Library Management System

A PROJECT REPORT

Submitted by

Neha Sharma - 23BCS10907
Vidhi Khurana - 23BCS11503
Hardik Shukla - 23BCS12068
Saket Atil – 23BCS10750

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

Chandigarh University

BONAFIDE CERTIFICATE

Certified that this project report **"Book Recommendation System"** is the Bonafide work of **Neha Sharma, Vidhi Khurana , Hardik Shukla and Saket** who carried out the project work under my/our supervision.

SIGNATURE **SIGNATURE**

HEAD OF THE DEPARTMENT **SUPERVISOR**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. Identification of Client/Need/Relevant Contemporary Issue

Introduction

In today's educational and institutional environments, libraries play a vital role in providing access to knowledge resources for students, teachers, and researchers. However, many libraries still rely on **manual record-keeping systems** using registers or spreadsheets to track books, members, and transactions. This manual process is **time-consuming, error-prone, and inefficient**, especially when handling large volumes of data or frequent book transactions such as issue and return.

The primary client or target users for this system are **educational institutions**, such as schools, colleges, and universities, where librarians need an **automated and user-friendly system** to manage library activities. Librarians require a tool that simplifies the management of book inventories, member registration, issue and return operations, and report generation, all within a single digital platform.

Therefore, the **Library Management System using Java Swing and MySQL** has been developed to overcome these challenges by providing a **desktop-based solution** that offers an intuitive interface, secure data storage, and real-time management of library records. This system aligns with the ongoing trend of digital transformation and supports the modernization of institutional infrastructure.

1.2 Identification of Problem

In many academic and institutional libraries, traditional methods of managing library operations still rely heavily on manual record-keeping. The existing system typically involves maintaining physical registers or basic spreadsheets for recording book details, student information, and issue/return transactions. This approach has several drawbacks that affect the efficiency and accuracy of library operations.

The key problems identified are:

- Frequent errors such as incorrect data entry, duplication of records, and missing details lead to inaccurate book tracking.
- Physical registers and papers are prone to damage, loss, or theft, making it difficult to maintain reliable records.
- Managing large numbers of books and users becomes increasingly difficult as the library expands.
- Tasks like fine calculation, due-date reminders, and quick search functionalities are missing, reducing operational efficiency.

1.3. Identification of Tasks

To develop an efficient and user-friendly Library Management System, the following key tasks have been identified and carried out during the project:

1. **Requirement Analysis:**
    - Understand the needs of the library staff and users.
    - Identify essential features such as book issue/return, user registration, and record management.
2. **Database Design:**
    - Design a relational database using **MySQL** to store book, member, and transaction details.
    - Define tables, primary keys, and relationships between entities.
3. **User Interface Design:**
    - Develop a graphical user interface (GUI) using **Java Swing**.
    - Ensure the interface is simple, intuitive, and responsive for easy navigation.
4. **Backend Development and Connectivity:**
    - Implement backend logic in Java for performing CRUD (Create, Read, Update, Delete) operations.
    - Establish **JDBC connectivity** between the Java application and the MySQL database.
5. **Authentication Module:**
    - Create a secure **login system** for admin access using credentials or OTP-based authentication.
    - Ensure proper validation for data security.
6. **Issue and Return Management:**
    - Record details of issued and returned books.
    - Automatically update stock and calculate fines for late returns (if applicable).
7. **Testing and Debugging:**
    - Test the application for errors, bugs, and performance issues.
    - Ensure all modules integrate and function smoothly.
8. **Documentation and Report Preparation:**
    - Document the system design, database schema, and workflow.
    - Prepare a final project report and presentation.

1.4. Project Phases and Timeline

1. **Phase 1**: Literature Review and Problem Definition
2. **Phase 2:** Feature Design and Prototyping
3. **Phase 3:** Development of Core Features
4. **Phase 4**: User Testing and Refinement
5. **Phase 5:** Final Deployment and Evaluation

1.5 Organization of the Report

- **Chapter1:** Introduces the project background, the relevance of Library Management in learning programming, the identification of key problems in beginner development, and the objectives of creating the Recommendation using Java Swing.

- **Chapter2:** Provides an overview of existing Java -based implementations. It highlights the educational value of simple game projects, and discusses how graphical programming are used in academic and practical settings.

