**LIBRARY MANAGEMENT SYSTEM CODE**  
**FRONT PAGE CODING**

CODE

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class LibraryManagement extends JFrame implements ActionListener {

    private JLabel label1, label2, label3, label4, label5, label6, label7;

    private JTextField textField1, textField2, textField3, textField4, textField5, textField6, textField7;

    public JButton addButton, viewButton, editButton, deleteButton, clearButton, exitButton, orderButton;

    private JPanel mainPanel;

    private Connection connection;

    public LibraryManagement() {

        setTitle("Library Management System");

        setSize(1000, 600);

        setDefaultCloseOperation(EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        mainPanel = new JPanel() {

            private Image backgroundImage = new ImageIcon("book.jpg").getImage();

            protected void paintComponent(Graphics g) {

                super.paintComponent(g);

                g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);

            }

        };

        mainPanel.setLayout(new GridBagLayout());

        Font font = new Font("Arial", Font.BOLD, 20);

        label1 = new JLabel("Book ID");

        label2 = new JLabel("Book Title");

        label3 = new JLabel("Author");

        label4 = new JLabel("Publisher");

        label5 = new JLabel("Year of Publication");

        label6 = new JLabel("ISBN");

        label7 = new JLabel("Number of Copies");

        label1.setFont(font);

        label2.setFont(font);

        label3.setFont(font);

        label4.setFont(font);

        label5.setFont(font);

        label6.setFont(font);

        label7.setFont(font);

        label1.setForeground(Color.BLACK);

        label2.setForeground(Color.BLACK);

        label3.setForeground(Color.BLACK);

        label4.setForeground(Color.BLACK);

        label5.setForeground(Color.BLACK);

        label6.setForeground(Color.BLACK);

        label7.setForeground(Color.BLACK);

        textField1 = new JTextField(20);

        textField2 = new JTextField(20);

        textField3 = new JTextField(20);

        textField4 = new JTextField(20);

        textField5 = new JTextField(20);

        textField6 = new JTextField(20);

        textField7 = new JTextField(20);

        textField1.setFont(font);

        textField2.setFont(font);

        textField3.setFont(font);

        textField4.setFont(font);

        textField5.setFont(font);

        textField6.setFont(font);

        textField7.setFont(font);

        addButton = createHoverButton("Add", new Color(102, 205, 170), font);

        viewButton = createHoverButton("View", new Color(100, 149, 237), font);

        editButton = createHoverButton("Edit", new Color(255, 165, 0), font);

        deleteButton = createHoverButton("Delete", new Color(255, 69, 0), font);

        clearButton = createHoverButton("Clear", new Color(210, 180, 140), font);

        exitButton = createHoverButton("Exit", new Color(220, 20, 60), font);

        orderButton = createHoverButton("Order", new Color(255, 215, 0), font);

        addButton.addActionListener(this);

        viewButton.addActionListener(this);

        editButton.addActionListener(this);

        deleteButton.addActionListener(this);

        clearButton.addActionListener(this);

        exitButton.addActionListener(this);

        orderButton.addActionListener(this);

        GridBagConstraints gbc = new GridBagConstraints();

        gbc.insets = new Insets(5, 5, 5, 5);

        gbc.gridx = 0;

        gbc.gridy = 0;

        mainPanel.add(label1, gbc);

        gbc.gridx++;

        mainPanel.add(textField1, gbc);

        gbc.gridx = 0;

        gbc.gridy++;

        mainPanel.add(label2, gbc);

        gbc.gridx++;

        mainPanel.add(textField2, gbc);

        JPanel buttonPanel = new JPanel(new FlowLayout());

        buttonPanel.setOpaque(false); // Transparent background for button panel

        buttonPanel.add(addButton);

        buttonPanel.add(viewButton);

        buttonPanel.add(editButton);

        buttonPanel.add(deleteButton);

        buttonPanel.add(clearButton);

        buttonPanel.add(orderButton);

        buttonPanel.add(exitButton);

        mainPanel.add(buttonPanel, gbc);

        add(mainPanel);

        setVisible(true);

        connectToDatabase();

    }

    private JButton createHoverButton(String text, Color color, Font font) {

        JButton button = new JButton(text);

        button.setBackground(color);

        button.setForeground(Color.WHITE);

        button.setFont(font);

        button.setFocusPainted(false);

        button.setBorderPainted(false);

        button.addMouseListener(new java.awt.event.MouseAdapter() {

            public void mouseEntered(java.awt.event.MouseEvent evt) {

                button.setBackground(color.darker());

            }

            public void mouseExited(java.awt.event.MouseEvent evt) {

                button.setBackground(color);

            }

        });

        return button;

    }

    private void connectToDatabase() {

        try {

            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/bookshopping", "root", "Ruhith@12115");

