VIT®
BHOPAL
www.vitbhopal.ac.in

PROJECT TITLE: -

Library Management System (LMS)

Course Code: CSE2006

Submitted By: Krish Mathur

Registration Number: 24BCE10068

Submitted To: Prof. Baseera A.

Date: 24-11-2025

Introduction

Libraries play a vital role in academic institutions by providing learning resources to students, faculty members and researchers. However, most university libraries still depend on manual record keeping for book and member management. Manual methods lead to slow processing, data inaccuracy, and difficulty in tracking the complete lending lifecycle of books.

The Library Management System (LMS) is developed to overcome these limitations by automating key library functions using Java. The project replaces handwritten registers with a structured, computer-based system that is reliable, scalable and easy to use. Through features such as book cataloguing, member registration, issue/return operations and transaction monitoring, the system ensures accurate record keeping and efficient book circulation.

This project also demonstrates the application of important Java programming concepts in a real-world scenario, such as object-oriented design, modular development and database interaction.

Problem Statement

University libraries handle thousands of books and multiple borrowing transactions daily. When these operations are performed manually, the following problems commonly arise:

- Difficulty in searching and locating available books quickly
- Errors in updating stock levels after issuing or returning books
- Missing or inconsistent transaction records
- Delays in calculating return dates and overdue fines
- No centralized history of book circulation

Because of these limitations, librarians require an automated solution that can maintain real-time and accurate data. The proposed system solves these challenges by providing digital record management for books, members and issuance details, ensuring efficiency and transparency in library workflows.

Functional Requirements

The system implements three major functional modules:

1. Book Inventory Management (Admin Module):
    - Allows the administrator to Add new books with details (Title, Author, Quantity).
    - Provides functionality to Update stock levels and delete obsolete records.
    - Enables Searching for books by ID or Title.
2. Circulation & Transaction Module:
    - Borrow Book: Validates stock availability and user eligibility before issuing a book. Automatically decrements stock in the database.
    - Return Book: Accepts returns and automatically increments the stock.
3. Automated Reporting & Alerts:
    - View All Books: Generates a dynamic list of all books and their current availability status.
    - Overdue Check: A background process (daemon thread) runs periodically to identify and log overdue books.

Non-Functional Requirements

1. Data Integrity: The system uses ACID properties via MySQL transactions to ensure that stock counts remain accurate even if multiple users access the system simultaneously.

2. Error Handling: Robust exception handling (using try-catch blocks) prevents the application from crashing during invalid inputs or database connection failures.

3. Maintainability: The code follows a modular structure (Model-DAO-Service pattern), making it easy to debug and upgrade.

4. Performance: Database queries are optimized using Prepared Statement to ensure fast retrieval of book records.

System Architecture

The system follows a Layered Architecture:

1. Presentation Layer: The Console UI (LibraryApp.java) where the user interacts with the menu.

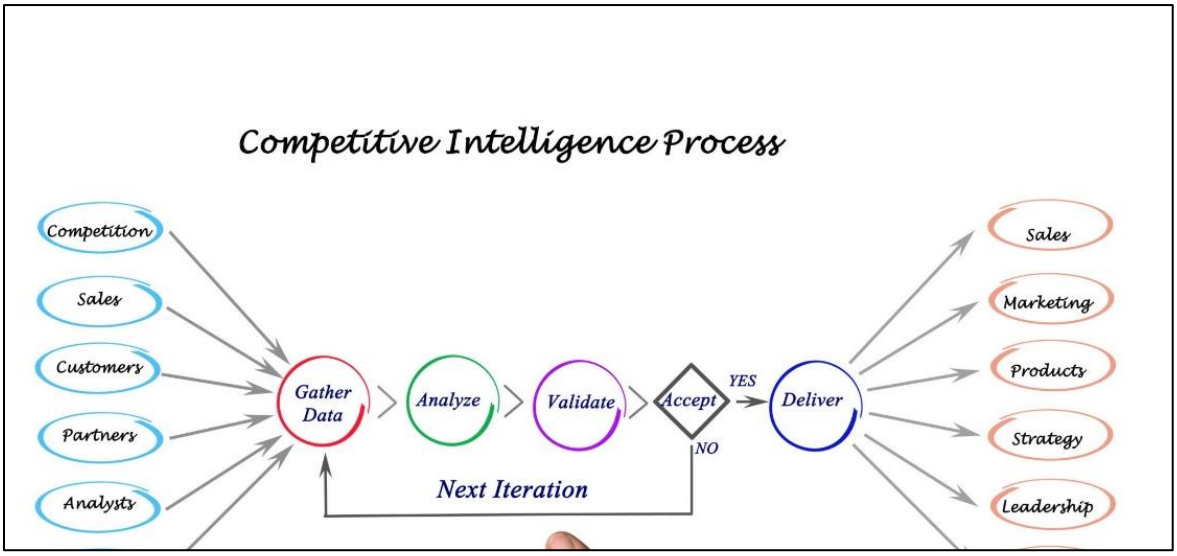
2. Business Logic Layer: The Service/DAO layer (BookDAO.java) that handles rules like "User cannot borrow if stock is 0".