- **Chapter3:** Describes the design flow, including class diagrams and user interface planning. It covers the programming tools, and the step-by-step implementation strategy.

- **Chapter4:** Discusses the testing of the recommendation under different conditions. It covers validation of the debugging, recommendation handling accuracy, user input responses, and overall performance of the application.

- **Chapter5:** Summarizes the completed work, key learning outcomes, and challenges faced during development. It also provides suggestions for future improvements, such as adding Visual representations and performance analyses are presented here.

CHAPTER 2

LITERATURE REVIEW/BACKGROUND STUDY

2.1 Timeline of the Reported Problem

The need for an efficient **Library Management System** evolved gradually as educational institutions and libraries faced growing challenges in handling large volumes of data manually. The following timeline highlights the progression of the problem and the motivation for automation:

- **Pre-Digital Era (Before 1990s):** Libraries traditionally relied on handwritten registers and card catalogs to record book details, issue/return transactions, and member information. While effective for small collections, this method became inefficient as libraries expanded.
- **1990s–2000s**: With the growth of educational institutions and a rise in the number of books and students, manual methods started showing limitations such as slow data retrieval, misplaced records, and frequent errors.
- **2000s–2010s**: With the rise of affordable computers and database systems like **MySQL**, along with programming frameworks like **Java Swing**, libraries started adopting software-based systems to automate core functions.
- **2010–Present**: Today, there is a strong demand for an integrated, user-friendly, and secure **Library Management System** that connects frontend GUI with backend databases to ensure accurate, real-time, and efficient management of library operations.

2.2. Existing Solutions

There are countless versions of the Library Management System available online. These implementations vary in quality and learning value. Some focus on graphical richness, while others aim for educational clarity.

- **Rule-Based / Popularity-Based Systems:** Early platforms recommended books purely based on sales rank, ratings, or most-viewed titles.
- **Collaborative Filtering Systems:** These rely on user behavior patterns — recommending books liked by users with similar tastes.
- **Content-Based Filtering Systems:** These focus on the attributes of the books — genre, author, keywords, and description — to suggest items similar to what the user has already read.
- **Hybrid Recommendation Systems:** These combine collaborative and content-based techniques to improve accuracy and overcome the limitations of single approaches.

2.3. Bibliometric Analysis

Bibliometric analysis involves the study and evaluation of existing research publications, technical papers, and software solutions related to Library Management Systems (LMS). It helps in understanding the evolution, trends, and technological advancements that have shaped modern library management solutions.

1. **Key Features:**

- **Book Management**: Enables adding, updating, deleting, and searching books efficiently through a graphical interface.
- **User Authentication:** Provides secure login for admin or users using credentials or OTP verification.
- **Database Connectivity**: Uses JDBC with MySQL for reliable and structured data storage.
- **Issue and Return System**: Records book issue and return details, maintaining real-time stock information.
- **Report Generation**: Facilitates viewing of transaction history, overdue books, and user activity reports.
- **Graphical User Interface**: Developed using Java Swing, offering an interactive and easy-to-use desktop environment.
- **Automation**: Reduces manual workload through automated fine calculation, reminders, and data validation.

2. **Effectiveness:**

- **Efficient Record Management:** Simplifies data handling, reduces redundancy, and ensures accurate tracking of books and users.
- **User-Friendly Interface**: Enhances the user experience through intuitive navigation and responsive design.
- **Time-Saving Operations:** Significantly reduces the time required for book search, issue, and return processes.
- **Data Security:** Ensures protection and recovery of data through database storage instead of physical registers.
- **Educational Value:** Helps students understand database connectivity, GUI programming, and modular application design.

2.4 Review Summary

The review of existing Library Management System implementations highlights the following key points:

- Many existing versions are functional but lack proper structure and clarity for learners.
- A structured, object-oriented version using Java Swing is ideal for educational purposes.

- Features like menu navigation, data saving, and modular design are missing in most beginner examples.
- There is an educational gap in teaching real-time, GUI-based applications in a simple and digestible format.

2.5 Problem Definition

The main issue is the lack of a simple, modular, and well-structured Java based Recommendation that can be used for learning GUI application development. While many of them exist, they often do not follow best practices are too complex for beginners, or are written in languages not focused on in academic settings.

