Annexure-I

LIBRARY MANAGEMENT SYSTEM

CSE PATHSALA

A training report

Submitted in partial fulfilment of the requirements for the award of degree of

B. TECH CSE

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

From 18/06/2025 to 03/08/2025

SUBMITTED BY

Name of student: ACSA ANNA TOMY

Registration Number: 12310876

Signature of the student:

[illegible]

List Of Figures

LOVELY PROFESSIONAL UNIVERSITY

Annexure-II: Student Declaration

To whom so ever it may concern

I, **ACSA ANNA TOMY,12310876,** hereby declare that the work done by me on "**LIBRARY MANAGEMENT SYSTEM**" from **June 2025** to **July 2025**, is a record of original work for the partial fulfilment of the requirements for the award of the degree, **B. TECH CSE.**

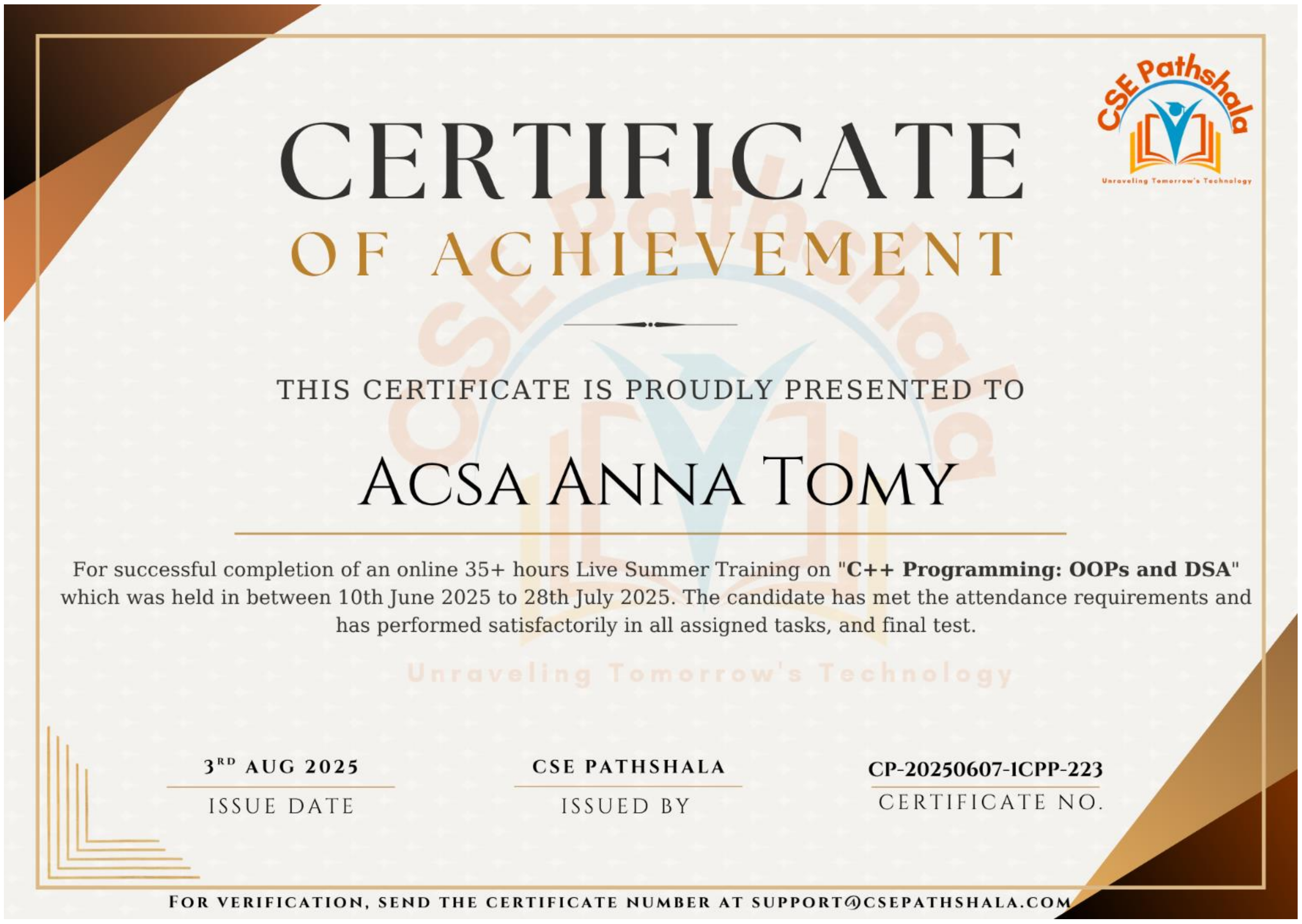
Name of the Student (Registration Number): **ACSA ANNA TOMY (12310876)**

Signature of the student

[illegible]

Dated: 12/08/2025

Training Certification :



CERTIFICATE

OF ACHIEVEMENT

THIS CERTIFICATE IS PROUDLY PRESENTED TO

ACSA ANNA TOMY

For successful completion of an online 35+ hours Live Summer Training on "**C++ Programming: OOPs and DSA**" which was held in between 10th June 2025 to 28th July 2025. The candidate has met the attendance requirements and has performed satisfactorily in all assigned tasks, and final test.

