Library Management System

Object Oriented Programming in Java- HNDIT 3012

KAN/IT/2022/F/0084

Department of Information Technology

Advanced Technological Institute

Kandy

2024

**Table of Contents**

1. Introduction
2. System Objectives
3. Project Scope
4. System Design
   1. Architecture
   2. Packages

5. Database design

5.1. Tables

6. Testing

OOP Concepts Applied

7. Results

8. Future Enhancement

9. Diagrams

9.1. ER Diagram

9.2. Class Diagram

10. Conclusion

11. GUI of the System

12. References

1. **Introduction**

The **Library Management System (LMS)** is an automated platform designed to streamline library functions such as book issuance, returns, member management, and database handling. This system uses Java Swing for its graphical user interface (GUI), MySQL for data persistence, and follows the **MVC (Model-View-Controller)** architectural pattern for separation of concerns.

1. **System Objectives**
   * **Automation of Library Functions**: Efficiently manage operations like book issuance, returns, member management, and more.

* **User-Friendly Interface**: Simplify the user experience for both library members and administrators.
* **Data Integrity**: Maintain accurate records for books, members, and transactions

1. **Project Scope**

The Library Management System covers the following functionalities:

* 1. **Book Management:**
     + **Add, Update, Remove, Search** books..
     + View books and track their availability (Issued/Returned status).
  2. **Member Management:**
     + Add new members, deactivate existing ones.
     + View member details such as contact information and borrowed details.
  3. **Login System:**
* Secure login system for both library administrators and regular members.

1. **Database Integration:**

o All system data (books, members, transactions) is stored in a **MySQL database** for persistent storage.

1. **System Design** 
   1. **Architecture**

The system is based on the **MVC architecture**, which consists of Three main components:

* + - **Model:** Represents the data and logic. It includes:
* Book class (manages book details)
* Member class (manages member data)
* Transaction class (manages book borrowing and returning details)
  + - **View:** Represents the user interface using Java Swing:
* GUI Components like LoginForm, BookManagementPanel, MemberDetailsPanel.
  + - **Controller or DAO:** Contains the business logic and database interaction:
* BookDAO (handles operations related to books)
* MemberDAO (handles member-related operations)
* IssueDAO & ReturnDAO (manages borrow/return operations)

* 1. **Packages**

1. **Model Package:**

* + - * **Classes:** Admin, Book, Member, MembershipCard, User, IssueBook, ReturnBook.
      * Represents Constructors and Variable declaration .

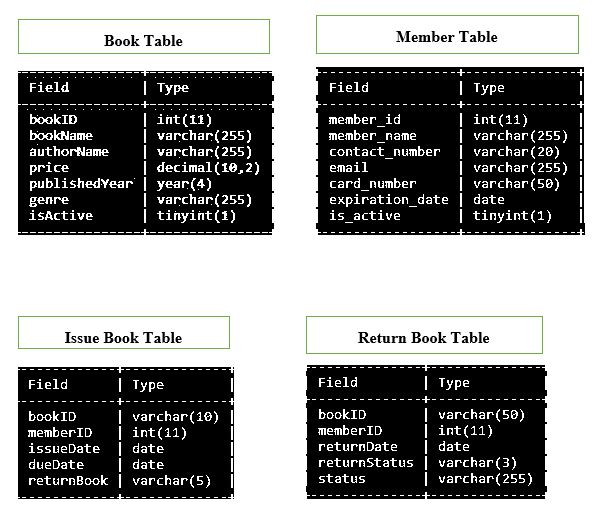
2. **View Package:**

* + - * **JFrames**: Welcome, LoginView, Menu, AddMembers, AddBooks, MemberManagement, BookManagement, Issuebooks, Returnbooks, Statistics.
      * Provides graphical user interface (GUI) for the system.

3. **Controller or DAO Package:**

* + - * **Classes**: BookDAO, MemberDAO, IssueBookDAO, ReturnDAO, StatisticsDAO LoginController.
      * Manages database operations and authentication logic.

1. **Database Design**



**5.1 Tables**

* + - **Books**
      * id (INT, Primary Key) o title (VARCHAR) o author (VARCHAR)
      * Genre(VARCHAR) o isIssued (BOOLEAN)
      * Price(Decimal)
    - **members**
      * id (INT, Primary Key) o name (VARCHAR) o contact (VARCHAR)
      * Email(varchar(255))
      * CardNumber(varchar(50))
    - **users**
      * id (INT, Primary Key) o username (VARCHAR) o password (VARCHAR)

* + - **borrowed**
      * id (INT, Primary Key) o book\_id (INT, Foreign Key) o member\_id (INT, Foreign Key) o borrow\_date (DATE) o return\_date (DATE)

1. **Testing** 
   1. **Unit Testing**:

Ensure the individual functionalities such as adding books, issuing books, etc., work as intended.