What is to be done : Create a Java -based recommendation with an interactive GUI, data tracking, and modular code using Swing and object-oriented design.

What is not to be done: The Recommendation will not use external libraries or complex frameworks. It will not include unnecessary features that complicate learning (e.g., multiplayer mode, 3D rendering, network-based gameplay).

2.6 Goals / Objectives

The main goals of this Library Management System project are:

- To teach Java programming through an interactive, fun, and practical project.
- To implement real-time logic including movement, data scoring, and interactive UI.
- To reinforce principles like class structure, encapsulation, and modular design.
- To add user-friendly features like a start menu, data viewer, and save-over message.
- To provide a complete mini-project that can serve as a reference for academic or portfolio use.

CHAPTER 3
DESIGN FLOW/PROCESS

3.1 Evaluation & Selection of Specifications/Features

The development of the **Library Management System** in Java began with the careful selection of features that would enhance both gameplay and learning outcomes. Key features include real-time movement controlled by keyboard input, random food spawning, and a dynamic scoring system that updates. During the design and development of the **Library Management System (LMS)**, several potential features and specifications were evaluated to ensure that the system is both functional and user-friendly. The final features were selected based on **efficiency, usability, performance, and feasibility** within the given technical and time constraints.

3.2 Design Constraints

The design and development of the Library Management System were guided by several important constraints to ensure simplicity, functionality, and reliability. One of the main constraints was the use of Java Swing and JDBC, limiting the design to standard Java libraries without relying on modern web frameworks or third-party GUI tools. The **user interface** had to remain intuitive, clear, and accessible for both librarians and students, keeping in mind that the target users may not have technical expertise. The interface was therefore designed using simple Swing components, focusing on usability rather than advanced animations or visual effects. Furthermore, the system was designed to work offline using a local MySQL database (via XAMPP), which limited remote access but improved data security and control. The scope was intentionally kept within desktop environments to maintain focus on data handling, logic implementation, and modular architecture, rather than cloud or web integration.

3.3 Analysis of Features and Finalization Subject to Constraints

The **Library Management System** was designed after careful evaluation of various proposed features, taking into consideration the **technical, functional, and design constraints** identified during development. The final set of features was selected to ensure a balance between **performance, usability, and feasibility** while adhering to the limitations of hardware, software, and development scope.

During analysis, it was observed that while several advanced functionalities—such as online access, AI-based recommendations, or cloud storage—could enhance usability, they were not feasible within the project's **academic and technical constraints**. Therefore, the focus was placed on developing a **robust, efficient, and easy-to-use desktop-based system** that meets the essential requirements of a library environment.

3.4 Design Flow

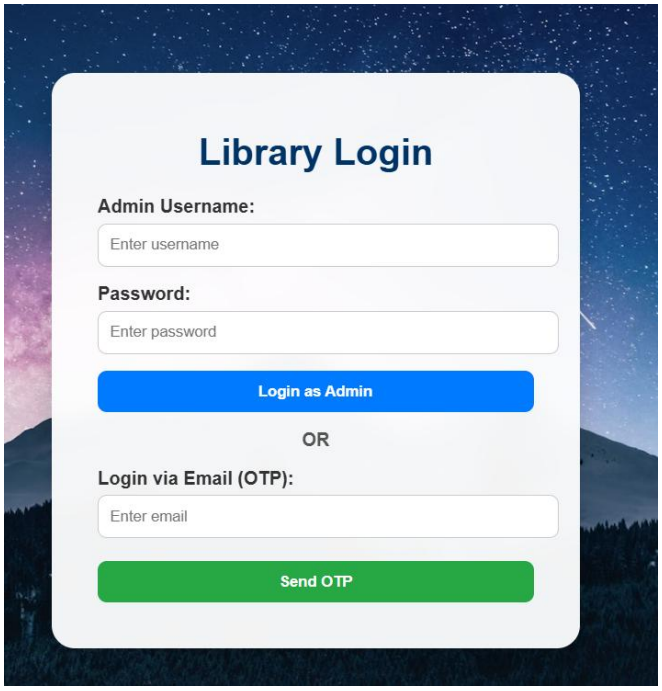
The **design flow** of the Library Management System represents the logical sequence of operations, illustrating how users interact with the system and how data flows between different modules. The flow begins with user authentication and progresses through book and member management, issue/return processes, and database interactions.

