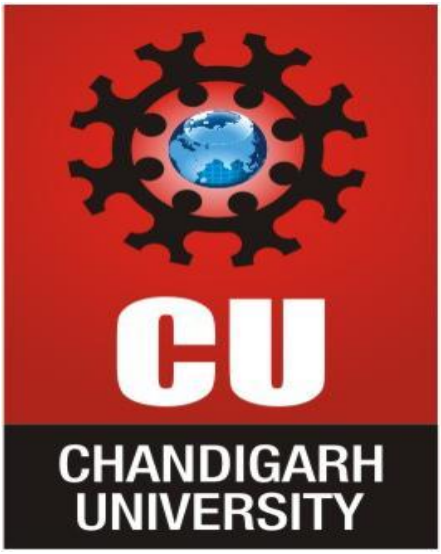
C PROGRAMMING PROJECT

Library Management System



Submitted By:-

Name: Bhuvi

UID: 25BCD10164

Section: 25BCD-3(B)

Submitted To:-

Name: Mr. Arvinder Singh

Acknowledgement

I express my heartfelt gratitude to my respected guide, Mr. Arvinder Singh, for his invaluable guidance, encouragement, and constant support throughout the completion of this project. Without his expert advice and insightful suggestions, this project would not have been possible.

I am also deeply thankful to the Head of the Department and all my faculty members for providing the facilities and resources necessary to complete this work. Their continuous motivation and feedback helped me improve my understanding of the subject.

A special note of thanks to my classmates and friends who supported me in testing and improving this program. Lastly, I would like to express my deepest appreciation to my parents and family for their constant encouragement, patience, and blessings, which have been my source of strength throughout this journey.

Certificate

This is to certify that the project titled “Library Management System in C” is a bonafide work carried out by Bhuvi of Chandigarh University towards the partial fulfillment of the requirements for the course Bachelor of Computer Applications (BCA) during the academic year 1. The project has been completed under the supervision and guidance of Mr. Arvinder Singh, who has verified that this work is original and prepared as per the guidelines.

This project successfully demonstrates the practical application of programming concepts in solving a real-world problem. The effort, dedication, and technical understanding shown by the student are commendable.

TABLE OF CONTENTS

1 INTRODUCTION

Libraries are a vital part of any educational or research institution. Traditionally, libraries maintain a manual record of books, members, and transactions.
As technology advanced, computerized systems became essential to manage data more effectively.

The **Library Management System** developed in **C language** helps automate basic library operations such as adding new books, issuing and returning books, and .maintaining book inventory records. It uses **file handling** concepts for persistent storage.

---

2. OBJECTIVES

- To automate the manual process of maintaining book records.
- To minimize errors and duplication of data.
- To make searching, issuing, and returning faster and efficient.
- To maintain accurate records of book availability.
- To understand and apply file handling and structure concepts in C.

---

3. PROBLEM DEFINITION

Manual record keeping in libraries leads to:

- Time delays during searching and updating.
- Misplacement or duplication of entries.
- Difficulty in tracking issued books and return dates.
- Lack of data security and integrity.

Hence, there is a need for an automated system that can efficiently manage all book-related operations.

---

4. EXISTING SYSTEM

The existing manual system uses registers or spreadsheets.
Operations like searching, adding, or modifying records are time-consuming.
Data redundancy and inconsistency occur frequently.

Drawbacks:

- Difficult to handle large data.
- Prone to human errors.
- No centralized control.
- Time-consuming updates.

---

5. PROPOSED SYSTEM

The proposed system is a **menu-driven C program** that manages library functions digitally.

Key Features:

- Add, search, issue, return, modify, and delete book records.
- Store data permanently in binary files.
- Provide user-friendly text menus.
- Generate quick and accurate results.

---

6. FEASIBILITY STUDY

6.1 Technical Feasibility

The system is designed in C language, which is platform-independent and lightweight. It runs efficiently on any standard desktop.

6.2 Operational Feasibility

The menu-driven interface is simple to use for beginners and non-programmers.

6.3 Economic Feasibility

It is a free and open-source solution, requiring no paid software or database.

7. SYSTEM ANALYSIS

The system comprises the following **entities**:

- **Book** – Contains attributes like ID, Title, Author, Quantity, and Availability.
- **Student** – Represents the person borrowing books.
- **Issue** – Stores details of issued and returned books.

Functional Requirements

1. Add Book
2. Display Book List
3. Search Book
4. Issue Book
5. Return Book
6. Modify Book Details
7. Delete Record

Non-Functional Requirements

- Efficient file storage and retrieval.
- User-friendly console interface.
- Error handling and validation.

8. SYSTEM DESIGN

The system is divided into modules to simplify programming and maintenance.

