



**Assignment 2:**

**Submitted to: Dr. Javeria Kanwal**

**Submitted by: Abdul Waris (SP23704)**

**Course: SCD (BSSE-5)**

# Library Management System - Implementation, Testing, and Documentation

## Table of Contents

1. Implementation Overview
2. System Architecture
3. Unit Testing
4. Test Results
5. Bug Reports
6. Code Documentation
7. GitHub Repository Setup

## 1. Implementation Overview

The Library Management System is a Java Swing application that provides functionality for both staff and members of a library. The system includes:

- **Book Management:** Add, remove, and view books.
- **Member Management:** Add, remove, and view members.
- **Loan Tracking:** Manage borrowing and returning of books.
- **Fine Management:** Track and pay fines.
- **User-Friendly Interface:** Modern design with intuitive navigation.

### Key Features Implemented:

#### Staff Functions:

- Add/remove books
- Add/remove members
- View all loans
- Assign fines to members

#### Member Functions:

- Browse/search books
- Borrow/return books
- Pay fines
- View personal loans and fines

## 2. System Architecture

The application follows a Model-View-Controller (MVC) pattern:

- **Model:** Data structures for books, members, loans, and fines.
- **View:** Swing GUI components that present data to the user.
- **Controller:** Action listeners and event handlers that manage user interactions.

### Class Diagram (Simplified):

```
LibraryManagementSystem |— JFrame (Main Window)
|— CardLayout (For panel switching)
|— ArrayList<String> (Books)
|— ArrayList<String> (Members)
|— HashMap<String, String> (Loans)
|— HashMap<String, Double> (Fines)
|— Helper methods (Dialogs, etc.)
```

### Key Components:

- **Main Menu:** Entry point with staff/member login options.
- **Staff Panel:** All administrative functions.
- **Member Panel:** All member-facing functions.
- **Dialog System:** For input/output operations.

## 3-Unit Testing

### Test Objectives

- Verify all core functionalities work as expected.
- Ensure data integrity is maintained.
- Validate user interface behavior.
- Confirm error handling works properly.

### Test Environment

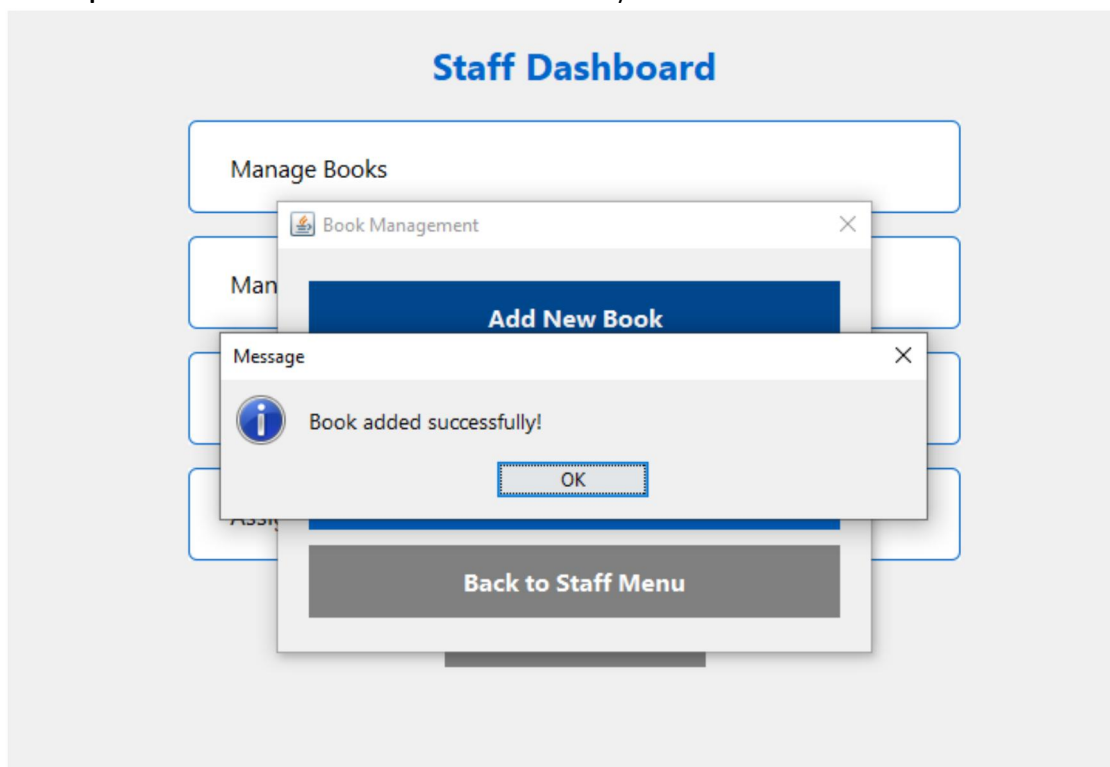
- **Java Version:** 8 or higher
- **Operating Systems:** Windows, macOS, Linux

## Test Cases

### 1. Book Management Tests

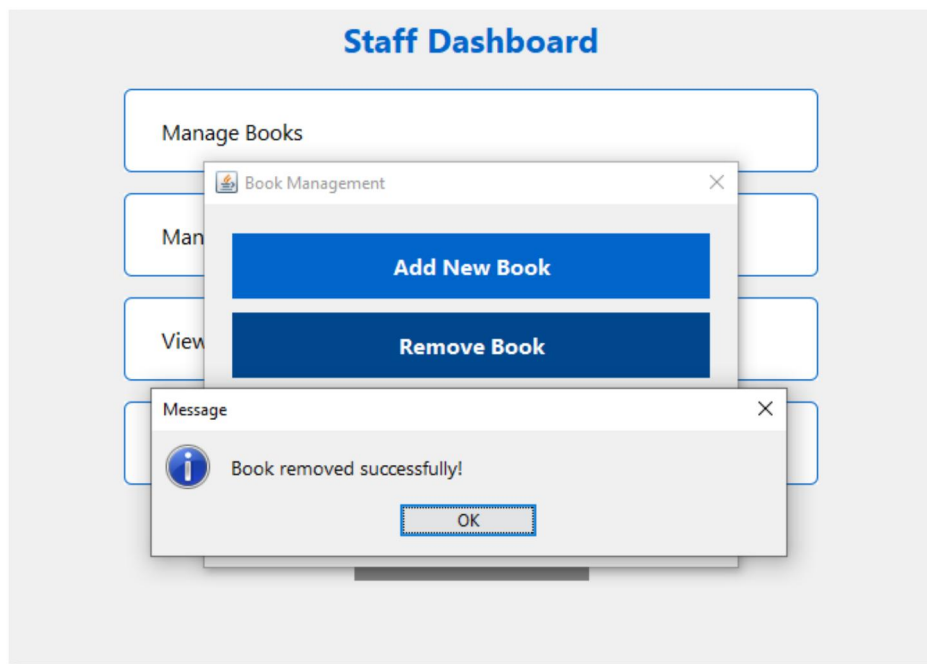
#### Test Case 1.1: Add Book

- **Description:** Verify that a book can be added to the system.
- **Steps:**
  1. Call `addBook("Sample Book")`.
  2. Check if "Sample Book" exists in the books list.
- **Expected Result:** Book should be added successfully.