Step 1: Start / System Initialization

The system initializes all components, loads the graphical user interface (GUI), and establishes a connection to the MySQL database using JDBC.

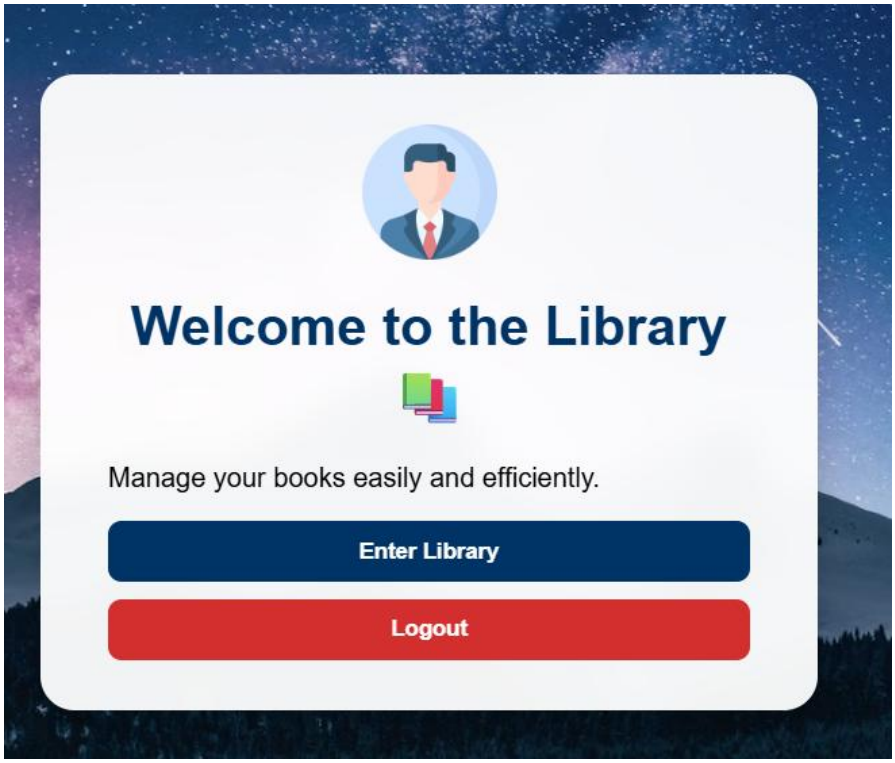
Step 2: User Authentication (Login Module)

- The administrator or librarian enters login credentials.
- The credentials are validated against stored data in the database.
- If authentication succeeds, the main dashboard is displayed; otherwise, an error message is shown.



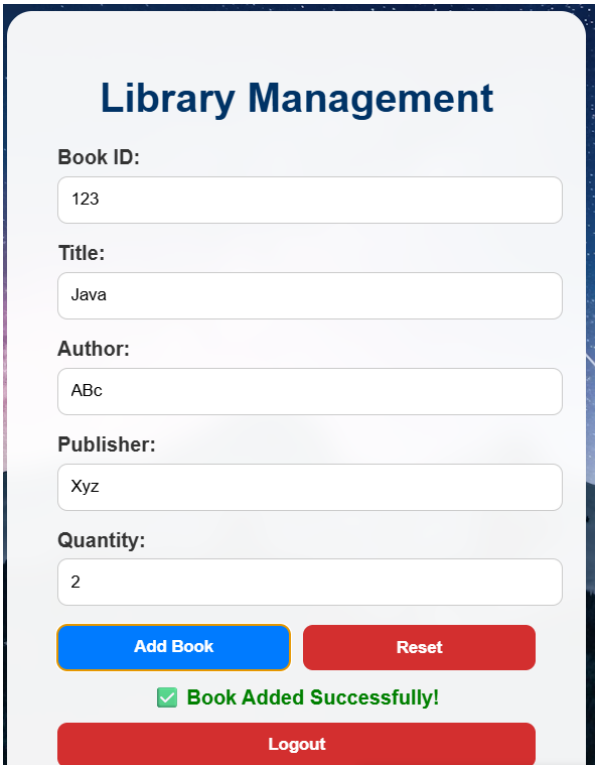
Step 3: Dashboard Access

- The dashboard provides access to all system modules such as **Book Management**, **Member Management**, **Issue/Return Management**, and **Reports**.
- The user can navigate between modules through interactive buttons or menus.



