Library Management System Project Report

Name: Pandi. Uma Bharathi

Course: CSE

College: SPMVV, Tirupati

Date: March 5,2025

Table of Contents

1. Introduction

2. Problem Statement

3. System Architecture & Technology

4. Source Code Explanation

5. Features Implemented

6. GitHub Repository Link

7. Output Screenshots

8.Conclusion

9.Future Scope

1.Introduction

The Library Management System is designed to streamline the process of managing books in a library. This project demonstrates the use of Java for developing the application and, optionally, MySQL for database management. The system provides functionalities for adding, viewing, removing, and issuing books, making library operations more efficient.

The Library Management System is a comprehensive software solution designed to simplify and automate the management of library resources. Developed using Java, this application streamlines the processes of adding, viewing, removing, and issuing books, thereby reducing manual errors and increasing operational efficiency. By addressing common challenges in library management, such as inventory tracking and administrative overhead, the system offers a practical and scalable approach for modern libraries. This project report outlines the system architecture, key functionalities, source code explanations, and potential future enhancements, providing a detailed insight into the development and impact of the solution.

3. System Architecture & Technology

**Frontend:** Command-Line Interface using Java

**Backend:** Java application logic

**Version Control:** GitHub

This project is implemented solely using two Java classes: Book.java and Library.java.

* **Book.java** defines the properties and behaviors of a book (such as ID, name, author, and issue status).
* **Library.java** contains the application logic and manages library operations like adding, viewing, removing, and issuing books via a command-line interface.

4. Source Code Explanation: Book.java:

public class Book { // Defines the Book class, which represents a single book entity

int id; // Unique identifier for the book.

String name; // The title of the book.

String author; // The author of the book.

boolean isIssued; // Flag indicating whether the book is currently issued.// Constructor to initialize a new Book object with id, name, and author.

public Book(int id, String name, String author) {

this.id = id; // Assigns the passed id to the book's id.

this.name = name; // Assigns the passed name to the book's name.

this.author = author; // Assigns the passed author to the book's author.

this.isIssued = false; // Initializes the book's issued status as false.

}

// Method to display the book's details in a readable format.

public void displayBook() {

// Prints book details along with issued status.

System.out.println(id + ". " + name + " by " + author + " | Issued: " + (isIssued ? "Yes" : "No"));

}

}

Library.javaThe Library.java file is the main driver of the application. It manages the collection of books and provides a text-based menu for users to interact with the system.Imports and Class Declaration:

import java.util.\*; // Imports Java utility classes for ArrayList and Scanner

public class Library {

static ArrayList<Book> books = new ArrayList<>(); // List to store Book objects

static Scanner sc = new Scanner(System.in); // Scanner for user input

Main Method and Menu Loop:

public static void main(String[] args) {

System.out.println("===== Library Management System =====");

while (true) { // Loop to keep the menu running

System.out.println("\n1. View Books");

System.out.println("2. Add Book");

System.out.println("3. Remove Book");

System.out.println("4. Issue Book");

System.out.println("5. Exit");

System.out.print("Enter choice: ");

int choice = sc.nextInt();

Menu Option Handling with Switch-Case:

switch (choice) {

case 1:

viewBooks();

case 2:

addBook();

break;

case 3:

removeBook();

break;

case 4:

issueBook();

break;

case 5:

System.out.println("Thank You!");

System.exit(0); // Exit the program

default:

System.out.println("Invalid Choice!");

}

}

}

Method: Viewing Books

public static void viewBooks() {

if (books.isEmpty()) {

System.out.println("No Books Available!");

} else {

for (Book b : books) {

b.displayBook(); // Display details of each book

}

}

}

Method: Adding a New Book

public static void addBook() {

System.out.print("Enter Book Name: ");

String name = sc.next(); // Reads the book name

System.out.print("Enter Author Name: ");

String author = sc.next(); // Reads the author's name

books.add(new Book(books.size() + 1, name, author)); // Creates and adds a new Book object

System.out.println("Book Added Successfully!");

}

**Method: Removing a Book**

public static void removeBook() {

System.out.print("Enter Book ID to Remove: ");

int id = sc.nextInt();

books.removeIf(b -> b.id == id); // Removes the book with the specified ID

System.out.println("Book Removed Successfully!");

}

Method: Issuing a Book

public static void issueBook() {

System.out.print("Enter Book ID to Issue: ");

int id = sc.nextInt();

for (Book b : books) {

if (b.id == id) {

if (!b.isIssued) {

b.isIssued = true;

System.out.println("Book Issued Successfully!");

} else {

System.out.println("Book Already Issued!");

}

return; // Exit after processing the correct book

}

}

System.out.println("Invalid Book ID!");

