

Library Management System

KAN/IT/2022/F/0097

P. A. D. P. PERUMUBULIARACHCHI

Department of Information Technology

Advanced Technological Institute

Kandy

December 2024

Acknowledgements

I would like to express my sincere gratitude to everyone who played a significant role in the success of this project. Their guidance, support, and collaboration were invaluable throughout the entire journey.

I extend my heartfelt thanks to my supervisor Mrs. Dilini Kumarihamy, whose expertise and mentorship were instrumental in steering this project towards success. Your insightful feedback and guidance provided the foundation for the development of my online ordering platform.

A special acknowledgment goes to my friends who provided unwavering support and encouragement. Your collaboration, constructive discussions, and shared enthusiasm made the project not only challenging but also enjoyable. Together, I faced obstacles and celebrated victories, creating a memorable and enriching experience.

I would like to express my appreciation to all for their technical assistance and valuable insights during critical phases of the project. Their expertise and willingness to share knowledge significantly contributed to overcoming challenges.

To all those who directly or indirectly contributed to the success of this project, your involvement did not go unnoticed. Thank you for being a part of this journey and for sharing your expertise, time, and encouragement.

Your collective efforts have shaped the outcome of this project, and I am truly grateful for the collaborative spirit that made it possible.

Sincerely,

P. A. D. P. PERUMUBULIARACHCHI

Table of Contents

	Page
Acknowledgement	ii
Table of Contents	iii
List of Figures / Tables	iv
Chapter 1 - Introduction	1
Introduction	1
Background & Motivation	2
Objectives	2
Scope	2
Critical Functionalities for Project	3
Chapter 2 - Technology Adapted	4
Problem Statement	4
Technology Adapted	4
Implementation in Action	5
Conclusion	6
Chapter 3 - Design	7
Introduction	7
Modules/Sections Overview	7
Chapter 4 - Conclusion & Further Work	13
Achievement	13
Problem Encounter & Limitations	13
Further Work	13
Chapter 5 – User Manual	14
References	17
Appendixes	18

List of Figures / Tables

	Page
Figure 1	7
Figure 2	6
Figure 3	7
Figure 4	8
Figure 5	9
Figure 6	10
Figure 7	11
Figure 8	12
Figure 9	18
Figure 10	18
Figure 11	19
Figure 12	19
Figure 13	20
Figure 14	20
Figure 15	21

Chapter 1 - Introduction

Introduction

The Simple Library Management System (LMS) is a comprehensive desktop application designed to streamline the management of library resources and user information. Developed using Java and the NetBeans Integrated Development Environment (IDE), this project focuses on providing an intuitive graphical user interface (GUI) for efficient interaction. The application leverages the Model-View-Controller (MVC) architecture to separate business logic from user interface design, ensuring scalability and maintainability.

The system caters to two primary user roles: **Admin** and **Member**. Admins have full access to manage library resources, including adding, updating, deleting, searching and viewing both member and book information. Members, on the other hand, are restricted to searching and viewing book details, promoting role-based access control.

This project emphasizes the use of Object-Oriented Programming (OOP) principles, including encapsulation, inheritance, polymorphism, abstraction, method overloading, aggregation, and composition. Furthermore, exception handling is implemented to enhance the robustness of the system by managing invalid user inputs, empty fields, and unauthorized access attempts.

The LMS offers a practical solution for small to medium-sized libraries to organize their operations, providing a user-friendly interface and ensuring data integrity. Through this project, developers gain valuable experience in designing and implementing a functional application while adhering to software engineering best practices.

Background & Motivation

Libraries are essential institutions that provide access to vast collections of knowledge, fostering education and research. Traditional library management systems often rely on manual operations that are time-consuming and prone to errors. A digital solution simplifies these operations, making the process more efficient, user-friendly, and accessible.

A Library Management System (LMS) allows libraries to automate tasks like cataloging books, managing memberships, and tracking borrow/return activities. This project aims to develop such a system using Java and its graphical user interface capabilities, ensuring a modern and interactive experience for users.

The motivation behind this project stems from the need to:

1. Enhance the operational efficiency of libraries by digitizing core processes.
2. Provide a role-based interface tailored for admins and members.
3. Leverage Java's object-oriented capabilities to develop a robust and scalable application.
4. Bridge the gap between technology and education by creating a tool that supports students and researchers in accessing resources.

Objectives

1. To develop a user-friendly library management system.
2. To implement role-based access control (Admin and Member roles).
3. To apply core OOP principles and design patterns.
4. To ensure data consistency and secure user authentication.

Scope

- **User Authentication:** A secure login system for Admin and Member roles.
 - **Admin:** Full access to manage books and members.
 - **Member:** Limited access to search and view books.
- **Member Management:**
 - Add, update, delete, and view member records.
 - Each member is associated with a membership card containing unique details.

- **Book Management:**
 - Add, update, delete, and search for books.
 - Support for detailed book information such as title, author, and publication year.
- **Search Functionality:**
 - Flexible search options for locating books by title, author, or both.
- **Exception Handling:**
 - Handles invalid inputs, empty fields, and unauthorized access.
- **Architecture:**
 - Based on the Model-View-Controller (MVC) pattern for clear separation of concerns.

Critical Functionalities for Project

- 1. User Authentication and Role Management:** To restrict system access and ensure data security.
- 2. Book Management:** To maintain accurate and up-to-date book inventory.
- 3 Member Management:** To manage library members and their activities.