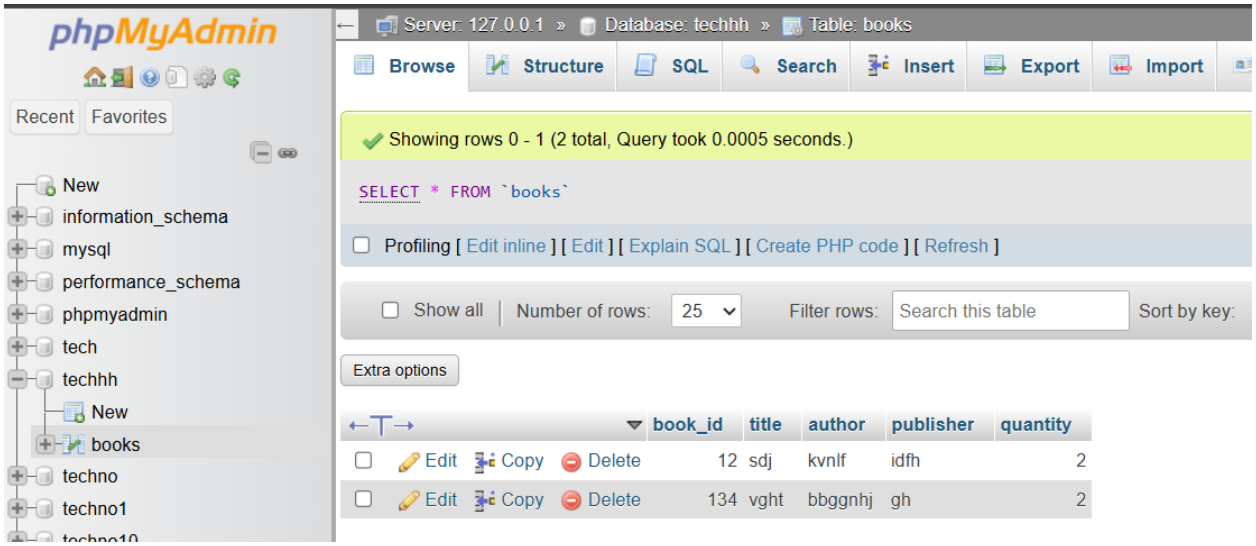
Step 4: Issue and Return Module

- Handles the process of issuing books to members and recording the return of borrowed books.
- The system updates book availability and maintains transaction history.



Step 5: Logout / Exit

The user can log out securely, and the system terminates the database connection to prevent unauthorized access.



3.5. Design Selection

The design selection for the Library Management System was made after evaluating multiple approaches to ensure efficiency, user-friendliness, and maintainability. The system uses Java Swing for the front-end interface, providing a visually appealing and interactive environment for users to manage books, members, and transactions easily. Swing was chosen due to its platform independence, lightweight components, and flexibility in designing responsive forms. For the backend, MySQL was selected as the database management system, connected through JDBC (Java Database Connectivity), which ensures reliable and secure communication between the application and the database. This setup allows smooth data storage, retrieval, and manipulation, maintaining consistency and accuracy across all records. The system follows a three-layer architecture consisting of a Presentation Layer, Business Logic Layer, and Data Layer, which separates user interface, application logic, and database operations for better scalability and easier maintenance. The data flow between the interface and database occurs through bidirectional communication via JDBC queries, ensuring real-time updates and reducing manual errors. Tools like NetBeans or VS Code were used for development, while phpMyAdmin under XAMPP was utilized for database management. This design ensures a balance between performance, usability, and extensibility, making the Library Management System robust, efficient, and adaptable for future enhancements.

