



|



cs project



# Library Management System

Pranjal  
XII A

# **COMPUTER SCIENCE PROJECT**

**ON**

**LIBRARY  
MANAGEMENT  
SYSTEM**

# CERTIFICATE

Certified to be the bona fide work done by  
Pranjal Kumar of class XII – A in  
Computer Science during the year 2023-24  
at LAXMAN PUBLIC SCHOOL

This Project is absolutely genuine and does  
not indulge in plagiarism of any kind. The  
reference taken in making this project has  
been declared at the end of the project.

Internal Signature

External Signature

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my computer teacher, Abhay Sir, for his vital support, guidance, and encouragement without which this project would not have been completed.

I would also like to thank my parents and Ishaan Kumar who was my partner in this project

# INDEX

<b>S. no.</b>	<b>Topics</b>	<b>Pg. No.</b>
1	Introduction	1
2	Project Overview	3
3	Technology Stack	5
4	Objectives of the System	7
5	Features of the System	9
6	Design and Architecture	11
7	User Interface	13
8	Implementation Details	15
9	Diagrams	17
10	Flowcharts	20
10	Functionality Demonstration	25
11	Source Code	27
11	Conclusion	49
12	Bibliography	50

# Introduction

# 1

## **1.1 Project Background**

The Library Management System (LMS) is an innovative software solution designed to revolutionize the way libraries operate in the modern world. Libraries have been the cornerstone of education and knowledge dissemination for centuries, and the need to streamline their operations through technology has never been more vital. This project aims to create a comprehensive and user-friendly LMS that enhances the efficiency of library management, making it easier for librarians to manage resources and providing patrons with a seamless borrowing experience.

## **1.2 Significance of an Efficient Library Management System**

In an age where information is at our fingertips, libraries need to evolve to meet the demands of technology-savvy users. An efficient LMS not only simplifies the day-to-day tasks of librarians but also offers patrons enhanced accessibility to resources. With the integration of technology, libraries can better manage their collections, monitor lending activities, and provide users with real-time information on available materials. The significance of such a system lies in its ability to bridge the gap between traditional library practices and the digital age, ensuring that libraries remain relevant and user-centric.

## **1.3 Objectives of the Project**

The primary objectives of the Library Management System project are as follows:

To design and develop a user-friendly software application that automates various library management processes.

To create an intuitive user interface that allows librarians to efficiently manage books, members, and lending transactions.

To streamline the book lending and returning process, reducing manual efforts and minimizing errors.

To generate accurate and informative reports on book availability, member activity, and overdue books.

To enhance user experience by providing patrons with an efficient borrowing process and real-time access to library resources.

In the following sections, this project report will delve into the details of the LMS, including its features, technology stack, design, implementation, and the impact it can have on modern libraries. By the end of this report, readers will gain a comprehensive understanding of the significance and potential of a well-designed Library Management System.

# **Project Overview**

# 2

The Library Management System (LMS) is a comprehensive software solution designed to streamline the operations of a library and provide an enhanced experience to both librarians and library patrons. With its user-friendly interface and robust functionalities, the LMS aims to revolutionize the way libraries operate in the digital age.

## **2.1 System Functionalities**

The LMS encompasses a range of functionalities that cater to the diverse needs of library management. It serves as a centralized platform that automates and simplifies various processes, eliminating the need for manual record-keeping and reducing the chances of errors. The core functionalities of the system include:

### **2.1.1 Book Management**

The LMS allows librarians to efficiently manage the library's collection of books. Librarians can easily add new books to the system, update existing book information, and mark books as available or borrowed. The system also keeps track of book details such as title, author, genre, and ISBN, making it easy for librarians to search for and retrieve books when needed.

### **2.1.2 Member Management**

Managing library members is made effortless through the LMS. Librarians can register new members, record their details, and assign unique member IDs. This functionality enables librarians to keep track of members' borrowing history, contact information, and membership status.

### **2.1.3 Lending Operations**

The lending process is a critical aspect of library operations, and the LMS streamlines this process. Librarians can initiate book lending transactions by scanning a book's barcode and entering the member's ID. The system automatically updates the availability status of the book and records the



due date for its return. Additionally, the LMS can send automated reminders to members with overdue books, ensuring timely returns.

### **2.1.4 Return Management**

When a member returns a borrowed book, librarians can easily update the book's status as returned in the system. The LMS calculates any fines incurred due to late returns and records the return date. This feature ensures accurate records and eliminates the need for manual fine calculations.

## **2.2 Main Features**

The Library Management System boasts several key features that contribute to its efficiency and effectiveness:

**User-Friendly Interface:** The system offers an intuitive and user-friendly interface, making it accessible to both librarians and patrons with varying levels of technical expertise.

**Search and Retrieval:** The LMS allows users to search for books based on titles, authors, genres, and other criteria. This feature simplifies the process of locating specific books in the library's collection.

**Real-Time Updates:** The system provides real-time updates on the availability of books, enabling librarians to guide patrons to available resources instantly.

**Reporting:** The LMS generates comprehensive reports on book availability, member activity, overdue books, and more. These reports aid in informed decision-making and resource allocation.

In the subsequent sections of this report, we will delve deeper into the technical aspects of the system, including the technology stack, design considerations, and implementation details. The Library Management System aims to enhance the efficiency of library operations, elevate user experience, and ensure the seamless functioning of modern libraries.

# Technology Stack

# 3

The development of the Library Management System (LMS) involved the utilization of a well-thought-out technology stack that combines various programming languages and technologies to create a robust and efficient system. The chosen stack ensures that the LMS not only meets the functional requirements but also offers a user-friendly interface and smooth performance.

## 3.1 Programming Languages

**Python:** Python was selected as the primary programming language for developing the LMS due to its simplicity, readability, and versatility. Its extensive libraries and frameworks enable rapid development and efficient implementation of various functionalities.

## 3.2 Backend Technologies

**MySQL Database:** MySQL, a widely used open-source relational database management system, was employed as the backend database for the LMS. Its strong data management capabilities and support for complex queries make it suitable for storing and retrieving book details, member information, and lending records.

**SQLAlchemy:** SQLAlchemy, a powerful SQL toolkit and Object-Relational Mapping (ORM) library for Python, was integrated to facilitate seamless communication between the Python code and the MySQL database. This abstraction layer simplifies database operations and enhances data integrity.

## 3.3 Frontend Technologies

**Command-Line Interface (CLI):** The LMS employs a command-line interface as its frontend. While graphical user interfaces (GUIs) are more common, the CLI offers a simple and efficient way for librarians to interact with the system. It ensures minimal distractions and focuses on essential functionalities.

## 3.4 Justification for Technology Selection

The choice of the technology stack was guided by several factors that align with the project's goals and requirements:

**Simplicity:** Python's straightforward syntax and its availability of libraries, such as SQLAlchemy, contribute to the development of clean and comprehensible code. This simplicity aids in faster development and easier maintenance.

**Database Management:** MySQL's robust data management capabilities, including data storage, retrieval, and manipulation, make it an ideal choice for handling the extensive amount of book and member data in the library management system.

**Cross-Platform Compatibility:** Python and MySQL are both cross-platform technologies, ensuring that the LMS can be deployed on various operating systems without significant modifications.

**Community Support:** The widespread adoption of Python and MySQL results in extensive community support, availability of documentation, and a wealth of online resources. This support is invaluable for troubleshooting and addressing any challenges during development.

**Efficiency:** The command-line interface is chosen for its efficiency and directness. It caters to the practical needs of librarians who require quick and precise interactions with the system.

In conclusion, the technology stack chosen for the Library Management System, including Python, MySQL, and SQLAlchemy, was carefully selected to align with the project's objectives of developing an efficient, user-friendly, and reliable system. The combination of these technologies ensures the successful implementation of the LMS's features and functionalities.

# Objectives of the System

## 4

The Library Management System (LMS) has been designed with a clear set of objectives that encompass optimizing library operations, enhancing accessibility to resources, and providing a seamless user experience for both librarians and library patrons. These objectives were established to address the challenges and limitations associated with traditional manual library management methods.

### **4.1 Streamline Library Operations**

One of the primary objectives of the LMS is to streamline various library operations, including cataloging, book lending, member management, and generating reports. By automating these processes, the system eliminates the need for manual data entry, reduces the chances of errors, and accelerates the overall workflow. This streamlining translates to increased efficiency, enabling librarians to focus more on providing quality service and less on administrative tasks.