Module Descriptions

1. **Book Module** – Handles storage and management of book records.
2. **Search Module** – Allows searching by title or author.
3. **Issue Module** – Manages issue operations and updates availability.
4. **Return Module** – Records returned books and restores availability count.
5. **Admin Module** – Modifies or deletes book data.

---

9. HARDWARE REQUIREMENTS

Component	Specification
Processor	Intel Core i3 or above
RAM	4 GB
Storage	500 MB free space
Display	1024×768 resolution
Keyboard	Standard 104 keys

---

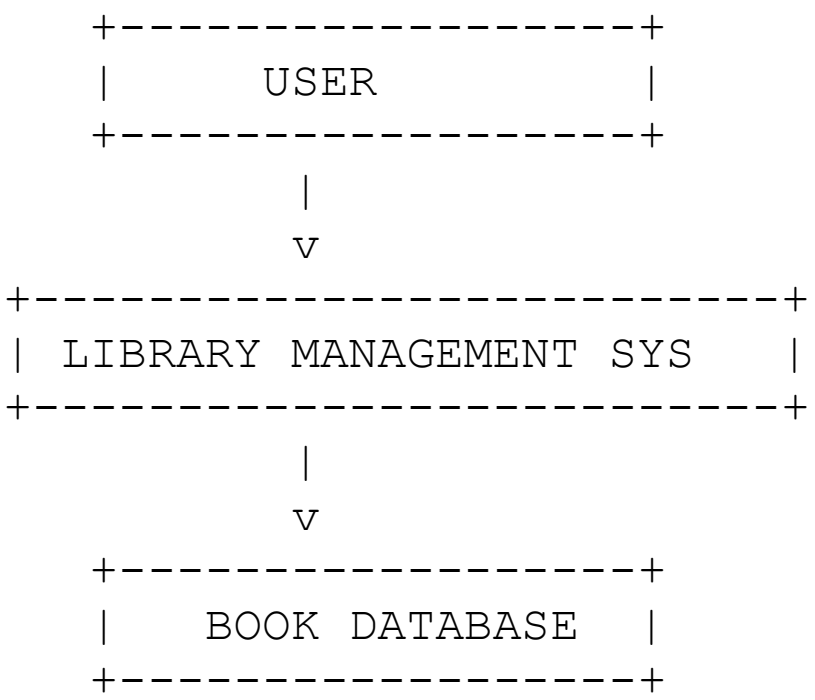
10. SOFTWARE REQUIREMENTS

Software	Version/Details
Operating System	Windows / Linux
Language	C
Compiler	GCC / Turbo C / Code::Blocks
Editor	Notepad / Visual Studio Code

---

11. DATA FLOW DIAGRAM (DFD)

Level 0 (Context Diagram)



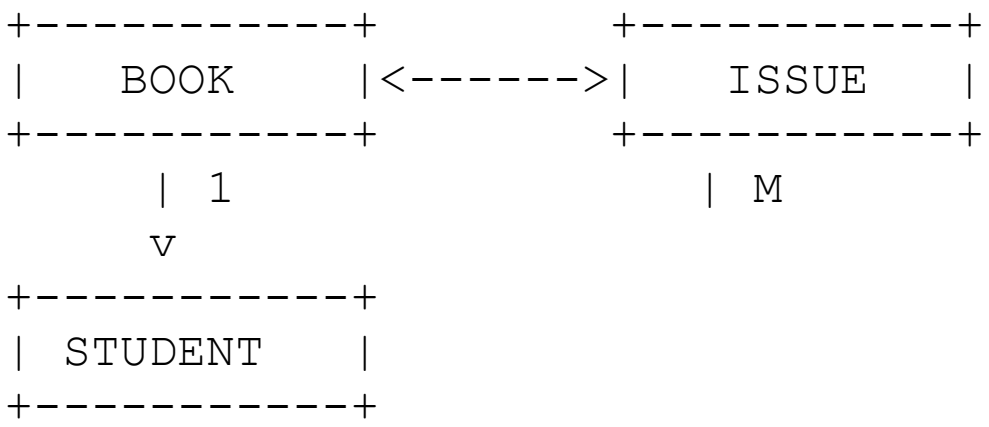
Level 1 (Detailed DFD)

- **Add Book** → Updates file

- **Issue Book** → Decreases available count
- **Return Book** → Increases count
- **Search Book** → Reads and displays matching records

---

12. ENTITY-RELATIONSHIP (ER) DIAGRAM



---

13. ALGORITHM (MAIN MODULE)

1. Start
2. Display main menu
3. Accept user choice
4. Based on choice:
    - Add book
    - Search book
    - Issue/Return book
    - Modify/Delete record
5. Repeat until Exit selected
6. Stop

