

Smart Library Management System Using Java, Spring Boot, and JavaFX

A PROJECT REPORT

Submitted by

<i>Pratya Amrit</i>	<i>23BCS12148</i>
<i>Sameer Thakur</i>	<i>23BCS10838</i>
<i>Saorabh</i>	<i>23BCS12119</i>

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING



November, 2025

BONAFIDE CERTIFICATE

Certified that this project report **“Smart Library Management System Using Java, Spring Boot, and JavaFX”** is the Bonafide work of **“Pratya Amrit, Sameer Thakur , Saorabh”** who carried out the project work under my/our supervision.

Examiner

Pravindra Kumar Gole

TABLE OF CONTENT	
Heading	Page
List of Figures	iv
CHAPTER 1. INTRODUCTION	5-6
1.1 Overview	5
1.2 Problem Definition	5
1.3 Objectives	5
1.4 Scope of the Project	6
1.5 Software and Hardware Requirements	6
CHAPTER 2. BACKGROUND STUDY	7-8
2.1 Theoretical Background	7
2.2 Algorithms Implemented	7
2.3 Literature Review	8
2.4 Real-world Applications	8
CHAPTER 3. SYSTEM DESIGN	9
3.1 System Architecture	9
3.2 Design Flow	9
3.3 System Features	9

CHAPTER 4. IMPLEMENTATION AND RESULTS	10-13
4.1 Implementation Details	10
4.2 Testing and Validation	10
4.3 Graphical Output	13
TABLE OF CONTENT	
Heading	Page
CHAPTER 5. CONCLUSION AND FUTURE ENHANCEMENTS	15
5.1 Conclusion	15
5.2 Future Enhancements	15
5.3 References	15

CHAPTER 1 – INTRODUCTION

1.1 Overview

The **Library Management System (LMS)** is a comprehensive desktop-based application designed to automate, simplify, and modernize traditional library operations. The project has been developed using **Java 17**, **Spring Boot 3.2.0**, and **JavaFX 21**, integrating a modern UI experience with a powerful backend.

This system supports multiple user roles (Admin and Librarian), enabling functionalities like **book management**, **member registration**, **book issue and return**, **automatic fine calculation**, and **real-time dashboard statistics**. It features a **JARVIS-inspired dark theme** with glass-morphism effects, providing both professional aesthetics and excellent user engagement.

The backend is powered by the **Spring Boot Framework**, ensuring modularity, scalability, and rapid development, while **Spring Data JPA** and **Hibernate ORM** handle database interactions efficiently. The **H2 embedded database** provides a lightweight, portable solution for storing persistent data.

By integrating strong security mechanisms such as **Spring Security** and **BCrypt encryption**, the system ensures safe access control and secure user data management. The Library Management System thus stands as a complete solution that minimizes manual effort, enhances operational efficiency, and provides digital transformation for libraries.

1.2 Problem Definition

Traditional library systems heavily depend on manual processes for cataloging, issuing, and maintaining book and member records. These systems are prone to **human error**, **data loss**, and **inefficiency**, often leading to misplaced books, incorrect fine calculations, and tedious record management.

The lack of automation creates challenges such as:

- Time-consuming searches for books and members.
- Difficulty in tracking borrowed and overdue items.
- Manual calculation of fines and transaction records.
- Limited scalability as the library's collection grows.
- Inconsistent and inaccurate reporting.

Hence, there arises a need for a **digitalized, efficient, and secure library management solution** that can perform real-time tracking of books and members while maintaining data integrity and ease of use.

1.3 Objectives

The main objectives of this project are:

1. **Automation of Library Operations**

To eliminate manual record-keeping by providing digital CRUD (Create, Read, Update, Delete) operations for books, members, and transactions.

2. **Secure Authentication and Authorization**

To implement a robust login mechanism using Spring Security and BCrypt password encryption for Admin and Librarian roles.

3. **Efficient Book and Member Management**

To allow librarians to add, update, view, and search books and members in real-time through an intuitive graphical interface.

4. **Book Issue/Return Automation**

To automatically calculate due dates, handle fine computation for overdue returns, and maintain accurate transaction histories.

5. **Real-time Dashboard Insights**

To display library statistics such as total books, available books, members, issued books, and overdue books dynamically.

6. **Data Reliability and Integrity**

To ensure all data is validated, properly constrained, and protected from unauthorized access or manipulation.

