**Library Management Systems**

**A MINI-PROJECT REPORT**

**Submitted by**

**Preethi G – 231501122 & Princess Darlene V – 231501123**

**In partial fulfillment of the award of the degree**

**of**

**BACHELOR OF TECHNOLOGY**

**IN**

**ARTIFICIAL INTELLIGENCE AND**

**MACHINE LEARNING**



**RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI**

**An Autonomous Institute**

**CHENNAI**

**NOVEMBER 2024**

BONAFIDE CERTIFICATE

Certified that this project report “**LIBRARY MANAGEMENT SYSTEM**” is the bonafide work of “Preethi G(231501122) and Princess Darlene(231501123)” who carried out the project work under my supervision.

**Submitted for the Practical Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SIGNATURE**

**Mr. U. Kumaran,**

**Assistant Professor (SS)**

**AIML,**

**Rajalakshmi Engineering College,   
 (Autonomous),**

**Thandalam, Chennai - 602 105**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# TABLEOFCONTENTS

1.INTRODUCTION

* 1.1 INTRODUCTION
* 1.2 IMPLEMENTATION
* 1.3 SCOPE OF THE PROJECT

1. SYSTEM SPECIFICATION
   * 2.1 HARDWARE SPECIFICATION
   * 2.2 SOFTWARE SPECIFICATION
2. SAMPLE CODE
   * 3.1 LOGIN
   * 3.2 DASHBOARD
   * 3.3 AVAILABLE BOOKS
   * 3.4 ADD BOOKS
   * 3.5 DELETE BOOKS
   * 3.6 AVAILABLE STAFF
   * 3.7 ADD STAFFS
   * 3.8 DELETE STAFFS
3. SNAPSHOTS
   * 4.1 HOME PAGE
   * 4.2 BOOK DETAILS
   * 4.3 ADD BOOKS
   * 4.4 REMOVE BOOKS
   * 4.5 STAFF DETAILS
   * 4.6 ADD STAFF
   * 4.7 REMOVE STAFFS
   * 4.8 EDIT ADMIN
   * 4.9 LOGIN PAGE
4. CONCLUSION
5. REFERENCES

## ABSTRACT

The Library Management System in JAVA is an advanced and

efficient solution designed to streamline library operations by providing a secure, user-friendly, and organized platform for managing library resources. This software system aims to enhance the overall library experience, ensuring efficient management of books, members, and inventory while simplifying administrative tasks.

The Library Management System in JAVA is a powerful tool for modernizing how libraries handle resources, increase operational efficiency, and reduce the administrative burden associated with traditional library management. It addresses the challenges of manual record-keeping and improves the accessibility of library services while maintaining the integrity and accuracy of data.

This system is not just limited to traditional libraries; it can be customized to accommodate various library types, such as educational institutions, corporate libraries, or specialized libraries. With its strong focus on inventory management, user experience, and ease of use, this system significantly enhances the way libraries function, making them more efficient and accessible to all users.

1.1 INTRODUCTION

The **Library Management System (LMS) in Java** effectively addresses several common challenges faced by libraries, offering a comprehensive, user-friendly, and secure solution. Here’s how it tackles key issues while enhancing the overall experience for both librarians and library members:

### 1. **Automation of Library Tasks**

* **Book Cataloging:** Librarians can automate the process of managing book records, including adding, updating, and deleting books. Instead of relying on manual entry or paper records, the system allows them to efficiently catalog books with key details (e.g., title, author, ISBN, genre, and availability). This automation reduces errors, improves accuracy, and ensures that the catalog is always up to date.
* **Member Registration:** New users can register for library membership through an easy-to-use interface. Once registered, members are automatically assigned a unique user ID and can log in to their accounts to manage their borrowing activities. This automated process eliminates the need for manual records and speeds up the onboarding process.
* **Overdue Fine Calculations:** The system automatically calculates fines for overdue books based on predefined rules (e.g., a fixed fine per day). It eliminates the manual calculation of overdue fees, saving librarians valuable time and reducing the risk of errors. The system also tracks payments and generates reports on fine status.

### 2. **Efficient Search and Book Reservation**

* **Search Functionality:** Users can quickly search for books using various filters such as title, author, genre, ISBN, or availability. The system’s powerful search engine ensures that members can find exactly what they are looking for without wading through lengthy lists. Advanced filters help users narrow down their search results, saving time and effort.
* **Check Availability:** Members can check if a book is available for borrowing or if it is currently on loan. If a book is unavailable, users can see the expected return date and plan accordingly. This feature ensures that members are always aware of a book's status and helps manage expectations.
* **Book Reservation:** If a book is currently borrowed by another user, the system can offer an option for members to reserve the book. Once the book is returned, the member who reserved it is notified, allowing them to borrow it immediately. This feature streamlines book borrowing and ensures that popular titles are accessible to users in an orderly and fair manner.

### 3. **User-Friendly Interface**

* **Member Dashboard:** The member interface is intuitive and simple to navigate. Users can view their borrowed books, check due dates, search the catalog, and manage their reservations. They can also view any fines associated with overdue books and make payments, all within the same dashboard.
* **Librarian Dashboard:** Librarians have access to a dedicated interface where they can manage book records, track borrowed books, monitor overdue items, and manage user accounts. The dashboard provides an overview of the library’s operations, including reports on books that are frequently borrowed, overdue books, and current inventory.
* **Responsive Design:** The user interface (UI) is designed to be clean, responsive, and easy to use. Whether on a desktop or a tablet, the design adapts to the screen size, providing a smooth user experience.