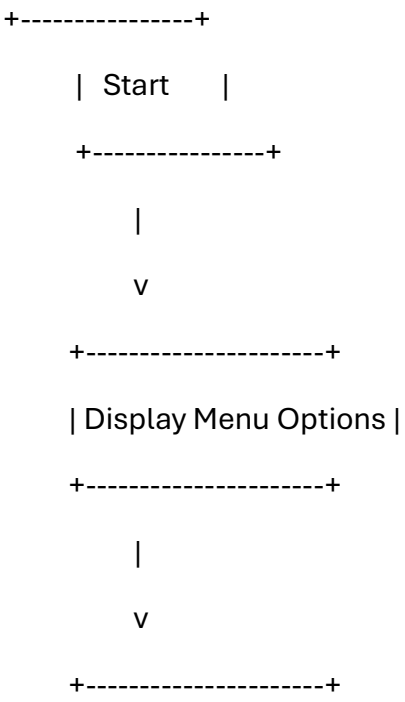
---

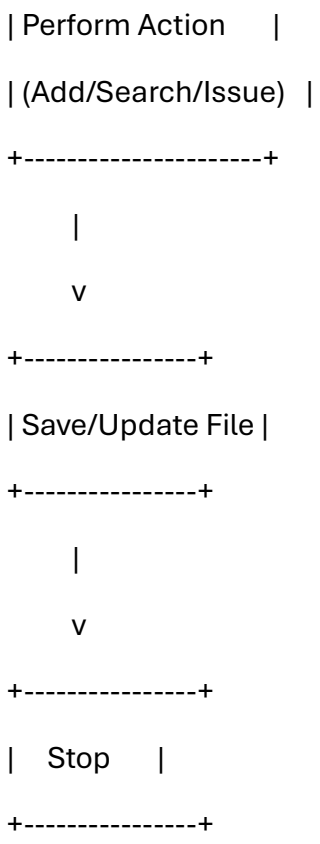
14. PSEUDOCODE

```
BEGIN
  DISPLAY Main Menu
  INPUT choice
  SWITCH(choice)
    CASE 1: call addBook()
    CASE 2: call displayBooks()
    CASE 3: call searchBook()
    CASE 4: call issueBook()
    CASE 5: call returnBook()
    CASE 6: call modifyBook()
    CASE 7: call deleteBook()
    CASE 8: EXIT
  END SWITCH
END
```

---

15. FLOWCHART





---

16. SOURCE CODE

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>


#define BOOKS_FILE “books.dat”

#define ISSUES_FILE “issues.dat”
```

```
#define TITLE_LEN 100
#define AUTHOR_LEN 50

Typedef struct {
    Int id;
    Char title[TITLE_LEN];
    Char author[AUTHOR_LEN];
    Int quantity;
    Int available;
} Book;

Typedef struct {
    Int bookId;
    Char studentName[50];
    Char studentId[20];
    Char issueDate[12];
    Char returnDate[12];
} Issue;
```

```
Void clear_input_buffer();
Void pause_console();
Void addBook();
Void displayBooks();
Void searchBook();
Void issueBook();
Void returnBook();
Void modifyBook();
Void deleteBook();
Int bookExists(int id, Book *out);
Void toLowerStr(char *s);

Int main() {
    Int choice;
    While (1) {
        System(“clear || cls”);
        Printf(“\n====== Library Management System
======\n”);
        Printf(“1. Add Book\n”);
```

```
        Printf("2. Display All Books\n");
        Printf("3. Search Book\n");
        Printf("4. Issue Book\n");
        Printf("5. Return Book\n");
        Printf("6. Modify Book\n");
        Printf("7. Delete Book\n");
        Printf("8. Exit\n");
        Printf("Enter choice: ");
        If (scanf("%d", &choice) != 1) { clear_input_buffer(); choice = 0; }
        Switch (choice) {
            Case 1: addBook(); break;
            Case 2: displayBooks(); break;
            Case 3: searchBook(); break;
            Case 4: issueBook(); break;
            Case 5: returnBook(); break;
            Case 6: modifyBook(); break;
            Case 7: deleteBook(); break;
            Case 8: printf("Exiting…\n"); exit(0);
```

```
        Default: printf("Invalid choice, try again.\n");
pause_console();
    }
  }
  Return 0;
}

Void clear_input_buffer() {
  Int c; while ((c = getchar()) != '\n' && c != EOF);
}
Void pause_console() {
  Printf("\nPress ENTER to continue…");
  Clear_input_buffer();
}

Int bookExists(int id, Book *out) {
  FILE *fp = fopen(BOOKS_FILE, "rb"); if (!fp) return 0;
Book b;
  While (fread(&b, sizeof(Book), 1, fp) == 1) {
```