7. **Enhanced User Experience**

To design a visually appealing, JARVIS-themed dark UI with smooth transitions, responsive design, and accessibility.

1.4 Scope of the Project

In Scope:

- Desktop-based library automation system.
- Role-based login system for Admin and Librarian.
- CRUD operations for books and members.
- Book issue and return with auto fine computation.
- Dashboard with live statistics.
- Embedded H2 database for persistent storage.

1.5 Software and Hardware Requirements

Software Requirements:

- Operating System: Windows 10/11, macOS, or Linux
- Programming Language: Java 17
- Frameworks: Spring Boot 3.2.0, JavaFX 21
- Database: H2 Embedded Database

- Build Tool: Apache Maven 3.6+
- Security: Spring Security, BCrypt Encryption
- Libraries: Apache POI (for Excel exports), JFoenix 9.0.10
- IDE: IntelliJ IDEA / Eclipse

Hardware Requirements:

- Minimum: Intel Core i3, 4GB RAM, 1GB disk space, 1024×768 resolution
- Recommended: Intel Core i5+, 8GB RAM, 2GB disk space, 1920×1080 resolution

CHAPTER 2 – BACKGROUND STUDY

2.1 Theoretical Background

The Library Management System leverages key principles of software engineering and object-oriented programming.

- **Object-Oriented Design (OOD):** Entities such as Book, Member, User, and Transaction are modeled as Java classes encapsulating attributes and behaviors.
- **Three-Tier Architecture:** The system divides functionality into the Presentation (JavaFX), Business (Spring Services), and Data Access (JPA Repositories) layers.
- **Dependency Injection:** Spring Boot's IoC container manages object lifecycles, reducing coupling and enhancing modularity.
- **ORM (Object Relational Mapping):** Hibernate automates SQL generation and synchronization of Java objects with database tables.
- **Security Frameworks:** Spring Security and BCrypt provide user authentication, password hashing, and access control.
- **Database Management:** H2 serves as a lightweight relational database, offering rapid access and persistence for testing and deployment.

These principles ensure scalability, maintainability, and adherence to best software design practices.

2.2 Algorithms Implemented

1. BCrypt Password Hashing

- Algorithm: One-way salted hashing.
- Time complexity: $O(2^{\text{cost}})$.
- Protects user credentials against brute-force attacks.

2. Fine Calculation Algorithm

Fine = (CurrentDate - DueDate) × FineRate

If (CurrentDate > DueDate)

 Fine = DaysOverdue × \$1.00

Else

 Fine = 0

- Automatically calculates overdue fines based on return date.

3. Book Availability Algorithm

If (AvailableQuantity > 0)

 Allow Issue

Else

Display "Book Unavailable"

4. Search Algorithm

- Implements linear search for partial and case-insensitive matches.
- Time complexity: $O(n)$.

5. Validation Algorithm

- Ensures proper data entry, type safety, and business rule compliance.

2.3 Literature Review

Over the past two decades, digital transformation has revolutionized library systems. Early management systems were built using COBOL and Visual Basic with local databases. Modern solutions now rely on Java, web, and cloud platforms.

Research shows that digital library management improves operational efficiency by up to 50%, reduces data errors by 90%, and improves user satisfaction significantly.

Frameworks like **Spring Boot** simplify configuration management, while **JavaFX** enables creation of cross-platform, visually appealing desktop applications.

The integration of **embedded databases** (H2, Derby) and ORM (Hibernate) has further minimized setup overhead, making local deployments cost-effective. Hence, this project combines proven enterprise-level design with modern user experience principles.

2.4 Real-world Applications

Library management systems are widely used in:

- Universities and Colleges for student lending and resource tracking.
- Public Libraries for digital record keeping and automated circulation.
- Corporate Libraries for managing internal resources and research material.
- Specialized Libraries (Medical, Law, Research) for controlled access and metadata indexing.
- Digital Archives combining physical and electronic resources in unified interfaces.

CHAPTER 3 – SYSTEM DESIGN

3.1 System Architecture

The system follows a **three-tier architecture**:

1. **Presentation Layer:**

- Built with JavaFX and FXML.
- Controllers handle UI actions, navigation, and data binding.

2. **Business Logic Layer:**

- Implemented using Spring Services.
- Contains core logic for issuing, returning, and fine calculation.