### 4. **Security and Data Integrity**

* **User Authentication and Authorization:** The system ensures that only authorized users (librarians and registered members) can access certain features. Members log in with unique usernames and passwords, and roles (admin, librarian, member) are clearly defined with role-based access controls (RBAC). This prevents unauthorized access to sensitive data and ensures that each user can only access the features relevant to their role.
* **Data Encryption:** Passwords and sensitive user data are securely stored using encryption techniques (e.g., hashing passwords). This ensures the integrity and confidentiality of user information. Additionally, secure communication protocols (e.g., SSL/TLS) are used for transmitting data between the application and the MySQL database, reducing the risk of data breaches.

### 5. **Streamlined Borrowing and Returning Process**

* **Book Borrowing and Returning:** The system allows members to borrow books and automatically updates the book’s availability. When a book is borrowed, the system records the due date and calculates any potential overdue fines. Returning a book is a simple process, where the system updates the book’s availability and tracks the return date.
* **Overdue Management:** Overdue books are automatically flagged, and the system sends notifications to both members and librarians about overdue items. This helps prevent books from being forgotten or lost, ensuring that library inventory is managed efficiently. Additionally, overdue fines are automatically applied, and members can view their fine status in real time.

### 6. **Real-Time Notifications**

* **Overdue Alerts:** Both members and librarians receive automated reminders when books are overdue. Members are notified of the fines they need to pay, helping to ensure that overdue items are returned on time.
* **Reservation Notifications:** Members who have reserved books are automatically notified when the book becomes available for pick-up. This ensures smooth transitions between borrowers and helps prevent long waiting times for popular titles.

### 7. **Reports and Analytics for Librarians**

* **Transaction Reports:** The system provides librarians with detailed transaction reports, including records of book borrowings, returns, and overdue items. These reports offer insights into library activity and usage patterns.
* **Inventory Reports:** Librarians can generate reports on the current status of the book inventory. These reports can include details such as which books are most frequently borrowed, which titles need to be reordered, or which books are no longer in circulation.
* **Fine Reports:** The system generates fine reports to help librarians track overdue fines, payments, and outstanding balances. This makes financial tracking easier and ensures transparency.

### 8. **Improved Efficiency and Reduced Workload**

* **Time Savings:** Automation of repetitive tasks, such as cataloging books, calculating fines, and updating availability, saves significant time for librarians. With fewer manual tasks to perform, librarians can focus on other activities, such as improving library services, assisting members, and managing the library’s overall operations.
* **Error Reduction:** Automating tasks like fine calculation and book tracking reduces the risk of human error, ensuring more accurate data and better management of library resources.

### 9. **Scalability and Flexibility**

* **Scalable System:** The LMS is designed to scale, making it suitable for libraries of all sizes—whether a small local library or a large university library. As the library grows, the system can accommodate more books, users, and transactions without performance degradation.
* **Customization:** The system is flexible enough to be customized or extended with additional features as needed. For example, new reporting capabilities, advanced search features, or integration with other systems can be added without disrupting existing operations.

1.2 IMPLEMENTATION

The **Library Management Systems** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

1.3 SCOPE OF THE PROJECT

The scope of the **Library Management System (LMS) in Java using MySQL** is quite comprehensive and involves several key functionalities to efficiently manage the entire lifecycle of books and user interactions within the library. Here's a more detailed breakdown of the system's scope, features, and operations:

### **User Management**

* **User Registration:** Allows new members (library users) to create accounts with relevant details such as name, contact information, and membership type (e.g., student, faculty, or general member).
* **Authentication:** Provides secure login for both library members and librarians. Users are authenticated based on usernames and passwords stored in the MySQL database.
* **Role-Based Access Control (RBAC):** Different user roles (e.g., member, librarian) with distinct access privileges:
  + **Members:** Can borrow, return books, view available books, and search the catalog.
  + **Librarians:** Can add, update, and delete books, manage member accounts, track overdue books, and monitor borrowing records.

### 2. **Book Management**

* **Add Books:** Librarians can add new books to the catalog with necessary details such as title, author, genre, ISBN, publisher, and available quantity.
* **Update Books:** The ability to update book details, including modifying existing records, changing availability status, and adjusting book quantities (e.g., after receiving new stock).
* **Delete Books:** Librarians can remove books from the catalog if they are no longer in circulation or are outdated.
* **Search and Filter Books:** Users can search for books based on title, author, genre, or ISBN. Filters enable efficient search for available books, specific authors, or genres.

### 3. **Book Borrowing & Returning**

* **Borrowing Books:** Members can borrow books, subject to availability. The system records the date of borrowing and tracks the due date for returns.
* **Returning Books:** Members return borrowed books, and the system updates the status of the books and logs the return date.
* **Overdue Tracking:** The system automatically tracks overdue books based on the due date and notifies both members and librarians of overdue items. Late returns may incur fines.
* **Book Availability:** The system dynamically updates the availability of books after they are borrowed or returned, ensuring that members can only borrow books that are currently available.

### 4. **Overdue Management**

* **Fines for Late Returns:** The system can calculate overdue fines based on the number of overdue days and apply the fine to the member's account.
* **Notifications:** Sends alerts to members regarding overdue books, providing them with the due date and any associated fines.

### 5. **Library Catalog and Search Functionality**

* **Search Interface:** A user-friendly search interface for browsing books based on various criteria like title, author, genre, and availability.
* **Advanced Filters:** Filters to help users narrow down their search, for example, books that are currently available, or books of a specific genre or author.
* **Book Details Page:** For each book, a detailed page displays key information (author, publisher, year of publication, genre, availability, and a brief description).

### 6. **Database Integration (MySQL)**

* **Book Database:** A MySQL database is used to store and manage all book-related data, such as title, author, genre, publication details, and status (available/borrowed).
* **User Database:** Stores user details, including login credentials, membership type, borrowing history, and current borrowing status.
* **Transaction Management:** Every borrowing and returning action is logged as a transaction in the database, which can be tracked for future reference.
* **Overdue and Fine Tracking:** Keeps a record of overdue books, fines, and payment status.