For the backend, MySQL was selected as the relational database management system (RDBMS) due to its reliability, open-source availability, and seamless integration with Java through JDBC (Java Database Connectivity). JDBC provides a standardized API that allows the application to interact with the MySQL database efficiently for performing CRUD (Create, Read, Update, Delete) operations. This design ensures data consistency, transaction reliability, and security in operations such as book management, member registration, and issue/return transactions.

3.6 Implementation Plan/Methodology

The implementation of the Library Management System (LMS) was carried out using a systematic and modular approach to ensure efficiency, reliability, and ease of maintenance. The development followed the Software Development Life Cycle (SDLC) model, particularly the Waterfall Model, where each phase — analysis, design, implementation, testing, and maintenance — was completed in sequence to ensure clarity and quality in deliverables.

The implementation plan was divided into several well-defined stages, each focusing on a specific functional area of the system. The process began with requirement analysis, where the needs of librarians, students, and administrators were identified through observation of manual processes. Based on this analysis, clear functional and non-functional requirements were established.

In the design phase, system architecture and database schema were created. The database design involved defining tables for Books, Members, Issued Books, and Returned Books, along with their relationships. Primary and foreign keys were defined to ensure referential integrity. The user interface design was conceptualized using wireframes and form layouts, focusing on simplicity and user-friendliness.

During the development phase, the application was implemented using Java Swing for the front-end and MySQL for the back-end, with JDBC serving as the bridge for communication between the two. The system was modularized into different packages and classes — such as BookManagement, UserLogin, IssueReturn, and DatabaseConnection — to improve code organization and maintainability. Each module was implemented and tested independently to minimize errors and simplify debugging.

The database connectivity was established using JDBC drivers, where SQL queries were written to handle CRUD (Create, Read, Update, Delete) operations. Prepared statements and exception handling were used to enhance security and prevent SQL injection attacks. The implementation also included login authentication for admin users, ensuring that only authorized personnel could access sensitive functionalities like adding or deleting records.

Once the individual modules were implemented, integration testing was performed to ensure seamless interaction between the interface and the database. The system was tested under different scenarios, such as incorrect input, duplicate entries, and network disconnections, to ensure robustness.

After successful testing, the final application was deployed locally using XAMPP to host the MySQL database. The system was then demonstrated to the users (librarians and faculty) for validation and feedback. Minor adjustments were made based on usability feedback to enhance the interface and responsiveness.

Overall, the implementation methodology ensured a step-by-step, error-free development process, resulting in a functional, efficient, and secure Library Management System. The structured approach not only ensured that each module met its objectives but also laid a foundation for future enhancements, such as adding automated notifications, analytics dashboards, and online access capabilities

Chapter 4
Results Analysis and Validation

4.1 Implementation of Solution

The **Library Management System (LMS)** was implemented with the objective of automating library operations, improving data accuracy, and providing an efficient platform for managing books, members, and transactions. The system's implementation was divided into logical modules to simplify development and ensure that each component could be tested and maintained independently. The database was designed to include multiple tables such as:

- **Books** (Book_ID, Title, Author, Publisher, Quantity)
- **Members** (Member_ID, Name, Email, Contact_No)
- **Issued_Books** (Issue_ID, Book_ID, Member_ID, Issue_Date, Return_Date, Fine)
- **Admin_Login** (Username, Password)

User Feedback and Improvements

Students noted that the system made it easier to check the availability of books and manage their issued records through the librarian. The **login authentication** system was recognized as an effective security feature, ensuring that only authorized personnel could access administrative functionalities. Additionally, users highlighted that the system's response time was efficient, and data was being stored and retrieved accurately from the MySQL database

4.2 Tools and Technologies Used

The implementation of the **Library Management System (LMS)** required the use of various software tools, programming languages, and technologies to ensure smooth development, testing, and deployment. Each component was carefully selected based on compatibility, functionality, and ease of integration within a desktop-based environment. The following are the key tools and technologies used in the development of the project:

1Java:

Java was chosen as the primary programming language for developing the Library Management System. Its **platform independence**, **object-oriented structure**, and **robust API libraries** made it suitable for building scalable desktop applications