Chapter 2 - Technology Adapted

The development of the Library Management System leveraged a combination of programming languages, frameworks, and architectural patterns to ensure robust functionality, scalability, and user-friendliness. Below is a detailed description of the technologies used:

1. Programming Language: Java

- **Why use java:** Java is a versatile, platform-independent language with robust support for object-oriented programming. It provides excellent libraries and tools for creating scalable and maintainable applications.
- **Use in the Project:**
 - Implementation of core logic for book, member, and transaction management.
 - Development of object-oriented modules such as Book, Member, and MembershipCard.

2. Integrated Development Environment: NetBeans

- **Why use Netbeans:** NetBeans is an open-source IDE with extensive support for Java development. Its intuitive interface and debugging tools simplify the development process.
- **Use in the Project:**
 - Designing the graphical user interface (GUI) using Java Swing.
 - Organizing the project into packages aligned with the MVC architecture.

3. Database: MySQL

- **Why use MySQL:** MySQL is a reliable, widely-used relational database management system that supports efficient data handling and querying.
- **Use in the Project:**
 - Storing book, member, and transaction data.
 - Ensuring data integrity and easy retrieval for system operations.

4. Framework: Java Swing

- **Why use Java Swing:** Swing is a GUI toolkit in Java that allows for the creation of cross-platform desktop applications with interactive and visually appealing interfaces.
- **Use in the Project:**
 - Building the login interface, dashboards, and forms for book and member management.
 - Ensuring a consistent and responsive design for user interaction.

5. Architectural Pattern: Model-View-Controller (MVC)

- **Why it use:** MVC separates the application logic into three interconnected components, promoting code organization, scalability, and maintainability.
- **Use in the Project:**
 - **Model:** Contains the business logic and data (e.g., classes for Book and Member).
 - **View:** Manages the graphical user interface and user interactions.
 - **Controller:** Acts as a mediator, handling user inputs and updating the View and Model.

6. Exception Handling

- **Why use Exception Handling:** Exception handling ensures the application can handle errors gracefully without crashing.
- **Use in the Project:**
 - Managing invalid login attempts, unauthorized access, and invalid data inputs.
 - Improving the robustness and reliability of the system.

Conclusion

The Library Management System (LMS) successfully addresses the core needs of modern library operations by automating book and member management, streamlining transactions, and providing a user-friendly interface. By leveraging Java, MySQL, and the MVC architecture, the system ensures scalability, reliability, and ease of use. This project not only enhances operational efficiency but also lays a strong foundation for future upgrades, such as advanced analytics and mobile app integration, to meet evolving library requirements.

Chapter 3 - Design

Introduction

The design of the Library Management System focuses on creating an intuitive, user-friendly interface and a well-structured backend to streamline library operations. Adopting the Model-View-Controller (MVC) architecture ensures a clear separation of concerns, making the system maintainable and scalable. The user interface, built using Java Swing, provides easy navigation for admins and members, while the backend logic and database ensure efficient data handling and secure operations. Each module is tailored to support critical functionalities such as book management, member management, and transaction tracking.