Unraveling Tomorrow's Technology

3RD AUG 2025	CSE PATHSHALA	CP-20250607-ICPP-223
ISSUE DATE	ISSUED BY	CERTIFICATE NO.

FOR VERIFICATION, SEND THE CERTIFICATE NUMBER AT SUPPORT@CSEPATHSHALA.COM

ACKNOWLEDGEMENT

I would like to thank all the persons who have helped to make this project, Library Management System, a success.

To start with, I would like to thank my project guide/mentor Professor Rahul Jain, whose expert advice, useful suggestions, and constant motivation have played an integral part in the development of the project.

I thank my institution and faculty for the availability of resources, technical assistance, and a research and development-friendly environment. Their collaboration enabled one to investigate graph algorithm concepts, greedy optimization methods, and practical payment systems.

I would also like to show appreciation for the contributions of my colleagues and friends, who shared ideas throughout the development process.

Lastly, I am also grateful to my family for their unwavering support and encouragement during the duration of this work.

Without the encouragement and support of all these people, this project would not be achievable.

CHAPTER 1

1. INTRODUCTION OF THE PROJECT UNDERTAKEN

1) Objectives of the work undertaken:

The main goal of this project is to develop and design an integrated Library Management System that can effectively manage different library operations. The system should be able to manage operations like cataloging books, tracking borrowing and return, maintaining accurate member records, and generating detailed reports on library usage and statistics. The system is designed to use appropriate data structures like linked lists, hash tables, trees, and queues to keep library information stored and arranged in an organized manner. Furthermore, efficient searching and sorting algorithms will be implemented to allow fast and accurate lookups of books and sorting of records. Another main aim is to manage full CRUD (Create, Read, Update, Delete) operations for books and members so that the system is dynamic and flexible. A simple-to-use interface will also be implemented to allow the system to be user-friendly for librarians and library members, hence making operations a daily breeze.

2) Scope of the Work:

The scope of this project is to fully automate library processes for physical as well as virtual collections. It encompasses member registration, book borrowing, returns, reservations, and waitlist management. The system will support search and sort facilities for books and member information, employing high-level algorithms to maximize speed and efficiency. It will be extensible to various types of libraries, including schools, public libraries, and digital-only libraries. The solution is scalable in the sense that it can support large collections and can have multiple consumers at once without affecting performance. Further, it can be made extensible to interact with advanced technology, such as barcode or RFID-based tracking, and can be made extensible to support e-book management for digital libraries.

3) Importance and Applicability:

Installation of a Library Management System is of high utility in modern information management. Automating the cataloging process, for example, is feasible in schools, while inventory management is maximized and preparation of reports for administrative purposes is facilitated. Public libraries, which typically handle large collections and high usage by members, can utilize the system to efficiently handle check-out and check-in processes, reducing the likelihood of errors and accelerating turnaround times. Electronic libraries can benefit from organizing e-books, online journals, and multimedia materials in a systematic way and making them accessible for fast and precise searches by users. The ability of the system to reduce labour, prevent the incidence of errors, and enhance accessibility makes it of high utility in real library operations, translating into greater productivity and user satisfaction.

- Educational institutions: Automates cataloging, inventory control, and report creation.
- Public libraries: Simplifies check-in/check-out processes and handles large collections efficiently.
- Digital libraries: Supports e-book, online resources, and electronic access systems management.
- Increases productivity, minimizes human error, and improves the user experience.

4) Role and Profile:

This project involves the work of a software developer and system architect who is in charge of creating a library management system from the ground up. For safe and long-lasting data management, this entails assessing requirements, organizing the system, developing strong data storage mechanisms, putting effective search and retrieval algorithms into practice, and integrating file or database systems. The position also entails creating an intuitive user interface to facilitate easy communication between librarians and library patrons.

The profile included:

Strong technical proficiency in programming, data structures, and file handling is required for the position, as is a thorough comprehension of library workflows. Managing book cataloging, lending, returns, fines, and search features are all included in this. In addition to being scalable to support both small institutional libraries and large-scale library networks, the system must guarantee accuracy, speed, and security when handling data. In order to maximize performance and guarantee that the software satisfies operational requirements in the real world, the profile also calls for problem-solving abilities.

5) Implementation:

The Library Management System translates the plan into a working software application. It starts by setting up relevant data structures and a database for book details, member information, and transaction records. A good database schema is created to define various relationships between books, members, and issued records for seamless and efficient data manipulation.