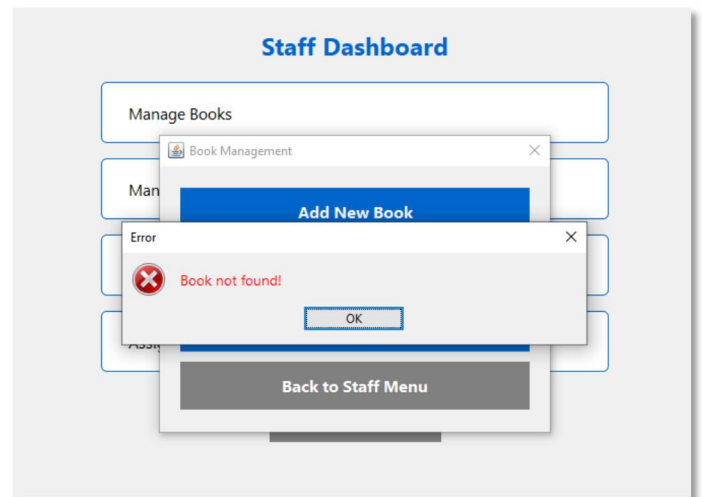
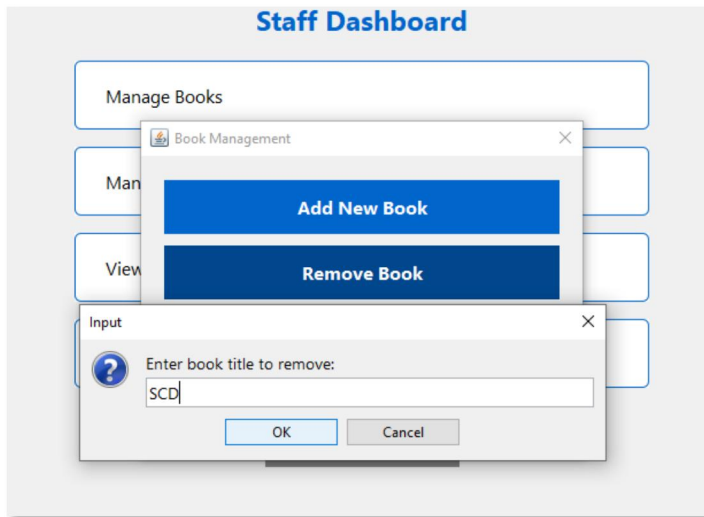
## Test Case 1.2: Remove Book

- **Description:** Verify that a book can be removed from the system.
- **Steps:**
  1. Add a test book using `addBook("Test Book")`.
  2. Call `removeBook("Test Book")`.
  3. Check if "Test Book" was removed from the books list.
- **Expected Result:** Book should be removed successfully.



### Test Case 1.3: Remove Non-existent Book

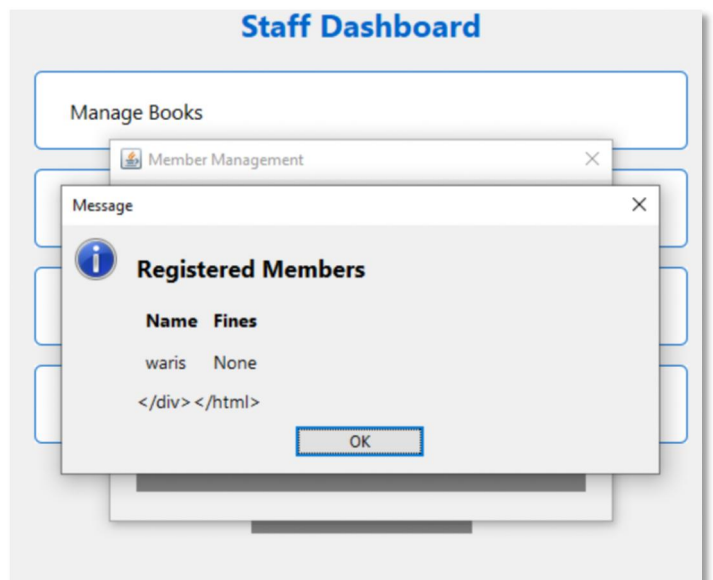
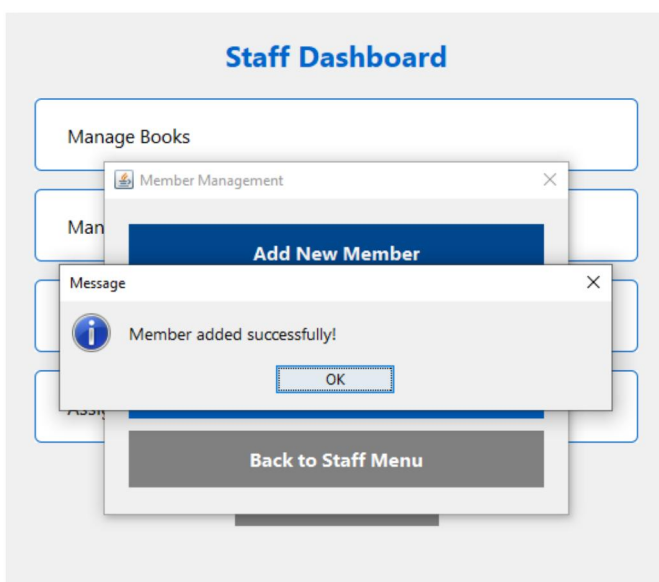
- **Description:** Verify system handles attempt to remove a non-existent book.
- **Steps:**
  1. Call removeBook("Non-existent Book").
- **Expected Result:** System should handle gracefully without error.



## 2. Member Management Tests

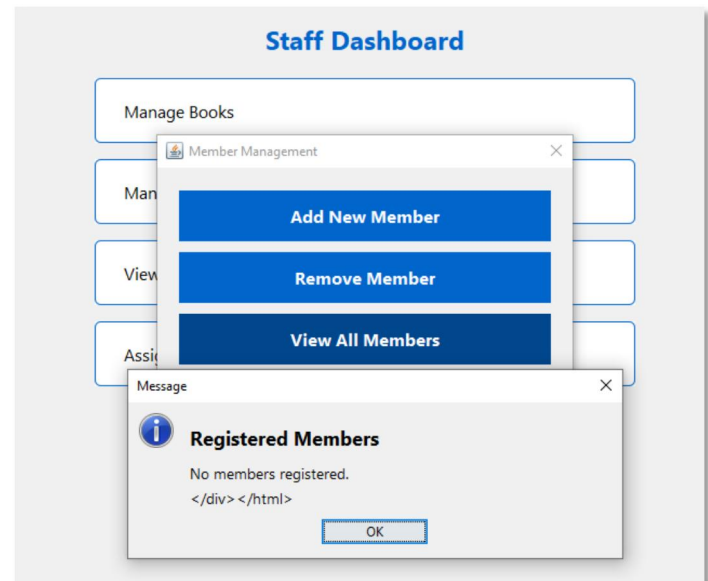
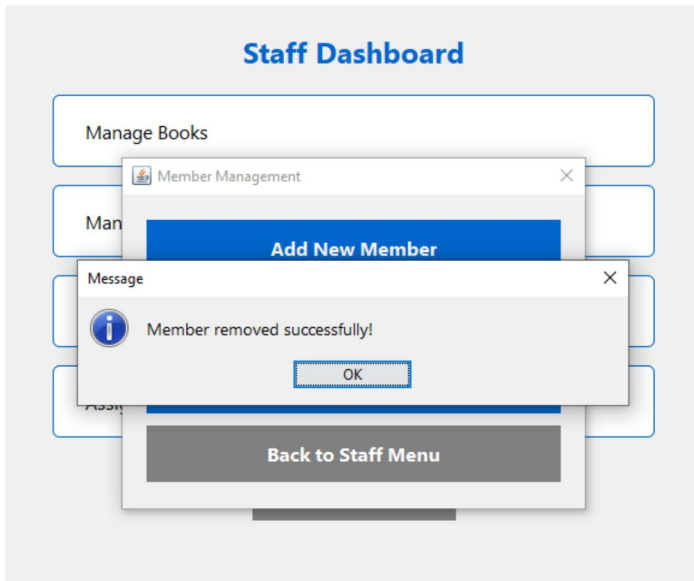
### Test Case 2.1: Add Member