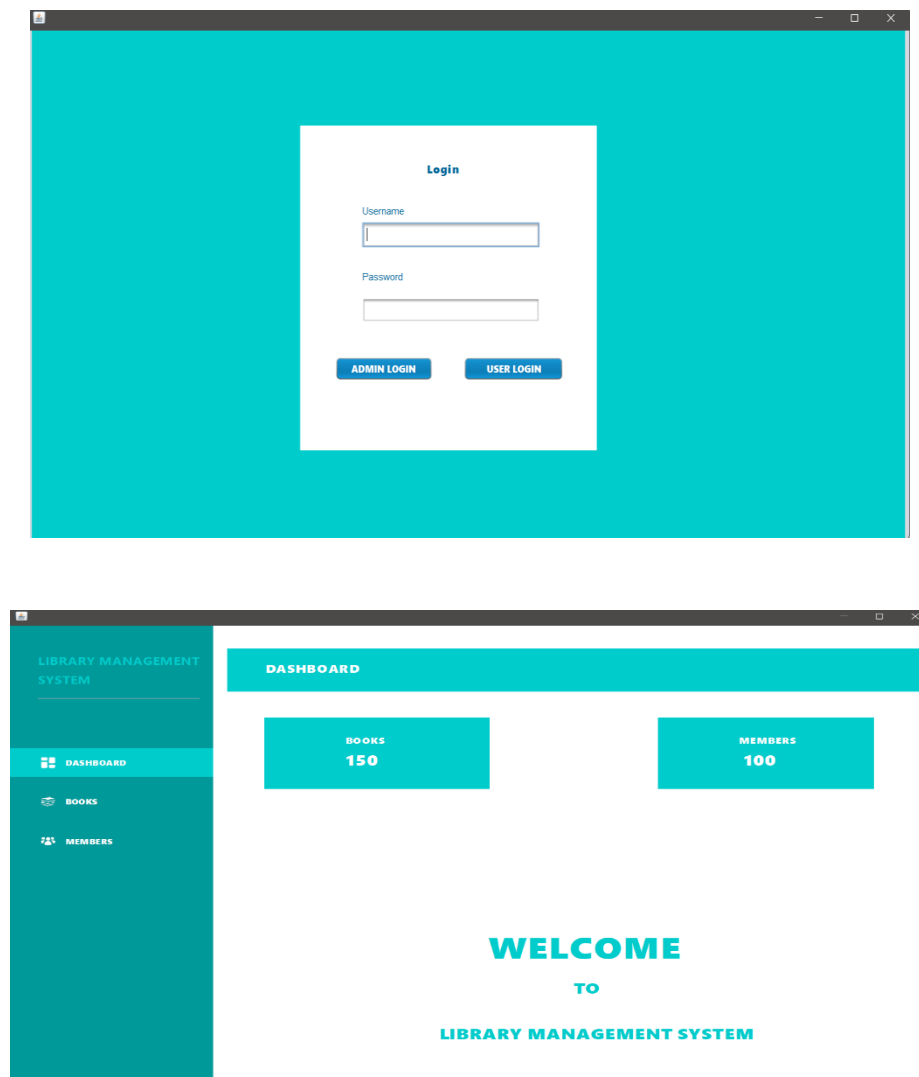
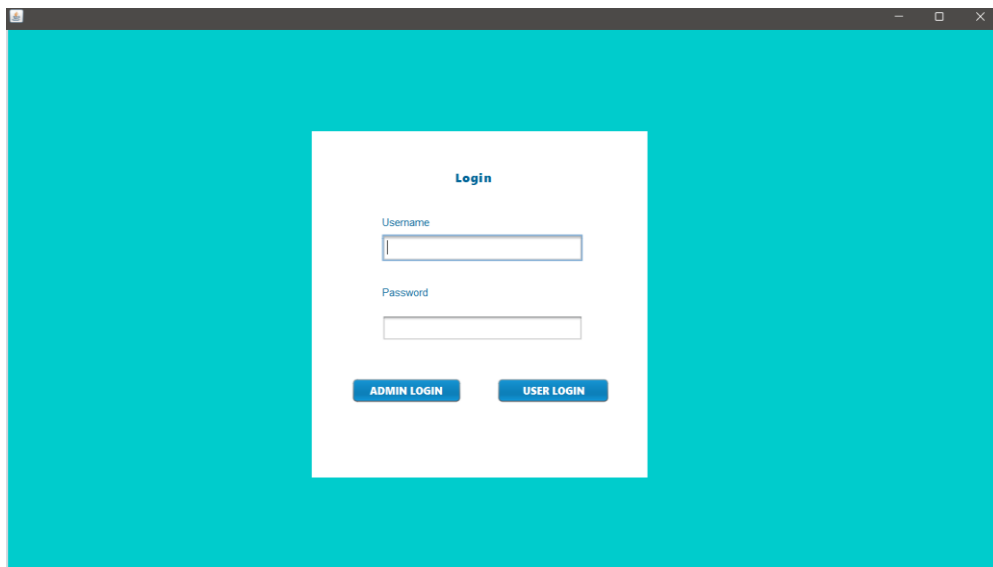


Figure1

Modules/Sections Overview

1. Login(user/admin)

- **Functionality:** Secure authentication system for the user and admin. Verify with the correct User name & Password.



The screenshot shows a web browser window with a solid teal background. In the center is a white rectangular login form. At the top of the form is the title "Login". Below it are two input fields: "Username" and "Password". At the bottom of the form are two blue buttons: "ADMIN LOGIN" and "USER LOGIN".

Figure2

- **Dash Board:** Welcome page for the System. Admin & User can see the dash board when they login to the system

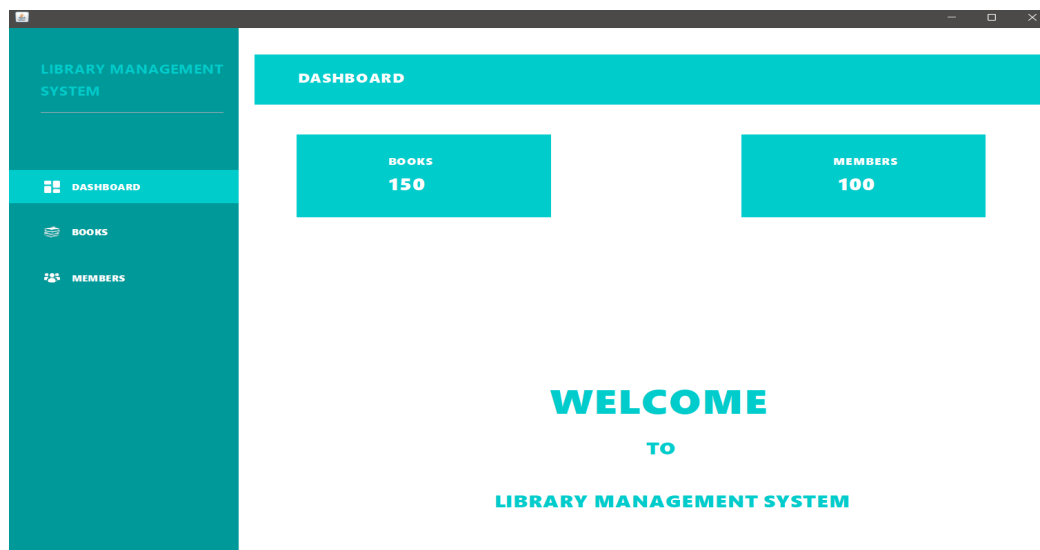


Figure3

- **Book:** Admin can Add, Update, Delete, Reset & Search book details

The screenshot shows the 'BOOKS' management page. On the left is a sidebar with 'LIBRARY MANAGEMENT SYSTEM' and navigation links for 'DASHBOARD', 'BOOKS', and 'MEMBERS'. The main content area has a teal header 'BOOKS'. Below it, there are input fields for 'Book ID', 'Title', 'Author', 'Published Year', 'Genre', 'Language', and 'Copies Available'. To the right of these fields is a search bar and a 'SEARCH' button. Below the input fields are four buttons: 'Add', 'Update', 'Delete', and 'Reset'. On the right side of the page is a table with the following columns: 'Book ID', 'Title', 'Author', 'Published Year', 'Genre', 'Language', and 'Copies Available'. The table body is currently empty.