### 7. **Admin and Librarian Functions**

* **Book Management (Add, Edit, Delete):** Librarians can perform CRUD operations (Create, Read, Update, Delete) on book records.
* **Member Management:** Librarians can view and manage member accounts, including activating/deactivating membership or adjusting member information.
* **Borrowing History:** Librarians can view borrowing history of all members, including current and past transactions.
* **Overdue Management:** Librarians can monitor overdue books and apply fines accordingly.

### 8. **User Interface (UI)**

* **Member Dashboard:** A member's interface for viewing their borrowed books, search results, and overdue status.
* **Librarian Dashboard:** An interface for librarians to manage books, track borrowed/returned items, and manage user accounts.
* **Responsive UI:** The system should have a user-friendly, intuitive design, possibly using JavaFX for the desktop application, to enhance the experience for both librarians and members.

### 9. **Reporting and Analytics**

* **Transaction Reports:** Provides reports on book borrowing and returning activity, including due dates and overdue items.
* **Fine Reports:** Generates reports related to overdue fines, the total amount owed by users, and fine payments.
* **Inventory Reports:** Displays current stock levels of books, newly added books, and books that are frequently borrowed.

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR : Intel i5

MEMORY SIZE : 4GB(Minimum)

HARD DISK : 500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

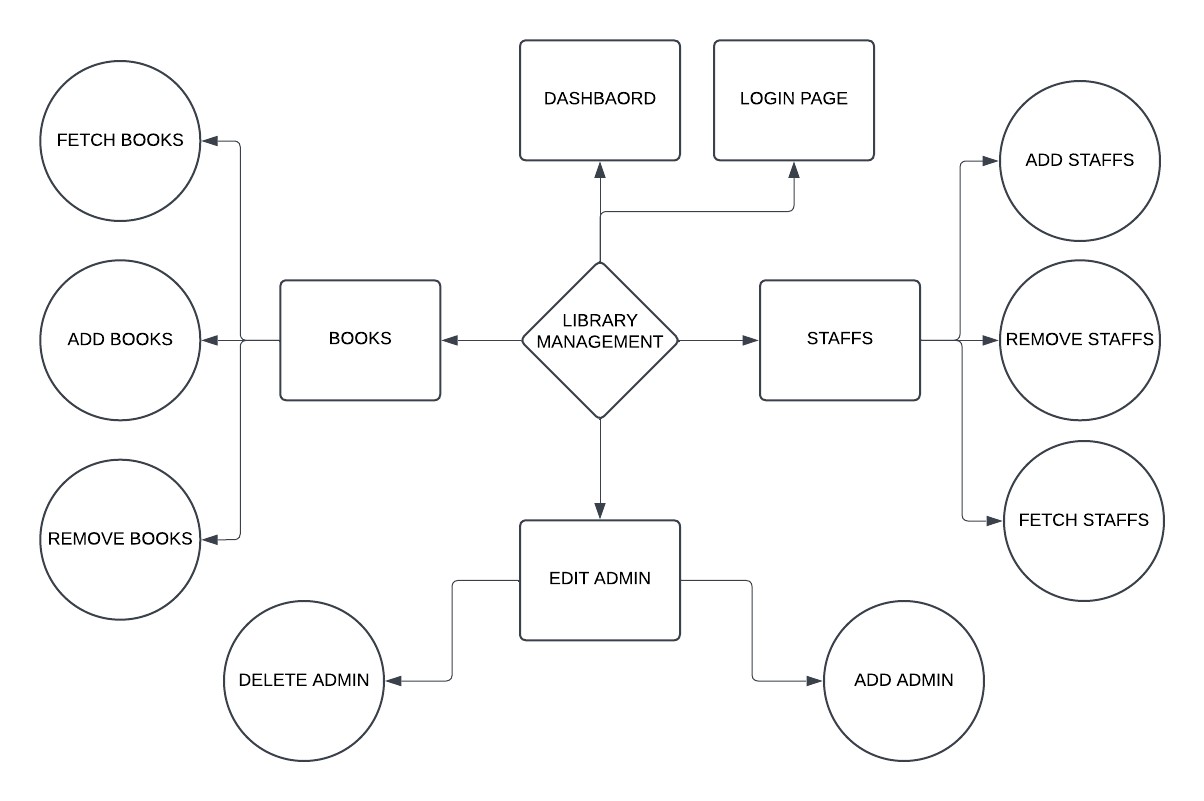
PROGRAMMING LANGUAGE : Java, MySQL

FRONT-END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 11

ER DIAGRAM



## Source code

1) LOGIN

import java.sql.\*; import javax.swing.\*;

public class LoginPage extends javax.swing.JFrame { public LoginPage() { initComponents(); login.addActionListener(new java.awt.event.ActionListener() { public void actionPerformed(java.awt.event.ActionEvent evt) { jButton1ActionPerformed(evt);

}

});

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

String url = "jdbc:mysql://localhost:3306/library";

String mysqluser = "root";

String mysqlpwd = "Taphasvi@266";

String pswrd = new String(password.getPassword());

String username = user.getText();

String query = "SELECT PASSWORD FROM ADMIN WHERE USER\_ID = '" +

username + "';"; try {

Connection conn = DriverManager.getConnection(url, mysqluser, mysqlpwd);

Statement stm = conn.createStatement();

ResultSet rs = stm.executeQuery(query);

if (rs.next()) {

String realpswrd = rs.getString("PASSWORD"); if (realpswrd.equals(pswrd)) {

Dashboard dsh = new Dashboard();

dsh.setVisible(true); this.dispose();

} else {

JOptionPane.showMessageDialog(this, "Username or Password is incorrect!");

}

} else {

JOptionPane.showMessageDialog(this, "Wrong Username!");

