# LIBRARY MANAGEMENT

# Introduction

The Library Management System is a simple desktop application developed in Java using Swing. It allows users to manage a list of books by adding, deleting, and searching for them efficiently. The system provides a graphical user interface to make operations user-friendly and easy to handle.

# Abstract

This project provides a basic library management application that stores and displays book information such as Book ID, Title, and Author. It includes options for adding new books, deleting existing ones, and searching books based on their title. The system is built using Java Swing and demonstrates GUI-based programming concepts.

# Existing System

In traditional systems, managing books in a library is done manually using registers or spreadsheets. This method is time-consuming and prone to errors. Searching for or deleting a specific book takes more effort, and maintaining large records becomes difficult as the library grows.

# Proposed System

The proposed Library Management Application provides an interactive GUI interface to manage library records easily. Users can perform CRUD operations (Create, Read, Update, Delete) efficiently. It is user-friendly, quick, and reduces the manual effort involved in book management.

# Software Requirements

1. Operating System: Windows 10 or later  
2. Programming Language: Java (JDK 8 or higher)  
3. IDE: Eclipse / IntelliJ IDEA / NetBeans  
4. Libraries: Java Swing and AWT packages

# Hardware Requirements

1. Processor: Intel i3 or higher  
2. RAM: Minimum 2 GB  
3. Hard Disk: Minimum 100 MB free space  
4. Display: Minimum 1024x768 resolution

# Source Code

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

class Book {

    int id;

    String title;

    String author;

    Book(int id, String title, String author) {

        this.id = id;

        this.title = title;

        this.author = author;

    }

}

public class LibraryManagementApp extends JFrame {

    private JTextField idField, titleField, authorField, searchField;

    private JTable table;

    private DefaultTableModel model;

    private ArrayList<Book> library = new ArrayList<>();

    public LibraryManagementApp() {

        setTitle("📚 Library Management Application");

        setSize(650, 500);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new BorderLayout(10, 10));

        setLocationRelativeTo(null);

        // ===== Top Panel for Adding Books =====

        JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

        topPanel.add(new JLabel("Book ID:"));

        idField = new JTextField(5);

        topPanel.add(idField);

        topPanel.add(new JLabel("Title:"));

        titleField = new JTextField(10);

        topPanel.add(titleField);

        topPanel.add(new JLabel("Author:"));

        authorField = new JTextField(10);

        topPanel.add(authorField);

        JButton addButton = new JButton("Add Book");

        topPanel.add(addButton);

        JButton deleteButton = new JButton("Delete Book");

        topPanel.add(deleteButton);

        add(topPanel, BorderLayout.NORTH);

        // ===== Center Table to Display Books =====

        String[] columns = {"Book ID", "Title", "Author"};

        model = new DefaultTableModel(columns, 0);

        table = new JTable(model);

        JScrollPane scrollPane = new JScrollPane(table);

        add(scrollPane, BorderLayout.CENTER);

        // ===== Bottom Panel for Searching =====

        JPanel bottomPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

        bottomPanel.add(new JLabel("Search by Title:"));

        searchField = new JTextField(12);

        bottomPanel.add(searchField);

        JButton searchButton = new JButton("Search");

        bottomPanel.add(searchButton);

        JButton refreshButton = new JButton("Show All");

        bottomPanel.add(refreshButton);

        add(bottomPanel, BorderLayout.SOUTH);

        // ====== Button Actions ======

        addButton.addActionListener(e -> addBook());

        deleteButton.addActionListener(e -> deleteBook());

        searchButton.addActionListener(e -> searchBook());

        refreshButton.addActionListener(e -> refreshTable());

        setVisible(true);

    }

    // Add book to list and table

    private void addBook() {

        try {

            int id = Integer.parseInt(idField.getText());

            String title = titleField.getText().trim();

            String author = authorField.getText().trim();

            if (title.isEmpty() || author.isEmpty()) {

                JOptionPane.showMessageDialog(this, "Please fill all fields!", "Error", JOptionPane.ERROR\_MESSAGE);

                return;

            }

            library.add(new Book(id, title, author));

            model.addRow(new Object[]{id, title, author});

            JOptionPane.showMessageDialog(this, "Book added successfully!");

            idField.setText("");

            titleField.setText("");

            authorField.setText("");

        } catch (NumberFormatException ex) {

            JOptionPane.showMessageDialog(this, "Book ID must be a number!", "Error", JOptionPane.ERROR\_MESSAGE);

        }

    }

    // Delete book by ID

    private void deleteBook() {

        int selectedRow = table.getSelectedRow();

        if (selectedRow != -1) {

            int id = (int) model.getValueAt(selectedRow, 0);

            library.removeIf(b -> b.id == id);

            model.removeRow(selectedRow);

            JOptionPane.showMessageDialog(this, "Book deleted successfully!");

        } else {

            JOptionPane.showMessageDialog(this, "Please select a book to delete.");

        }

    }

    // Search book by title

    private void searchBook() {

        String searchText = searchField.getText().trim().toLowerCase();

        if (searchText.isEmpty()) {

            JOptionPane.showMessageDialog(this, "Please enter a title to search!");

            return;

        }

        DefaultTableModel searchModel = new DefaultTableModel(new String[]{"Book ID", "Title", "Author"}, 0);

        for (Book b : library) {

            if (b.title.toLowerCase().contains(searchText)) {

                searchModel.addRow(new Object[]{b.id, b.title, b.author});

            }

        }

        table.setModel(searchModel);

    }

    // Show all books

    private void refreshTable() {

        model.setRowCount(0);

        for (Book b : library) {

            model.addRow(new Object[]{b.id, b.title, b.author});

        }

        table.setModel(model);

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(LibraryManagementApp::new);

    }

}

Output

When the application runs, a graphical interface appears with options to:  
1. Add new books with their ID, title, and author.  
2. View all books in a table format.  
3. Search for books by title.  
4. Delete any selected book from the table.

# Conclusion

The Library Management System provides an efficient way to manage library records digitally. It helps users maintain a list of books, search them easily, and perform operations quickly using a user-friendly GUI interface.