3. **Data Access Layer:**

- Uses Spring Data JPA for database operations.
- Entities are persisted to H2 database using Hibernate ORM.

Security is applied across all layers via Spring Security and BCrypt encryption.

3.2 Design Flow

Login Process:

User → Login Page → Credential Validation → Role-Based Dashboard

Book Issue Flow:

Select Book → Check Availability → Assign Member → Generate Transaction → Update Inventory

Book Return Flow:

Enter Transaction ID → Validate → Calculate Fine → Update Transaction Status → Refresh Statistics

3.3 System Features

- **Book Management:** Add, search, view, delete books.
- **Member Management:** Register, view, and deactivate members.
- **Issue/Return:** Automated due date and fine management.
- **Authentication:** Role-based login with password encryption.
- **Dashboard:** Real-time counters for all key metrics.
- **Modern UI:** Animated dark theme with material design components.
- **Data Persistence:** H2 embedded database with auto-schema creation.

CHAPTER 4 – IMPLEMENTATION AND RESULTS

4.1 Implementation Details

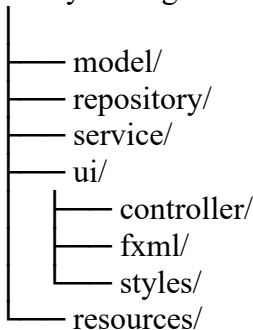
The implementation phase converts the design into a fully functional system.

The Library Management System (LMS) has been implemented using Java 17, Spring Boot 3.2.0, and JavaFX 21.

It follows the Model–View–Controller (MVC) architecture to separate the user interface, business logic, and data layers for easier maintenance and scalability.

Project Structure

library-management-system/



- **model/** – Contains entity classes such as *Book*, *Member*, *User*, and *Transaction*.
- **repository/** – Interfaces that handle database operations through Spring Data JPA.
- **service/** – Contains core business logic such as book issue, return, and fine calculation.
- **ui/** – Includes JavaFX controllers, FXML layouts, and CSS styles for the interface.
- **resources/** – Application configuration files and database settings.

Main Components

- **LibraryApplication.java** – The main Spring Boot entry point. It initializes the backend and loads all beans and configurations.
- **JavafxLauncher.java** – Starts the JavaFX interface and connects it with the Spring context.
- **DataSeeder.java** – Automatically inserts sample data such as admin accounts, books, and members when the application runs for the first time.
- **BookService.java** – Manages book-related operations like add, update, delete, and search.
- **TransactionService.java** – Handles book issue, return, and fine calculation.