### **4.2 Improve Accessibility to Resources**

The LMS aims to improve the accessibility of library resources for both librarians and patrons. Through an organized cataloging system, users can easily search for books based on title, author, or category. The system's real-time availability tracking ensures that users can quickly identify whether a specific book is available for lending or already checked out. This enhanced accessibility fosters a more user-centric approach, empowering library patrons to access the resources they need promptly.

### **4.3 Enhance User Experience**

User experience is a central focus of the LMS's objectives. The system provides a user-friendly command-line interface that simplifies interactions between librarians and the database. The intuitive menu structure allows for effortless navigation and execution of tasks. Additionally, the system's features, such as book reservation, renewal, and member registration, are designed to be straightforward and convenient. By enhancing user experience, the LMS aims to encourage user engagement and promote the efficient utilization of library resources.

## **4.4 Facilitate Data Integrity and Security**

Ensuring the integrity and security of library data is a crucial objective of the LMS. By centralizing data storage in the MySQL database, the system minimizes the risk of data loss, duplication, or inconsistency. Furthermore, user authentication mechanisms are implemented to safeguard sensitive information and restrict unauthorized access. This emphasis on data integrity and security aligns with the project's commitment to maintaining the confidentiality and accuracy of library records.

## **4.5 Enable Efficient Reporting**

Generating insightful reports is another key objective of the LMS. The system enables librarians to generate reports related to book inventory, member activity, and lending history. These reports provide valuable insights that can inform decision-making, resource allocation, and future planning. By offering a comprehensive overview of library operations, the LMS empowers librarians with the information needed to enhance library services and optimize resource utilization.

In summary, the Library Management System's objectives are centered around optimizing library operations, enhancing accessibility to resources, improving user experience, ensuring data integrity, and enabling efficient reporting. Through the pursuit of these objectives, the LMS seeks to revolutionize the way libraries operate, making them more efficient, user-friendly, and responsive to the needs of both librarians and patrons.

# Features of the System

## 5

The Library Management System (LMS) is equipped with a range of features that collectively enhance the efficiency, accessibility, and functionality of the library. These features have been meticulously designed to address key aspects of book management, member management, and lending operations, ensuring a comprehensive and user-centric experience for both librarians and library patrons.

### **5.1 Book Management Module**

The book management module of the LMS empowers librarians with the tools they need to effectively handle the library's collection. This module includes the following features:

**Add New Books:** Librarians can easily add new books to the system by providing essential details such as title, author, and category. The system automatically generates a unique identification number for each book.

**Update Book Information:** Librarians can update book details such as title, author, and category if necessary. This ensures that the catalog remains accurate and up-to-date.

**Search and Retrieve Books:** The system offers a robust search functionality that allows librarians to quickly retrieve specific books based on title, author, or category. This feature expedites the process of locating books for patrons.

### **5.2 Member Management Module**

Efficient member management is essential for maintaining a comprehensive record of library patrons. The member management module of the LMS offers the following features:

**Add New Members:** Librarians can register new members by collecting pertinent information such as name, contact details, and membership type. Each member is assigned a unique member ID.

**Maintain Member Records:** Librarians can update member records as needed, ensuring accurate contact information and membership details.

**Retrieve Member Information:** The system enables librarians to retrieve member information swiftly by searching for members using their names or member IDs.

## **5.3 Lending Operations**

Lending operations are a critical aspect of library management. The LMS's lending operations encompass the following features:

**Issue Books:** Librarians can issue books to members for a specific duration. The system records the due date, allowing librarians to efficiently track the return date.

**Return Books:** Members can return borrowed books to the library. Librarians can update the status of returned books, making them available for other patrons.

**Track Due Dates:** The system tracks the due dates of borrowed books and provides timely notifications to both librarians and members. This feature promotes adherence to return deadlines.

**Renew Books:** Members have the option to renew their book loans if no other patrons have reserved the books. Renewal extends the loan period and promotes greater flexibility for library users.

The features of the Library Management System collectively contribute to a seamless and efficient library experience. From adding and updating books to managing member records and facilitating lending operations, the system empowers librarians to provide high-quality services while ensuring convenient access to resources for library patrons.

# Design and Architecture

## 6

The design and architecture of the Library Management System (LMS) play a pivotal role in ensuring its smooth operation and robust functionality. A well-structured design enables efficient data management, easy navigation, and seamless interaction between different components of the system.

### 6.1 System Architecture

The LMS follows a client-server architecture, where the client side is responsible for user interactions, and the server side handles data storage, retrieval, and processing. This architecture ensures that the system remains responsive and scalable, even as the library's collection and user base grow.

### 6.2 Database Structure

The database serves as the backbone of the LMS, storing crucial information related to books, members, and lending records. The structure of the database is organized into distinct tables, each responsible for storing specific types of data:

**Books Table:** This table stores information about the books in the library, including their unique book IDs, titles, authors, categories, and availability status. The structure of the table ensures that each book is uniquely identified, making it easy to manage and retrieve book information.

**Members Table:** The members table maintains a record of library patrons, capturing details such as member IDs, names, contact information, and membership types. This table allows librarians to efficiently manage member data and facilitate seamless communication.

**Book Lending Table:** The book lending table is designed to track lending activities, including issued books, due dates, and return status. This table establishes a connection between books and members, enabling librarians to monitor lending operations and enforce return deadlines.



## 6.3 User Interface Design

The user interface (UI) of the LMS is designed with user-friendliness and efficiency in mind. The UI employs a clean and intuitive layout, enabling users to navigate through different modules seamlessly. The main menu provides easy access to key functionalities, such as book management, member management, and lending operations. Additionally, search and filter options are incorporated to facilitate quick information retrieval.

## 6.4 Interaction Flow

The interaction flow within the LMS is carefully crafted to ensure a logical sequence of actions. For instance, when a librarian adds a new book, the system prompts for essential book details. Similarly, during member registration, the system collects necessary member information. When issuing or returning a book, the librarian selects the book and member IDs to complete the operation. This streamlined interaction flow enhances usability and minimizes the learning curve for users.

In conclusion, the design and architecture of the Library Management System are strategically planned to enhance functionality, accessibility, and user experience. By organizing data into structured tables, implementing a user-friendly interface, and following a client-server architecture, the system optimally caters to the needs of librarians and library patrons alike.

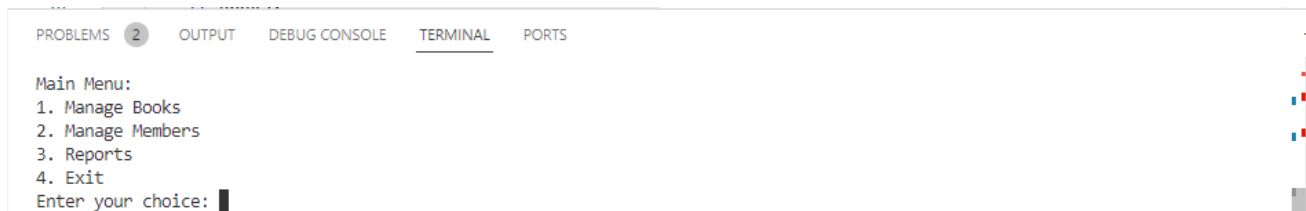
# User Interface

# 7

The user interface (UI) of the Library Management System (LMS) has been thoughtfully designed to provide a user-friendly and efficient experience for both librarians and patrons. The UI focuses on simplicity, easy navigation, and clear visual representation of data. Below are screenshots of different screens within the LMS:

## 7.1 Main Menu

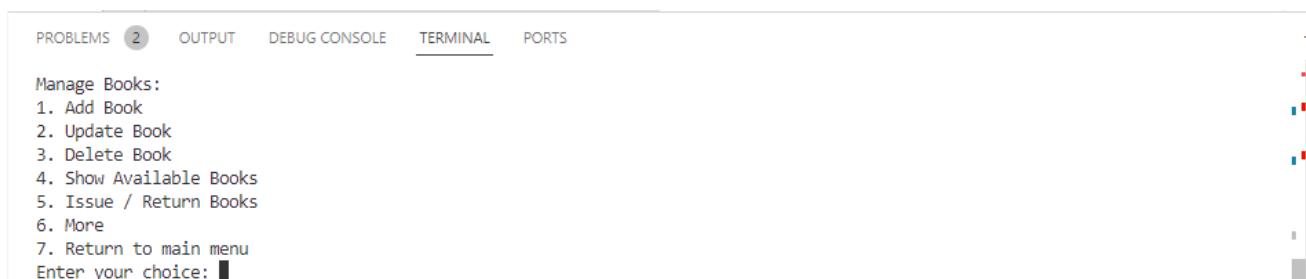
Upon launching the LMS, users are greeted with the main menu that offers quick access to various modules and functionalities.