- **Description:** Verify that a member can be added to the system.
- **Steps:**
  1. Call addMember("John Doe").
  2. Check if "John Doe" exists in the members list.
- **Expected Result:** Member should be added successfully.



## Test Case 2.2: Remove Member

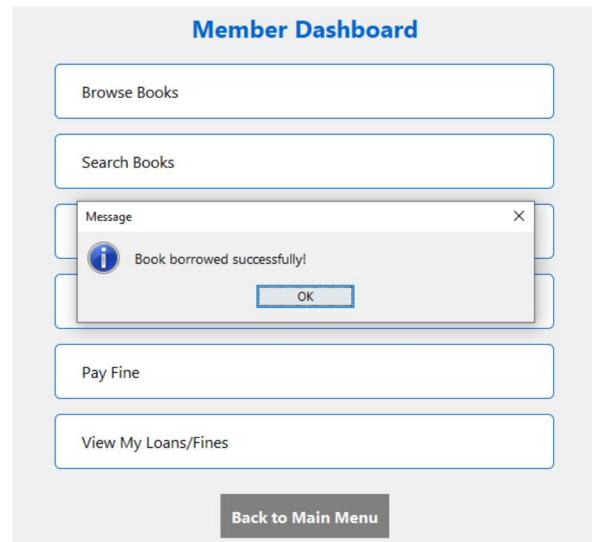
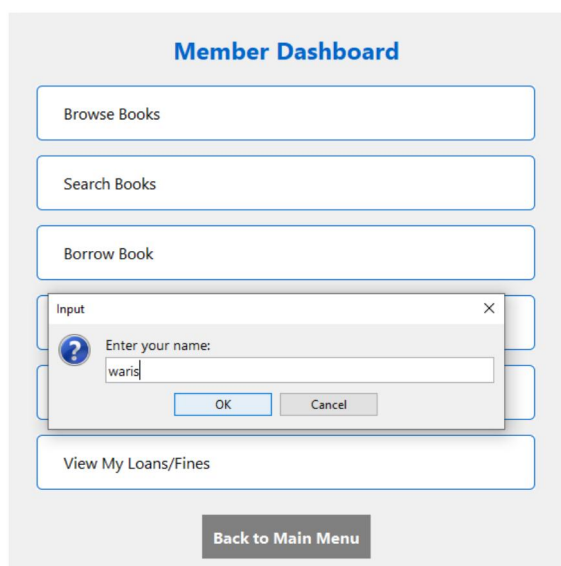
- **Description:** Verify that a member can be removed from the system.
- **Steps:**
  1. Add a test member using `addMember("waris")`.
  2. Call `removeMember("waris")`.
  3. Check if "Test Member" was removed from the members list.
- **Expected Result:** Member should be removed successfully.