Having completed the data-related aspect, the dispensation of the core modules ensues with the book management module. With this, the book can be added, updated, removed, or tracked for availability by the librarian. The member management module is created to enrol users, manage their profiles, and handle unique identification numbers per member. The borrowing and returning module further facilitate book issuance with due dates, the calculation of fines for late return, and keel tracking thereabout.

Next, upon those core functions come search and sorting programs to allow speedy access to book details by title, author, or category and orderly rendition of the library catalog. The reservation and waitlist features are also designed.

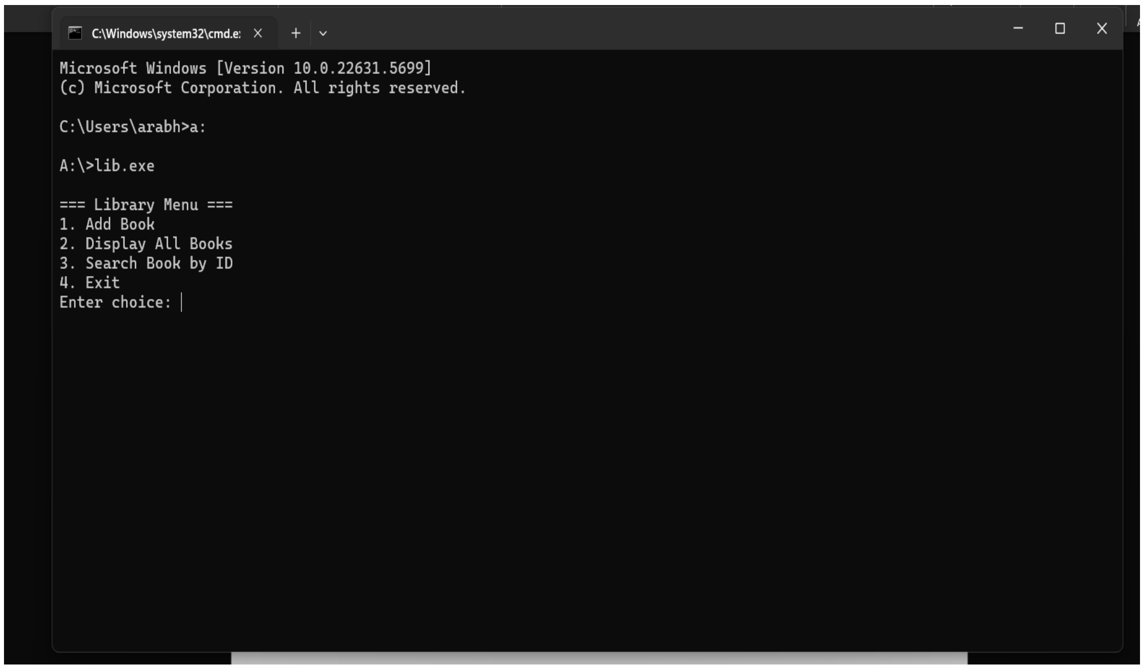
6) Working:

1. Purpose:

This is a simple console-based C++ program for acquiring, displaying, and searching data about a book in a library.
It uses file handling to maintain the details of a book perhaps forever.

2. Features:

- **Add Book** → Store new book details into a file.
- **Display All Books** → Read and show all stored books.
- **Search Book by ID** → Find a specific book using its ID.
- **Exit** → Close the program.



```
Microsoft Windows [Version 10.0.22631.5699]
(c) Microsoft Corporation. All rights reserved.

C:\Users\arabh>a:

A:\>lib.exe

=== Library Menu ===
1. Add Book
2. Display All Books
3. Search Book by ID
4. Exit
Enter choice:
```

Figure 6.1 All functions

3. How It Works Internally:

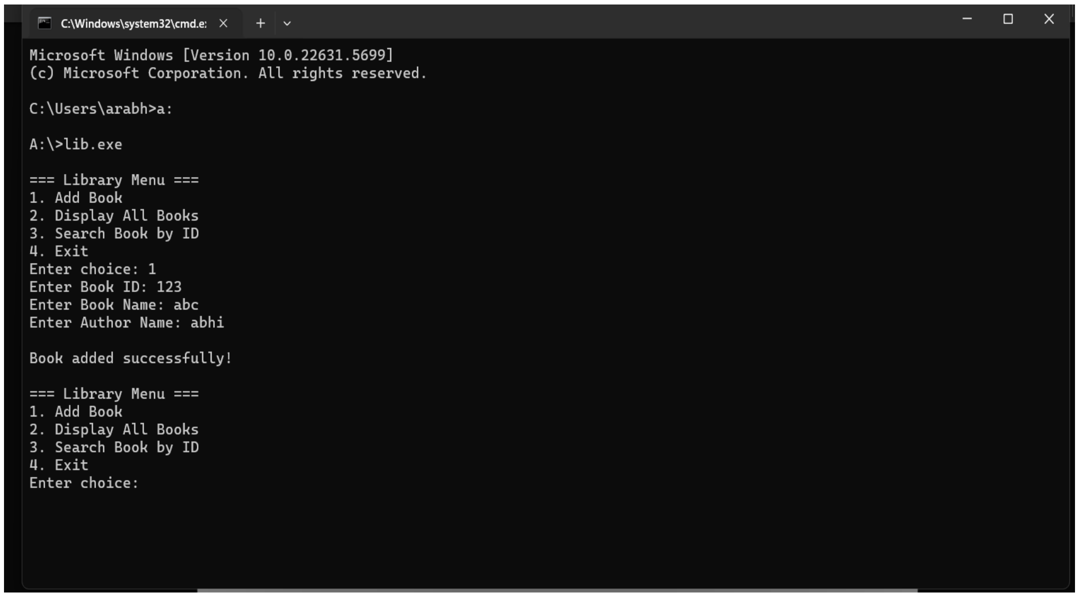
a) Data Storage:

- All book details are saved inside a file **library.dat**.
- Instead of saving raw objects in binary (which causes string issues), we save each field (ID, bookName, author) in **text form** or using a **struct with fixed char arrays**.

b) Adding a Book:

1. User chooses "Add Book" from the menu.
2. Program asks for:
    - Book ID
    - Book Name
    - Author Name

3. Details are written to the file using ofstream.

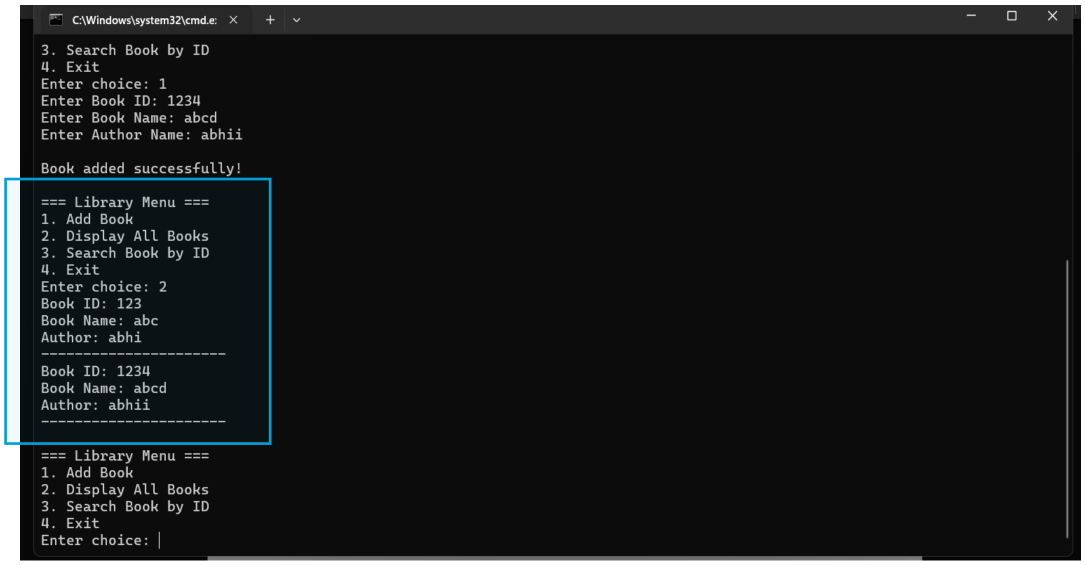


```
Microsoft Windows [Version 10.0.22631.5699]
(c) Microsoft Corporation. All rights reserved.

C:\Users\arabh>a:

A:\>lib.exe

=== Library Menu ===
1. Add Book
2. Display All Books
3. Search Book by ID
4. Exit
Enter choice: 1
Enter Book ID: 123
Enter Book Name: abc
Enter Author Name: abhi

Book added successfully!

=== Library Menu ===
1. Add Book
2. Display All Books
3. Search Book by ID
4. Exit
Enter choice:
```

Figure 6.2 Adding book

c) Displaying All Books:

1. Opens **library.dat** using ifstream.
2. Reads each book record.
3. Prints them one by one.

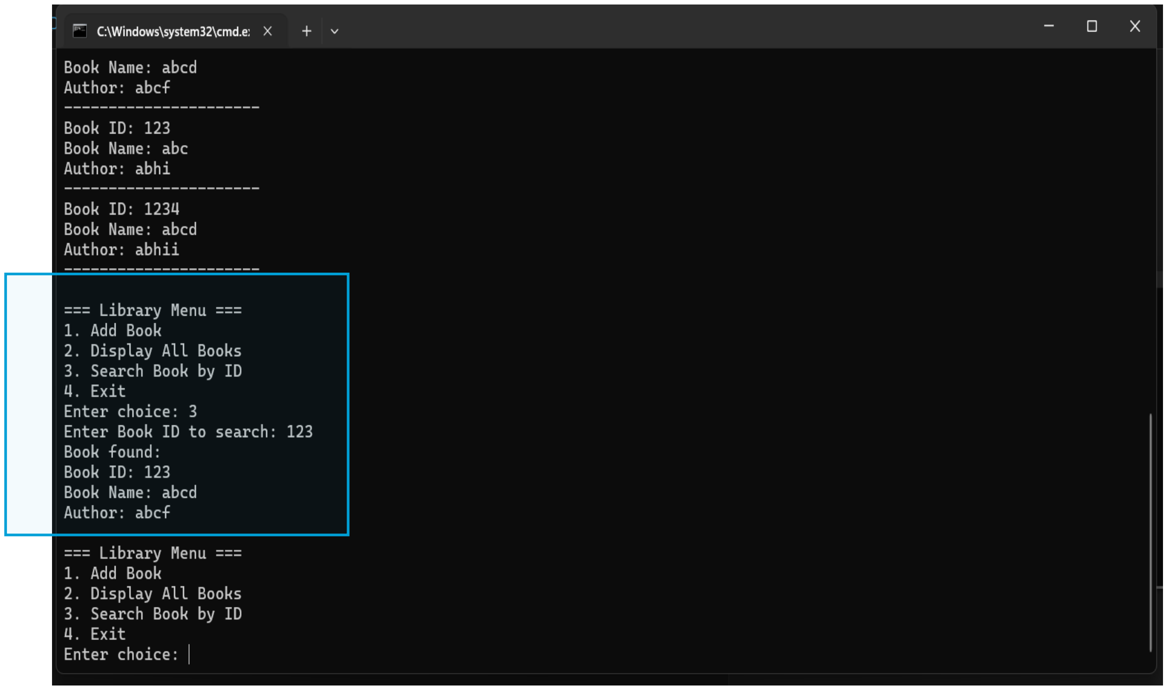


```
3. Search Book by ID
4. Exit
Enter choice: 1
Enter Book ID: 1234
Enter Book Name: abcd
Enter Author Name: abhii

Book added successfully!

=== Library Menu ===
1. Add Book
2. Display All Books
3. Search Book by ID
4. Exit
Enter choice: 2
Book ID: 123
Book Name: abc
Author: abhi
----------------------
Book ID: 1234
Book Name: abcd
Author: abhii
----------------------

=== Library Menu ===
1. Add Book
2. Display All Books
3. Search Book by ID
4. Exit
Enter choice: |
```

Figure 6.3 Displaying books

d) <u>Searching by Book ID:</u>

1. User enters a book ID.
2. Program scans each record from file.
3. If ID matches, it shows the book details.
4. If not found, displays "Book not found".



```
Book Name: abcd
Author: abcf
----------------------
Book ID: 123
Book Name: abc
Author: abhi
----------------------
Book ID: 1234
Book Name: abcd
Author: abhii
----------------------

=== Library Menu ===
1. Add Book
2. Display All Books
3. Search Book by ID
4. Exit
Enter choice: 3
Enter Book ID to search: 123
Book found:
Book ID: 123
Book Name: abcd
Author: abcf

=== Library Menu ===
1. Add Book
2. Display All Books
3. Search Book by ID
4. Exit
Enter choice:
```

Figure 6.4 Searching books

CHAPTER 2

2. INTRODUCTION OF THE COMPANY/WORK

1) Company's Vision and Mission:

CSE Pathshala aims to remove barriers of geography, affordability, and background and make quality computer science education universally accessible. The organization sets out to be a catalyst in bringing forth a generation of technologists with relevant skills, creativity, and ethical concerns.

Computer Science Pathshala aims at equipping students with conceptual knowledge, application skills, and the right amount of professionalism to allow them to thrive in a highly competitive technology-driven society. They suppose this by:

- Offering value-added experiential learning activities to marry practical activities with theoretical knowledge.
- Affording learners individual attention and mentoring them through their paths.
- Ensuring the curriculum meets industrial requirements so that their graduates find their path into jobs.
- Establishing a nurturing and collaborative learning environment that fosters growth and innovation.
- Pledging to constantly innovate on teaching methods with state-of-the-art tools and approaches to retain maximum relevance and impact.

2) Origin and growth of company:

C.S.E Pathshala was conceived by **Nirmal Gaud** as an educational channel on the YouTube platform so that top-notch computer science education could reach learners in every corner of India and across the border. After initiation with just a handful of tutorials on programming basics, the channel gained great publicity all because of its opposition to clarity, depth, and learner-friendly methods.

As the audience grew, C.S.E-Pathshala expanded beyond YouTube into its own online learning platform where it could host a wider array of courses offered live, interactively, and recorded with mentor backing. The courses have further diversified into more advanced topics such as machine learning, data science, and full-stack web development.