        } catch (SQLException ex) {

            JOptionPane.showMessageDialog(this, "Database connection failed: " + ex.getMessage());

        }

    }

    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == addButton) {

            addBook();

        } else if (e.getSource() == viewButton) {

            viewBooks();

        } else if (e.getSource() == editButton) {

            editBook();

        } else if (e.getSource() == deleteButton) {

            deleteBook();

        } else if (e.getSource() == clearButton) {

            clearFields();

        } else if (e.getSource() == exitButton) {

            System.exit(0);

        } else if (e.getSource() == orderButton) {

            orderBook();

        }

    }

    private void addBook() {

        if (textField1.getText().isEmpty() || textField2.getText().isEmpty() || textField3.getText().isEmpty() ||

        textField4.getText().isEmpty() || textField5.getText().isEmpty() || textField6.getText().isEmpty() |

        textField7.getText().isEmpty()) {

        JOptionPane.showMessageDialog(this, "Please fill in all fields before adding a book");

        return;

    }

    int yearOfPublication, numberOfCopies;

    try {

        yearOfPublication = Integer.parseInt(textField5.getText());

        numberOfCopies = Integer.parseInt(textField7.getText());

    } catch (NumberFormatException ex) {

        JOptionPane.showMessageDialog(this, "Year of Publication and Number of Copies must be valid integers");

        return;

    }

    String query = "INSERT INTO shoping (book\_id, title, author, publisher, year\_of\_publication, isbn, number\_of\_copies) VALUES (?, ?, ?, ?, ?, ?, ?)";

    try (PreparedStatement statement = connection.prepareStatement(query)) {

        statement.setString(1, textField1.getText());

        statement.setString(2, textField2.getText());

        statement.setString(3, textField3.getText());

        statement.setString(4, textField4.getText());

        statement.setInt(5, yearOfPublication);

        statement.setString(6, textField6.getText());

        statement.setInt(7, numberOfCopies);

        statement.executeUpdate();

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

        clearFields();

    } catch (SQLException ex) {

        JOptionPane.showMessageDialog(this, "Error adding book: " + ex.getMessage());

    }

    }

    private void viewBooks() {

        String query = "SELECT \* FROM shoping";

        try (Statement statement = connection.createStatement();

             ResultSet resultSet = statement.executeQuery(query)) {

            String[] columns = {"Book ID", "Book Title", "Author", "Publisher", "Year of Publication", "ISBN", "Number of Copies"};

            DefaultTableModel model = new DefaultTableModel(columns, 0);

            while (resultSet.next()) {

                String[] book = {

                    resultSet.getString("book\_id"),

                    resultSet.getString("title"),

                    resultSet.getString("author"),

                    resultSet.getString("publisher"),

                    resultSet.getString("year\_of\_publication"),

                    resultSet.getString("isbn"),

                    resultSet.getString("number\_of\_copies")

                };

                model.addRow(book);

            }

            JTable table = new JTable(model);

            table.setFont(new Font("Arial", Font.PLAIN, 18));

            table.setRowHeight(30);

            JFrame frame = new JFrame("View Books");

            frame.add(new JScrollPane(table));

            frame.setSize(1000, 700);

            frame.setLocationRelativeTo(this);

            frame.setVisible(true);

        } catch (SQLException ex) {

            JOptionPane.showMessageDialog(this, "Error retrieving books: " + ex.getMessage());

        }

    }

    private void editBook() {

        String bookID = textField1.getText();

    if (bookID.isEmpty()) {

        JOptionPane.showMessageDialog(this, "Please enter a Book ID to edit.");

        return;

    }

    String query = "UPDATE shoping SET title = ?, author = ?, publisher = ?, year\_of\_publication = ?, isbn = ?, number\_of\_copies = ? WHERE book\_id = ?";

    try (PreparedStatement statement = connection.prepareStatement(query)) {

        // Validate and parse integer fields for Year of Publication and Number of Copies

        int yearOfPublication, numberOfCopies;

        try {

            yearOfPublication = Integer.parseInt(textField5.getText());

            numberOfCopies = Integer.parseInt(textField7.getText());

        } catch (NumberFormatException ex) {

            JOptionPane.showMessageDialog(this, "Year of Publication and Number of Copies must be valid integers");

            return;

        }

        statement.setString(1, textField2.getText()); // title

        statement.setString(2, textField3.getText()); // author

        statement.setString(3, textField4.getText()); // publisher

        statement.setInt(4, yearOfPublication);       // year\_of\_publication

        statement.setString(5, textField6.getText()); // isbn

        statement.setInt(6, numberOfCopies);          // number\_of\_copies

        statement.setString(7, bookID);               // book\_id

        int rowsUpdated = statement.executeUpdate();

        if (rowsUpdated > 0) {

            JOptionPane.showMessageDialog(this, "Book updated successfully");