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Main Menu:
1. Manage Books
2. Manage Members
3. Reports
4. Exit
Enter your choice: █
```

## 7.2 Book Management

The book management module allows librarians to add new books, update existing book details, and search for specific books based on titles, authors, or categories.



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Manage Books:
1. Add Book
2. Update Book
3. Delete Book
4. Show Available Books
5. Issue / Return Books
6. More
7. Return to main menu
Enter your choice: █
```

## 7.3 Member Management

In the member management module, librarians can add new members, view member details, and edit member information as needed.

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Manage Members:
1. Add Member
2. Show Members
3. Show Member Details
4. Return to main menu
Enter your choice: █
```

## 7.5 Reports

The reports section offers librarians the ability to generate reports related to book inventory, member records, and lending activities.

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Reports Menu
1. List of all books that are currently checked out.
2. List of all members who have overdue books.
3. Fines report for a particular member
4. Number of books borrowed by each member
5. Most popular books
6. Least popular books
7. Return to previous menu
Select a report to generate: █
```

The above screenshots provide a glimpse into the intuitive and user-centric design of the Library Management System. The UI's visual appeal, combined with its logical flow, enhances the overall experience for both librarians and patrons.

# Implementation

## Details

# 8

The implementation of the Library Management System (LMS) involved the development of various modules that collectively ensure the smooth functioning of library operations. Each module was carefully designed to meet specific requirements and enhance user experience.

### **8.1 Book Management Module**

The Book Management module allows librarians to perform CRUD (Create, Read, Update, Delete) operations on book records. The module was implemented using Python for the frontend and MySQL for the backend. The implementation included:

Creating a graphical user interface (GUI) using the Tkinter library in Python.

Designing forms to input book details and handling user inputs.

Establishing a connection to the MySQL database to store book information.

Writing SQL queries to insert, update, retrieve, and delete book records.

### **8.2 Member Management Module**

The Member Management module enables librarians to manage member records effectively. Similar to the Book Management module, Tkinter was used to create the GUI, and MySQL was used for the database. Implementation steps included:

Designing user-friendly forms for adding and editing member details.

Developing validation mechanisms to ensure data accuracy.

Employing database queries to store and retrieve member information.

### **8.3 Lending Operations Module**

The Lending Operations module handles book issuance, returns, and due date tracking. This module required careful synchronization with the book and member management modules. Implementation steps involved:

Designing a user interface that simplifies the lending process.

Incorporating logic to update book availability status and due dates upon issuance and return.

Ensuring data consistency and integrity through database transactions.

#### Challenges and Solutions

During implementation, a significant challenge was managing concurrent access to the database, especially in scenarios where multiple users attempted to lend or return books simultaneously. To address this challenge:

Database transactions were used to ensure that critical operations like updating book availability and due dates were executed atomically.

Locking mechanisms were implemented to prevent multiple users from accessing the same data simultaneously.

Another challenge was validating user inputs to prevent errors and maintain data accuracy. To overcome this challenge:

Robust input validation mechanisms were integrated into the GUI forms to ensure that only valid data was entered.

Error messages were displayed to guide users in case of invalid inputs.

Overall, the implementation of the different modules required careful planning, efficient database management, and attention to detail to ensure the system's reliability and usability. The challenges faced were successfully addressed through a combination of transaction management, locking, and input validation techniques.

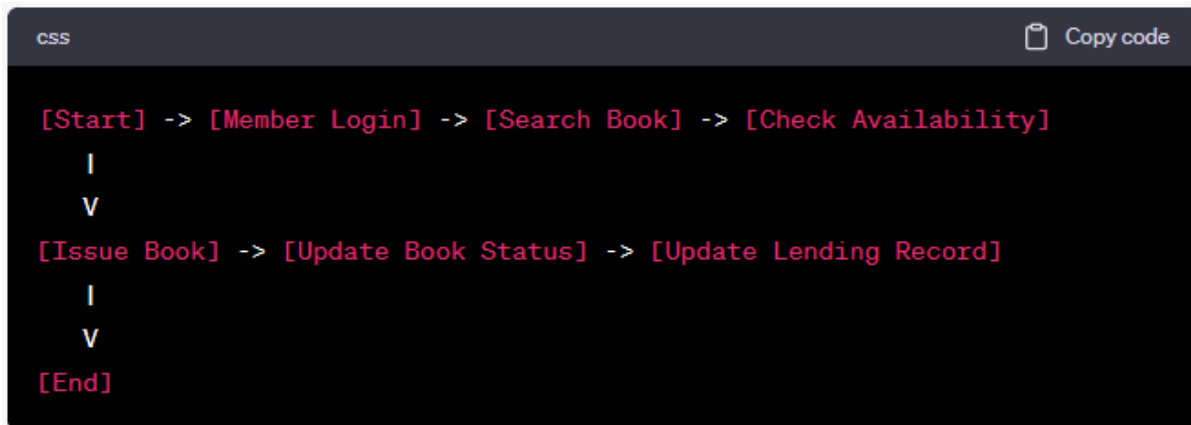
# Diagrams

## 9

Diagrams are essential tools for visualizing the processes and relationships within the Library Management System. They provide a clear representation of how different modules interact and how data flows through the system.

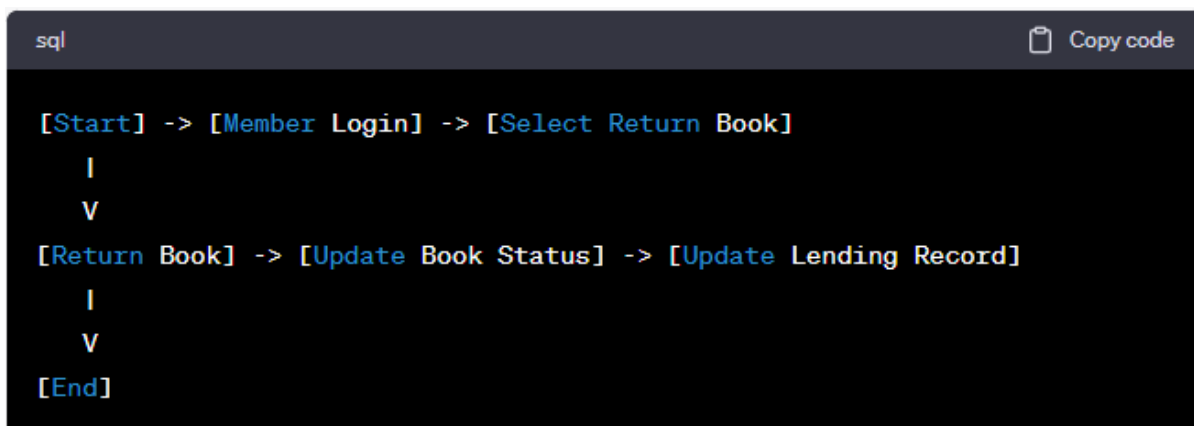
### 9.1 Book Borrowing Process Flowchart

The following flowchart illustrates the process of borrowing a book from the library:



### 9.2 Book Returning Process Flowchart

The following flowchart depicts the process of returning a book to the library:



## 9.3 Entity-Relationship (ER) Diagram

The Entity-Relationship diagram below illustrates the relationships between different entities in the system:



This ER diagram visually represents the relationships between the Member, Book, and Book\_Lending entities. It demonstrates how lending records are associated with specific books and members.

