - Select this option by entering 6.
  - **Clear Display:** Clear the terminal display.
    - Select this option by entering 7.
  - **Exit:** Exit the system.
    - Select this option by entering 8.
- **Available Student Options:**
  - **View Book List (1):** Browse through all available books.
  - **Search for a Book (2):** Locate a specific book by entering its name or author.
  - **Take (borrow) a Book (3):** Select a book to borrow based on its ID.
  - **Return a Book (4):** Return a borrowed book using its ID.
  - **View Borrowing History (5):** See your history of borrowed and returned books.

- **Clear Display (6):** Clear the terminal display.
- **Exit (7):** Exit this system.

7.2 System Requirements

To run the Library Management System, ensure that your system meets the following requirements:

- **Operating System:** Windows, macOS, or Linux
- **Compiler:** C compiler (e.g., GCC) or IDE supporting C (e.g., Code::Blocks, VS Code, Dev-C++)
- **Memory:** Minimum 512 MB RAM
- **Disk Space:** Minimum 100 MB free disk space
- **Additional Software:** None required (console-based application)

7.3 Installation and Setup

To install and set up the Library Management System, follow these steps:

- **Obtain the Source Code:**
  - Download the source code folder or files for the system (`1_DashBoard.c`, `2_Student.c`, `3_Manger.c`, etc.) along with the supporting text files (`students.txt`, `managers.txt`, `books.txt`).
- **Place Supporting Files:**
  - Ensure the `students.txt`, `managers.txt`, `books.txt` , etc files are in the same directory as the executable.
- **Run the System:**
  - On a Compiler(Code::Blocks, others), run the executable:
    - Open project folder on Compiler
    - Run the 1_DashBoard.c

End