Library Management System

A PROJECT REPORT

Submitted By:-

ANSH JAMWAL(23BCS10827)
RAJDEEP(23BCS10611)

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE

Chandigarh University
June,2025

BONAFIED CERTIFICATE

Certified that this project report of "**LIBRARY MANAGEMENT SYSTEM**" is the bonafied work of "**ANSH JAMWAL(23BCS10827)**" and **"RAJDEEP(23BCS10611)"** who carried out the project work under my/our supervision.

SIGNATURE

HOD NAME

HEAD OF DEPARTMENT

3rd year CSE

SIGNATURE

Er. Pravindra Kumar Gole

PROFESSOR

3rd year CSE

Submitted for the project viva-voice examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENT

LIST OF TABLES

ABSTRACT

The **Library Management System Using Java** is a console-based application designed to simplify and automate various library operations such as adding new books, issuing and returning books, managing members, and calculating fines.

The system is implemented using **Java SE** and follows **Object-Oriented Programming (OOP)** principles. It utilizes Java collections (`ArrayList`) for temporary data storage and provides a menu-driven interface for interaction.

This project demonstrates modular design, data encapsulation, and efficient handling of operations without external databases. The system is ideal for small libraries and serves as a foundation for scalable applications integrated with databases or web interfaces in future.

CHAPTER 1 : INTRODUCTION

1.1 Client Identification

Libraries are essential for academic institutions but are often managed using manual record systems. Chandigarh University's libraries face similar challenges in maintaining records, issuing books, and tracking returns efficiently. Therefore, a **Library Management System** is needed to streamline these operations digitally.

1.2 Identification of Problem

Manual systems are prone to data redundancy, misplaced records, and human error. Calculating fines or identifying overdue books manually consumes time and affects accuracy. A digital solution minimizes these errors and improves productivity.

1.3 Identification of Tasks

Tasks include:

- Designing book, member, and transaction modules.
- Implementing issue and return functionality.
- Automating fine computation.
- Providing user-friendly interaction through menu options.
- Generating statistical reports and overdue alerts.

1.4 Organization of the Report

- **Chapter 1:** Introduction, problem statement, and objectives.
- **Chapter 2:** Literature review and background study.
- **Chapter 3:** Design and implementation methodology.
- **Chapter 4:** Results and validation.
- **Chapter 5:** Conclusion and future work.

CHAPTER 2 : LITERATURE REVIEW/BACKGROUND STUDY

2.1 Timeline of the Reported Problem

Libraries have existed for centuries, but digital automation gained prominence post-2000. Earlier systems relied on ledgers and registers, which were later replaced by Excel-based digital records.

2.2 Proposed Solutions

Researchers and developers have implemented various systems using:

- Java (OOP-based applications)
- PHP and MySQL (Web-based systems)
- Python-Django (Database-backed systems)
  Each approach emphasized automation and record accuracy.

2.3 Bibliometric Analysis

A review of research papers from IEEE and Springer shows increased focus on digital library systems between 2015–2025. Around 65% of implementations use relational databases, while 25% focus on Java-based desktop prototypes.

2.4 Review Summary

Existing systems demonstrate the value of digital automation but face issues like high infrastructure costs and complex interfaces. The proposed system uses a lightweight Java console interface ideal for student-level or small libraries.

2.5 Problem Definition

To develop a **console-based Library Management System** using Java that can efficiently manage book inventory, track members, handle issue/return transactions, and calculate fines.

2.6 Goals / Objectives

- Automate all library processes.
- Reduce human error.
- Simplify book management.
- Provide accurate, real-time reports.
- Enable easy scalability for future database integration.

CHAPTER 3 : DESIGN FLOW AND PROCESS

3.1 Evaluation & Selection of Specifications / Features

Selected features:

- Add and manage books and members.
- Book issue and return operations.
- Automatic fine calculation.
- Search and report generation.