}

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e.getMessage());

} }

public static void main(String args[]) { java.awt.EventQueue.invokeLater(new Runnable() { public void run() { new LoginPage().setVisible(true);

}

});

}

}

2) DASHBOARD

import javax.swing.\*;

public class Dashboard extends javax.swing.JFrame {

public Dashboard() { initComponents(); setDefaultCloseOperation(Dashboard.DISPOSE\_ON\_CLOSE);

}

private void initComponents() { add = new javax.swing.JButton(); update = new javax.swing.JButton(); delete = new javax.swing.JButton(); search = new javax.swing.JButton(); logout = new javax.swing.JButton(); home = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE); getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

add.setText("Add Books"); add.addActionListener(evt -> addActionPerformed(evt));

getContentPane().add(add, new org.netbeans.lib.awtextra.AbsoluteConstraints(250,

310, 280, 110));

update.setText("Update Books"); update.addActionListener(evt -> updateActionPerformed(evt));

getContentPane().add(update, new

org.netbeans.lib.awtextra.AbsoluteConstraints(250, 460, 280, 110));

delete.setText("Delete Books"); delete.addActionListener(evt -> deleteActionPerformed(evt));

getContentPane().add(delete, new org.netbeans.lib.awtextra.AbsoluteConstraints(730,

310, 280, 110));

search.setText("Search Books"); search.addActionListener(evt -> searchActionPerformed(evt));

getContentPane().add(search, new

org.netbeans.lib.awtextra.AbsoluteConstraints(730, 460, 280, 110));

logout.setText("Logout"); logout.addActionListener(evt -> logoutActionPerformed(evt));

getContentPane().add(logout, new

org.netbeans.lib.awtextra.AbsoluteConstraints(960, 40, 140, 60));

home.setFont(new java.awt.Font("Georgia", 1, 48)); home.setHorizontalAlignment(javax.swing.SwingConstants.CENTER); home.setText("DASHBOARD");

getContentPane().add(home, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 20, 1160, 120));

pack(); }

private void addActionPerformed(java.awt.event.ActionEvent evt) { new AddBooks().setVisible(true);

}

private void updateActionPerformed(java.awt.event.ActionEvent evt) {

// Code to open Update Books frame

}

private void deleteActionPerformed(java.awt.event.ActionEvent evt) {

// Code to open Delete Books frame

}

private void searchActionPerformed(java.awt.event.ActionEvent evt) {

// Code to open Search Books frame

}

private void logoutActionPerformed(java.awt.event.ActionEvent evt) { this.dispose();

}

public static void main(String args[]) { java.awt.EventQueue.invokeLater(() -> new Dashboard().setVisible(true));

}

private javax.swing.JButton add; private javax.swing.JButton update; private javax.swing.JButton delete; private javax.swing.JButton search; private javax.swing.JButton logout; private javax.swing.JLabel home;

}

3) AVAILABLE BOOKS

import java.sql.\*; import javax.swing.\*; import javax.swing.table.DefaultTableModel;

public class BooksAvailable extends javax.swing.JFrame { public BooksAvailable() { initComponents(); setDefaultCloseOperation(BooksAvailable.DISPOSE\_ON\_CLOSE);

}

private void initComponents() { jTable2 = new javax.swing.JTable(); fetch = new javax.swing.JButton(); back = new javax.swing.JButton();

jTable2.setModel(new javax.swing.table.DefaultTableModel( new Object [][] {}, new String [] { "Book ID", "Category", "Name", "Author", "Copies" }

)); fetch.setText("Fetch"); fetch.addActionListener(evt -> fetchActionPerformed(evt)); back.setText("Back"); back.addActionListener(evt -> backActionPerformed(evt)); pack();

}

private void fetchActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel)jTable2.getModel();

String url = "jdbc:mysql://localhost:3306/library";

String user = "root";

String pwd = "Taphasvi@266";

String query = "SELECT \* FROM BOOKS;";

try {

Connection conn = DriverManager.getConnection(url, user, pwd);

Statement stm = conn.createStatement(); ResultSet rs = stm.executeQuery(query); while (rs.next()) { model.addRow(new Object[] { rs.getString("BOOK\_ID"), rs.getString("CATEGORY"), rs.getString("TITLE"), rs.getString("AUTHOR"), rs.getString("COPIES")

});

}

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e.getMessage());

} }

private void backActionPerformed(java.awt.event.ActionEvent evt) {

this.dispose();

}

public static void main(String args[]) { java.awt.EventQueue.invokeLater(() -> new BooksAvailable().setVisible(true));

}

private javax.swing.JButton back; private javax.swing.JButton fetch; private javax.swing.JTable jTable2;

}

4) 1) ADD BOOKS

import java.sql.\*; import javax.swing.\*;

public class AddBooks extends javax.swing.JFrame { public AddBooks() { initComponents(); setDefaultCloseOperation(AddBooks.DISPOSE\_ON\_CLOSE);

}

private void addActionPerformed(java.awt.event.ActionEvent evt) {

String url = "jdbc:mysql://localhost:3306/library";

String user = "root";

String pwd = "Taphasvi@266";

String insertQuery = "INSERT INTO BOOKS VALUES(?,?,?,?,?)";

String updateQuery = "UPDATE BOOKS SET COPIES=COPIES+? WHERE

TITLE=? AND CATEGORY=? AND AUTHOR=?";

String id = t1.getText();

String category = t2.getText();

String title = t3.getText(); String author = t4.getText(); int copies = Integer.parseInt(t5.getText()); try {

Connection conn = DriverManager.getConnection(url, user, pwd); PreparedStatement checkStm = conn.prepareStatement(updateQuery); checkStm.setInt(1, copies); checkStm.setString(2, title); checkStm.setString(3, category); checkStm.setString(4, author); int rows = checkStm.executeUpdate(); if (rows > 0) {