* 1. **Integration Testing**:

Verify that the database operations (CRUD) interact seamlessly with the GUI components.

* 1. **Performance Testing**:

Monitor the system for speed, especially during queries or user actions, ensuring efficient performance.

**OOP Concepts Applied**

The **Library Management System (LMS)** is designed using key Object-Oriented Programming (OOP) principles, ensuring modularity, scalability, and maintainability::

**Encapsulation:** The system’s classes (e.g., Management, Database) encapsulate internal logic and data, exposing only necessary methods and properties. This protects data integrity and promotes modularity.

**Inheritance**: The LMS uses inheritance to create specialized data entities (Admin, Member) that inherit common properties and methods from a base class(User class), enabling efficient management and code reuse.

**Composition**: MembershipCard class cannot work without the member class

By applying these OOP principles, the LMS is adaptable, easy to extend, and well-suited for future development.

1. **Results**

The Library Management System successfully provides:

1. **User-Friendly Interface**:

The interface is designed for ease of use, allowing administrators to manage books, members, and transactions effortlessly.

1. **Data Persistence**:

All data is stored securely in the MySQL database.

1. **Security**: User authentication ensures only authorized access to sensitive operations.

.

1. **Future Enhancements**

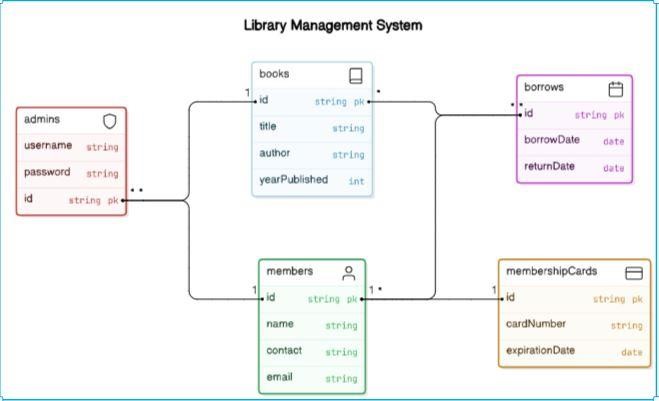
 **Search Features**: Add the ability to search books and members by various criteria (e.g., title, name, category)..

 **Reports**: Generate reports such as overdue books, transaction histories, and active members.

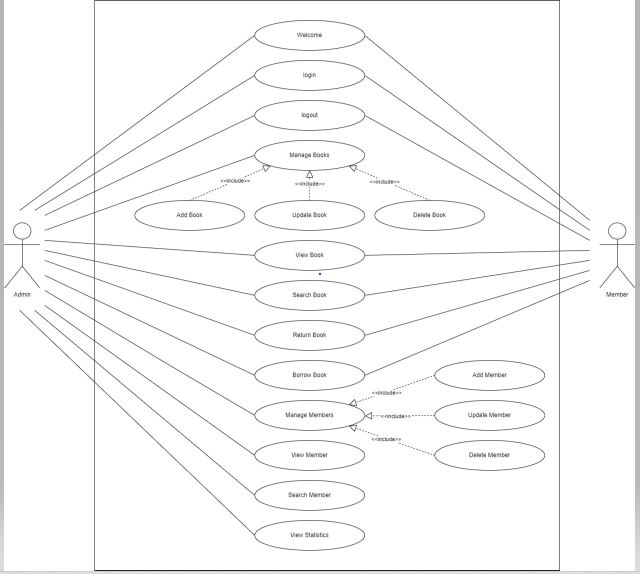
 **Advanced UI**: Transition from Java Swing to JavaFX for enhanced styling and responsiveness.

1. **Diagrams**

**9.1. ER Diagram**



**9.2. Class Diagram**

****

1. **Conclusion**

The **Library Management System (LMS)** developed in this project is a feature-rich application that effectively applies object-oriented programming principles. Its modular architecture, with a clear separation of concerns between the Controller, Model, and View components, ensures both maintainability and scalability.

Key components include the **Management class** for coordinating system operations, the **Database class** for handling database logic, and the **Constructor and Variable Declaration class** for managing data objects. The **Graphical User Interface (GUI)** provides an intuitive and user-friendly experience.