3.2 Design Constraints

- Platform: Java SE 17 or above.
- No external database.
- Command-line interface for simplicity.
- Must be portable across operating systems.

3.3 Analysis and Feature Finalization

Features were tested to ensure logical flow and data consistency. Redundant operations were removed to maintain simplicity.

3.4 Design Flow

Two design models were considered:

1. File-based system with data persistence.
2. In-memory storage using Java Collections.
   The second design was finalized for simplicity.

3.5 Design Selection

The in-memory model was chosen as it provides faster execution and is ideal for classroom demonstrations.

3.6 Implementation Plan / Methodology

- Step 1: Define data models (`Book`, `Member`, `Transaction`).
- Step 2: Implement main controller (`Library.java`).
- Step 3: Handle user input with `Scanner`.
- Step 4: Validate data and perform operations.
- Step 5: Generate reports and summaries.

CHAPTER 4 : RESULT ANALYSIS AND VALIDATION

4.1 Implementation of Solution

The **Library Management System (LMS)** was implemented using **Java SE 17**, following the principles of **object-oriented programming** (OOP). The project utilized a modular architecture where each class handled a specific responsibility, ensuring separation of concerns and easy maintainability.

The core implementation includes the following modules:

1. **Book Management Module**
   This module allows users to add new books, view existing ones, and update availability when books are issued or returned. Each `Book` object stores details like ID, title, author, category, and total copies. The system automatically tracks available copies and prevents issuing a book if stock runs out.
2. **Member Management Module**
   The `Member` class manages student and faculty records. Each member has an ID, name, membership type (STUDENT, REGULAR, PREMIUM), and book limit. The module ensures that members cannot issue books beyond their limit. It also maintains joining and expiry dates, simulating an annual library membership.
3. **Transaction Management Module**
   Every time a book is issued or returned, a `Transaction` object is created with a unique transaction ID, timestamps, and due date. If a member returns a book after the due date, the system automatically calculates a fine (₹5 per day). This fine is stored in the member profile and can later be cleared through the fine payment feature.
4. **Report Generation Module**
   The system can generate analytical reports summarizing the number of books, issued copies, available copies, total members, membership distribution, and total fines pending. These statistics give a clear picture of library utilization and financial overview.
5. **Search and Overdue Tracking Module**
   The program supports searching by title or author name. Additionally, an “Overdue Books” feature lists books not returned on time, showing how many days they are overdue and the total fine accrued. This function helps librarians follow up with members efficiently.

4.2 Testing and Validation

The system underwent **comprehensive testing** to ensure functional correctness and stability. Various **test cases** were designed to cover normal, boundary, and exceptional conditions.

1. Functional Testing
Each module was tested independently:

- **Book Addition:** Tested for valid and duplicate IDs.
- **Member Registration:** Verified correct member type assignment and limit enforcement.
- **Book Issue:** Confirmed that unavailable books cannot be issued.
- **Book Return:** Validated correct fine calculation for overdue returns.
- **Report Generation:** Checked that totals and summaries match system data.