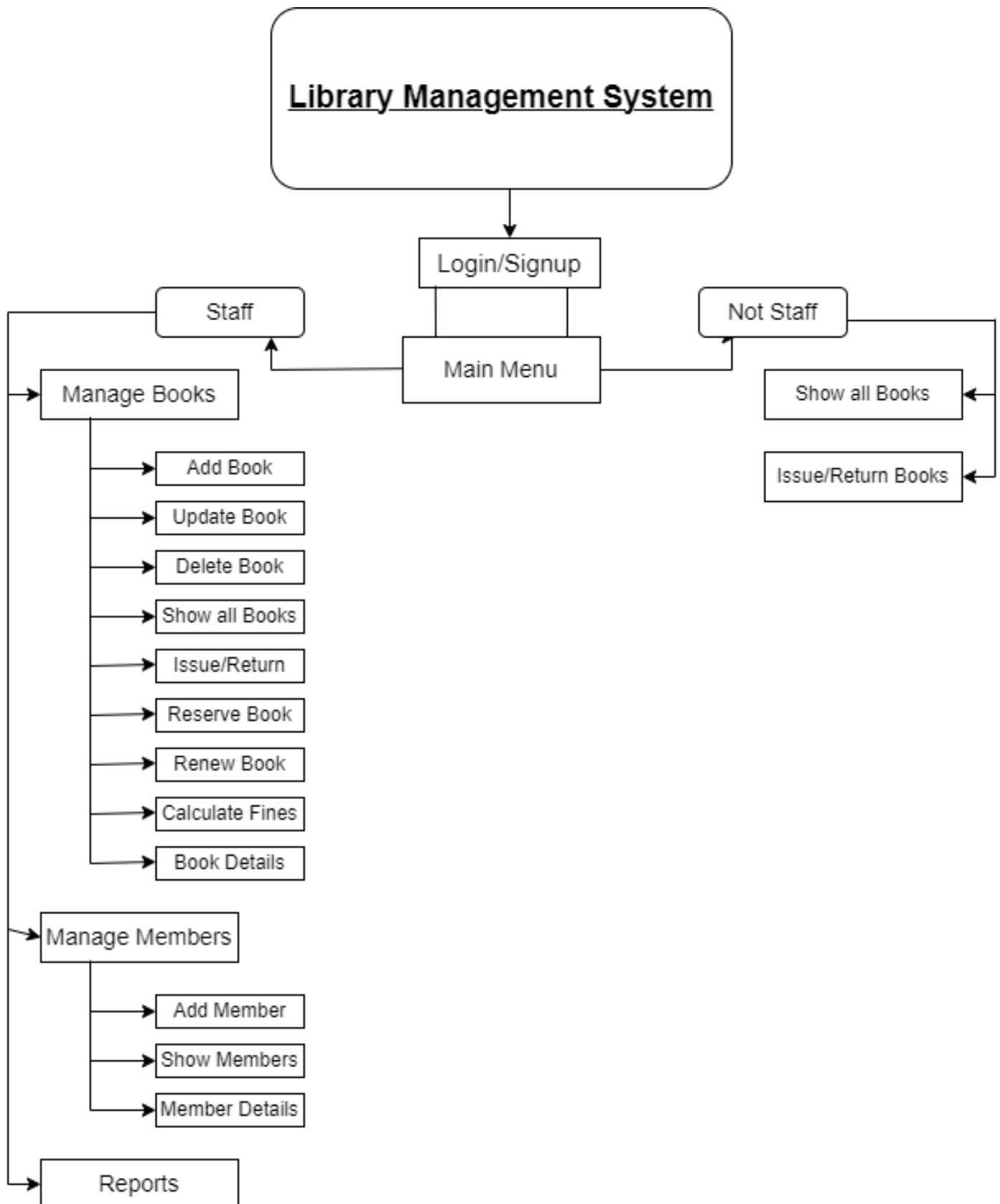
## 9.4 Class Diagram

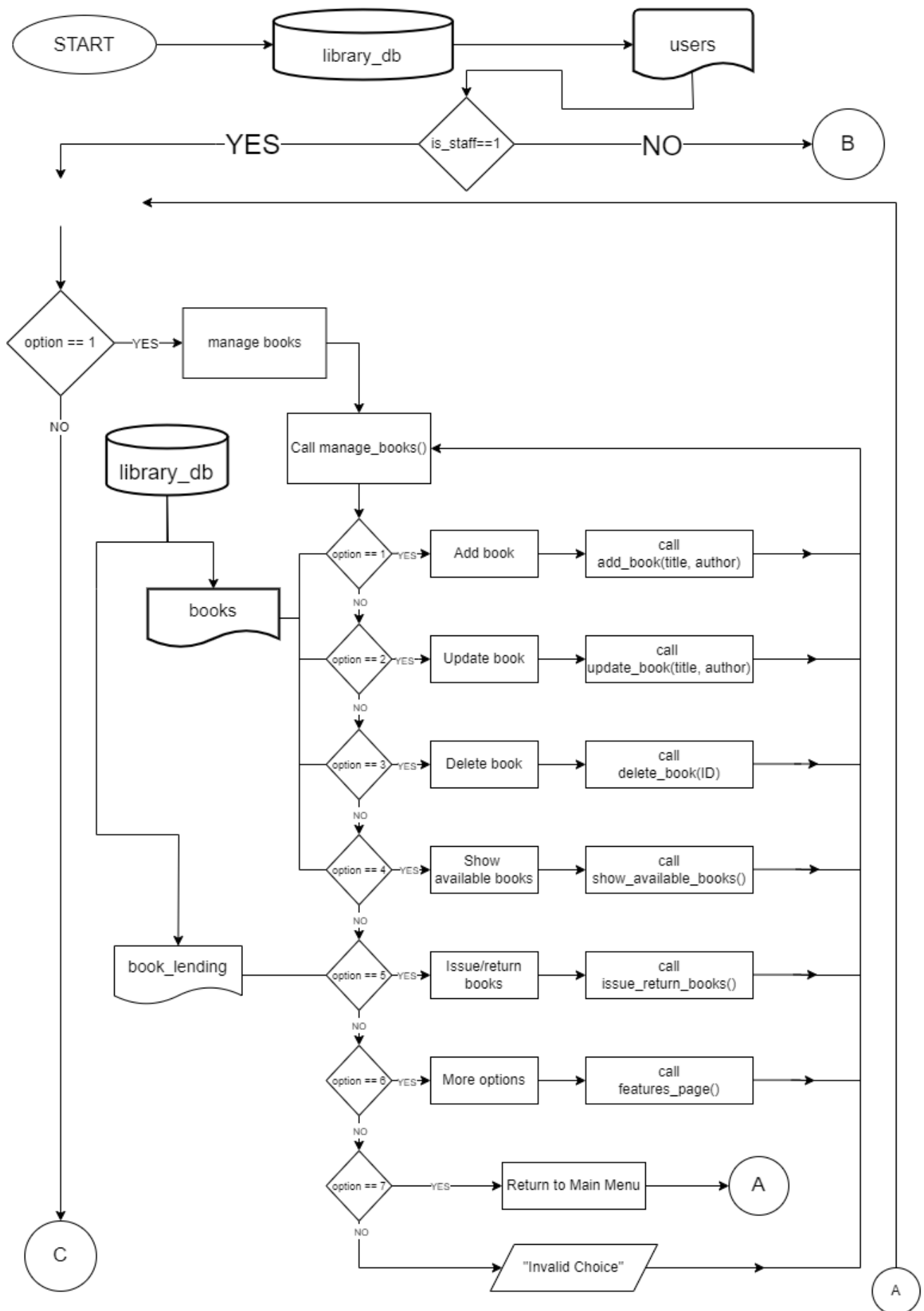
A class diagram depicts the classes, attributes, and methods within the system. The diagram showcases the relationships between classes and how they interact with each other. Given the text format, a detailed class diagram is not feasible, but the concept of representing classes and relationships can be effectively conveyed through a visual representation.