Figure4

- **Member:** Admin can Add, Update, Delete, Reset & Search member details

The screenshot shows the 'MEMBERS' management page. On the left is a sidebar with 'LIBRARY MANAGEMENT SYSTEM' and navigation links for 'DASHBOARD', 'BOOKS', and 'MEMBERS'. The main content area has a teal header 'MEMBERS'. Below it, there are input fields for 'Member ID', 'Name', 'Address', 'Telephone No', 'Whatsapp No', 'Membership Card', and 'MC Expire Date'. To the right of these fields is a search bar and a 'SEARCH' button. Below the input fields are four buttons: 'Add', 'Update', 'Delete', and 'Reset'. On the right side of the page is a table with the following columns: 'Member ID', 'Name', 'Address', 'Telephone No', 'Whatsapp No', 'Membership Card', and 'MC Expire Date'. The table body is currently empty.

Figure5

User Case Diagram

The use case diagram represents the interactions between the user roles (Admin) and the system's main functionalities

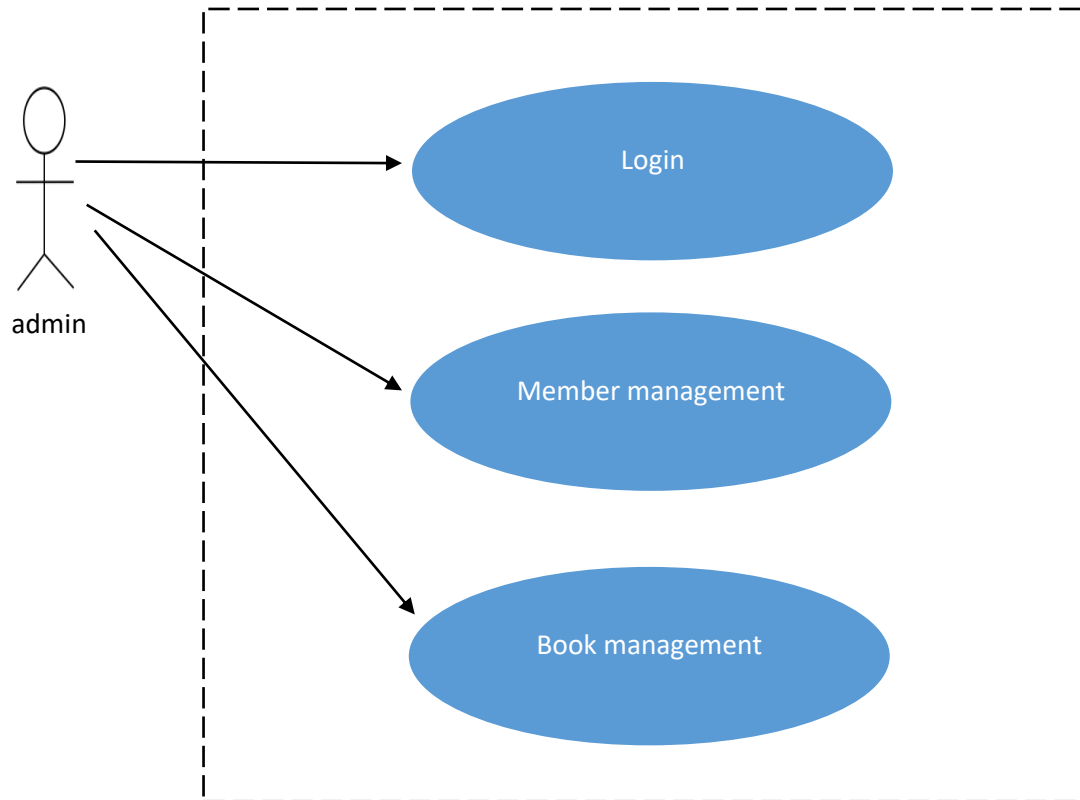


Figure6

ER Diagram

The ER diagram represents the entities in the Library Management System and how they are related