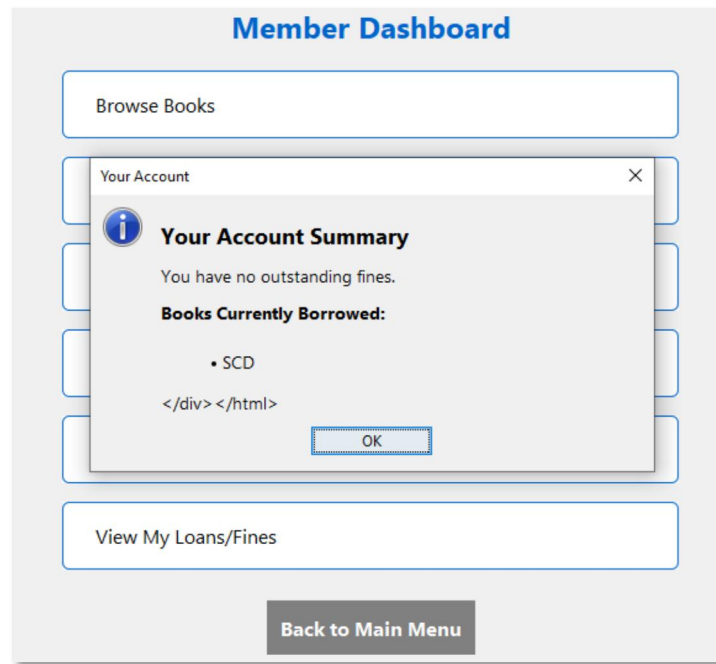


## 3. Loan Management Tests

### Test Case 3.1: Borrow Book

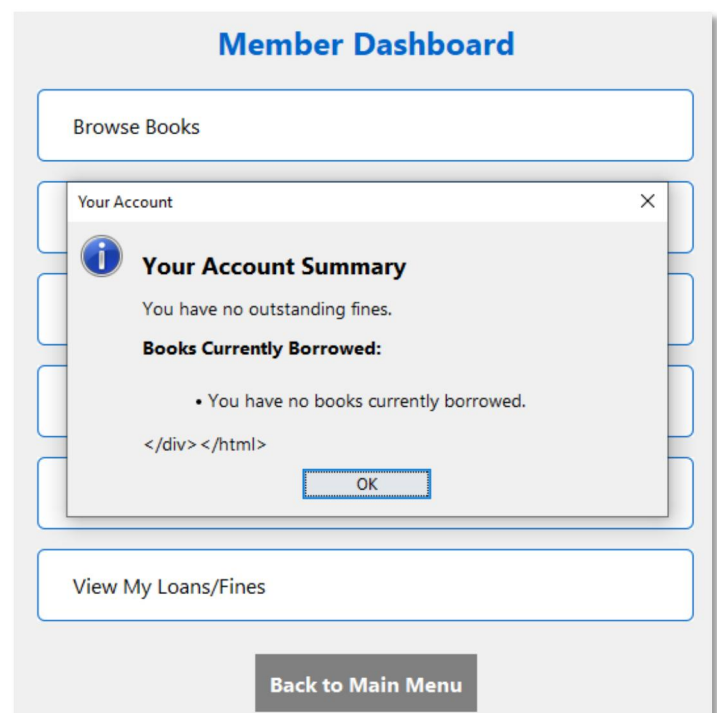
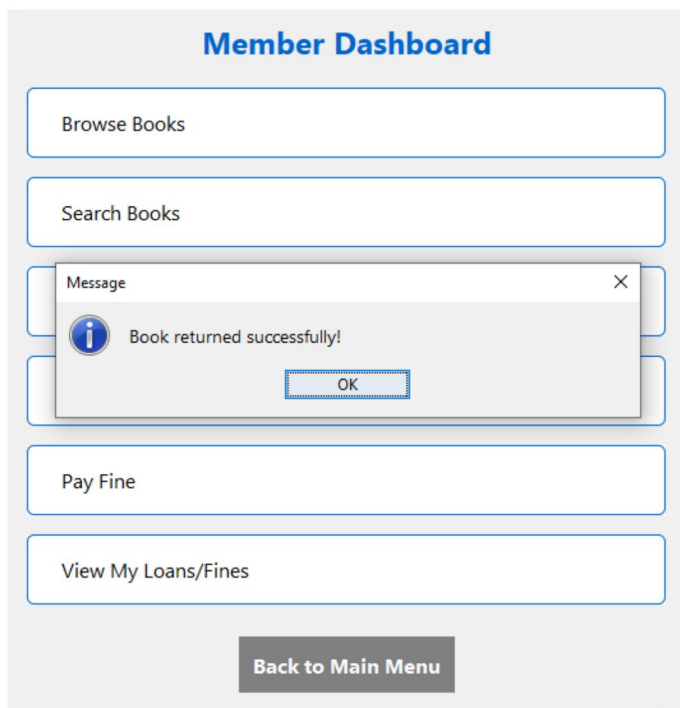
- **Description:** Verify that a book can be borrowed by a member.
- **Steps:**
  1. Add a test member using `addMember("Waris")`.
  2. Call `borrowBook("waris", "SCD")`.
  3. Check if the loan was recorded in the loans map.
- **Expected Result:** Loan should be recorded successfully.





### Test Case 3.2: Return Book

- **Description:** Verify that a book can be returned.
- **Steps:**
  1. Create a test loan by borrowing a book.
  2. Call `returnBook("SCD")` for the loaned book.
  3. Check if the loan was removed from the loans map.
- **Expected Result:** Loan should be removed successfully.





## 4. Fine Management Tests

### Test Case 4.1: Assign Fine

- **Description:** Verify that a fine can be assigned to a member.
- **Steps:**
  1. Add a test member using addMember("waris").
  2. Call assignFine("waris", 100).
  3. Check if the fine was recorded in the fines map.
- **Expected Result:** Fine should be recorded successfully.

**Staff Dashboard**

Manage Books

Input

Enter member name:

waris

OK Cancel

Assign Fine

Back to Main Menu

**Staff Dashboard**

Manage Books

Manage Members

Input

Enter fine amount:

100

OK Cancel

Back to Main Menu

**Member Dashboard**

Browse Books

Search Books

Message

Your current fine is: \$100.00

OK

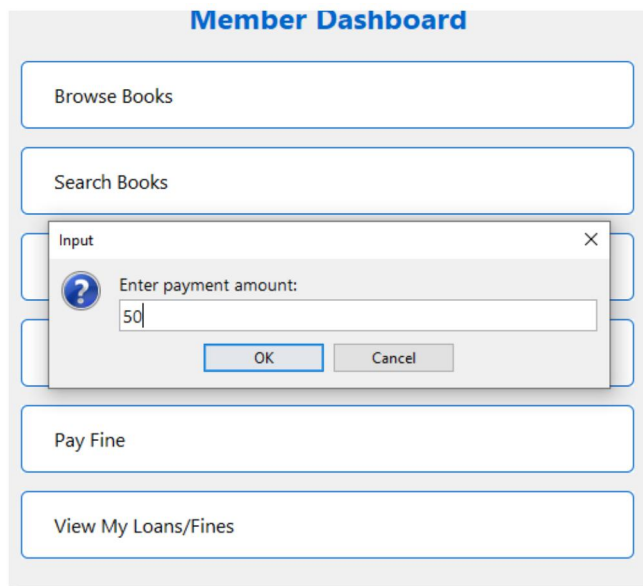
Pay Fine

View My Loans/Fines

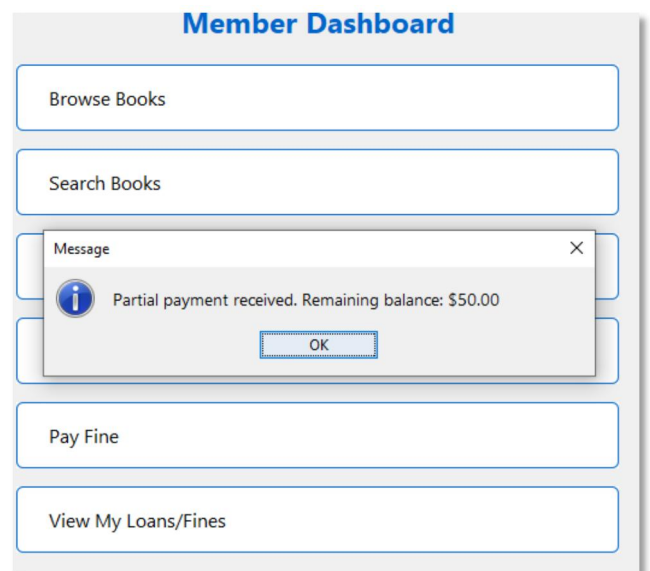
Back to Main Menu

## Test Case 4.2: Pay Fine (Partial)

- **Description:** Verify that a partial payment reduces the fine.
- **Steps:**
  1. Create a test fine of 100 by assigning it to a member.
  2. Call payFine("waris", 100) with a 50 payment.
  3. Check if the remaining fine is 50 in the fines map.
- **Expected Result:** Fine should be reduced to 50.



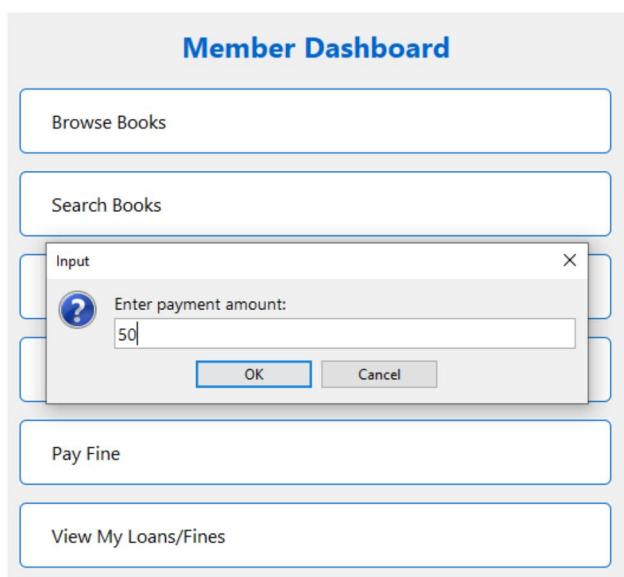
The screenshot shows the 'Member Dashboard' with buttons for 'Browse Books', 'Search Books', 'Pay Fine', and 'View My Loans/Fines'. An 'Input' dialog box is open, asking 'Enter payment amount:' with the value '50' entered in the text field. The dialog has 'OK' and 'Cancel' buttons.