What began as a free tutorials channel has flourished into a full-fledged learning ecosystem catering to a varied set of students, backed by spawn-and-foster-a-thousand great partnerships in content creation along with a dedicated culture of good teaching.

3) Various departments and their functions:

Somehow, fewer in numbers if compared with large-scale educational entities, currently, C.S.E-Pathshala operates through several core functional areas:

- Content Development Department - Design and recording of course material ensure structured tutorials and accurate content.
- Teaching and Mentorship Department - Conducts live classes, doubt clearance, personalized guidance, and one-on-one mentorship for learners.
- Technical Department - Take care of the eLearning platform, video production, uploading courses, and maintenance of the website and associated tools.
- Student Support and Community Management - Respond to queries, manage forums, arrange study groups, and keep them active.
- Marketing and Outreach Department - Promote courses and resources via social media, partnerships, and other digital marketing channels.
- Administration and Operations - Takes care of financial matters, scheduling, interdepartmental coordination, and overall flow of the organization.

4) Course Overview and Experience:

➢ **Information about the course**

The subject taken was Data Structures and Algorithms (DSA) in C++, which aimed to enhance problem-solving abilities and algorithmic thinking through the C++ programming language. The subject delivered an ideal blend of theoretical knowledge and practical problem-solving, allowing participants to implement learned principles in actual programming issues. Focus was given to typing effective, optimized, and clean code as well as enhancing logical reasoning to solve competitive programming and technical interview problems.

➢ Topics Covered

During the course, a wide range of concepts were covered to provide in-depth insight into fundamental DSA concepts with practical implementation in C++. The key topics covered were:

- Introduction to C++ fundamentals and syntax refreshers
- Arrays and Strings
- Linked Lists (Singly, Doubly, and Circular)
- Stacks and Queues (including Priority Queues)
- Trees and Binary Search Trees
- Graphs and Traversal Algorithms (BFS, DFS)
- Recursion and Backtracking
- Sorting and Searching Algorithms
- Dynamic Programming Basics
- Complexity Analysis (Time and Space Complexity)
- Along with lectures, problem-solving exercises and multiple-choice questions (MCQs) were incorporated into the sessions in order to strengthen comprehension.

➢ Method of Teaching

The course was taught in a live online format under the supervision of Professor Rahul Jain, a seasoned educator with expertise in computer science topics. Classes were conducted daily during the week, with the session time ranging from 1 to 3 hours based on the complexity of the topic discussed.

The approach integrated:

- Concept Explanation – Simplifying topics into easy-to-understand pieces.
- MCQ Discussions – Developing critical thinking and fast problem-solving.
- Hands-On Practice – Writing and running C++ programs in sessions.
- Doubt Clarification – Chasing down questions in the moment to maintain concept clarity.
- The interactive method kept students engaged while giving flexibility to delve more into hard subjects

➢ **Course duration and schedule**

- The DSA with C++ course started on 18th June and ended on 25th July, so it was a 38-day training. We had classes on all days of the week without any break to maintain continuous engagement and learning momentum.
- The schedule differed from day to day to suit the varying intensity of learning:
- There were 1-hour sessions on some days with only theory and small exercises.
- On other days, we had 2-hour sessions that involved concept explanation, MCQ discussion, and coding practice.

- There were also intensive 3-hour sessions focused on breaking down difficult problems, dealing with multiple topics, or rehashing important concepts before exams.

- The balance between short, intense lessons and long, detailed problem-solving classes permitted the absorption of knowledge and applying it. Furthermore, the constant daily learning kept us grounded with the concepts and prevented long gaps that would hinder understanding.

CHAPTER 3

3. BRIEF DESCRIPTION OF THE WORK DONE

The Library Management System was separated into discrete, organized modules in order to achieve the project's goals. This modular design methodology made it easier to manage complexity, allowed for seamless component integration, and made sure that every development stage could be thoroughly tested before moving on to the next.

➢ Module 1:

System Design & Requirement Analysis:
Goal: Determine the functional requirements, comprehend a library's workflow, and adjust the system architecture accordingly.

Completed Work:

gathered requirements to comprehend the needs for fine calculation, borrowing, returning, and cataloging books.

created a draft system architecture for long-term record-keeping that included file-based data storage.

Interactions between the system, members, and the librarian were mapped.

Result: A well-defined system architecture with distinct data flows and functionalities.

- Module 2:

File Management & Data Storage:
Goal: Use files to store book records in a secure and long-lasting manner.

Completed Work:

created a Library class to hold author, title, and book ID information.

Added, stored, and retrieved book data using binary file operations.

facilitated rapid access to extensive book lists by guaranteeing effective read/write operations.

Result: A strong file-handling system that makes it possible to store and retrieve library records securely.

- Module 3:

Functions of Book Management:

The goal is to create the essential features for adding, showing, and searching books.

Completed Work:

To enter and store book details in the file, add Book() was created.