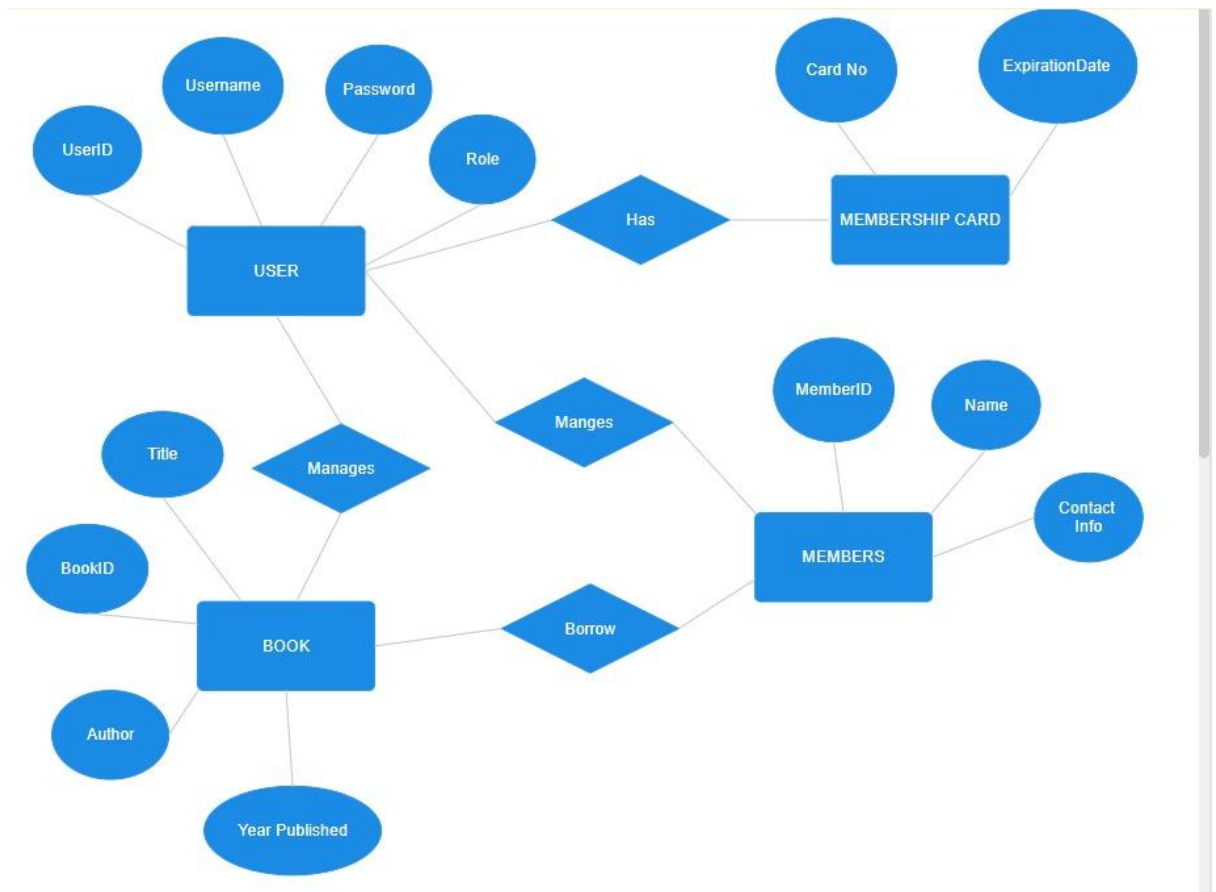


Figure7

Class Diagram

The class diagram shows the classes in the system, their attributes, constructors, methods and relationships.

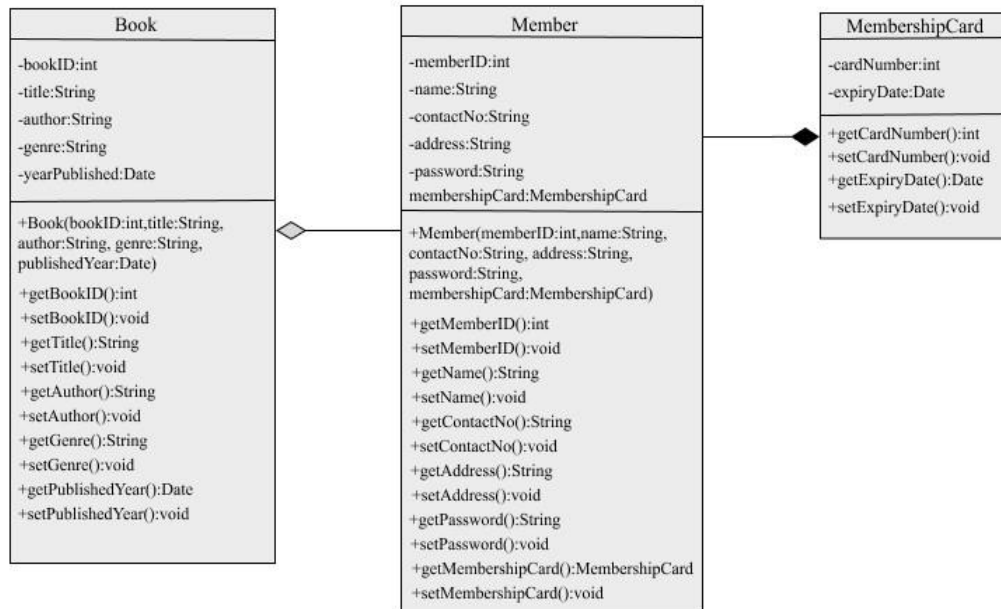


Figure8

Chapter 4 - Conclusion & Further Work

Achievement

The implementation of the Library Management System (LMS) has significantly streamlined library operations by automating book and member management and providing a user-friendly platform for borrowing and returning books. The system enhances efficiency, reduces manual workload, and ensures secure role-based access for users.

Problem Encounter & Limitations

While the system is functional, certain limitations were identified:

1. **Interface Customization:** Designing a visually appealing and fully responsive user interface posed challenges. Some components lack the desired level of polish, impacting user experience.
2. **Advanced Features:** Features like detailed reporting and analytics or automated late return notifications are currently not implemented, limiting the system's capability to provide in-depth operational insights.