```
    If (b.id == id) { if (out) *out = b; fclose(fp); return 1; }
  } fclose(fp); return 0;
}

Void addBook() {
  Book b; system("clear || cls"); printf("\n--- Add New Book ---\n");
  Printf("Enter Book ID: "); if (scanf("%d", &b.id) != 1) { clear_input_buffer(); printf("Invalid ID.\n"); pause_console(); return; }
  If (bookExists(b.id, NULL)) { printf("Book with ID %d exists.\n", b.id); pause_console(); return; }
  Clear_input_buffer(); printf("Enter Title: "); fgets(b.title, TITLE_LEN, stdin); b.title[strcspn(b.title, "\n")] = '\0';
  Printf("Enter Author: "); fgets(b.author, AUTHOR_LEN, stdin); b.author[strcspn(b.author, "\n")] = '\0';
  Printf("Enter Quantity: "); scanf("%d", &b.quantity); b.available = b.quantity;
  FILE *fp = fopen(BOOKS_FILE, "ab"); fwrite(&b, sizeof(Book), 1, fp); fclose(fp);
  Printf("Book added successfully.\n"); pause_console();
}
```

```
Void displayBooks() {
   Book b; system("clear || cls"); FILE *fp =
fopen(BOOKS_FILE, "rb");
   If (!fp) { printf("No records found.\n"); pause_console();
return; }
   Printf("ID\tTitle\tAuthor\tQty\tAvail\n"); printf("----------
--------------------------\n");
   While (fread(&b, sizeof(Book), 1, fp) == 1) {
      Printf("%d\t%s\t%s\t%d\t%d\n", b.id, b.title, b.author,
b.quantity, b.available);
   } fclose(fp); pause_console();
}
```

---

17. OUTPUT SCREENS

```
===== Library Management System =====
1. Add Book
2. Display Books
3. Search Book
```

4. Update Book

5. Delete Book

6. Issue Book

7. Return Book

8. Display Issued Books

9. Exit

Enter your choice: 1

Enter book title: C Programming

Enter author name: Brian Kernighan

Enter quantity (number of copies): 3

Book added successfully with ID: 1001

Enter your choice: 2

ID Title Author Qty Avail

1001 C Programming Brian Kernighan 3 3

Enter your choice: 6

Enter book ID to issue: 1001

Enter student ID: 2001

Enter student name: Alice

Enter issue date (dd-mm-yyyy): 10-11-2025

Book issued successfully to Alice (Student ID: 2001)

Enter your choice: 2

ID Title Author Qty Avail

1001 C Programming Brian Kernighan 3 2

Enter your choice: 8

BookID StudentID StudentName IssueDate ReturnDate

1001 2001 Alice 10-11-2025 -

Enter your choice: 7

Enter Book ID: 1001

Enter Student ID: 2001

Enter return date (dd-mm-yyyy): 15-11-2025

Book returned successfully.

Enter your choice: 2

ID Title Author Qty Avail

1001 C Programming Brian Kernighan 3 3

---

18. TESTING

Testing Methods

- **Unit Testing**: Tested each function separately.
- **Integration Testing**: Verified module interaction.
- **System Testing**: Validated entire application.

- **User Acceptance Testing**: Confirmed correct output with sample data.

Test Cases

Case	Input	Expected Output	Result
1	Add Book (101)	Success	Pass
2	Search “C”	Display result	Pass
3	Issue Book (101)	Availability = 4	Pass
4	Return Book (101)	Availability = 5	Pass

---

19. VALIDATION

All inputs are validated:

- Book IDs are unique.
- Quantities must be positive integers.
- Invalid entries display error messages.

---

20. LIMITATIONS

- No GUI (text-based only).
- No user login system.
- Data storage limited to local machine.
- No network connectivity.

21. FUTURE SCOPE

- Integration with MySQL or SQLite database.
- GUI-based interface (C++/Java/Python frontend).
- Multi-user login and role-based access.
- Fine calculation for late returns.
- Barcode integration for real libraries.

22. CONCLUSION

The Library Management System developed in C language is a simple yet effective application for managing the day-to-day activities of a library. It automates tasks such as adding, searching, issuing, and returning books, which helps reduce manual effort and errors.

Through this project, I gained practical knowledge of important C programming concepts like file handling, structures, loops, and modular programming. The system provides a reliable and user-friendly way to store and manage data efficiently.

Overall, this project not only demonstrates the use of programming for solving real-life problems but also strengthens the understanding of how logical thinking and coding can be combined to build useful applications.

23. BIBLIOGRAPHY

Programming in ANSI C – E. Balagurusamy

Let Us C – Yashavant Kanetkar

www.geeksforgeeks.org

www.tutorialspoint.com