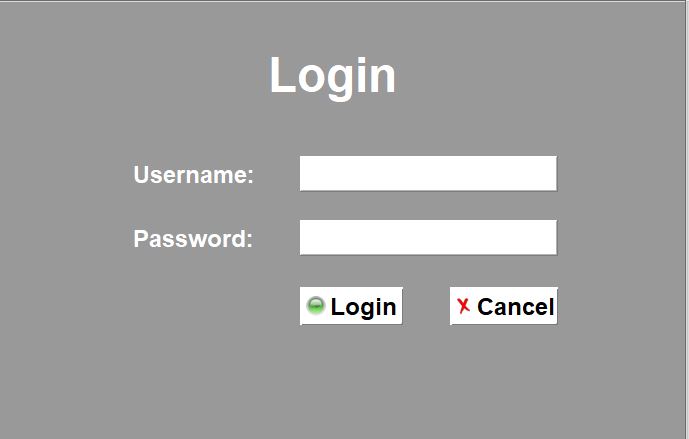
By leveraging OOP concepts such as **encapsulation**, **inheritance**, and **composition** the LMS achieves high modularity and flexibility, making it adaptable for future development. This report offers a comprehensive overview of the system's architecture, providing valuable insights for understanding and maintaining the LMS.

1. **GUI of the System**

* **Welcome:**

****

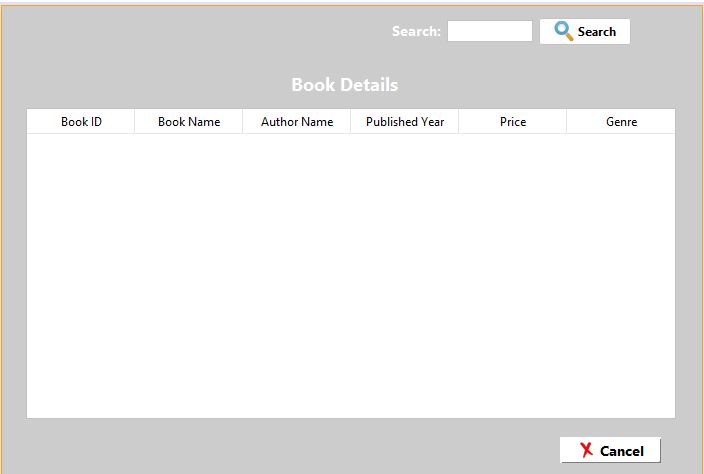
* **Login**

****

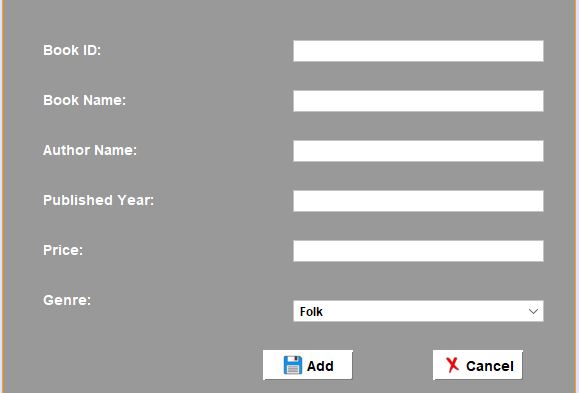
* **Menu**