}

}

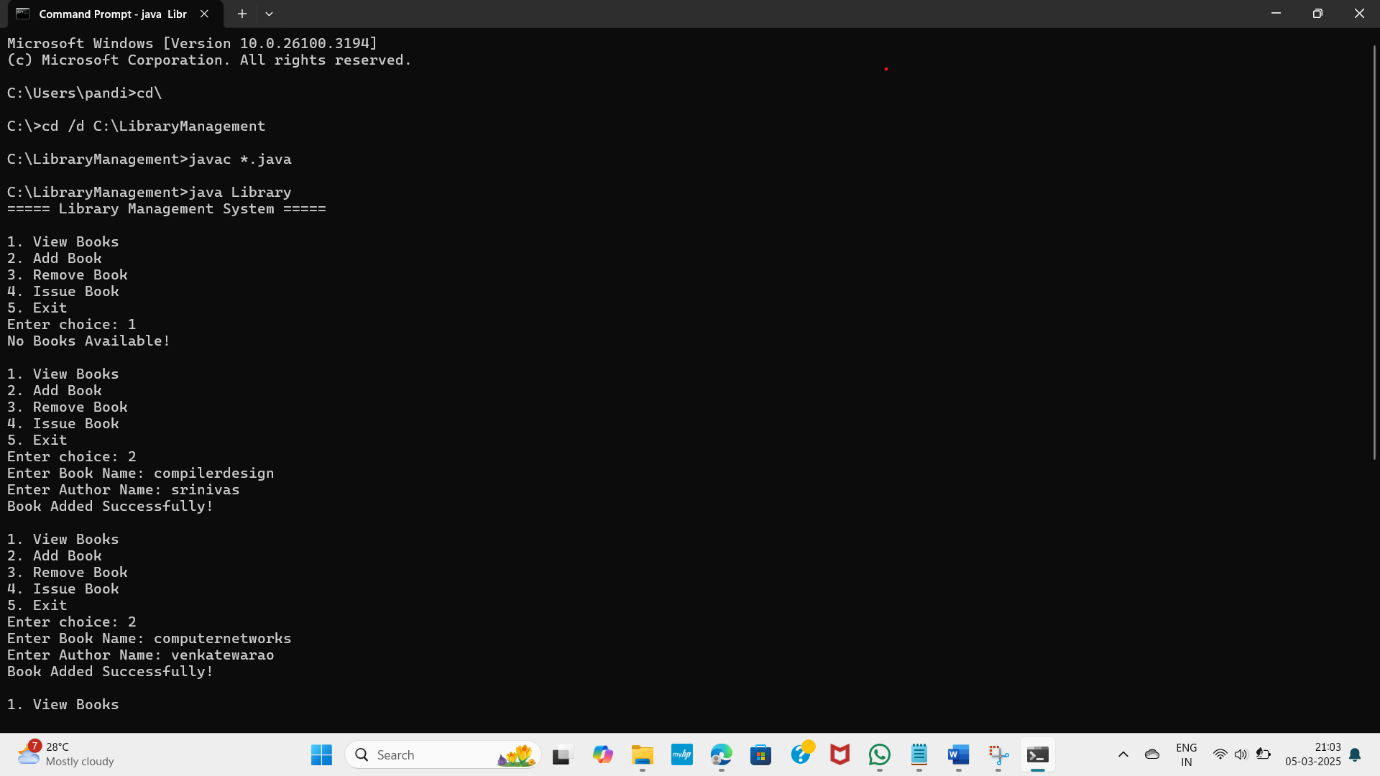
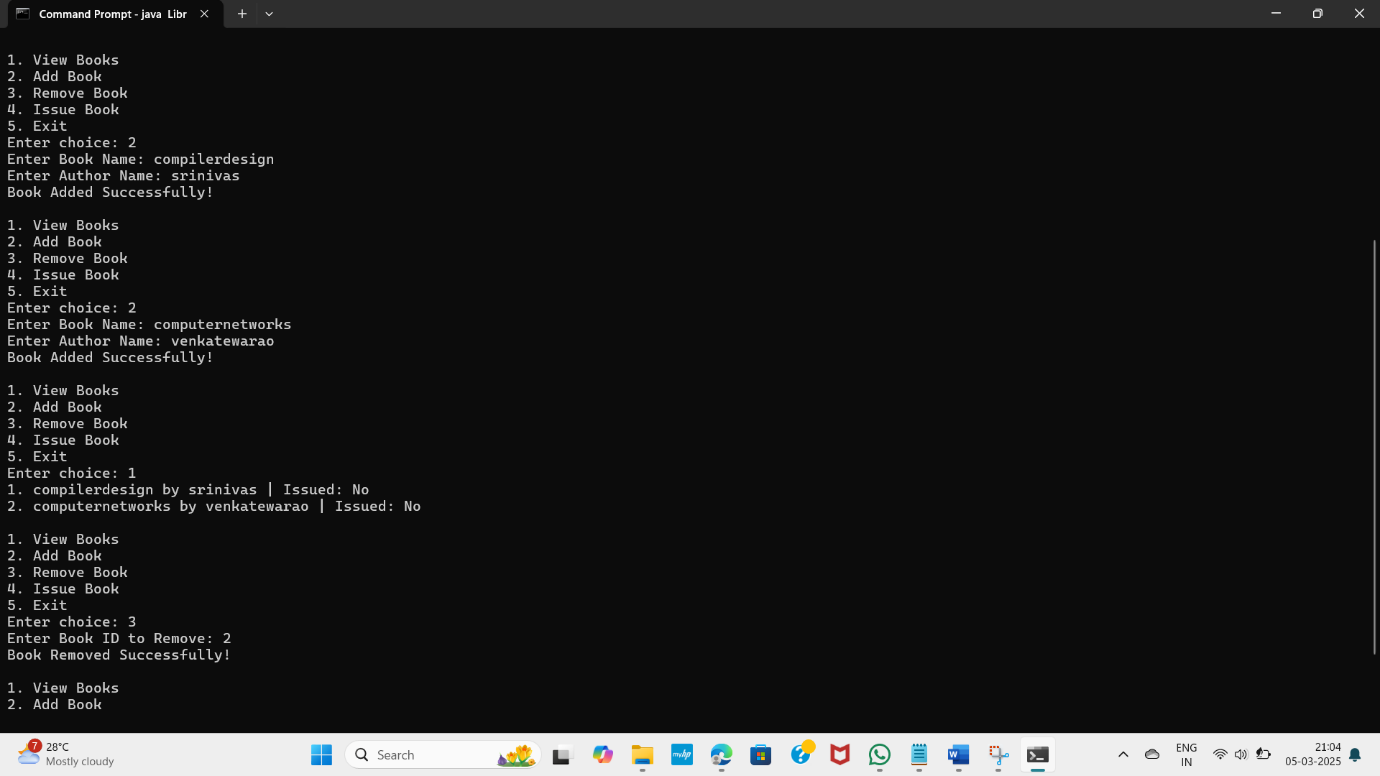
**5. Features Implemented**