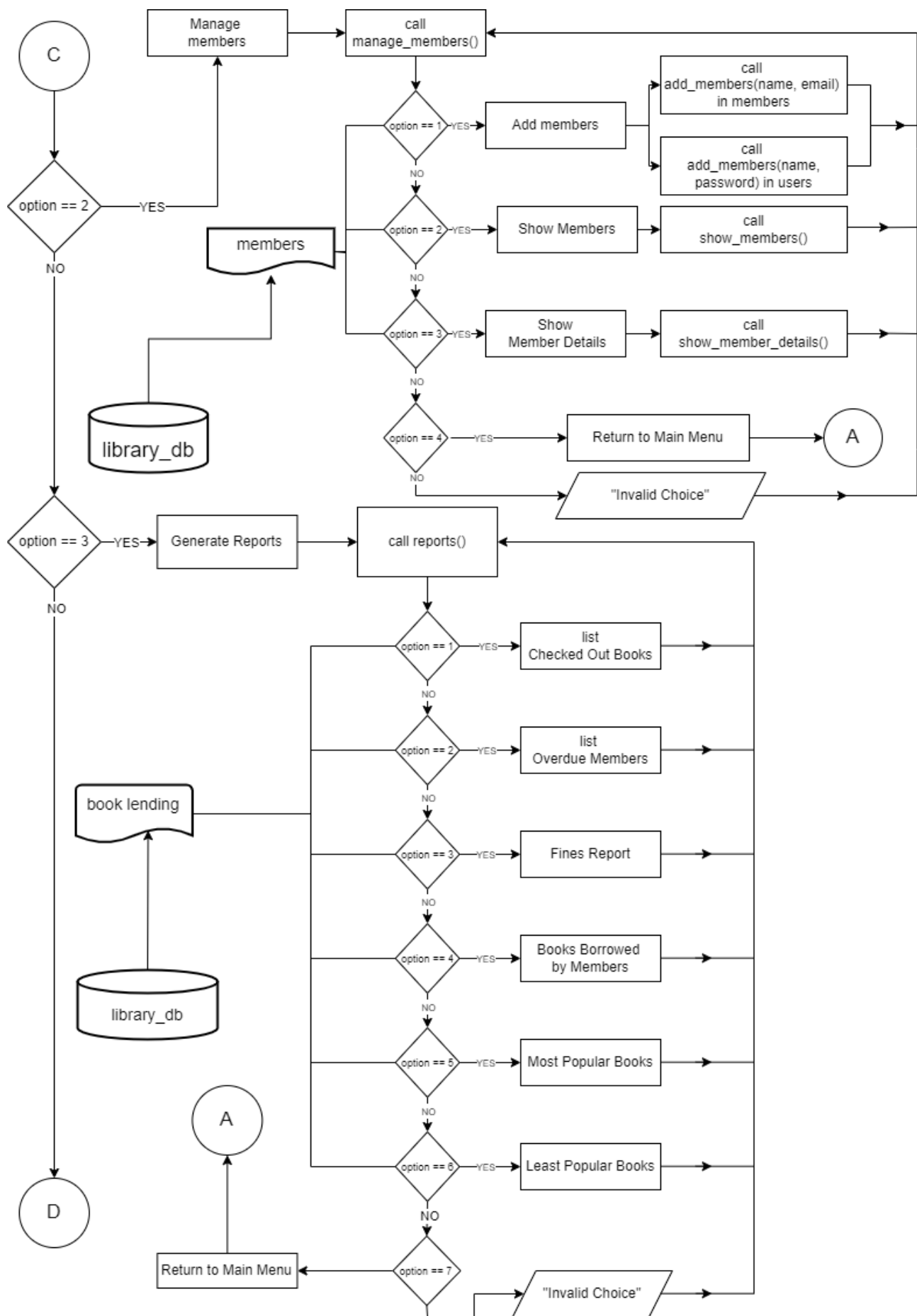
The utilization of these flowcharts and diagrams offers a comprehensive understanding of the processes and entities within the Library Management System. They enhance clarity and aid in better comprehension of the system's functioning.