        } else {

            JOptionPane.showMessageDialog(this, "No book found with the provided Book ID");

        }

        clearFields();

    } catch (SQLException ex) {

        JOptionPane.showMessageDialog(this, "Error updating book: " + ex.getMessage());

    }

    }

    private void deleteBook() {

        String bookID = textField1.getText();

    if (bookID.isEmpty()) {

        JOptionPane.showMessageDialog(this, "Please enter a Book ID to delete.");

        return;

    }

    int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this book?", "Confirm Deletion", JOptionPane.YES\_NO\_OPTION);

    if (confirm != JOptionPane.YES\_OPTION) {

        return;

    }

    String query = "DELETE FROM shoping WHERE book\_id = ?";

    try (PreparedStatement statement = connection.prepareStatement(query)) {

        statement.setString(1, bookID); // book\_id

        int rowsDeleted = statement.executeUpdate();

        if (rowsDeleted > 0) {

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

        } else {

            JOptionPane.showMessageDialog(this, "No book found with the provided Book ID");

        }

        clearFields();

    } catch (SQLException ex) {

        JOptionPane.showMessageDialog(this, "Error deleting book: " + ex.getMessage());

    }

    }

    private void orderBook() {

        Bookshop bookShopApp = new Bookshop();

        bookShopApp.setVisible(true);

        this.dispose();

    }

    private void clearFields() {

        textField1.setText("");

        textField2.setText("");

    }

    public static void main(String[] args) {

        new LibraryManagement();

    }

}

**ORDER PAGE CODING**

**CODING**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

public class Bookshop extends JFrame {

    private JTable bookTable;

    private DefaultTableModel model;

    private JTextField quantityField;

    private JButton orderButton, orderInfoButton;

    private static final String DB\_URL\_ORDERS = "jdbc:mysql://localhost:3306/book\_orders";

    private static final String USER = "root";

    private static final String PASSWORD = "Ruhith@12115";

    private static final String DB\_URL\_BOOKSHOP = "jdbc:mysql://localhost:3306/bookshopping";

    public Bookshop() {

        setTitle("Book Shopping System");

        setSize(1000, 600);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel() {

            private Image backgroundImage = new ImageIcon("order.jpg").getImage();

            protected void paintComponent(Graphics g) {

                super.paintComponent(g);

                g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);

            }

        };

        mainPanel.setLayout(null);

        model = new DefaultTableModel(new String[]{"ID", "Title", "Author", "Publisher", "Year", "ISBN", "Copies"}, 0);

        bookTable = new JTable(model);

        loadBooksFromDatabase();

        JScrollPane scrollPane = new JScrollPane(bookTable);

        scrollPane.setBounds(30, 30, 900, 300);

        mainPanel.add(scrollPane);

        JLabel quantityLabel = new JLabel("Quantity:");

        quantityLabel.setBounds(30, 350, 100, 30);

        quantityLabel.setForeground(Color.WHITE); // Make label text visible over the background

        mainPanel.add(quantityLabel);

        quantityField = new JTextField(5);

        quantityField.setBounds(130, 350, 50, 30);

        mainPanel.add(quantityField);

        orderButton = new JButton("Order Book");

        orderButton.setBounds(200, 350, 150, 30);

        orderButton.setBackground(Color.GREEN);

        orderButton.setForeground(Color.WHITE);

        orderButton.addActionListener(e -> openOrderDetails());

        mainPanel.add(orderButton);

        orderInfoButton = new JButton("Order Info");

        orderInfoButton.setBounds(400, 350, 150, 30);

        orderInfoButton.setBackground(Color.BLUE);

        orderInfoButton.setForeground(Color.WHITE);

        orderInfoButton.addActionListener(e -> viewOrderInfo());

        mainPanel.add(orderInfoButton);

        add(mainPanel);

    }

    private void loadBooksFromDatabase() {

        model.setRowCount(0);  // Clear the table

        try (Connection conn = DriverManager.getConnection(DB\_URL\_BOOKSHOP, USER, PASSWORD)) {

            String query = "SELECT \* FROM shoping";

            Statement stmt = conn.createStatement();

            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {

                int bookId = rs.getInt("book\_id");

                String title = rs.getString("title");

                String author = rs.getString("author");

                String publisher = rs.getString("publisher");

                int year = rs.getInt("year\_of\_publication");

                String isbn = rs.getString("isbn");

                int copies = rs.getInt("number\_of\_copies");

                model.addRow(new Object[]{bookId, title, author, publisher, year, isbn, copies});

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    private void openOrderDetails() {

        int selectedRow = bookTable.getSelectedRow();

        if (selectedRow == -1) {

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

            return;

        }

        String bookId = model.getValueAt(selectedRow, 0).toString();

        int quantity = Integer.parseInt(quantityField.getText());