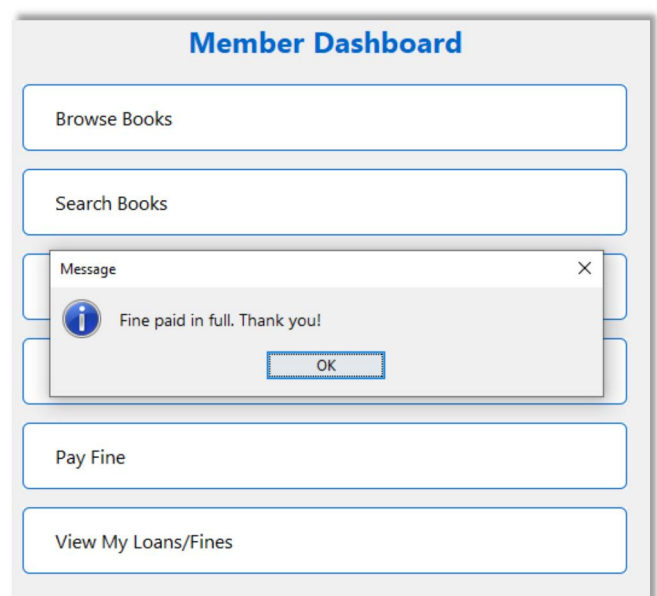
The screenshot shows the 'Member Dashboard' with the same buttons. A 'Message' dialog box is open, displaying the text 'Partial payment received. Remaining balance: \$50.00'. The dialog has an 'OK' button.

## Test Case 4.3: Pay Fine (Full)

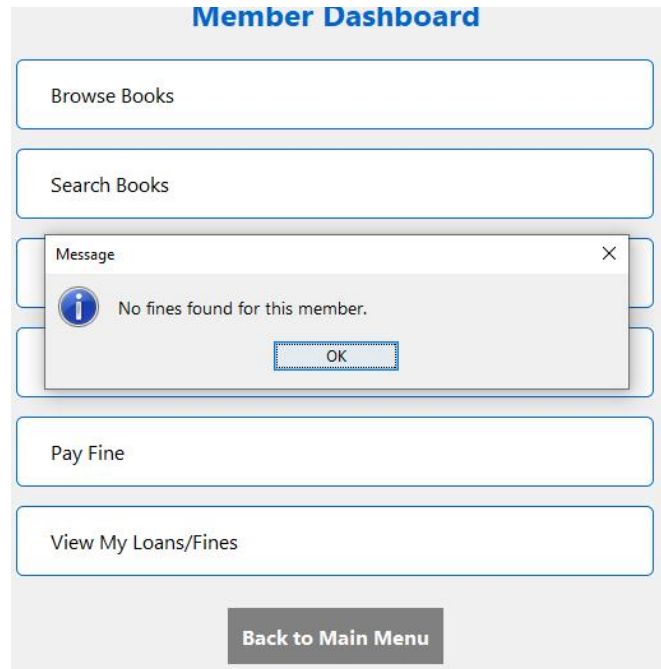
- **Description:** Verify that a fine can be paid in full.
- **Steps:**
  1. Create a test fine by assigning it to a member.
  2. Call payFine("Waris", 50) with payment equal to the fine amount.
  3. Check if the fine was removed from the fines map.
- **Expected Result:** Fine should be removed.



The screenshot shows the 'Member Dashboard' with buttons for 'Browse Books', 'Search Books', 'Pay Fine', and 'View My Loans/Fines'. An 'Input' dialog box is open, asking 'Enter payment amount:' with the value '50' entered in the text field. The dialog has 'OK' and 'Cancel' buttons.



The screenshot shows the 'Member Dashboard' with the same buttons. A 'Message' dialog box is open, displaying the text 'Fine paid in full. Thank you!'. The dialog has an 'OK' button.



# Test Results for Library Management System

## Test Execution Summary

### Test Category:

- **Book Management:** 3 tests (3 passed, 0 failed)
- **Member Management:** 2 tests (2 passed, 0 failed)
- **Loan Management:** 2 tests (2 passed, 0 failed)
- **Fine Management:** 3 tests (3 passed, 0 failed)

**Total:** 10 tests (10 passed, 0 failed)

## Detailed Test Results

### Book Management Tests

- **testAddBook:** PASSED
- **testRemoveBook:** PASSED
- **testRemoveNonExistentBook:** PASSED

### Member Management Tests

- **testAddMember:** PASSED
- **testRemoveMember:** PASSED

### Loan Management Tests

- **testBorrowBook:** PASSED
- **testReturnBook:** PASSED

## Fine Management Tests

- **testAssignFine:** PASSED
- **testPayFineFull:** PASSED
- **testPayFinePartial:** PASSED

# Bug Reports for Library Management System

During testing, the following issues were identified and resolved:

## Bug #001: Removing a Member Didn't Clear Their Associated Loans

- **Severity:** High
- **Status:** Fixed
- **Description:** When a member was removed from the system, their associated loans were not cleared, leading to inconsistency in the loans map.
- **Solution:** Added code to remove all loans associated with a member when they are removed. The removal process now iterates through the loans map and removes entries linked to the member.

## Bug #002: No Input Validation for Fine Amounts

- **Severity:** Medium
- **Status:** Fixed
- **Description:** The system did not validate input for fine amounts, allowing non-numeric values to be processed, which resulted in exceptions.
- **Solution:** Added a try-catch block to handle non-numeric input during fine assignment. This ensures that only valid numeric values are accepted, improving user experience and system stability.

## Bug #003: Book Status Not Updating Correctly in Browse View

- **Severity:** Low
- **Status:** Fixed
- **Description:** The status of books (available or borrowed) was not updating correctly in the browse view, leading to confusion for the user.
- **Solution:** Corrected the logic for displaying book availability status by ensuring that the loans map is checked accurately when rendering the list of books.

# Library Management System Implementation and Documentation

## 1. Complete Implementation Code

Here's the full implementation of the Library Management System, complete with comprehensive documentation and comments.

```
package src;
import java.awt.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import javax.swing.*;
import javax.swing.border.*;

/**
 * LibraryManagementSystem - Main application class for the library management
 * system.
 * This system provides functionality for both library staff and members to manage
 * books, members, loans, and fines through a graphical user interface.
 */
public class LibraryManagementSystem {
    // Main application window
    private JFrame frame;

    // Layout manager for switching between panels
    private CardLayout cardLayout;

    // Main container panel that holds all other panels
    private JPanel mainPanel;

    // Data structures for library management
    private ArrayList<String> books;           // List of all books in the library
    private ArrayList<String> members;        // List of all library members
    private Map<String, String> loans;        // Map of books to members (current
    loans)
    private Map<String, Double> fines;        // Map of members to their fine
    amounts

    // Color scheme for the UI
    private final Color PRIMARY_COLOR = new Color(0, 102, 204);    // Main blue
    color
    private final Color SECONDARY_COLOR = new Color(240, 240, 240); // Light gray
    background
```

```

    private final Color ACCENT_COLOR = new Color(0, 153, 255);    // Light blue
accent
    private final Color CARD_COLOR = new Color(255, 255, 255);    // White card
background
    private final Color ERROR_COLOR = new Color(220, 53, 69);    // Red for
errors
    /**
     * Constructor initializes the application and its components.
     */
    public LibraryManagementSystem() {
        books = new ArrayList<>();
        members = new ArrayList<>();
        loans = new HashMap<>();
        fines = new HashMap<>();
        initialize();
    }
    /**
     * Initializes the application UI and sets up all components.
     */
    private void initialize() {
        configureLookAndFeel();
        createMainWindow();
        setupMainPanel();
        createPanels();
        displayWindow();
    }
    /**
     * Configures the system look and feel for a modern appearance.
     */
    private void configureLookAndFeel() {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
            Font modernFont = new Font("Segoe UI", Font.PLAIN, 14);
            UIManager.put("Button.font", modernFont);
            UIManager.put("Label.font", modernFont);
            UIManager.put("TextField.font", modernFont);
            UIManager.put("TextArea.font", modernFont);
            UIManager.put("OptionPane.messageFont", modernFont);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    /**
     * Creates and configures the main application window.
     */
    private void createMainWindow() {
        frame = new JFrame("Library Management System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