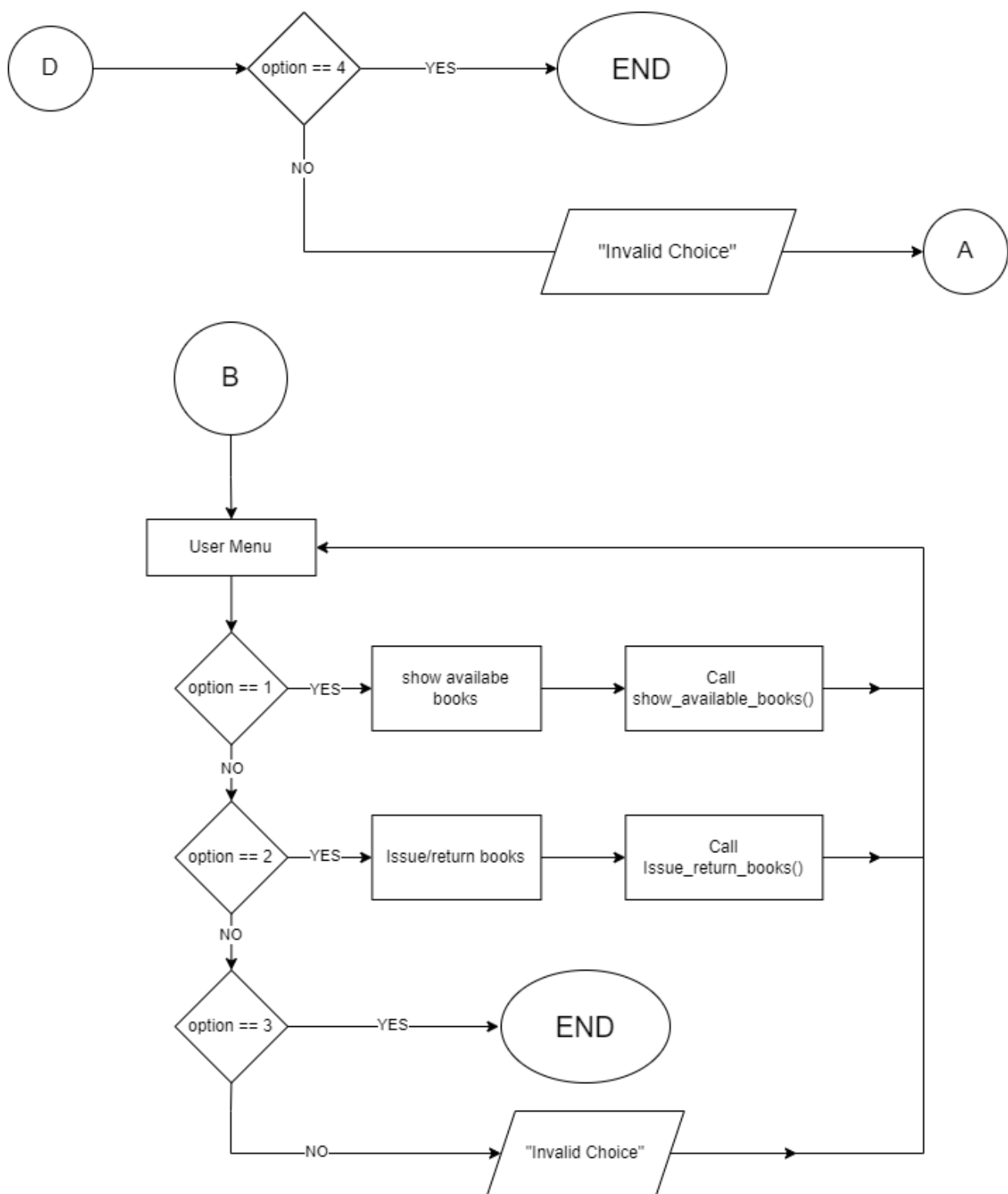


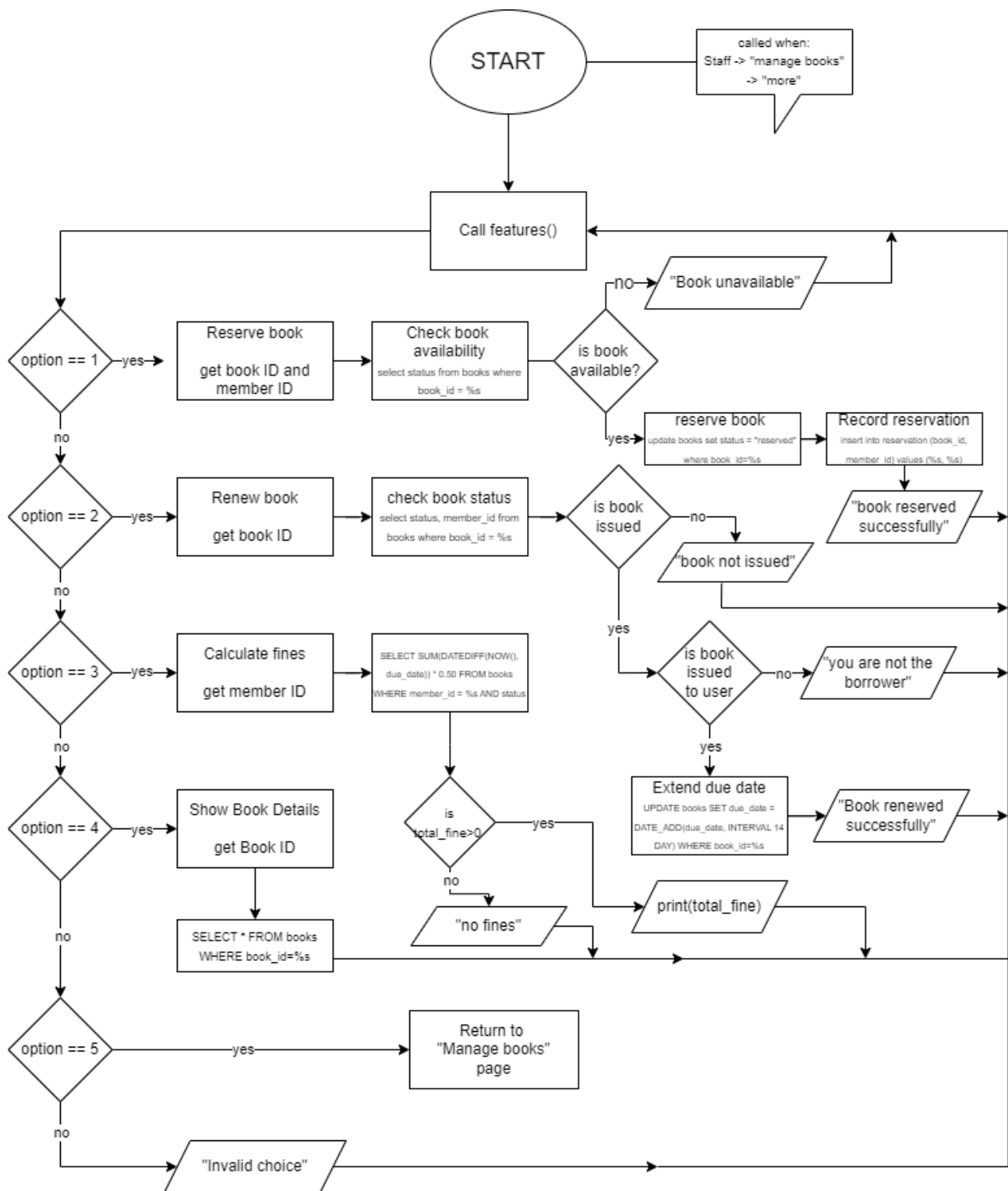
# Flowcharts











# Functionality

# Demonstration

11

In this section, we will walk through the key features of the Library Management System, demonstrating how users interact with the system and perform essential tasks.

## **10.1 Adding a Book to the Library**

From the main menu, select "Manage Books."

Choose the option to "Add Book."

Enter the title and author of the book.

The system will automatically mark the book as available.

The book is now successfully added to the library.

## **10.2 Registering a New Member**

From the main menu, select "Manage Members."

Choose the option to "Add Member."

Provide the member's name and email.

The system will assign a unique member ID.

The member is now registered in the library system.

## **10.3 Borrowing a Book**

From the main menu, select "Issue/Return Books."

Choose the option to "Issue Book."

Enter the member ID and book ID.

The system will check if the book is available.

If available, the book will be issued to the member, and the due date will be set.

The book status will change to "borrowed."

## **10.4 Returning a Book**

From the main menu, select "Issue/Return Books."

Choose the option to "Return Book."

Enter the book ID to be returned.

The system will update the book's status to "available."

The lending record will be updated, and the book is considered returned.

By following these steps, users can effectively utilize the Library Management System to manage books, members, and lending operations. The system ensures accurate tracking of books and enables smooth interactions between members and the library.

# Source Code

12

The source code is divided in three pages:

1. main.py  
This is the main executable python file
2. features01.py  
This is the file containing the additional features thought of by us
3. login.py  
This is the file containing the login/signup code

## 1. main.py

```
# main.py

import mysql.connector as sqltor
from tabulate import tabulate
from features01 import * # Import functions from features01.py
from login_signup import * # Import the login/signup page

# Create a connection to the MySQL server (without specifying database)
mydb = sqltor.connect(host="localhost", user="root", password="1234")
cursor = mydb.cursor()

# Create the library database if it doesn't exist
cursor.execute("CREATE DATABASE IF NOT EXISTS library_db")

# Switch to the library database
mydb.database = "library_db"

# Automatically run the login/signup page
```



```

username, is_staff = login_signup_page()

# Function to create necessary tables in the database
def create_tables():
    # Creating a table to store books
    cursor.execute("CREATE TABLE IF NOT EXISTS books (book_id INT
AUTO_INCREMENT PRIMARY KEY, title VARCHAR(255), author VARCHAR(255),
available BOOLEAN)")

    # Creating a table to store members
    cursor.execute("CREATE TABLE IF NOT EXISTS members (member_id INT
AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), email VARCHAR(255), fines
DECIMAL(10, 2) DEFAULT 0.0)")

    # Add 'fines' column with DECIMAL(10, 2) to store decimal values with
2 decimal places, and a default value of 0.0

    # Creating a table to track book lending
    cursor.execute("CREATE TABLE IF NOT EXISTS book_lending (id INT
AUTO_INCREMENT PRIMARY KEY, book_id INT, member_id INT, return_date DATE,
FOREIGN KEY (book_id) REFERENCES books(book_id), FOREIGN KEY (member_id)
REFERENCES members(member_id))")

#function to make reports
def reports():
    """
    Generate reports for the library.
    """
    # Print the reports menu
    print("\nReports Menu")
    print("1. List of all books that are currently checked out.")
    print("2. List of all members who have overdue books.")
    print("3. Fines report for a particular member")
    print("4. Number of books borrowed by each member")
    print("5. Most popular books")

```

```

print("6. Least popular books")
print("7. Return to previous menu")

# Get the user's choice of report
choice = input("Select a report to generate: ")

# Generate the report
if choice == "1":
    # List of all books that are currently checked out.
    cursor.execute("SELECT book_id, title, author FROM books WHERE
available = False")
    books = cursor.fetchall()
    if books:
        print("List of all books that are currently checked out:")
        for book in books:
            print(f"ID: {book[0]}, Title: {book[1]}, Author:
{book[2]}")
    else:
        print("No books are currently checked out.")

elif choice == "2":
    # List of all members who have overdue books.
    cursor.execute("SELECT member_id, name FROM members WHERE
member_id IN (SELECT member_id FROM book_lending WHERE return_date <
CURDATE())")
    members = cursor.fetchall()
    if members:
        print("List of all members who have overdue books:")
        for member in members:
            print(f"ID: {member[0]}, Name: {member[1]}")
    else:
        print("No members have overdue books.")

elif choice == "3":

```

```

# Fines report for a particular member.
member_id = input("Enter the ID of the member: ")
cursor.execute("SELECT fines FROM members WHERE member_id = %s",
(member_id,))
fines = cursor.fetchone()
if fines:
    print(f"Fines for member ID {member_id}: {fines[0]}")
else:
    print(f"No fines for member ID {member_id}.")

elif choice == "4":
    # Number of books borrowed by each member.
    cursor.execute("SELECT member_id, COUNT(*) AS num_books FROM
book_lending GROUP BY member_id")
    borrowed_books = cursor.fetchall()
    if borrowed_books:
        print("Number of books borrowed by each member:")
        for borrowed_book in borrowed_books:
            print(f"ID: {borrowed_book[0]}, Number of books:
{borrowed_book[1]}")
    else:
        print("No books have been borrowed.")

elif choice == "5":
    # Most popular books.
    cursor.execute("SELECT book_id, COUNT(*) AS num_borrows FROM
book_lending GROUP BY book_id ORDER BY num_borrows DESC LIMIT 10")
    most_popular_books = cursor.fetchall()
    if most_popular_books:
        print("Most popular books:")
        for book in most_popular_books:
            print(f"ID: {book[0]}, Number of borrows: {book[1]}")
    else:
        print("No books have been borrowed.")