To read and display every book record, display Books() was created.

To find a book using its distinct Book ID, search Book() was implemented.

Result: An effective core system that enables librarians to effectively manage the book catalog.

- Module 4:

User Interface Driven by Menus:

The goal is to give librarians an intuitive interface through which they can engage with the system.

Completed Work:

created a text-based menu with add, display, search, and exit options.

made certain that input/output functions were user-friendly and error-proof.

permitted constant interaction until the user made a conscious decision to stop.

Result: An intuitive menu-driven interface that even non-technical users can use.

FlowDiagram

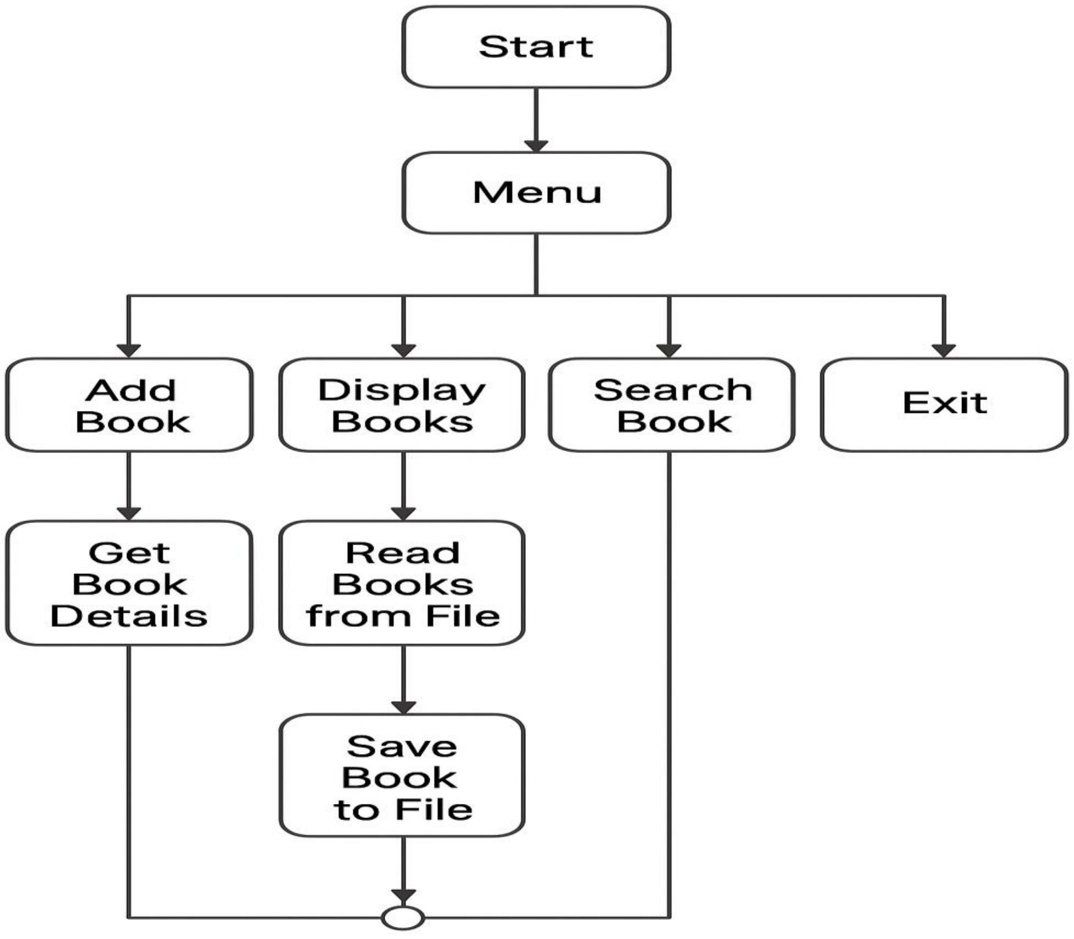


Figure 3.1 flow diagram

CHAPTER 4

CODE AND EXAMPLES

```
#include <iostream>

#include <fstream>

#include <string>

using namespace std;


class Library {

private:

   string bookName, author;

   int bookID;


public:

   void getBookDetails() {

      cout << "Enter Book ID: ";

      cin >> bookID;

      cin.ignore();

      cout << "Enter Book Name: ";
getline(cin, bookName);


  cout << "Enter Author Name: ";

        getline(cin, author);

    }


    void showBookDetails() const {

        cout << "Book ID: " << bookID << "\n";

        cout << "Book Name: " << bookName << "\n";

        cout << "Author: " << author << "\n";

    }


    int getBookID() const {

        return bookID;

    }


    string getBookName() const { return bookName; }

    string getAuthor() const { return author; }

};


void addBook() {
 Library lib;
    ofstream fout("library.txt", ios::app);
    lib.getBookDetails();
    fout << lib.getBookID() << "\n"
         << lib.getBookName() << "\n"
         << lib.getAuthor() << "\n";
    fout.close();
    cout << "\nBook added successfully!\n";
}

void displayBooks() {
    ifstream fin("library.txt");
    if (!fin) {
        cout << "No books found.\n";
        return;
    }

    int id;
    string name, author;
    while (fin >> id) {
```

```
  fin.ignore();
        getline(fin, name);
        getline(fin, author);
        cout << "Book ID: " << id << "\nBook Name: " << name
            << "\nAuthor: " << author << "\n----------------------\n";
    }
    fin.close();
}

void searchBook() {
    ifstream fin("library.txt");
    if (!fin) {
        cout << "No books found.\n";
        return;
    }

    int searchID, id;
    string name, author;
    bool found = false;
  cout << "Enter Book ID to search: ";
   cin >> searchID;

   while (fin >> id) {
      fin.ignore();
      getline(fin, name);
      getline(fin, author);

      if (id == searchID) {
         cout << "Book found:\n";
         cout << "Book ID: " << id << "\nBook Name: " << name
             << "\nAuthor: " << author << "\n";
         found = true;
         break;
      }
   }

   if (!found) cout << "Book not found.\n";
   fin.close();
}
```

```
int main() {
    int choice;
    do {
        cout << "\n=== Library Menu ===\n";
        cout << "1. Add Book\n";
        cout << "2. Display All Books\n";
        cout << "3. Search Book by ID\n";
        cout << "4. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
        case 1: addBook(); break;