Further Work

To address these limitations and improve the Library Management System, the following steps are proposed:

1. **Customization Refinement:** Explore additional design tools or frameworks to improve the user interface. This will enhance aesthetics and usability, ensuring a more professional and engaging experience for users.
2. **Integration of Advanced Features:** Implement features like overdue notifications, advanced reporting, and analytics to enhance functionality and provide deeper insights into library operations.
3. **Mobile Application Support:** Develop a mobile application for greater accessibility and convenience for users.
4. **Cloud Integration:** Introduce cloud storage for data synchronization and remote access, making the system scalable and adaptable for larger libraries.

Chapter 5 – User Manual

Introduction	15
System Requirements	15
Login Instructions	15
Features and Functions	15
Admin User Functions	15
Step-by-Step Instructions	16
Managing Books	16
Managing Members	16
Troubleshooting	16
Contact Information	16

5.1 – Introduction

The Library Management System (LMS) is a desktop application designed to simplify the management of books, members, and transactions in a library. This user manual provides step-by-step guidance for using the system effectively.

5.2 – System Requirements

Hardware:

- Processor: Dual-core or higher
- RAM: 4GB or more
- Storage: 500MB free space

Software:

- Java Runtime Environment (JRE) 8+
- MySQL Database
- Operating System: Windows

5.3 – Login Instructions

1. Open the application.
2. Enter your **Username** and **Password**.
3. Click the **Login** button.
4. Based on your role (Admin or Member), you will be directed to the respective dashboard.

5.4 – Features and Functions

Admin User Functions

1. Manage books: Add, edit, delete, and search books.
2. Manage members: Register, update, delete, and view member details.

5.5– Step-by-Step Instructions

Managing Books (Admin Role Only)

1. Go to the **Books** section in the dashboard.
2. Use the **Add Book** button to input book details (title, author, year, etc.).
3. Use the **Edit Book** option to update book information.
4. To delete a book, select it from the list and click **Delete**.
5. Use the **Search** field to find books by title or author.

Managing Members (Admin Role Only)

1. Navigate to the **Members** section.
2. Click **Add Member** to register a new user.
3. Use the **Edit Member** option to update member details.
4. To remove a member, select their record and click **Delete**.

5.5– Troubleshooting

Problem: Unable to log in.

Solution: Ensure your username and password are correct. Contact the admin if you forget your credentials.

Problem: Book or member details not updating.

Solution: Check for missing or invalid fields during data entry.

5.5– Contact Information

For technical support or further assistance:

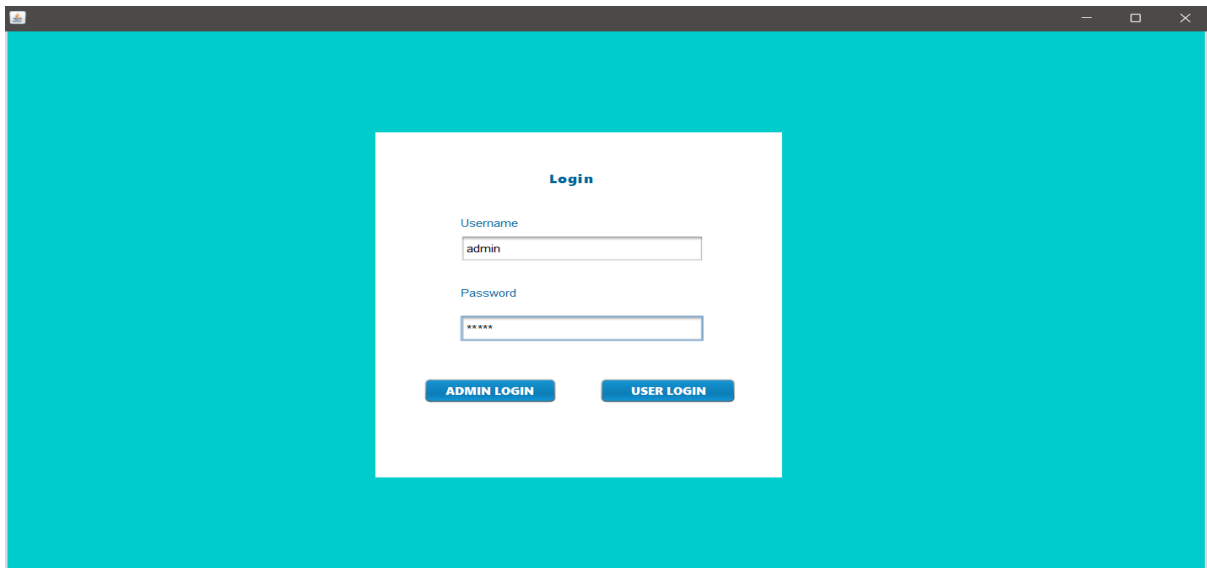
- **Email:** support@lms.com
- **Phone:** +94 77 35 524 896
- **Website:** librarymanagementsystem

References

- Java Programming Documentation: [Java Docs](#)
- NetBeans IDE Guide: [NetBeans Documentation](#)
- MySQL Database Documentation: [MySQL Docs](#)

Appendixes

Appendix A – Admin Login



A screenshot of a web browser window displaying a login form. The form is centered on a solid teal background. It has a title "Login" in bold. Below the title are two input fields: "Username" with the text "admin" and "Password" with masked characters "*****". At the bottom of the form are two buttons: "ADMIN LOGIN" and "USER LOGIN".