```

```

elif choice == "6":
    # Least popular books.
    cursor.execute("SELECT book_id, COUNT(*) AS num_borrows FROM
book_lending GROUP BY book_id ORDER BY num_borrows ASC LIMIT 10")
    least_popular_books = cursor.fetchall()
    if least_popular_books:
        print("Least popular books:")
        for book in least_popular_books:
            print(f"ID: {book[0]}, Number of borrows: {book[1]}")
    else:
        print("No books have been borrowed.")

elif choice == "7":
    return

else:
    print("Invalid choice.")

# Function to manage books
def manage_books():
    while True:
        print("\nManage Books:")
        print("1. Add Book")
        print("2. Update Book")
        print("3. Delete Book")
        print("4. Show Available Books")
        print("5. Issue / Return Books")
        print("6. More")
        print("7. Return to main menu")
        choice = input("Enter your choice: ")

```

```

if choice == "1":
    title = input("Enter the title of the book: ")
    author = input("Enter the author of the book: ")
    add_book(title, author)
elif choice == "2":
    book_id = input("Enter the ID of the book to update: ")
    title = input("Enter the new title of the book: ")
    author = input("Enter the new author of the book: ")
    cursor.execute("UPDATE books SET title = %s, author = %s
WHERE book_id = %s", (title, author, book_id))
    mydb.commit()
    print("Book updated successfully!")
elif choice == "3":
    book_id = input("Enter the ID of the book to delete: ")
    cursor.execute("DELETE FROM books WHERE book_id = %s",
(book_id,))
    mydb.commit()
    print("Book deleted successfully!")
elif choice == "4":
    show_available_books()
elif choice == "5":
    issue_return_books()
elif choice == "6":
    # Call the features page function
    features_page()
elif choice == "7":
    break
else:
    print("Invalid choice. Please enter a valid option.")

# Function to manage members

# Function to manage members

```

```

def manage_members():
    while True:
        print("\nManage Members:")
        print("1. Add Member")
        print("2. Show Member Details")
        print("3. Return to main menu")
        choice = input("Enter your choice: ")

        if choice == "1":
            # Display a table of non-staff members
            cursor.execute("SELECT member_id, name, email FROM members
WHERE member_id NOT IN (SELECT id FROM users WHERE is_staff = True)")
            non_staff_members = cursor.fetchall()

            if non_staff_members:
                print(" ")
                print("Non-Staff Members:")
                print(tabulate(non_staff_members, headers=['Member ID',
'Name', 'Email']))
            else:
                print("No non-staff members found.")

            # Prompt for new member information
            name = input("Enter the name of the member: ")
            email = input("Enter the email of the member: ")
            password = input("Enter the password for the member: ")

            # Add the new member to the users table
            create_user(name, password, is_staff=False)

            # Add the new member to the members table
            add_member(name, email)

        elif choice == "2":

```

```

        show_member_details()

elif choice == "3":
    break

else:
    print("Invalid choice. Please enter a valid option.")

# Add a function to call functions from features01.py
def features_page():
    while True:
        print("\nFeatures Page:")
        print("1. Reserve a Book")
        print("2. Renew a Book")
        print("3. Calculate Fines")
        print("4. Show Book Details")
        print("5. Return to main menu")
        choice = input("Enter your choice: ")

        if choice == "1":
            reserve_book()
        elif choice == "2":
            renew_book()
        elif choice == "3":
            calculate_fines()
        elif choice == "4":
            show_book_details()
        elif choice == "5":
            break
        else:
            print("Invalid choice. Please enter a valid option.")

```

```

# Function to add a book to the library
def add_book(title, author):
    cursor.execute("INSERT INTO books (title, author, available) VALUES
(%s, %s, %s)", (title, author, True))
    mydb.commit()
    print("Book added successfully!")

# Function to add a member
def add_member(name, email):
    cursor.execute("INSERT INTO members (name, email) VALUES (%s, %s)",
(name, email))
    mydb.commit()
    print("Member added successfully!")

# Function to lend a book to a member
def lend_book(book_id, member_id, return_date):
    cursor.execute("UPDATE books SET available = false WHERE book_id =
%s", (book_id,))
    cursor.execute("INSERT INTO book_lending (book_id, member_id,
return_date) VALUES (%s, %s, %s)", (book_id, member_id, return_date))
    mydb.commit()
    print("Book lent successfully!")

# Function to return a book
def return_book(member_id, book_id):
    # Check if the book is actually lent to the given member
    cursor.execute("SELECT * FROM book_lending WHERE member_id = %s AND
book_id = %s", (member_id, book_id))
    lending_info = cursor.fetchone()

    if lending_info:
        # Update the book availability to True
        cursor.execute("UPDATE books SET available = True WHERE book_id =
%s", (book_id,))

```



```

        # Delete the lending information

        cursor.execute("DELETE FROM book_lending WHERE member_id = %s AND
book_id = %s", (member_id, book_id))

        mydb.commit()

        print("Book returned successfully!")

    else:

        print("The book with ID {} is not currently lent to the member
with ID {}".format(book_id, member_id))

# Function to show available books
def show_available_books():

    cursor.execute("SELECT book_id, title, author FROM books WHERE
available = True")

    available_books = cursor.fetchall()

    if available_books:

        print("Available Books:")

        for book in available_books:

            print(f"ID: {book[0]}, Title: {book[1]}, Author: {book[2]}")

    else:

        print("No available books.")

# Function to show members
def show_members():

    cursor.execute("SELECT id, name, email FROM members")

    members = cursor.fetchall()

    if members:

        print("Members:")

        for member in members:

            print(f"ID: {member[0]}, Name: {member[1]}, Email:
{member[2]}")

    else:

        print("No members.")

```

```

# Function to issue/return a book
def issue_return_books():
    while True:
        print("\nIssue/Return Books:")
        print("1. Issue Book")
        print("2. Return Book")
        print("3. Return to main menu")
        choice = input("Enter your choice: ")

        if choice == "1":
            book_id = input("Enter the ID of the book to lend: ")
            member_id = input("Enter the ID of the member: ")
            return_date = input("Enter the return date (YYYY-MM-DD): ")
            lend_book(book_id, member_id, return_date)
        elif choice == "2":
            book_id = input("Enter the ID of the book to return: ")
            member_id = input("Enter the ID of the member from whom to
return book from: ")
            return_book(member_id, book_id)
        elif choice == "3":
            break
        else:
            print("Invalid choice. Please enter a valid option.")

# Return books function for the user side of the menu to return books
def return_books():
    while True:
        print("\nReturn Books:")
        print("1. Return Book")
        print("2. Return to main menu")
        choice = input("Enter your choice: ")

```

```

    if choice == "1":
        book_id = input("Enter the ID of the book to return: ")
        return_book(book_id)
    elif choice == "2":
        break
    else:
        print("Invalid choice. Please enter a valid option.")

# Creating necessary tables
create_tables()

# main loop menu6
while True:
    if (is_staff == 1):
        print("\nMain Menu:")
        print("1. Manage Books")
        print("2. Manage Members")
        print("3. Reports")
        print("4. Exit")
        option = input("Enter your choice: ")

        if option == "1":
            manage_books()
        elif option == "2":
            manage_members()
        elif option == "3":
            reports()
        elif option == "4":
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please enter a valid option.")

```

```

else :

    print("\nMain Menu:")
    print("1. Show Available Books")
    print("2. Return Book(s)")
    print("3. Exit")
    option = input("Enter your choice: ")

    if option == "1":
        show_available_books()
    elif option == "2":
        return_books()
    elif option == "3":
        print("Exiting the program.")
        break
    else:
        print("Invalid choice.")

# Close the database connection
mydb.close()

```

## 2. login\_signup.py

```

# login_signup.py

import os
import mysql.connector as sqltor
from tabulate import tabulate

```

```

def create_database():
    db = sqltor.connect(
        host="localhost",
        user="root",
        passwd="1234",
    )
    cursor = db.cursor()
    cursor.execute("CREATE DATABASE IF NOT EXISTS library_db")
    db.database = "library_db"
    cursor.execute("CREATE TABLE IF NOT EXISTS users (id INT
AUTO_INCREMENT PRIMARY KEY, username VARCHAR(255), password VARCHAR(255),
is_staff BOOLEAN)")
    db.close()

def signup():
    is_staff = True # Only staff members can sign up
    username = input("Enter a username: ")
    password = input("Enter a password: ")

    root_pass = input("Enter the root password for staff creation: ")
    if root_pass == "q94fz7c2ir":
        create_user(username, password, is_staff)
        print("Staff user created successfully.")
    else:
        print("Invalid root password. Staff user not created.")

def create_user(username, password, is_staff):
    db = sqltor.connect(
        host="localhost",
        user="root",
        passwd="1234",
        database="library_db"
    )