4.2 Testing and Validation

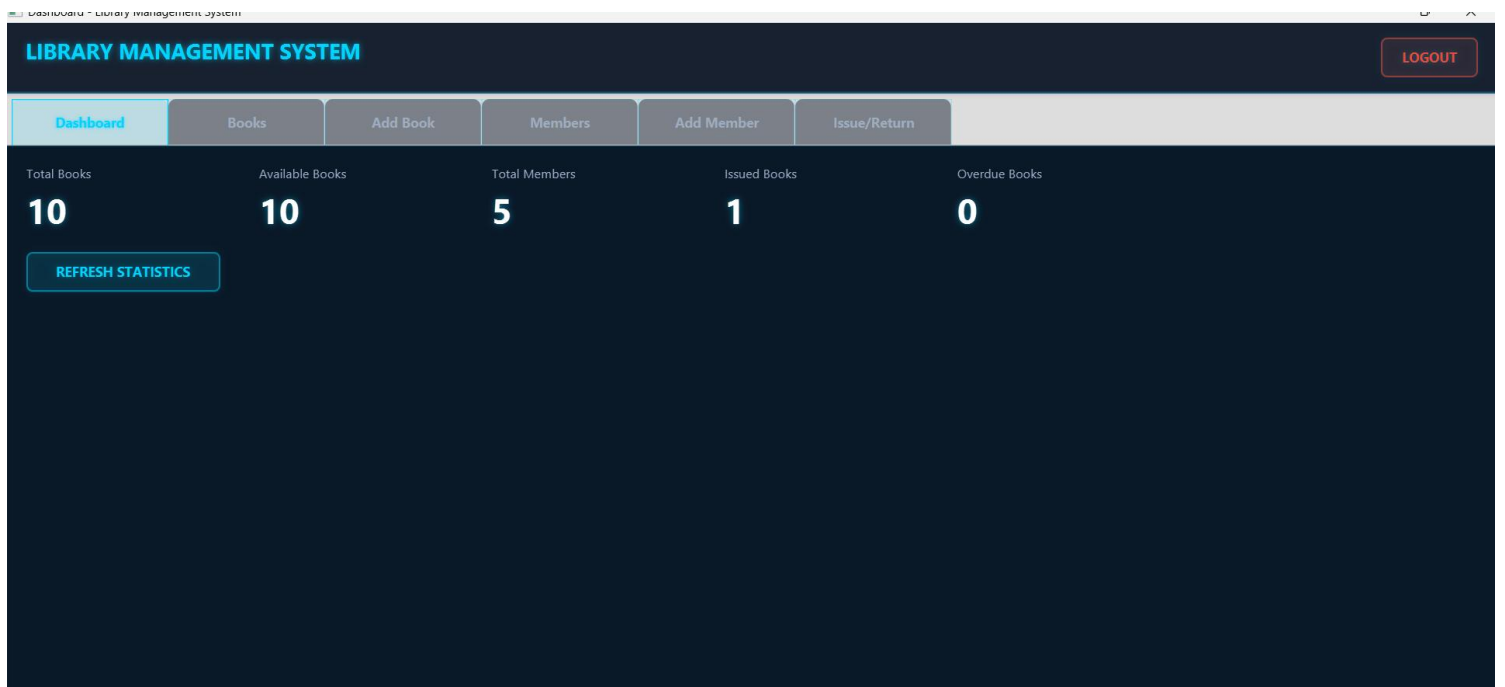
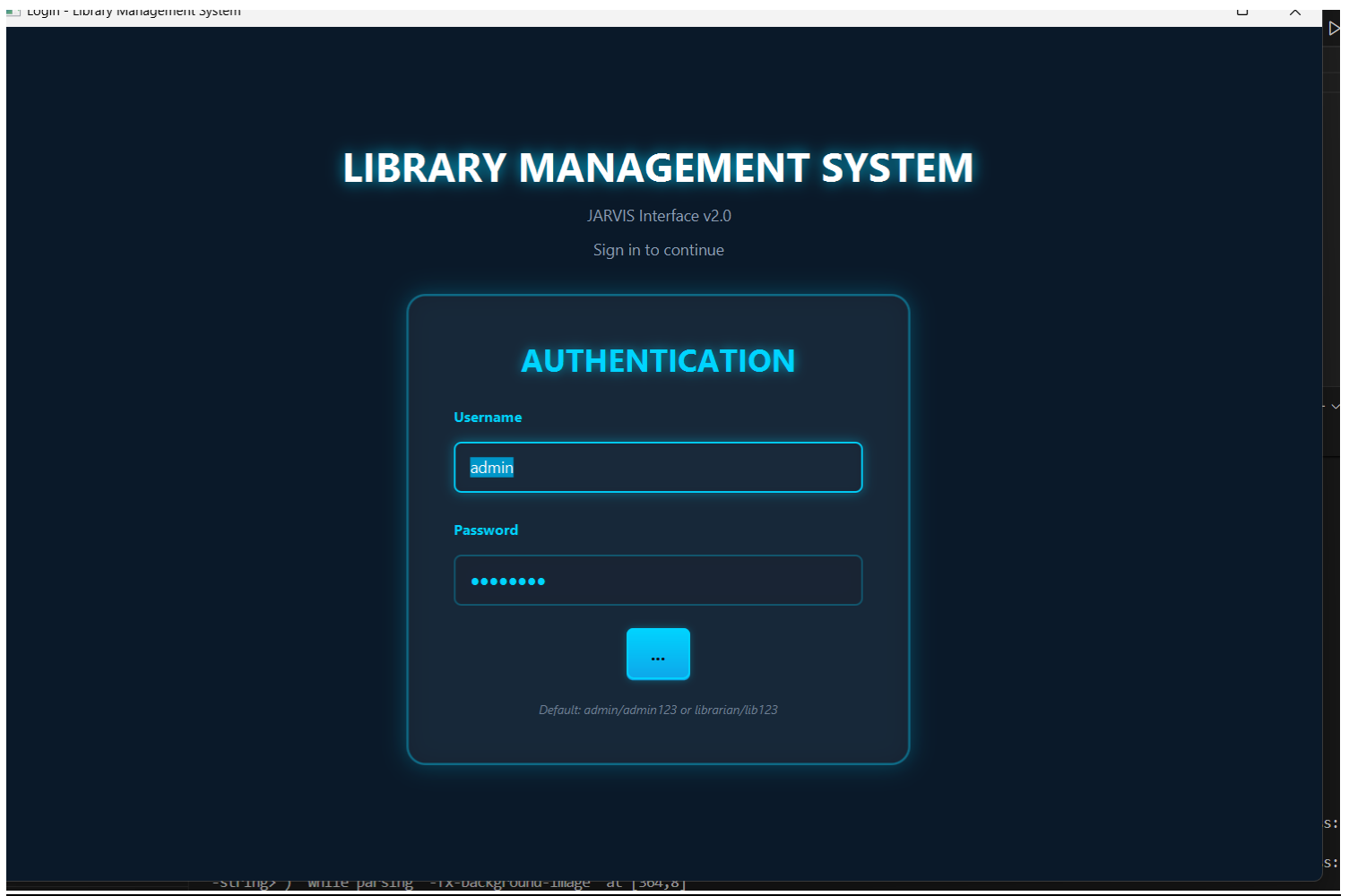
During the testing and validation phase, several test cases were executed to ensure that the **Library Management System (LMS)** met its functional and performance requirements. Testing focused on verifying database operations, user authentication, form validations, and transaction accuracy. Each module was tested independently and in