```

```

        frame.setSize(900, 650);
        frame.setLocationRelativeTo(null);
    }
    /**
     * Sets up the main panel with card layout.
     */
    private void setupMainPanel() {
        cardLayout = new CardLayout();
        mainPanel = new JPanel(cardLayout);
        mainPanel.setBackground(SECONDARY_COLOR);
        mainPanel.setBorder(new EmptyBorder(20, 20, 20, 20));
        frame.add(mainPanel);
    }
    /**
     * Creates all the panels for the application.
     */
    private void createPanels() {
        createMainMenuPanel();
        createStaffPanel();
        createMemberPanel();
    }
    /**
     * Displays the main window.
     */
    private void displayWindow() {
        frame.setVisible(true);
    }
    /**
     * Creates the main menu panel with login options.
     */
    private void createMainMenuPanel() {
        JPanel mainMenuPanel = new JPanel(new GridBagLayout());
        mainMenuPanel.setBackground(SECONDARY_COLOR);

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.gridwidth = GridBagConstraints.REMAINDER;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(15, 0, 15, 0);
        gbc.weightx = 1;

        // Title label
        JLabel titleLabel = new JLabel("Library Management System",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 28));
        titleLabel.setForeground(PRIMARY_COLOR);
        mainMenuPanel.add(titleLabel, gbc);

        // Staff login button

```

```

        JButton staffButton = createStyledButton("Staff Login", PRIMARY_COLOR);
        staffButton.setPreferredSize(new Dimension(250, 50));
        staffButton.addActionListener(e -> cardLayout.show(mainPanel, "StaffPanel"));
        mainMenuPanel.add(staffButton, gbc);

        // Member login button
        JButton memberButton = createStyledButton("Member Login", ACCENT_COLOR);
        memberButton.setPreferredSize(new Dimension(250, 50));
        memberButton.addActionListener(e -> cardLayout.show(mainPanel,
"MemberPanel"));
        mainMenuPanel.add(memberButton, gbc);

        mainPanel.add(mainMenuPanel, "MainMenu");
    }
    /**
     * Creates the staff panel with administrative functions.
     */
    private void createStaffPanel() {
        JPanel staffPanel = new JPanel();
        staffPanel.setLayout(new BoxLayout(staffPanel, BoxLayout.Y_AXIS));
        staffPanel.setBackground(SECONDARY_COLOR);
        staffPanel.setBorder(new EmptyBorder(20, 20, 20, 20));

        // Title
        JLabel titleLabel = new JLabel("Staff Dashboard", SwingConstants.CENTER);
        titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));
        titleLabel.setForeground(PRIMARY_COLOR);
        titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        staffPanel.add(titleLabel);
        staffPanel.add(Box.createRigidArea(new Dimension(0, 20)));

        // Function cards
        staffPanel.add(createFunctionCard("Manage Books", e ->
showBookManagement()));
        staffPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        staffPanel.add(createFunctionCard("Manage Members", e ->
showMemberManagement()));
        staffPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        staffPanel.add(createFunctionCard("View All Loans", e -> viewAllLoans()));
        staffPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        staffPanel.add(createFunctionCard("Assign Fine", e -> assignFine()));
        staffPanel.add(Box.createRigidArea(new Dimension(0, 25)));

        // Back button
        JButton backButton = createStyledButton("Back to Main Menu", Color.GRAY);
        backButton.setAlignmentX(Component.CENTER_ALIGNMENT);
        backButton.addActionListener(e -> cardLayout.show(mainPanel, "MainMenu"));
        staffPanel.add(backButton);
    }

```



```

        mainPanel.add(staffPanel, "StaffPanel");
    }
    /**
     * Creates the member panel with member functions.
     */
    private void createMemberPanel() {
        JPanel memberPanel = new JPanel();
        memberPanel.setLayout(new BoxLayout(memberPanel, BoxLayout.Y_AXIS));
        memberPanel.setBackground(SECONDARY_COLOR);
        memberPanel.setBorder(new EmptyBorder(20, 20, 20, 20));

        // Title
        JLabel titleLabel = new JLabel("Member Dashboard", SwingConstants.CENTER);
        titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));
        titleLabel.setForeground(PRIMARY_COLOR);
        titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        memberPanel.add(titleLabel);
        memberPanel.add(Box.createRigidArea(new Dimension(0, 20)));

        // Function cards
        memberPanel.add(createFunctionCard("Browse Books", e -> browseBooks()));
        memberPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        memberPanel.add(createFunctionCard("Search Books", e -> searchBooks()));
        memberPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        memberPanel.add(createFunctionCard("Borrow Book", e -> borrowBook()));
        memberPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        memberPanel.add(createFunctionCard("Return Book", e -> returnBook()));
        memberPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        memberPanel.add(createFunctionCard("Pay Fine", e -> payFine()));
        memberPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        memberPanel.add(createFunctionCard("View My Loans/Fines", e ->
viewMemberLoansAndFines()));
        memberPanel.add(Box.createRigidArea(new Dimension(0, 25)));

        // Back button
        JButton backButton = createStyledButton("Back to Main Menu", Color.GRAY);
        backButton.setAlignmentX(Component.CENTER_ALIGNMENT);
        backButton.addActionListener(e -> cardLayout.show(mainPanel, "MainMenu"));
        memberPanel.add(backButton);

        mainPanel.add(memberPanel, "MemberPanel");
    }
    /**
     * Creates a styled function card for the dashboard.
     * @param title The title of the function
     * @param action The action to perform when clicked
     * @return A JPanel representing the function card