JOptionPane.showMessageDialog(this, "One record updated successfully");

} else {

PreparedStatement insertStm = conn.prepareStatement(insertQuery); insertStm.setString(1, id); insertStm.setString(2, category); insertStm.setString(3, title); insertStm.setString(4, author); insertStm.setInt(5, copies); insertStm.execute();

JOptionPane.showMessageDialog(this, "One record added successfully");

}

t1.setText(null); t2.setText(null); t3.setText(null); t4.setText(null); t5.setText(null);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e);

} }

public static void main(String args[]) { java.awt.EventQueue.invokeLater(() -> new AddBooks().setVisible(true));

}

}

2) DELETE BOOKS

import java.sql.\*; import javax.swing.\*;

public class RemoveBooks extends javax.swing.JFrame { public RemoveBooks() { initComponents(); setDefaultCloseOperation(RemoveBooks.DISPOSE\_ON\_CLOSE);

}

private void deleteActionPerformed(java.awt.event.ActionEvent evt) {

String url = "jdbc:mysql://localhost:3306/library";

String user = "root";

String pwd = "Taphasvi@266";

String input = t1.getText();

String query = "DELETE FROM BOOKS WHERE book\_id ='" + input + "' OR title

='" + input + "';"; try {

Connection conn = DriverManager.getConnection(url, user, pwd); Statement stm = conn.createStatement(); int rows = stm.executeUpdate(query); if (rows > 0)

JOptionPane.showMessageDialog(this, "Book removed from library!"); else

JOptionPane.showMessageDialog(this, "No such book available!"); stm.close();

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e.getMessage());

} }

private void cancelActionPerformed(java.awt.event.ActionEvent evt) { this.dispose();

}

public static void main(String args[]) { java.awt.EventQueue.invokeLater(() -> new RemoveBooks().setVisible(true));

}

}

3) AVAILABLE STAFF

import java.sql.\*; import javax.swing.\*; import javax.swing.table.DefaultTableModel;

public class StaffAvailable extends javax.swing.JFrame { public StaffAvailable() { initComponents(); setDefaultCloseOperation(StaffAvailable.DISPOSE\_ON\_CLOSE);

}

private void fetchActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel) jTable2.getModel();

String url = "jdbc:mysql://localhost:3306/library";

String user = "root";

String pwd = "Taphasvi@266";

String query = "SELECT \* FROM STAFF;";

try {

Connection conn = DriverManager.getConnection(url, user, pwd);

Statement stm = conn.createStatement(); ResultSet rs = stm.executeQuery(query); while (rs.next()) {

String staffid = rs.getString("STAFF\_ID"); String name = rs.getString("name"); long contact = rs.getLong("CONTACT"); model.addRow(new Object[] { staffid, name, contact });

}

rs.close();

stm.close();

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e.getMessage());

} }

private void backActionPerformed(java.awt.event.ActionEvent evt) { this.dispose();

}

public static void main(String args[]) { java.awt.EventQueue.invokeLater(() -> new StaffAvailable().setVisible(true));

}

private javax.swing.JButton back, fetch; private javax.swing.JTable jTable2;

}

4) ADD STAFF

import java.sql.\*; import javax.swing.\*;

public class AddStaff extends javax.swing.JFrame { public AddStaff() { initComponents(); setDefaultCloseOperation(AddStaff.DISPOSE\_ON\_CLOSE);

} private void addActionPerformed(java.awt.event.ActionEvent evt) {

String url = "jdbc:mysql://localhost:3306/library";

String user = "root";

String pwd = "Taphasvi@266";

String query = "INSERT INTO STAFF VALUES(?,?,?)";

String id = t1.getText(); String name = t2.getText(); int contact = Integer.parseInt(t3.getText()); try {

Connection conn = DriverManager.getConnection(url, user, pwd); PreparedStatement stm = conn.prepareStatement(query); stm.setString(1, id); stm.setString(2, name); stm.setInt(3, contact); stm.execute();

JOptionPane.showMessageDialog(this, "One staff added successfully"); t1.setText(null); t2.setText(null); t3.setText(null);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e);

} } private void cancelActionPerformed(java.awt.event.ActionEvent evt) { this.dispose();

}

public static void main(String args[]) { java.awt.EventQueue.invokeLater(() -> new AddStaff().setVisible(true));

} private javax.swing.JButton add, cancel; private javax.swing.JTextField t1, t2, t3;

}

5) DELETE STAFF

import java.sql.\*; import javax.swing.\*;

public class RemoveStaff extends javax.swing.JFrame { public RemoveStaff() { initComponents(); setDefaultCloseOperation(RemoveStaff.DISPOSE\_ON\_CLOSE);

}

private void deleteActionPerformed(java.awt.event.ActionEvent evt) {

String url = "jdbc:mysql://localhost:3306/library";

String user = "root";

String pwd = "Taphasvi@266";

String input = t1.getText();

String query = "DELETE FROM STAFF WHERE staff\_id ='" + input + "' OR name

='" + input + "';"; try (Connection conn = DriverManager.getConnection(url, user, pwd); Statement stm = conn.createStatement()) { int rows = stm.executeUpdate(query);

JOptionPane.showMessageDialog(this, rows > 0 ? "Staff removed from library!" : "No staff available!");

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e.getMessage());

} }

private void cancelActionPerformed(java.awt.event.ActionEvent evt) { this.dispose();

}

public static void main(String args[]) { java.awt.EventQueue.invokeLater(() -> new RemoveStaff().setVisible(true));

}

private javax.swing.JButton cancel; private javax.swing.JButton delete; private javax.swing.JTextField t1;