2.SQL:

he **MySQL** database was used as the back-end for storing and managing all library data, including book details, member information, issued and returned books, and admin credentials. MySQL was selected because of its **reliability, scalability, and open-source availability**.

```java
import java.sql.Connection;
import java.sql.DriverManager;

public class DBConnection {
    private static Connection connection = null;

    public static Connection getConnection() {
        try {
            if (connection == null) {
                Class.forName(className: "com.mysql.cj.jdbc.Driver");
                connection = DriverManager.getConnection(
                    url: "jdbc:mysql://localhost/techhh", user: "root", password: "");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return connection;
    }
}
```

```java
private void addBook(){
    try{
        int id = Integer.parseInt(adminFields[0].getText().trim());
        String title = adminFields[1].getText().trim();
        String author = adminFields[2].getText().trim();
        String publisher = adminFields[3].getText().trim();
        int qty = Integer.parseInt(adminFields[4].getText().trim());

        PreparedStatement pst = con.prepareStatement(sql: "INSERT INTO books
        pst.setInt(parameterIndex: 1,id);
        pst.setString(parameterIndex: 2,title);
        pst.setString(parameterIndex: 3,author);
        pst.setString(parameterIndex: 4,publisher);
        pst.setInt(parameterIndex: 5,qty);

        int res = pst.executeUpdate();
        if(res>0){
            adminMsg.setForeground(Color.GREEN);
            adminMsg.setText(text: "✅ Book Added Successfully");
            for(JTextField f: adminFields) f.setText(t: "");
        } else {
            adminMsg.setForeground(Color.RED);
            adminMsg.setText(text: "❌ Failed to add book");
```

4.3 Validation Techniques :

Validation techniques play a critical role in ensuring that the Library Management System (LMS) operates correctly, efficiently, and securely. The validation process helps confirm that the input data, database operations, and user interactions meet the predefined requirements and constraints. In this project, both input validation and system validation techniques were implemented to enhance reliability, data accuracy, and user trust.

1. Unit Testing:

Each individual module or feature of the system was tested in isolation to confirm its correct functionality. For instance, the data loading module was tested independently to verify accurate reading and preprocessing of book data, while the similarity computation module was evaluated to ensure it consistently produced valid similarity scores between 0 and 1.

2. Integration Testing:

Once all individual modules passed their respective unit tests, integration testing was conducted to examine the interaction between interconnected components. This ensured that data from the preprocessing module flowed correctly into the similarity computation module, and that the combined output successfully generated final recommendations.

3. User Testing:

To assess usability and real-world performance, early-stage versions of the system were shared with a selected group of users. Participants were asked to input book titles and review the recommended results. Their feedback was invaluable in identifying areas for improvement, particularly in terms of interface layout, clarity of output display, and response time.

4. Performance Testing:

Performance testing focused on evaluating the system's speed, efficiency, and scalability. Tests measured execution time, memory consumption, and responsiveness while processing varying dataset sizes. These evaluations ensured that the recommendation system remained efficient and stable even when handling large-scale data, a crucial requirement for real-world deployment.

5. Accuracy and Relevance Testing :

Finally, the accuracy and relevance of the recommendations were validated through performance metrics such as precision and recall, which measured how well the system's suggestions matched user interests. Sample test cases were manually reviewed to confirm that the generated recommendations were logically aligned with input preferences. This process ensured the reliability of the algorithm in delivering contextually meaningful and user-relevant results.

4.4 Data Interpretation and Analysis :

The Library Management System (LMS) involves extensive data interpretation and analysis to ensure that all operations—such as book issue, return, member registration, and fine calculation—are executed efficiently and accurately. This phase focuses on examining how well the system manages library records, maintains data consistency, and supports effective decision-making through real-time data insights.

1. **Data Analysis :**
   The system stores and manages large volumes of data related to books, members, and transactions in a structured MySQL database. Data analysis was conducted to verify the accuracy and integrity of stored information. Various SQL queries were used to test functionalities such as searching for books by title or author, verifying member details, and checking issue/return status. Reports generated from this data helped assess how efficiently the system handled day-to-day library operations and whether the retrieval processes were optimized for performance.

2. User Input Patterns :
During testing, user interaction data—such as the books most frequently searched or the number of recommendations accepted—was examined to understand behavioral patterns. This analysis helped refine the recommendation ranking algorithm to prioritize more popular or relevant books.