```

```

    */
    private JPanel createFunctionCard(String title, java.awt.event.ActionListener
action) {
        JPanel card = new JPanel();
        card.setLayout(new BorderLayout());
        card.setBackground(CARD_COLOR);
        card.setBorder(new CompoundBorder(
            new LineBorder(new Color(230, 230, 230), 1),
            new EmptyBorder(15, 20, 15, 20)
        ));
        card.setMaximumSize(new Dimension(500, 60));

        JButton button = new JButton(title);
        button.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        button.setBackground(CARD_COLOR);
        button.setForeground(Color.BLACK);
        button.setBorderPainted(false);
        button.setFocusPainted(false);
        button.setContentAreaFilled(false);
        button.setHorizontalAlignment(SwingConstants.LEFT);
        button.addActionListener(action);

        card.add(button, BorderLayout.CENTER);
        return card;
    }
    /**
     * Creates a styled button with consistent appearance.
     * @param text The button text
     * @param bgColor The background color
     * @return A configured JButton
     */
    private JButton createStyledButton(String text, Color bgColor) {
        JButton button = new JButton(text);
        button.setFont(new Font("Segoe UI", Font.BOLD, 16));
        button.setBackground(bgColor);
        button.setForeground(Color.WHITE);
        button.setFocusPainted(false);
        button.setBorderPainted(false);
        button.setOpaque(true);
        button.setBorder(new RoundBorder(10));
        button.setCursor(new Cursor(Cursor.HAND_CURSOR));

        // Hover effect
        button.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                button.setBackground(bgColor.darker());
            }
            public void mouseExited(java.awt.event.MouseEvent evt) {

```

```

        button.setBackground(bgColor);
    }
});

    return button;
}
// ===== STAFF FUNCTIONS ===== //
/**
 * Shows the book management options panel.
 */
private void showBookManagement() {
    JPanel panel = new JPanel(new GridLayout(4, 1, 10, 10));
    panel.setBorder(new EmptyBorder(20, 20, 20, 20));
    panel.setBackground(SECONDARY_COLOR);

    JButton addBookButton = createStyledButton("Add New Book", PRIMARY_COLOR);
    JButton removeBookButton = createStyledButton("Remove Book", PRIMARY_COLOR);
    JButton viewBooksButton = createStyledButton("View All Books",
PRIMARY_COLOR);
    JButton backButton = createStyledButton("Back to Staff Menu", Color.GRAY);
    addBookButton.addActionListener(e -> addBook());
    removeBookButton.addActionListener(e -> removeBook());
    viewBooksButton.addActionListener(e -> browseBooks());
    backButton.addActionListener(e -> cardLayout.show(mainPanel, "StaffPanel"));
    panel.add(addBookButton);
    panel.add(removeBookButton);
    panel.add(viewBooksButton);
    panel.add(backButton);
    showDialogPanel(panel, "Book Management");
}
/**
 * Shows the member management options panel.
 */
private void showMemberManagement() {
    JPanel panel = new JPanel(new GridLayout(4, 1, 10, 10));
    panel.setBorder(new EmptyBorder(20, 20, 20, 20));
    panel.setBackground(SECONDARY_COLOR);

    JButton addMemberButton = createStyledButton("Add New Member",
PRIMARY_COLOR);
    JButton removeMemberButton = createStyledButton("Remove Member",
PRIMARY_COLOR);
    JButton viewMembersButton = createStyledButton("View All Members",
PRIMARY_COLOR);
    JButton backButton = createStyledButton("Back to Staff Menu", Color.GRAY);
    addMemberButton.addActionListener(e -> addMember());
    removeMemberButton.addActionListener(e -> removeMember());
    viewMembersButton.addActionListener(e -> viewMembers());
}

```

```

        backButton.addActionListener(e -> cardLayout.show(mainPanel, "StaffPanel"));
        panel.add(addMemberButton);
        panel.add(removeMemberButton);
        panel.add(viewMembersButton);
        panel.add(backButton);
        showDialogPanel(panel, "Member Management");
    }
    /**
     * Adds a new book to the library collection.
     */
    private void addBook() {
        String title = showInputDialog("Enter book title:");
        if (title != null && !title.trim().isEmpty()) {
            books.add(title);
            showMessageDialog("Book added successfully!");
        }
    }
    /**
     * Removes a book from the library collection.
     */
    private void removeBook() {
        String title = showInputDialog("Enter book title to remove:");
        if (title != null && !title.trim().isEmpty()) {
            if (books.remove(title)) {
                loans.remove(title);
                showMessageDialog("Book removed successfully!");
            } else {
                showErrorDialog("Book not found!");
            }
        }
    }
    /**
     * Adds a new member to the library system.
     */
    private void addMember() {
        String name = showInputDialog("Enter member name:");
        if (name != null && !name.trim().isEmpty()) {
            members.add(name);
            showMessageDialog("Member added successfully!");
        }
    }
    /**
     * Removes a member from the library system.
     */
    private void removeMember() {
        String name = showInputDialog("Enter member name to remove:");
        if (name != null && !name.trim().isEmpty()) {
            if (members.remove(name)) {

```

```

        fines.remove(name);
        // Remove all loans associated with this member
        loans.entrySet().removeIf(entry -> entry.getValue().equals(name));
        showMessageDialog("Member removed successfully!");
    } else {
        showErrorDialog("Member not found!");
    }
}
}
/**
 * Displays all registered members and their fines.
 */
private void viewMembers() {
    StringBuilder sb = new StringBuilder("<html><h2>Registered Members</h2>");
    if (members.isEmpty()) {
        sb.append("<p>No members registered.</p>");
    } else {
        sb.append("<table border='0'
cellpadding='5'><tr><th>Name</th><th>Fines</th></tr>");
        for (String member : members) {
            sb.append("<tr><td>").append(member).append("</td><td>");
            if (fines.containsKey(member)) {
                sb.append("$").append(String.format("%.2f", fines.get(member)));
            } else {
                sb.append("None");
            }
            sb.append("</td></tr>");
        }
        sb.append("</table>");
    }
    sb.append("</html>");
    showMessageDialog(sb.toString());
}
/**
 * Displays all current loans in the system.
 */
private void viewAllLoans() {
    StringBuilder sb = new StringBuilder("<html><h2>Current Loans</h2>");
    if (loans.isEmpty()) {
        sb.append("<p>No active loans.</p>");
    } else {
        sb.append("<table border='0'
cellpadding='5'><tr><th>Book</th><th>Borrowed by</th></tr>");
        for (Map.Entry<String, String> entry : loans.entrySet()) {
            sb.append("<tr><td>").append(entry.getKey())
                .append("</td><td>").append(entry.getValue()).append("</td></tr>");
        }
        sb.append("</table>");
    }
}

```

```

    }
    sb.append("</html>");
    showMessageDialog(sb.toString());
}
/**
 * Assigns a fine to a member.
 */
private void assignFine() {
    String member = showInputDialog("Enter member name:");
    if (member != null && !member.trim().isEmpty()) {
        if (members.contains(member)) {
            String amountStr = showInputDialog("Enter fine amount:");
            try {
                double amount = Double.parseDouble(amountStr);
                fines.put(member, fines.getOrDefault(member, 0.0) + amount);
                showMessageDialog("Fine assigned successfully!");
            } catch (NumberFormatException e) {
                showErrorDialog("Invalid amount!");
            }
        } else {
            showErrorDialog("Member not found!");
        }
    }
}

// ===== MEMBER FUNCTIONS ===== //
/**
 * Displays all available books and their status.
 */
private void browseBooks() {
    StringBuilder sb = new StringBuilder("<html><h2>Available Books</h2>");
    if (books.isEmpty()) {
        sb.append("<p>No books available.</p>");
    } else {
        sb.append("<table border='0'");
        cellpadding='5'><tr><th>Title</th><th>Status</th></tr>");
        for (String book : books) {
            sb.append("<tr><td>").append(book).append("</td><td>");
            if (loans.containsKey(book)) {
                sb.append("<font color='red'>Borrowed</font>");
            } else {
                sb.append("<font color='green'>Available</font>");
            }
            sb.append("</td></tr>");
        }
        sb.append("</table>");
    }
    sb.append("</html>");
    showMessageDialog(sb.toString());
}