3. Data Persistence Layer: The Database (MySQL) connected via JDBC Driver.

Flow: User Input → Java Application → JDBC Driver → MySQL Database.

Design Diagrams

- 7.1 Use Case Diagram:



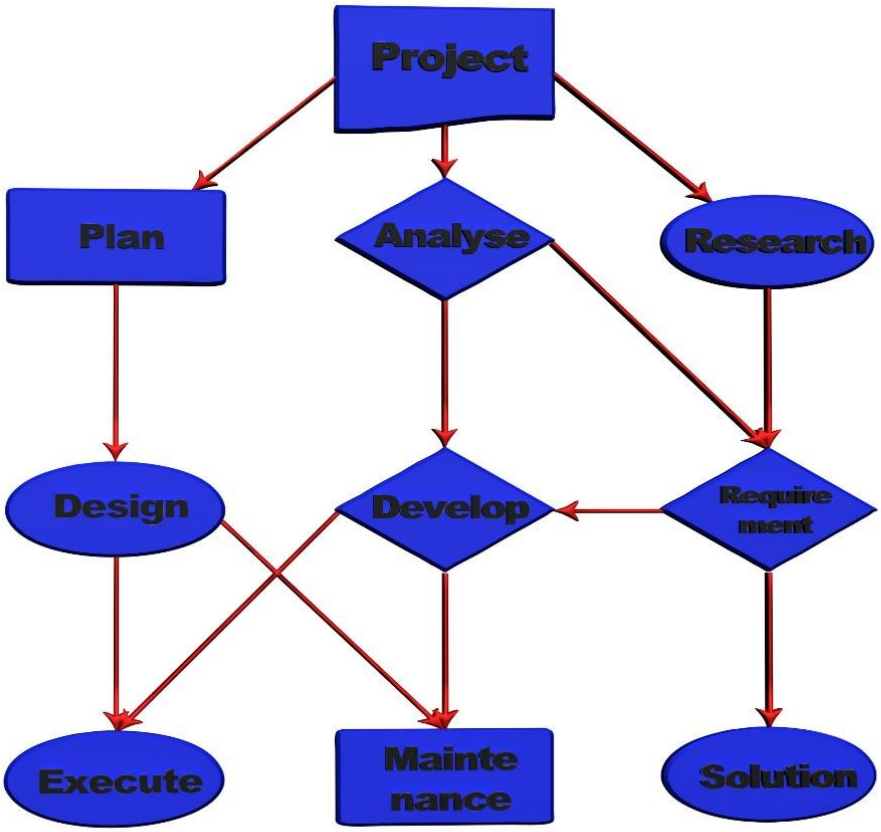
- 7.2 Class Diagram:

. Class Book: Attributes (id, title, author, qty). Methods (getDetails()).

. Class BookDAO: Methods (addBook(), borrowBook(), getAllBooks()).

. Class DBConnection: Method (getConnection()).

- 7.3 ER Diagram (Database Design):



Design Decisions & Rationale

- Why Java? Java was chosen for its platform independence and robust Exception Handling capabilities, which are essential for a stable business application.

- Why JDBC? Java Database Connectivity (JDBC) provides a standard API for connecting to relational databases, allowing direct execution of SQL queries from Java code.

- Why MySQL? It is an open-source, reliable relational database that handles structured data efficiently.