Figure9

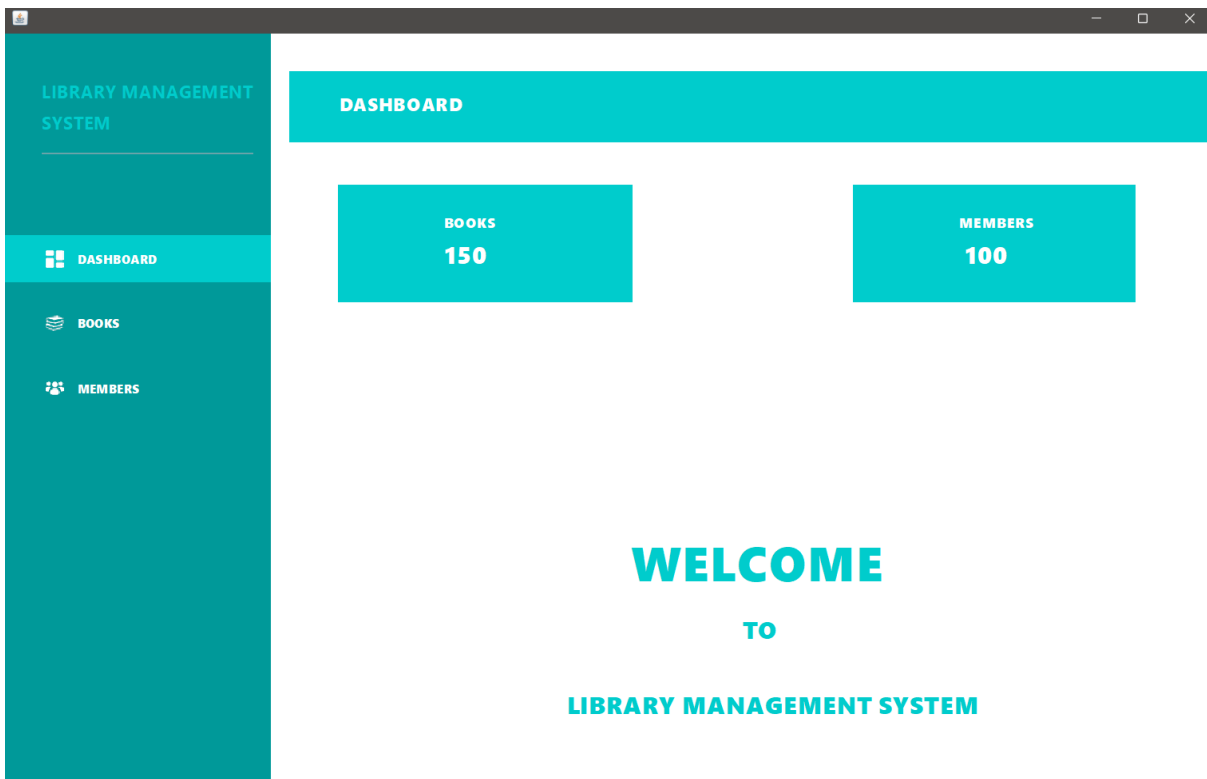
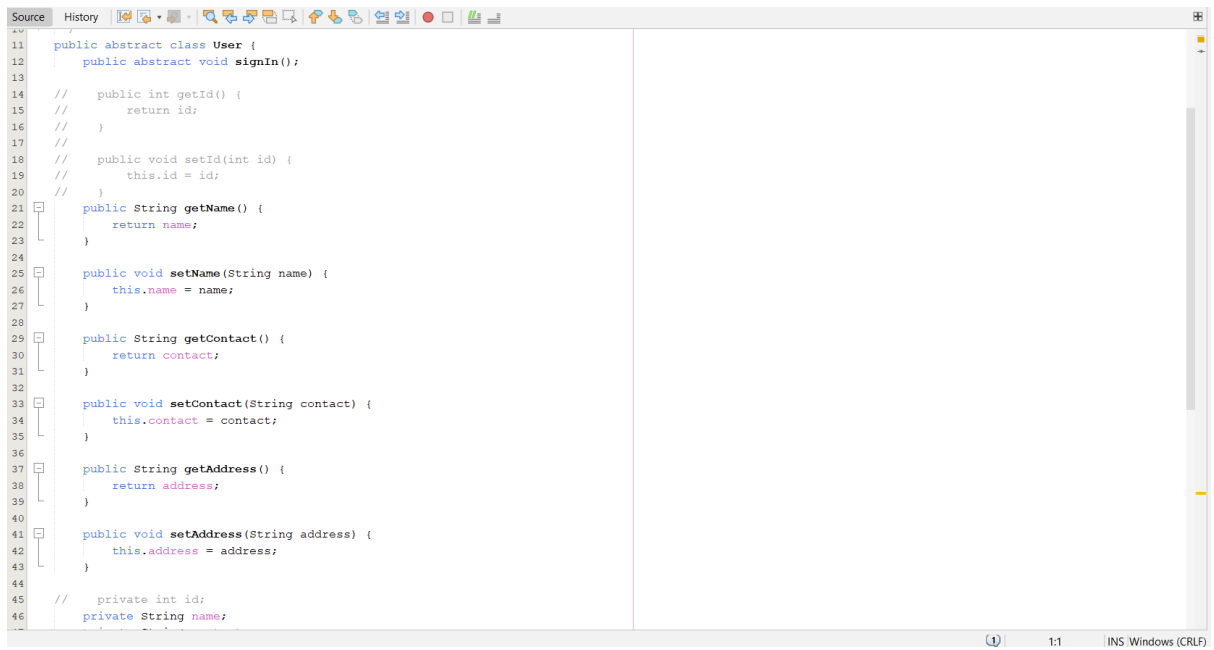