}

6) EDIT ADMIN

import java.sql.\*; import javax.swing.\*;

public class EditAdmin extends javax.swing.JFrame { public EditAdmin() { initComponents(); setDefaultCloseOperation(EditAdmin.DISPOSE\_ON\_CLOSE);

}

private void updateActionPerformed(java.awt.event.ActionEvent evt) {

String url = "jdbc:mysql://localhost:3306/library";

String user = "root";

String pwd = "Taphasvi@266";

String id = t1.getText();

String column = jComboBox1.getSelectedItem().toString();

String query = "UPDATE admin SET " + column + " = '" + id + "';";

try (Connection conn = DriverManager.getConnection(url, user, pwd); Statement stmt = conn.createStatement()) { int rows = stmt.executeUpdate(query); if (rows > 0) {

JOptionPane.showMessageDialog(this, "Credentials Updated Successfully");

}

t1.setText("");

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e);

} }

private void cancelActionPerformed(java.awt.event.ActionEvent evt) { this.dispose();

}

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(() -> new EditAdmin().setVisible(true));

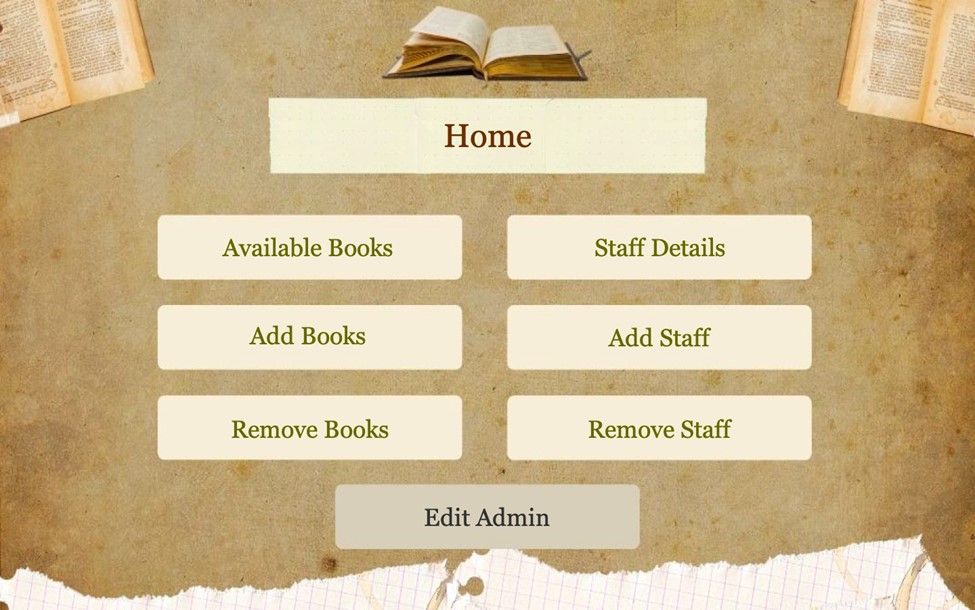
}

private javax.swing.JButton cancel; private javax.swing.JComboBox<String> jComboBox1; private javax.swing.JTextField t1; private javax.swing.JButton update;