* **Add Book:** Users can add new books by entering the book details.
* **View Books:** The application displays a list of all available books.
* **Remove Book:** Books can be removed by specifying their ID.
* **Issue Book:** Books can be marked as issue

**6. GitHub Repository Link**

[Library Management System GitHub Repository](https://github.com/PandiUmaBharathi/LibraryManagement)

**7. Output Screenshots**

* **Figure 1:** Main menu of the application.  
  
* **Figure 2:** Example of adding a new book.  
  

**8. Conclusion**

The Library Management System successfully automates many manual tasks associated with library management. It demonstrates the use of Java programming to create an efficient and interactive system, simplifying the process of managing books.

The Library Management System successfully automates many manual tasks involved in library management, demonstrating the effectiveness of Java programming in creating an interactive and efficient application. By using two main classes—**Book.java** for handling book attributes and **Library.java** for managing library operations—the project provides a solid foundation for further enhancements.

Although this version operates through a command-line interface and maintains data in memory, it meets the initial objectives by allowing users to add, view, remove, and issue books with ease. This streamlined approach not only reduces human errors but also improves the overall efficiency of managing library resources.

In summary, the project has achieved its goal of simplifying library operations and sets the stage for future developments, such as integrating a database for persistent storage, implementing a graphical user interface, and adding user authentication. This work demonstrates both practical coding skills and an understanding of system design, making it a valuable contribution to modern library management solutions.

**9. Future Scope**

* Develop a graphical user interface (GUI) for improved usability.
* Integrate with MySQL for persistent storage of data.
* Implement user authentication and role-based access control.
* Add advanced search and filtering functionalities.
* The Library Management System can be enhanced with additional features and improvements to make it more efficient and user-friendly. Some potential future enhancements include:
* **Graphical User Interface (GUI):** Implementing a GUI using Java Swing or JavaFX to improve usability instead of a command-line interface.
* **Database Integration:** Connecting the system to a MySQL database for persistent data storage, allowing books and records to be saved permanently.
* **User Authentication:** Adding user roles such as librarian and members with login credentials to control access and track issued books.
* **Advanced Search & Filters:** Implementing search functionality based on book name, author, or availability to improve efficiency.
* **Book Return System:** Extending the issue functionality to allow book returns and track due dates with possible fines for late returns.
* **Report Generation:** Generating reports for issued books, available books, and user activity to improve management efficiency.
* These enhancements will make the system more robust, scalable, and suitable for real-world library management.
* Top of Form
* Bottom of Form