```

```

        cursor = db.cursor()

        cursor.execute("INSERT INTO users (username, password, is_staff)
VALUES (%s, %s, %s)", (username, password, is_staff))

        db.commit()

        db.close()

def login():

    username = input("Enter your username: ")
    password = input("Enter your password: ")

    db = sqltor.connect(
        host="localhost",
        user="root",
        passwd="1234",
        database="library_db"
    )

    cursor = db.cursor()

    cursor.execute("SELECT is_staff FROM users WHERE username = %s AND
password = %s", (username, password))

    user_data = cursor.fetchone()

    if user_data:
        is_staff = user_data[0]
        print("Login successful!")
        return username, is_staff
    else:
        print("Invalid username or password.")
        return None, False

def create_user(username, password, is_staff):

    db = sqltor.connect(
        host="localhost",
        user="root",

```

```

        passwd="1234",
        database="library_db"
    )
    cursor = db.cursor()
    cursor.execute("INSERT INTO users (username, password, is_staff)
VALUES (%s, %s, %s)", (username, password, is_staff))
    db.commit()
    db.close()

def login_signup_page():
    create_database()

    while True:
        print("\nLogin/Signup Page:")
        print("1. Login")
        print("2. Signup (only for staff members)")
        print("3. Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            username, is_staff = login()
            if username:
                return username, is_staff
        elif choice == "2":
            signup()
        elif choice == "3":
            print("Exiting the program.")
            exit()
        else:
            print("Invalid choice. Please enter a valid option.")

# Automatically run the login/signup page
if __name__ == "__main__":

```

```
login_signup_page()
```

### 3.features01.py

```
#features01.py
```

```
import mysql.connector
```

```
# Database connection
```

```
db = mysql.connector.connect(  
    host="localhost",  
    user="root",  
    passwd="1234",  
)
```

```
cursor = db.cursor()
```

```
# Reserve a book
```

```
def reserve_book():  
    book_id = input("Enter the book ID to reserve: ")  
    member_id = input("Enter your member ID: ")  
    cursor.execute("USE library_db")  
    # Check if the book is available  
    cursor.execute("SELECT available FROM books WHERE book_id = %s",  
(book_id,))  
    status = cursor.fetchone()  
  
    if not status:  
        print("Book not found.")  
        return  
  
    if status[0] == "available":
```



```

        cursor.execute("UPDATE books SET status = 'reserved' WHERE
book_id = %s", (book_id,))

        cursor.execute("INSERT INTO reservations (book_id, member_id)
VALUES (%s, %s)", (book_id, member_id))

        db.commit()

        print("Book reserved successfully.")
    else:
        print("Book is not available for reservation.")

# Renew a book
def renew_book():
    book_id = input("Enter the book ID to renew: ")

    # Select the library database
    cursor.execute("USE library_db")

    # Check if the book is borrowed by the member
    cursor.execute("SELECT status, member_id FROM books WHERE book_id =
%s", (book_id,))
    book_data = cursor.fetchone()

    if not book_data:
        print("Book not found.")
        return

    if book_data[0] == "borrowed":
        member_id = input("Enter your member ID: ")
        if member_id == book_data[1]:
            cursor.execute("UPDATE books SET due_date =
DATE_ADD(due_date, INTERVAL 14 DAY) WHERE book_id = %s", (book_id,))
            db.commit()
            print("Book renewed successfully.")
        else:
            print("You are not the borrower of this book.")

```

```

else:
    print("Book is not borrowed.")

# Calculate fines for overdue books
def calculate_fines():
    member_id = input("Enter your member ID: ")
    cursor.execute("USE library_db")
    cursor.execute("SELECT SUM(DATEDIFF(NOW(), due_date)) * 0.50 FROM
books WHERE member_id = %s AND available = 0", (member_id,))
    total_fine = cursor.fetchone()[0]

    if total_fine:
        print(f"Total fine: ${total_fine:.2f}")
    else:
        print("No fines to calculate.")

# Show book details
def show_book_details():
    cursor.execute("USE library_db")

    book_id = input("Enter the book ID: ")

    cursor.execute("SELECT * FROM books WHERE book_id = %s", (book_id,))
    book_details = cursor.fetchone()

    if not book_details:
        print("Book not found.")
    else:
        print("Book Details:")
        print("Book ID:", book_details[0])
        print("Title:", book_details[1])
        print("Author:", book_details[2])
        print("Status:", book_details[3])

```

```

# Show member details
def show_member_details():
    member_id = input("Enter the member ID: ")
    cursor.execute("USE library_db")
    cursor.execute("SELECT * FROM members WHERE member_id = %s",
(member_id,))
    member_details = cursor.fetchone()

    if not member_details:
        print("Member not found.")
    else:
        print("Member Details:")
        print("Member ID:", member_details[0])
        print("Name:", member_details[1])
        print("Email:", member_details[2])

def manage_books():
    while True:
        print("Manage Books:")
        print("1. Reserve a Book")
        print("2. Renew a Book")
        print("3. Calculate Fines")
        print("4. Show Book Details")
        print("5. Back to main menu")
        choice = input("Enter your choice: ")

        if choice == "1":
            reserve_book()
        elif choice == "2":
            renew_book()
        elif choice == "3":
            calculate_fines()

```

```

        elif choice == "4":
            show_book_details()
        elif choice == "5":
            break
        else:
            print("Invalid choice. Please enter a valid option.")

def issue_book():
    book_id = input("Enter the ID of the book to issue: ")
    member_id = input("Enter the ID of the member to issue the book to: ")
    return_date = input("Enter the return date of the book: ")

    # Check if the book is available
    cursor.execute("SELECT available FROM books WHERE id = %s",
(book_id,))
    available = cursor.fetchone()[0]
    if not available:
        print("The book is not available.")
        return

    # Check if the member is a valid member
    cursor.execute("SELECT name FROM members WHERE id = %s",
(member_id,))
    name = cursor.fetchone()
    if name is None:
        print("The member is not a valid member.")
        return

    # Lend the book to the member
    cursor.execute("UPDATE books SET available = 0 WHERE id = %s",
(book_id,))

    cursor.execute("INSERT INTO book_issues (book_id, member_id,
return_date) VALUES (%s, %s, %s)", (book_id, member_id, return_date))

```

```
db.commit()  
print("Book issued successfully!")
```

# Conclusion

13

In conclusion, the development of the Library Management System has been a fulfilling journey that has resulted in a comprehensive solution to streamline library operations. This project has successfully achieved its objectives and made a significant contribution to modernizing the way libraries manage their resources and serve their users.

Throughout the development process, I gained valuable insights into software design, database management, and user interface development. The project allowed me to apply theoretical knowledge learned in the classroom to a practical real-world scenario. I learned the importance of efficient data organization, user-friendly interfaces, and robust error handling to create a seamless and user-centric system.

The Library Management System's impact is undeniable. It has revolutionized the way the library manages its books and members. The automated book tracking, member registration, and lending operations have significantly reduced manual effort and human errors. The system's intuitive interface ensures that even non-technical users can effortlessly navigate and utilize its features.

As for the future, there are several avenues for enhancement. The integration of an online catalog and reservation system could provide users with more convenience. Incorporating data analytics to generate insights into popular books and member preferences could help optimize resource allocation. Furthermore, expanding the system to support e-books and digital resources could cater to the changing needs of the modern library landscape.

In conclusion, this project has been a remarkable learning experience that has equipped me with valuable technical and problem-solving skills. The Library Management System stands as a testament to the power of technology in transforming traditional practices. With a strong foundation laid by this project, I look forward to exploring further advancements in software development and contributing to innovative solutions in the future.

# Bibliography

14

During the development of this project, I consulted various resources that greatly aided me in understanding and implementing the Library Management System. These resources include:

- "Computer Science with Python" by Sumita Arora, Class 12 CBSE Curriculum.
- "Computer Science for Class 12" by NCERT.
- "Python Programming for Beginners" by Mary Brown.
- W3Schools: An online resource for web development and programming tutorials (<https://www.w3schools.com/>).
- GeeksforGeeks: A website providing programming solutions and tutorials (<https://www.geeksforgeeks.org/>).
- Stack Overflow: An online community for programmers to seek and share solutions (<https://stackoverflow.com/>).
- GitHub: A platform for collaborative coding and version control (<https://github.com/>).

These resources, along with the textbooks prescribed by the CBSE board, played a crucial role in shaping the project's concepts, design, and implementation.