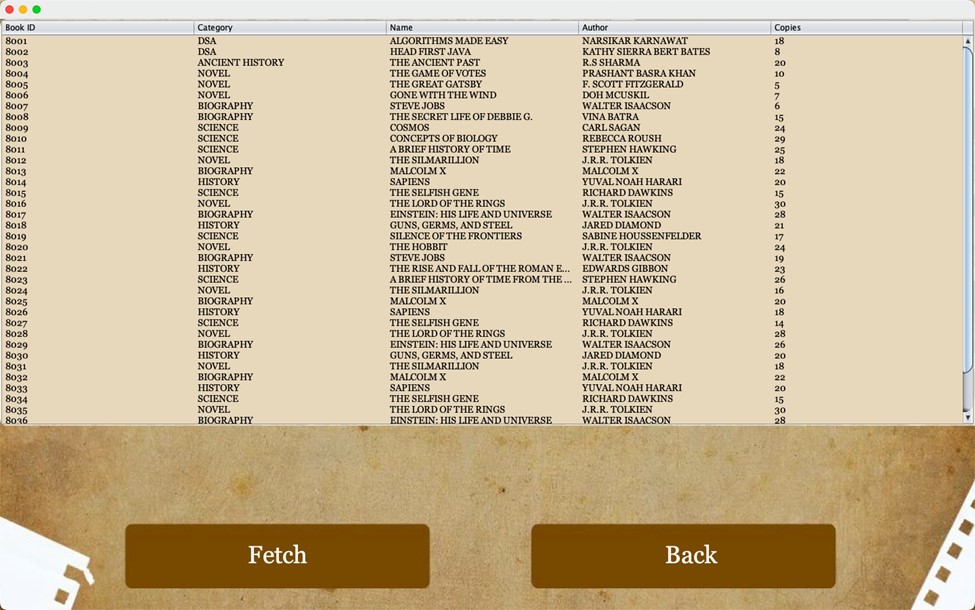
}

SNAPSHOTS

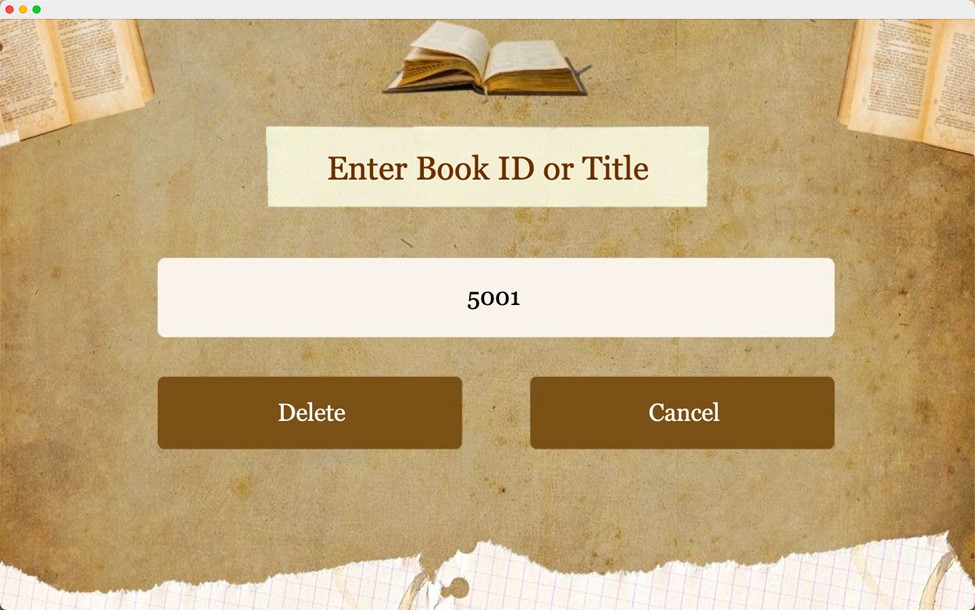
4.1 HOME PAGE



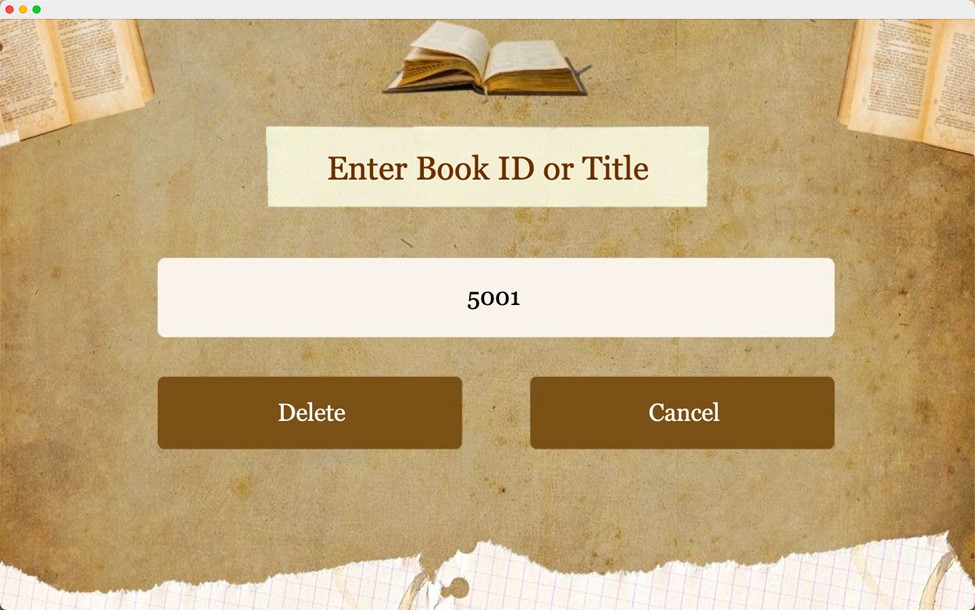
4.2 BOOK DETAILS



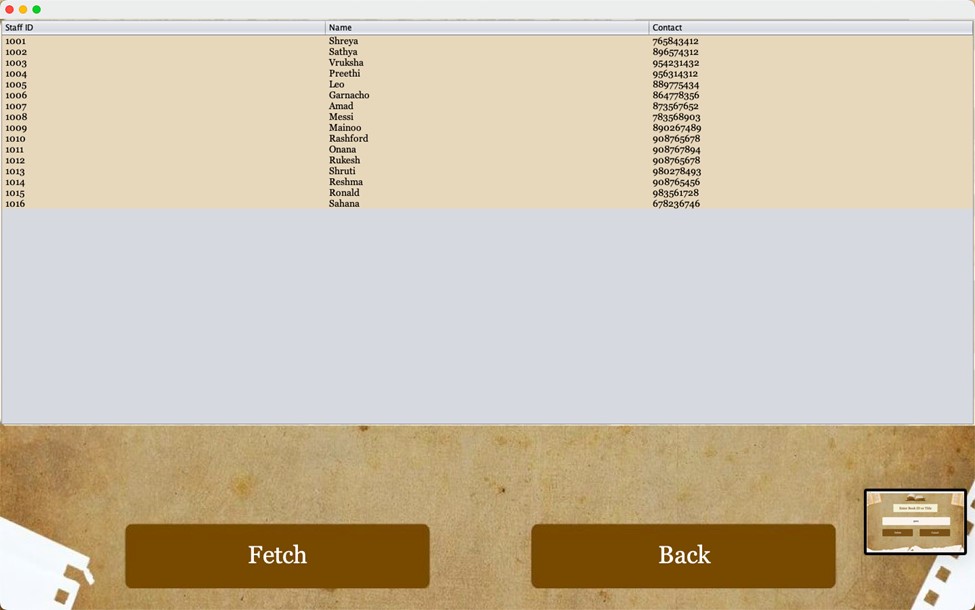
4.3 ADD BOOKS



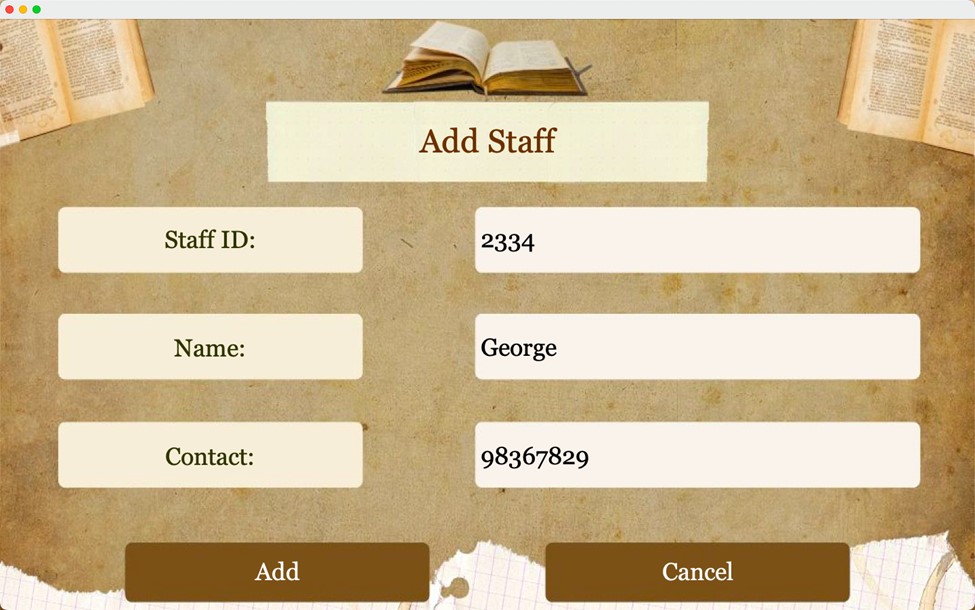
4.4 REMOVE BOOKS



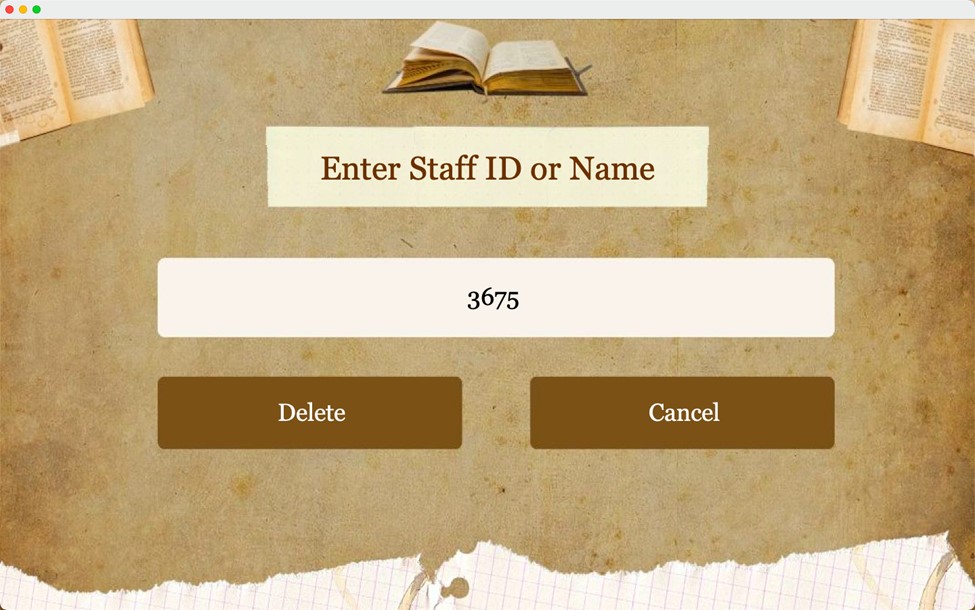
4.5 STAFF DETAILS



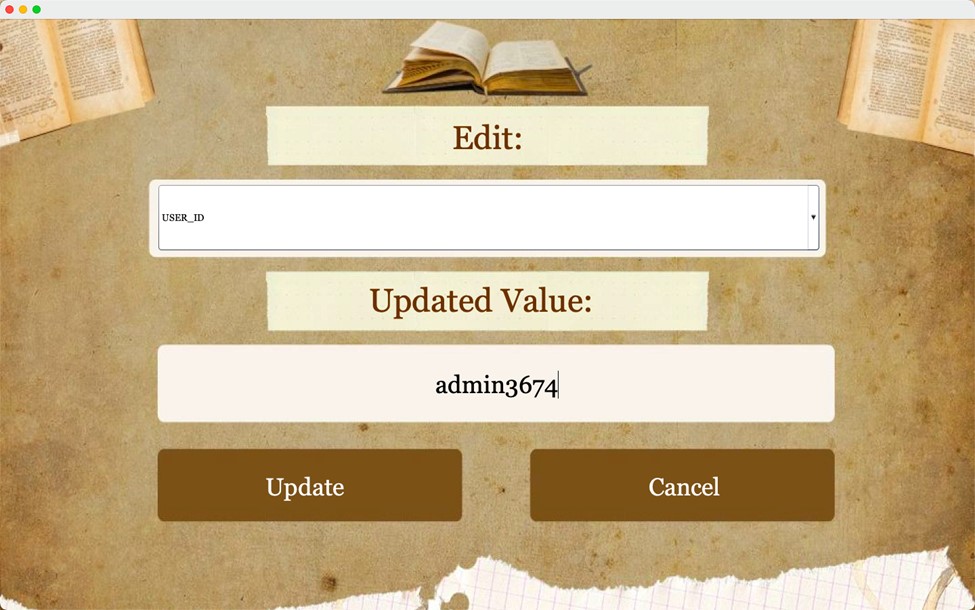
4.6 ADD STAFFS



4.7 REMOVE STAFFS



4.8 EDIT ADMIN



## CONCLUSION

The Java-based Library Management System is a robust application designed to streamline and automate library operations such as book lending, returns, catalog management, and member tracking. This project demonstrates the efficient use of Java's object-oriented programming features and database integration to enhance library workflows.

REFERENCES

1. <https://www.javatpoint.com/java-tutorial>
2. <https://www.wikipedia.org/>
3. [https://www.w3schools.com/sql/](https://www.wikipedia.org/)