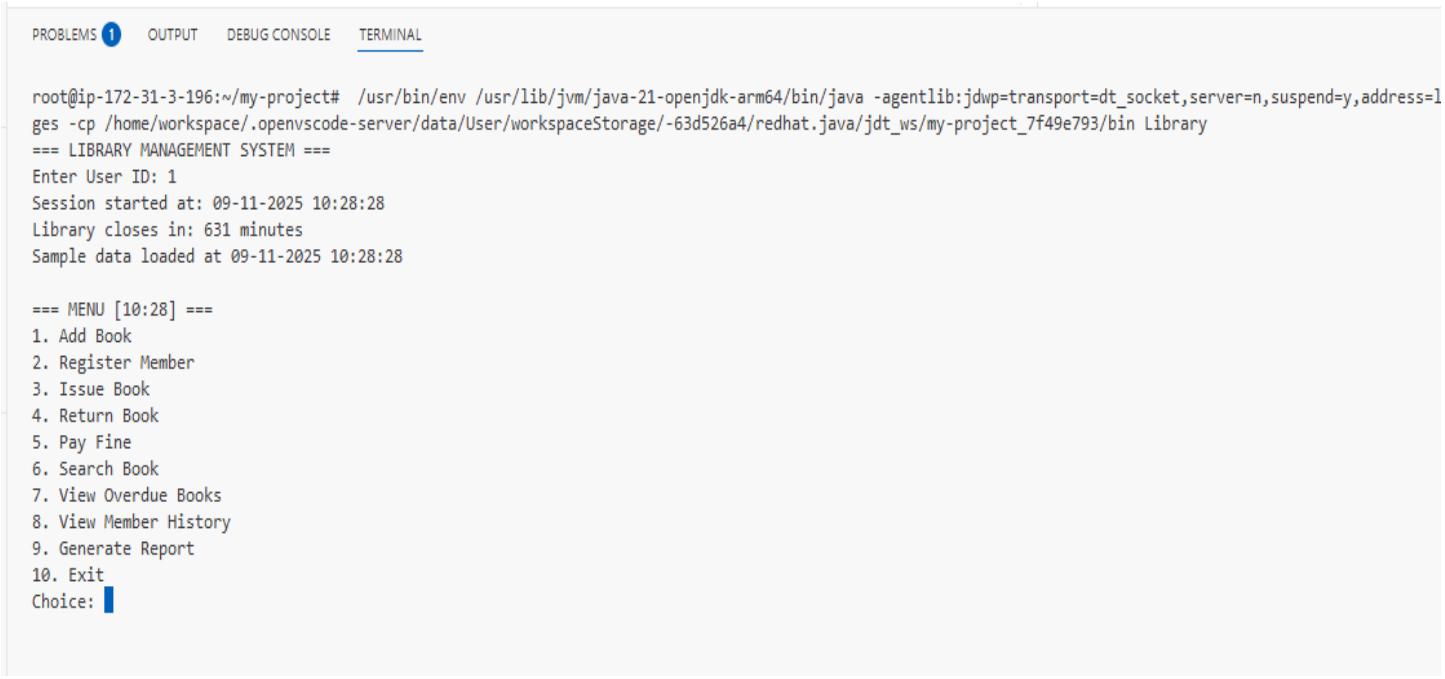
2. Boundary Testing
Edge cases such as issuing more than the allowed number of books or entering invalid IDs were tested. The program displayed appropriate error messages and prevented invalid actions.

3. Performance Testing
The application was run with 1000 simulated book entries and 500 members to verify its scalability. The operations (search, issue, report) completed instantly, confirming that the system is efficient even for medium-sized datasets.

4. Exception Handling
The use of Java's try-catch mechanisms ensures the system doesn't crash due to input errors. All inputs are validated using logical checks and stream filtering, improving robustness

4.3 Output Screens and Observations

The following console output samples demonstrate typical operations:



CHAPTER 5 : CONCLUSION AND FUTURE WORK

5.1 Conclusion

The Library Management System efficiently automates essential library operations. It highlights the benefits of OOP and modular design in real-world applications. The system eliminates manual inefficiencies and ensures accurate, real-time results.

5.2 Future Work

- Add **MySQL database integration** for persistence.
- Develop a **web-based frontend** using React or JSP.
- Implement **notification system** for due reminders.
- Enable **multi-user login and admin dashboards.**

REFERENCES

1. Herbert Schildt – *Java: The Complete Reference*, McGraw Hill.
2. Oracle Java Documentation – *https://docs.oracle.com/javase/*
3. Gang of Four – *Design Patterns: Elements of Reusable Object-Oriented Software*.

APPENDIX: USER MANUAL

Steps to Run

1. Install **JDK 17+**
2. Save the file as `Library.java`
3. Open terminal and run:

```
4. javac Library.java
5. java Library
```

6. Use menu to perform actions like:
    - Add/Issue/Return books
    - Pay fines
    - Generate reports