Figure10

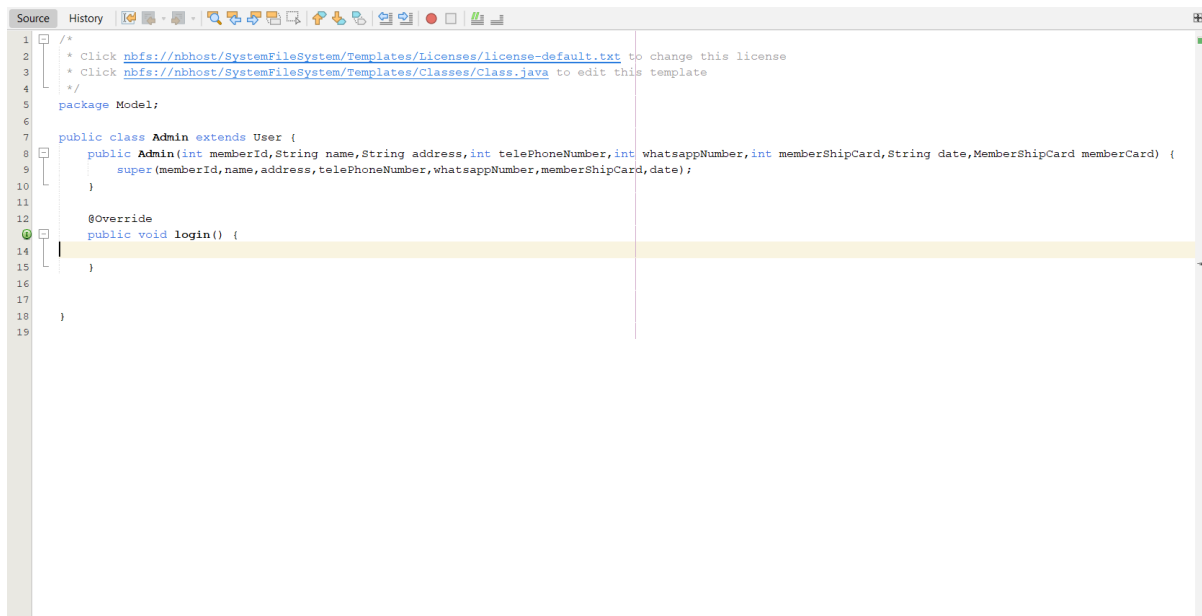
OPP CONSEPTS



The screenshot shows an IDE window with a source code editor. The code defines an abstract class `User` with several methods. The methods include `signIn()`, `getId()`, `setId()`, `getName()`, `setName()`, `getContact()`, `setContact()`, `getAddress()`, and `setAddress()`. The class also has private attributes `id` and `name`.

```
11 public abstract class User {
12     public abstract void signIn();
13
14     // public int getId() {
15     //     return id;
16     // }
17     //
18     // public void setId(int id) {
19     //     this.id = id;
20     // }
21     public String getName() {
22         return name;
23     }
24
25     public void setName(String name) {
26         this.name = name;
27     }
28
29     public String getContact() {
30         return contact;
31     }
32
33     public void setContact(String contact) {
34         this.contact = contact;
35     }
36
37     public String getAddress() {
38         return address;
39     }
40
41     public void setAddress(String address) {
42         this.address = address;
43     }
44
45     // private int id;
46     private String name;
```

Figure13



The screenshot shows an IDE window with a source code editor. The code defines a class `Admin` that extends the `User` class. The `Admin` class has a constructor that takes several parameters and a `login()` method. The `login()` method is highlighted with a yellow background.

```
1 /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5 package Model;
6
7 public class Admin extends User {
8     public Admin(int memberId, String name, String address, int telePhoneNumber, int whatsappNumber, int memberShipCard, String date, MemberShipCard memberCard) {
9         super(memberId, name, address, telePhoneNumber, whatsappNumber, memberShipCard, date);
10     }
11
12     @Override
13     public void login() {
14         //
15     }
16
17
18 }
19
```

Figure14


```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package Model;
6
7  public abstract class User {
8      int memberId;
9      private String name;
10     String address;
11     int telephoneNumber;
12     int whatsappNumber;
13     String date;
14
15
16     public User(int memberId,String name,String address,int telePhoneNumber,int whatsappNumber,int memberShipCard,String date){
17         this.memberId = memberId;
18         this.name = name;
19         this.address = address;
20         this.telePhoneNumber = telePhoneNumber;
21         this.whatsappNumber = whatsappNumber;
22         //this.memberShipCard = memberShipCard;
23         this.date = date;
24     }
25
26     public abstract void login();
27
28     public int getmemberId(){
29         return memberId;
30     }
31
32     public String getName(){
33         return name;
34     }
35
36     public void setName(String name){
```

Figure15