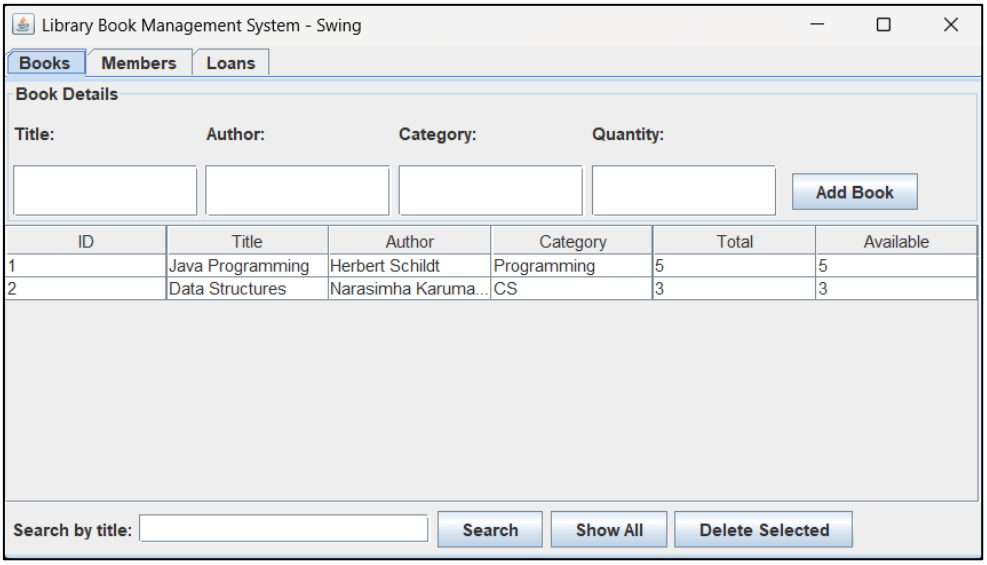
- Why Multithreading? A background thread was implemented to simulate real-time monitoring of overdue books without freezing the main user interface.