        JFrame orderFrame = new JFrame("Order Details");

        orderFrame.setSize(500, 400);

        orderFrame.setLayout(null);

        orderFrame.setLocationRelativeTo(null);

        orderFrame.getContentPane().setBackground(new Color(250, 235, 215)); // Light vibrant color

        JLabel nameLabel = new JLabel("Customer Name:");

        nameLabel.setBounds(50, 50, 150, 30);

        JTextField nameField = new JTextField();

        nameField.setBounds(200, 50, 200, 30);

        JLabel addressLabel = new JLabel("Address:");

        addressLabel.setBounds(50, 100, 150, 30);

        JTextField addressField = new JTextField();

        addressField.setBounds(200, 100, 200, 30);

        JLabel phoneLabel = new JLabel("Phone Number:");

        phoneLabel.setBounds(50, 150, 150, 30);

        JTextField phoneField = new JTextField();

        phoneField.setBounds(200, 150, 200, 30);

        JLabel paymentLabel = new JLabel("Payment Type:");

        paymentLabel.setBounds(50, 200, 150, 30);

        String[] paymentTypes = {"Credit Card", "Debit Card", "PayPal", "Cash"};

        JComboBox<String> paymentTypeComboBox = new JComboBox<>(paymentTypes);

        paymentTypeComboBox.setBounds(200, 200, 200, 30);

        JButton confirmOrderButton = new JButton("Confirm Order");

        confirmOrderButton.setBounds(150, 270, 200, 40);

        confirmOrderButton.setBackground(Color.GREEN);

        confirmOrderButton.setForeground(Color.WHITE);

        confirmOrderButton.addActionListener(e -> {

            String customerName = nameField.getText();

            String address = addressField.getText();

            String phone = phoneField.getText();

            String paymentType = (String) paymentTypeComboBox.getSelectedItem();

            if (customerName.isEmpty() || address.isEmpty() || phone.isEmpty()) {

                JOptionPane.showMessageDialog(orderFrame, "Please fill all fields.");

                return;

            }

            placeOrder(bookId, quantity, customerName, address, phone, paymentType);

            orderFrame.dispose(); // Close the order details frame

        });

        orderFrame.add(nameLabel);

        orderFrame.add(nameField);

        orderFrame.add(addressLabel);

        orderFrame.setVisible(true);

    }

    private void viewOrderInfo() {

        JFrame infoFrame = new JFrame("Order Information");

        infoFrame.setSize(600, 400);

        infoFrame.setLayout(new BorderLayout());

        infoFrame.setLocationRelativeTo(null);

        DefaultTableModel orderModel = new DefaultTableModel(new String[]{"Order ID", "Book ID", "Quantity", "Customer Name", "Address", "Phone", "Payment Type"}, 0);

        JTable orderTable = new JTable(orderModel);

        try (Connection conn = DriverManager.getConnection(DB\_URL\_ORDERS, USER, PASSWORD)) {

            String query = "SELECT \* FROM orders";

            Statement stmt = conn.createStatement();

            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {

                int orderId = rs.getInt("order\_id");

                int bookId = rs.getInt("book\_id");

                int quantity = rs.getInt("quantity");

                String customerName = rs.getString("customer\_name");

                String address = rs.getString("address");

                String phone = rs.getString("phone");

                String paymentType = rs.getString("payment\_type");

                orderModel.addRow(new Object[]{orderId, bookId, quantity, customerName, address, phone, paymentType});

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        infoFrame.add(new JScrollPane(orderTable), BorderLayout.CENTER);

        infoFrame.setVisible(true);

    }

    private void placeOrder(String bookId, int quantity, String customerName, String address, String phone, String paymentType) {

        // Connect to the 'book\_orders' database and insert the order details

        try (Connection conn = DriverManager.getConnection(DB\_URL\_ORDERS, USER, PASSWORD)) {

            String query = "INSERT INTO orders (book\_id, quantity, customer\_name, address, phone, payment\_type) VALUES (?, ?, ?, ?, ?, ?)";

            PreparedStatement pstmt = conn.prepareStatement(query);

            pstmt.setInt(1, Integer.parseInt(bookId));

            pstmt.setInt(2, quantity);

            pstmt.setString(3, customerName);

            pstmt.setString(4, address);

            pstmt.setString(5, phone);

            pstmt.setString(6, paymentType);

int rowsInserted = pstmt.executeUpdate();

            if (rowsInserted > 0) {

                JOptionPane.showMessageDialog(this, "Order placed successfully!");

            } else {

                JOptionPane.showMessageDialog(this, "Failed to place the order.");

            }

        } catch (SQLException e) {

            JOptionPane.showMessageDialog(this, "Error placing the order: " + e.getMessage());

            e.printStackTrace();

        }

    }

public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> new Bookshop().setVisible(true));

    }

}