```

```

}
/**
 * Searches for books matching a query.
 */
private void searchBooks() {
    String query = showInputDialog("Enter book title to search:");
    if (query != null && !query.trim().isEmpty()) {
        StringBuilder sb = new StringBuilder("<html><h2>Search Results</h2>");
        boolean found = false;
        sb.append("<table border='0'
cellpadding='5'><tr><th>Title</th><th>Status</th></tr>");
        for (String book : books) {
            if (book.toLowerCase().contains(query.toLowerCase())) {
                sb.append("<tr><td>").append(book).append("</td><td>");
                if (loans.containsKey(book)) {
                    sb.append("<font color='red'>Borrowed</font>");
                } else {
                    sb.append("<font color='green'>Available</font>");
                }
                sb.append("</td></tr>");
                found = true;
            }
        }
        sb.append("</table>");
        if (!found) {
            sb.append("<p>No books found matching
'").append(query).append("</p>");
        }
        sb.append("</html>");
        showMessageDialog(sb.toString());
    }
}
/**
 * Allows a member to borrow a book.
 */
private void borrowBook() {
    String member = showInputDialog("Enter your name:");
    if (member != null && !member.trim().isEmpty()) {
        if (members.contains(member)) {
            String book = showInputDialog("Enter book title:");
            if (book != null && !book.trim().isEmpty()) {
                if (books.contains(book)) {
                    if (!loans.containsKey(book)) {
                        loans.put(book, member);
                        showMessageDialog("Book borrowed successfully!");
                    } else {
                        showErrorDialog("Book is already borrowed!");
                    }
                }
            }
        }
    }
}

```

```

        } else {
            showErrorDialog("Book not found!");
        }
    }
} else {
    showErrorDialog("Member not found!");
}
}
}
/**
 * Allows a member to return a book.
 */
private void returnBook() {
    String book = showInputDialog("Enter book title to return:");
    if (book != null && !book.trim().isEmpty()) {
        if (loans.containsKey(book)) {
            loans.remove(book);
            showMessageDialog("Book returned successfully!");
        } else {
            showErrorDialog("This book wasn't borrowed or doesn't exist!");
        }
    }
}
/**
 * Allows a member to pay fines.
 */
private void payFine() {
    String member = showInputDialog("Enter your name:");
    if (member != null && !member.trim().isEmpty()) {
        if (fines.containsKey(member)) {
            double amount = fines.get(member);
            showMessageDialog("Your current fine is: $" + String.format("%.2f",
amount));

            String paymentStr = showInputDialog("Enter payment amount:");
            try {
                double payment = Double.parseDouble(paymentStr);
                if (payment >= amount) {
                    fines.remove(member);
                    showMessageDialog("Fine paid in full. Thank you!");
                } else {
                    fines.put(member, amount - payment);
                    showMessageDialog("Partial payment received. Remaining
balance: $" +
                        String.format("%.2f", (amount - payment)));
                }
            } catch (NumberFormatException e) {
                showErrorDialog("Invalid payment amount!");
            }
        }
    }
}

```



```

        } else {
            showMessageDialog("No fines found for this member.");
        }
    }
}
/**
 * Displays a member's loans and fines.
 */
private void viewMemberLoansAndFines() {
    String member = showInputDialog("Enter your name:");
    if (member != null && !member.trim().isEmpty()) {
        StringBuilder sb = new StringBuilder();
        sb.append("<html><h2>Your Account Summary</h2>");

        if (fines.containsKey(member)) {
            sb.append("<p><b>Outstanding Fines:</b>");
            sb.append(String.format("%.2f", fines.get(member))).append("</p>");
        } else {
            sb.append("<p>You have no outstanding fines.</p>");
        }

        sb.append("<h3>Books Currently Borrowed:</h3><ul>");
        boolean hasLoans = false;
        for (Map.Entry<String, String> entry : loans.entrySet()) {
            if (entry.getValue().equals(member)) {
                sb.append("<li>").append(entry.getKey()).append("</li>");
                hasLoans = true;
            }
        }
        if (!hasLoans) {
            sb.append("<li>You have no books currently borrowed.</li>");
        }
        sb.append("</ul></html>");

        showMessageDialog(sb.toString());
    }
}
// ===== HELPER METHODS ===== //
/**
 * Shows an input dialog with formatted message.
 * @param message The message to display
 * @return User input or null if canceled
 */
private String showInputDialog(String message) {
    return JOptionPane.showInputDialog(frame,
        "<html><div style='width:300px;'>" + message + "</div></html>");
}
/**

```

```

    * Shows a message dialog with formatted message.
    * @param message The message to display
    */
    private void showMessageDialog(String message) {
        JOptionPane.showMessageDialog(frame,
            "<html><div style='width:300px;'>" + message + "</div></html>",
            "Information", JOptionPane.INFORMATION_MESSAGE);
    }
    /**
    * Shows an error dialog with formatted message.
    * @param message The error message to display
    */
    private void showErrorDialog(String message) {
        JOptionPane.showMessageDialog(frame,
            "<html><div style='width:300px;color:red;'>" + message + "</div></html>",
            "Error", JOptionPane.ERROR_MESSAGE);
    }
    /**
    * Shows a custom panel in a dialog.
    * @param panel The panel to display
    * @param title The dialog title
    */
    private void showDialogPanel(JPanel panel, String title) {
        JDialog dialog = new JDialog(frame, title, true);
        dialog.setContentPane(panel);
        dialog.setSize(400, 300);
        dialog.setLocationRelativeTo(frame);
        dialog.setVisible(true);
    }
    /**
    * Custom border class for rounded corners.
    */
    class RoundBorder extends AbstractBorder {
        private Color color;
        private int radius;

        public RoundBorder(int radius) {
            this(Color.GRAY, radius);
        }

        public RoundBorder(Color color, int radius) {
            this.color = color;
            this.radius = radius;
        }

        @Override
        public void paintBorder(Component c, Graphics g, int x, int y, int width,
int height) {

```

```

        Graphics2D g2 = (Graphics2D) g.create();
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setColor(color);
        g2.drawRoundRect(x, y, width - 1, height - 1, radius, radius);
        g2.dispose();
    }

    @Override
    public Insets getBorderInsets(Component c) {
        return new Insets(radius + 1, radius + 1, radius + 1, radius + 1);
    }

    @Override
    public Insets getBorderInsets(Component c, Insets insets) {
        insets.left = insets.right = radius + 1;
        insets.top = insets.bottom = radius + 1;
        return insets;
    }
}

/**
 * Main method to launch the application.
 * @param args Command line arguments (not used)
 */
public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new LibraryManagementSystem());
}
}

```

## 7-Github Repository:

Link:

<https://github.com/abdulwaris707/LibraryManagementSystem.git>