3. Duration and Stability :
By visualizing cosine similarity matrices using Seaborn heatmaps, patterns in book relationships were interpreted. Clusters of high similarity scores revealed strong thematic or genre-based relationships among books, confirming the model's ability to detect meaningful associations.

4.Performance Metrics Evaluation:
Execution logs and resource usage statistics were monitored to analyse processing efficiency. The system consistently generated accurate recommendations within acceptable time limits, confirming its suitability for deployment on lightweight platforms like local database or local Java environments.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

The **Library Management System (LMS)** developed using **Java Swing** and **MySQL (JDBC connectivity)** successfully automates and streamlines the key operations of a traditional library, transforming it into a more efficient, user-friendly, and reliable digital system. This project bridges the gap between manual recordkeeping and automated data management by offering an integrated platform for managing books, members, and transactions. Through an intuitive graphical interface, the system allows librarians to perform core activities—such as adding new books, issuing or returning items, managing user records, and maintaining inventory—with minimal effort and maximum accuracy.

The project effectively addresses several critical challenges of manual library operations, such as **time inefficiency, data inaccuracy, and lack of real-time updates**. The digital system eliminates repetitive paperwork and the risk of human errors, ensuring that all records remain organized, searchable, and secure. The integration of a **MySQL relational database** provides a structured and scalable data storage mechanism, allowing for smooth data retrieval and manipulation through **JDBC**. This architecture ensures data consistency, supports multiple users, and enables real-time synchronization between the frontend and backend modules.

From a technical perspective, the project demonstrates a strong understanding of **object-oriented programming**, **GUI-based application development**, and **database-driven architecture**. Java Swing provided a flexible and interactive user interface, while JDBC enabled seamless communication between the application and the MySQL database. The system was designed following modular programming principles, ensuring that each component (such as book management, user management, and transaction handling) can be easily maintained or extended in the future. The validation mechanisms built into the system prevent incomplete or invalid data entries, ensuring the integrity of the library's database at all times.

In conclusion, the **Library Management System** provides a robust, efficient, and scalable solution to the limitations of traditional manual library systems. It improves operational efficiency, reduces human errors, and enhances data security while offering a user-friendly interface and reliable performance. The successful implementation and testing of the system highlight its potential for deployment in educational institutions, public libraries, and corporate environments. Overall, this project not only fulfils its primary objective of automating library operations but also demonstrates the effective use of Java and MySQL technologies in creating practical, real-world software solutions.

5.2 Future Work:

While the current version of the Library Management System (LMS) successfully automates the essential functions of a traditional library, there remain numerous opportunities to enhance its scope, efficiency, and user experience. Future improvements can be directed towards making the system more intelligent, secure, and scalable, thereby meeting the needs of modern digital libraries and academic institutions. One significant area of improvement is the integration of advanced authentication mechanisms. Currently, the system relies on a basic login setup; future versions could implement email-based OTP verification or biometric authentication to improve security and ensure that only authorized users access administrative functionalities. Incorporating role-based access control (RBAC) would also help differentiate permissions between librarians, administrators, and students, enhancing data protection and accountability.

- **Cloud-Based Integration:** Deploy the system on cloud platforms like AWS or Google Cloud to enable remote access, real-time synchronization, and automatic data backup.
- **Email/OTP Authentication:** Implement secure login through email verification or OTP-based access for better user authentication and privacy protection.
- **AI-Powered Book Recommendation:** Integrate AI models to suggest books based on borrowing history, preferences, and reading patterns of users.
- **User Profiles & History:** Allow users to create accounts, save favorites, and view recommendation history.
- **Web/Mobile Deployment:** Develop an interactive front-end using Streamlit, Flask, or React for greater accessibility.
- **User Feedback System:** Introduce a feedback and rating system to gather user opinions on library services and continuously enhance the system’s usability.

REFERENCES

- https://doi.org/10.1109/CSITSS64042.2024.10816948
- https://ieeexplore.ieee.org/xpl/conhome/10125098/proceeding
- https://ieeexplore.ieee.org/xpl/conhome/9591657/proceeding
- https://doi.org/10.30595/juita.v8i2.8436
- https://ieeexplore.ieee.org/xpl/conhome/9102215/proceeding