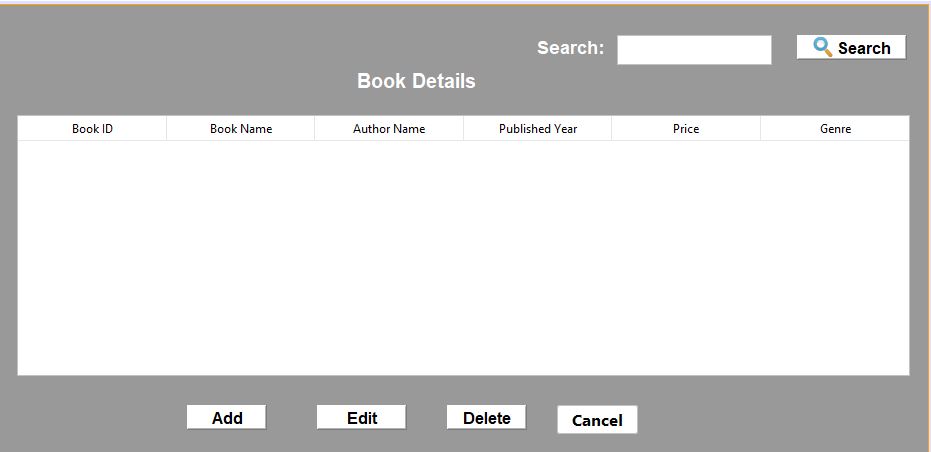
* **Member View**



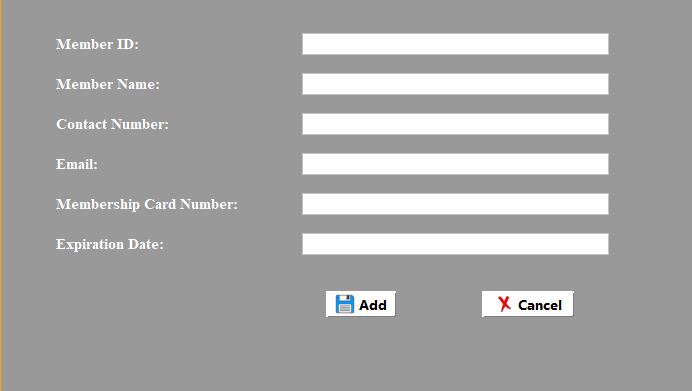
* **Add Book**



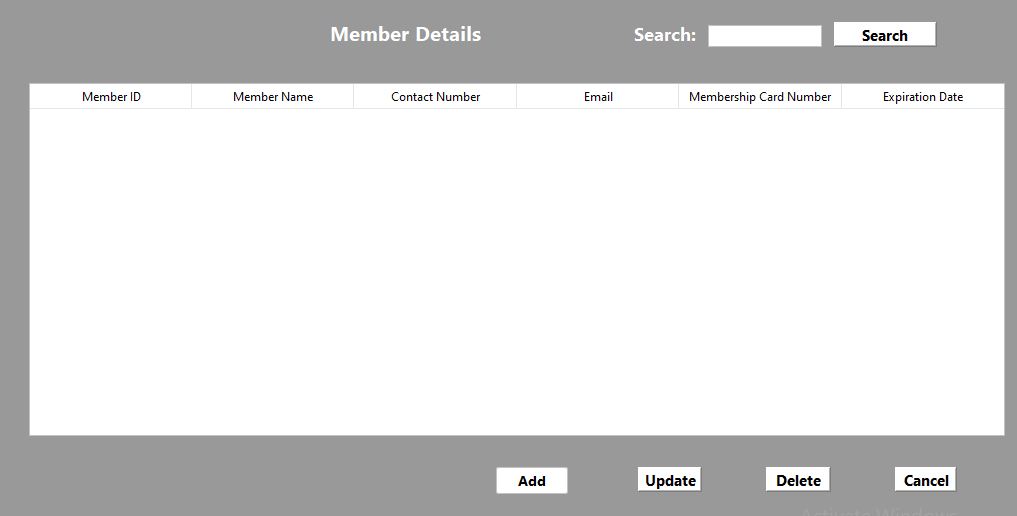
* **Book Management**

****

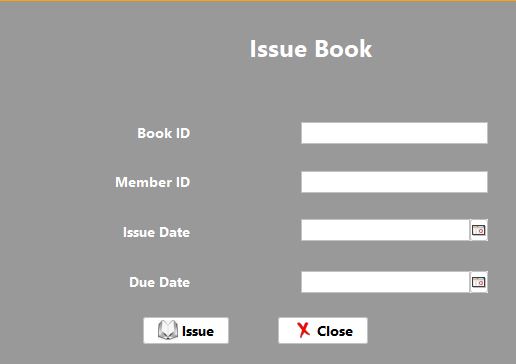
* **Add Member**

****

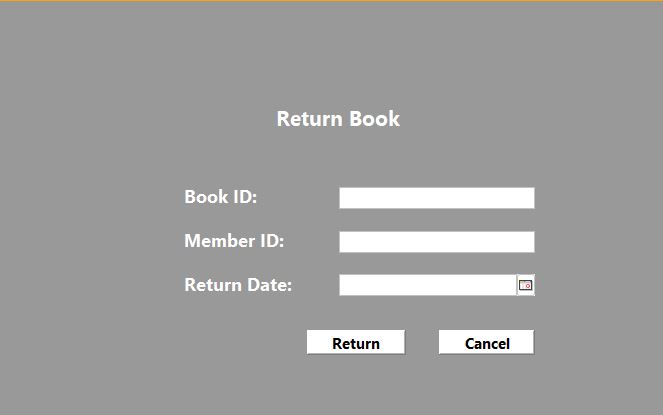
* **Member Management**

****

* **Issue Books**

****

* **Return Books**

****

1. **References**

* "Java: The Complete Reference" by Herbert Schildt
* Oracle Java Documentation
* Oracle Java Docs
* W3 School
* BTech youtube channel