        case 2: displayBooks(); break;
        case 3: searchBook(); break;
        case 4: cout << "Exiting...\n"; break;
        default: cout << "Invalid choice!\n";
        }
} while (choice != 4);



    return 0;
   }
```

4.1 EXAMPLE -1

Adding Books

```
=== Library Menu ===
1. Add Book
2. Display All Books
3. Search Book by ID
4. Exit
Enter choice: 1
Enter Book ID: 101
Enter Book Name: The C++ Programming Language
Enter Author Name: Bjarne Stroustrup

Book added successfully!
```

```
=== Library Menu ===

1. Add Book

2. Display All Books


3. Search Book by ID

4. Exit

Enter choice: 1

Enter Book ID: 102

Enter Book Name: Clean Code

Enter Author Name: Robert C. Martin


Book added successfully!
```

4.2 EXAMPLE -2

Displaying All Books

```
=== Library Menu ===

1. Add Book

2. Display All Books

3. Search Book by ID

4. Exit

Enter choice: 2


Book ID: 101

Book Name: The C++ Programming Language
```

```
Author: Bjarne Stroustrup

----------------------

Book ID: 102

Book Name: Clean Code


Author: Robert C. Martin

----------------------
```

4.3 EXAMPLE -3

Searching for a Book

Case A – Book Found

```
=== Library Menu ===

1. Add Book

2. Display All Books

3. Search Book by ID

4. Exit

Enter choice: 3

Enter Book ID to search: 102


Book found:

Book ID: 102

Book Name: Clean Code

Author: Robert C. Martin
```

Case B – Book Not Found

```
=== Library Menu ===

1. Add Book

2. Display All Books

3. Search Book by ID


4. Exit

Enter choice: 3

Enter Book ID to search: 200


Book not found.
```

4.4 EXAMPLE -4

Exiting

```
=== Library Menu ===

1. Add Book

2. Display All Books

3. Search Book by ID

4. Exit

Enter choice: 4

Exiting...
```

CHAPTER 5

CONCLUSION

Adding, storing, retrieving, and searching for books in a library setting are all made easier by the Library Management System created for this project. The system guarantees persistent data storage by implementing file handling in C++, enabling book records to be accessed even after the program

has ended. Librarians can easily manage operations thanks to the menu-driven interface's intuitive user experience. Modular design, effective coding techniques, and precise data handling were carefully considered throughout the development process to guarantee dependability and maintainability.

A lightweight yet useful application that satisfies the fundamental requirements of small to medium-sized libraries is the end result, with room for future improvements like member management, database integration, and sophisticated search capabilities. This project emphasizes the value of user-friendly software in real-world situations in addition to showing how programming concepts can be applied practically.

REFERENCES

https://cplusplus.com/doc/tutorial/files/#google_vignette (accessed on July 26, 2025).

https://www.geeksforgeeks.org/software-engineering/library-management-system/ (accessed on july26, 2025).

https://egenius.in/blogs/what-is-a-library-management-system/(accessed on July 26, 2025).

https://www.academia.edu/37726542/Library_Management_System_Mini_Project_Report_On_LIBRARY_MANAGEMENT_SYSTEM#outer_page_6 (accessed on July 26, 2025).

https://www.scribd.com/document/655318769/PROJECT-REPORT-ON-LIBRARY-MANAGEMENT-SYSTEM (accessed on July 27, 2025).