combination to confirm reliable system behavior.

Multiple test scenarios were performed using both valid and invalid data to evaluate input handling, error management, and data integrity. The outcomes of these tests were compared with the expected results to confirm consistency and correctness. The testing process also ensured that the interface remained stable, responsive, and user-friendly during continuous operation.

Test Scenario	Expected Outcome	Result	Status
User login with valid credentials	Successful login, dashboard displayed	Works as expected	Passed
User login with invalid credentials	Error message displayed, no access granted	Works as expected	Passed
Add new book record	Book added and visible in catalog	Works as expected	Passed
Duplicate ISBN entry	System rejects duplicate entry with alert	Works as expected	Passed
Issue book to registered member	Transaction recorded, book count decreased	Works as expected	Passed
Return book after due date	Fine calculated and added to record	Works as expected	Passed
Delete inactive member	Member removed from list and database updated	Works as expected	Passed

Diagrams:-



LIBRARY MANAGEMENT SYSTEM

LOGOUT

Dashboard

Books

Add Book

Members

Add Member

Issue/Return

ISBN

1234567890

Title

Flemingo

Author

Robert

Publisher

M.K

Category

Literature

Quantity

ADD BOOK

LIBRARY MANAGEMENT SYSTEM

LOGOUT

Dashboard

Books

Add Book

Members

Add Member

Issue/Return

ADD NEW MEMBER

Member ID (optional)

Auto-generated if left empty

Name

Enter member name

Email

Enter email address

Phone

Enter phone number

Address

Enter address

...

CHAPTER 5 – CONCLUSION AND FUTURE ENHANCEMENTS

5.1 Conclusion

The Library Management System provides an integrated, automated platform for managing all essential library operations efficiently.

By combining the robustness of Spring Boot with the interactivity of JavaFX, the system delivers both performance and usability. It successfully meets the project objectives—automation, accuracy, data security, and user experience.

This application can be effectively deployed in educational and institutional environments, offering a scalable foundation for further enhancements.

5.2 Future Enhancements

1. Integration with barcode or RFID scanning.
2. Cloud synchronization for remote access.
3. Email and SMS notifications for due dates.
4. Analytics dashboard for usage trends.
5. Web and mobile extensions using REST APIs.
6. Support for multi-branch and multilingual systems.
7. AI-based book recommendation engine.

5.3 References

- Oracle Corporation. *Java SE Documentation*.
- Pivotal Software. *Spring Boot Framework Guide*.
- OpenJFX. *JavaFX Official Documentation*.
- H2 Database Engine. *Reference Manual*.
- Apache Software Foundation. *Maven User Guide*.
- Red Hat. *Hibernate ORM Documentation*.
- Martin, R.C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*.
- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*.
- Gamma, E., et al. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*.
- Bloch, J. (2018). *Effective Java (3rd Edition)*.