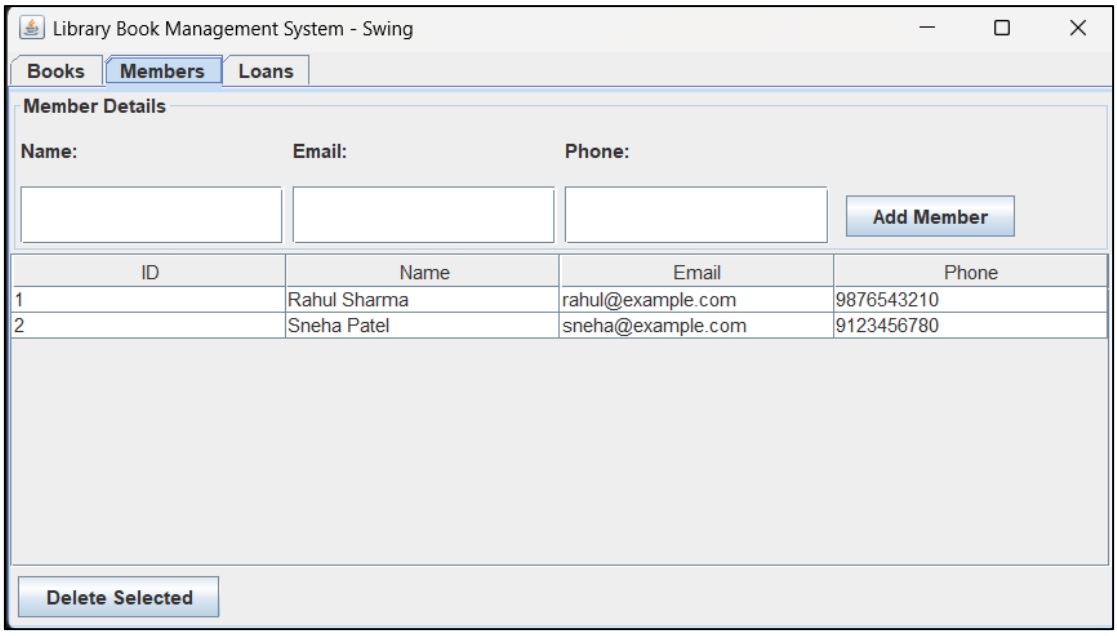
Implementation Details

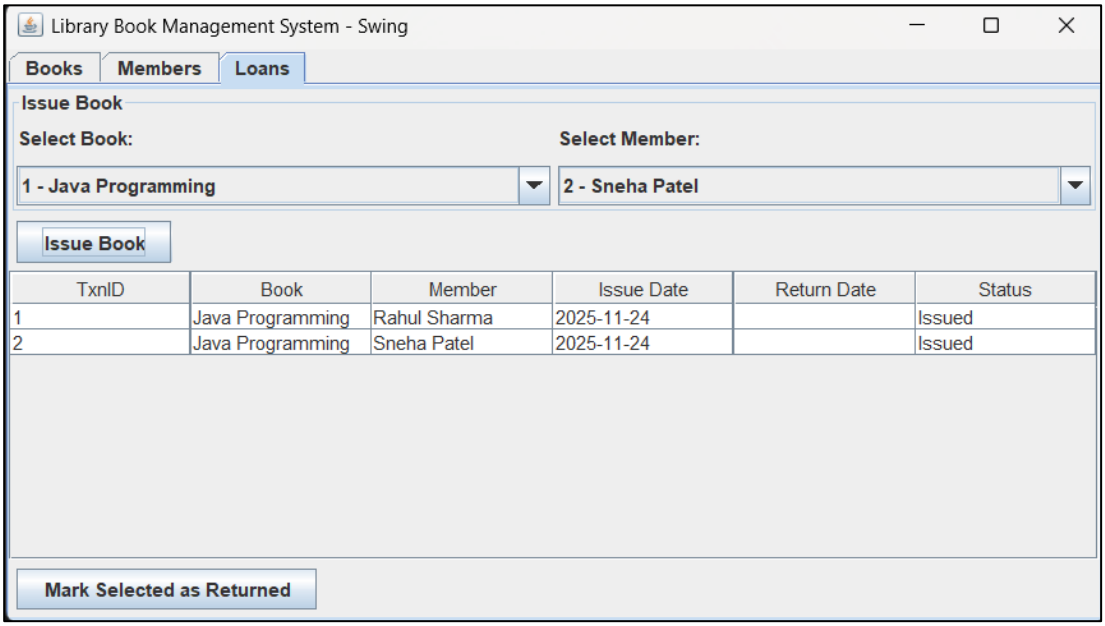
The project is implemented using Java 17 and MySQL 8.0. Key concepts applied include:

- Encapsulation: The Book class uses private variables with public getters/setters to protect data.
- Collections Framework: An ArrayList<Book> is used to store and manipulate search results temporarily in RAM before display.
- Exception Handling: Custom exceptions (e.g., BookNotFoundException) and standard SQL exceptions are handled to ensure a smooth user experience.
- Multithreading: The ReminderTask class extends Thread to run background checks.

SCREENSHOT


Library Book Management System - Swing
Books
Members
Loans
Book Details
Title:
Author:
Category:
Quantity:
Add Book
ID
Title
Author
Category
Total
Available
1
Java Programming
Herbert Schildt
Programming
5
5
2
Data Structures
Narasimha Karuma...
CS
3
3
Search by title:
Search
Show All
Delete Selected


Library Book Management System - Swing
Books
Members
Loans
Member Details
Name:
Email:
Phone:
Add Member
ID
Name
Email
Phone
1
Rahul Sharma
rahul@example.com
9876543210
2
Sneha Patel
sneha@example.com
9123456780
Delete Selected


Library Book Management System - Swing
Books
Members
Loans
Issue Book
Select Book:
Select Member:
1 - Java Programming
2 - Sneha Patel
Issue Book
TxnID
Book
Member
Issue Date
Return Date
Status
1
Java Programming
Rahul Sharma
2025-11-24
Issued
2
Java Programming
Sneha Patel
2025-11-24
Issued
Mark Selected as Returned

Testing Approach

The system was tested using Black Box Testing techniques:

- Unit Testing: Each method in BookDAO (add, borrow, view) was tested individually with dummy data.
- Validation Testing:
  - Test Case 1: Enter a negative number for quantity. Result: System prompts for valid integer.
  - Test Case 2*:* Try to borrow a non-existent Book ID. Result: System throws BookNotFoundException.
  - Test Case 3: Disconnect Database. *Result:* System catches SQL Exception and prints a user-friendly error.

Challenges Faced

- JDBC Configuration: Initially, faced ClassNotFoundException because the MySQL Connector JAR was not correctly added to the classpath. This was resolved by configuring the library dependencies in IntelliJ.

- Database Synchronization: ensuring that the "Issued" count did not exceed "Quantity" during rapid transactions. This was solved by adding logic checks before the SQL UPDATE command.

Learnings & Key Takeaways

- Gained practical experience in integrating Java with MySQL using JDBC.
- Understood the importance of checked vs. unchecked exceptions in real-world applications.
- Learned how to use Git for version control to manage project changes.
- Improved understanding of Object-Oriented Design principles like cohesion (separating DAO from Model).

Future Enhancements

- GUI Implementation: Upgrade the console interface to a Graphical User Interface (GUI) using JavaFX or Swing.
- Web Integration: Convert the application into a web app using Servlets and JSP.
- Barcode Scanning: Integrate a barcode scanner API to read ISBNs directly for faster book entry.
- User Authentication: Add a login system for different roles (Student vs. Librarian).

References

- Schildt, H. (2018). *Java: The Complete Reference* (11th ed.). Oracle Press.

- MySQL Documentation: https://dev.mysql.com/doc/

- Oracle Java Documentation: https://docs.oracle.com/en/java/

- Course Syllabus: CSE2006 Programming in Java.