LIBRARY MANAGEMENT AND BOOK EXCHANGE SYSTEM

A MINI-PROJECT BY:

Sivaruhith 230701320

Soundaryalakshmi 230701326

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project **"LIBRARY MANAGEMENT AND BOOK EXCHANGE SYSTEM"** is the
bonafide work of **"SIVA RUHITH S, SOUNDARYALAKSHMI S"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Mr.G.Saravana Gokul
Assistant Professor(SS)
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

SIGNATURE

Ms.V.JANANEE
Assistant Professor(SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINE

ABSTRACT

The Library Management and Book Exchange System is an integrated digital solution designed to enhance the efficiency of library operations while promoting community-driven book sharing. This platform offers a comprehensive approach to managing library resources, allowing administrators to handle tasks such as cataloging, book lending, member management, and generating reports, all within a centralized system. Users benefit from easy access to library materials, real-time book availability updates, and the ability to borrow, reserve, or exchange books with fellow members.

The system includes key features such as secure login and sign-in pages for both administrators and users, ensuring safe and personalized access. The Book Detail Page provides in-depth information on each book, including title, author, genre, availability, and user reviews, making it easy for users to discover books of interest. Administrators can add new books through the Add Book Design feature, inputting key details such as ISBN, publication date, and description, while the Delete Book Design allows for efficient management of outdated or damaged books in the catalog.

The View Book Design enables users to search and browse through the library's collection with advanced filters for genre, author, or availability, facilitating a smooth and tailored book discovery experience. The Order Page Design allows users to place orders for borrowing or exchanging books, ensuring a streamling process for book transactions.

By integrating the concept of a book exchange within the library management system, the platform not only increases the accessibility of books but also fosters sustainability by reducing waste and encouraging resource-sharing among community members. With features that prioritize ease of use, security, and engagement, the Library Management and Book Exchange System provides an innovative solution for modern libraries, enhancing both administrative efficiency and user experience.

TABLE OF CONTENTS

INTRODUCTION

1.1 INTRODUCTION

The **Library Management and Book Exchange System** is a modern platform designed to streamline library operations and provide a community-driven approach to sharing books. This system integrates traditional library management functions, such as cataloging, borrowing, and user management, with a novel book exchange feature, allowing users to exchange books with fellow members. The system aims to improve the efficiency of library operations by automating common tasks, reduce the manual effort of tracking books, and increase the accessibility of library resources. It also promotes a sustainable reading culture by facilitating the exchange of books, allowing users to enjoy a wider variety of books without having to purchase new ones.

The platform supports both library administrators and users, ensuring a smooth and seamless experience for both parties. For administrators, the system provides tools to manage the library catalog, track book transactions, and generate reports. For users, it offers an intuitive interface to search for books, borrow or exchange books, and maintain a personalized reading profile.

1.1 IMPLEMENTATION

The **LIBRARY MANAGEMENT AND BOOK EXCHANGE SYSTEM** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

1.2 WEBSITE FEATURES

- User Authentication
- Book Management
- Administrator Features
- Notifications and Reminders
- Search and Filter Functionality
- Responsive Design

.

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR	:	Inteli7
MEMORY SIZE	:	4GB(Minimum)
HARD DISK	:	500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE	:	Java, MySQL
FRONT-END	:	Java Swings
BACK-END	:	MySQL
OPERATING SYSTEM	:	Windows 11

SAMPLE CODE

3.1 FRONTEND 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();
    }
}
```

3.2 ORDER PAGE 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();
        }
    }
```

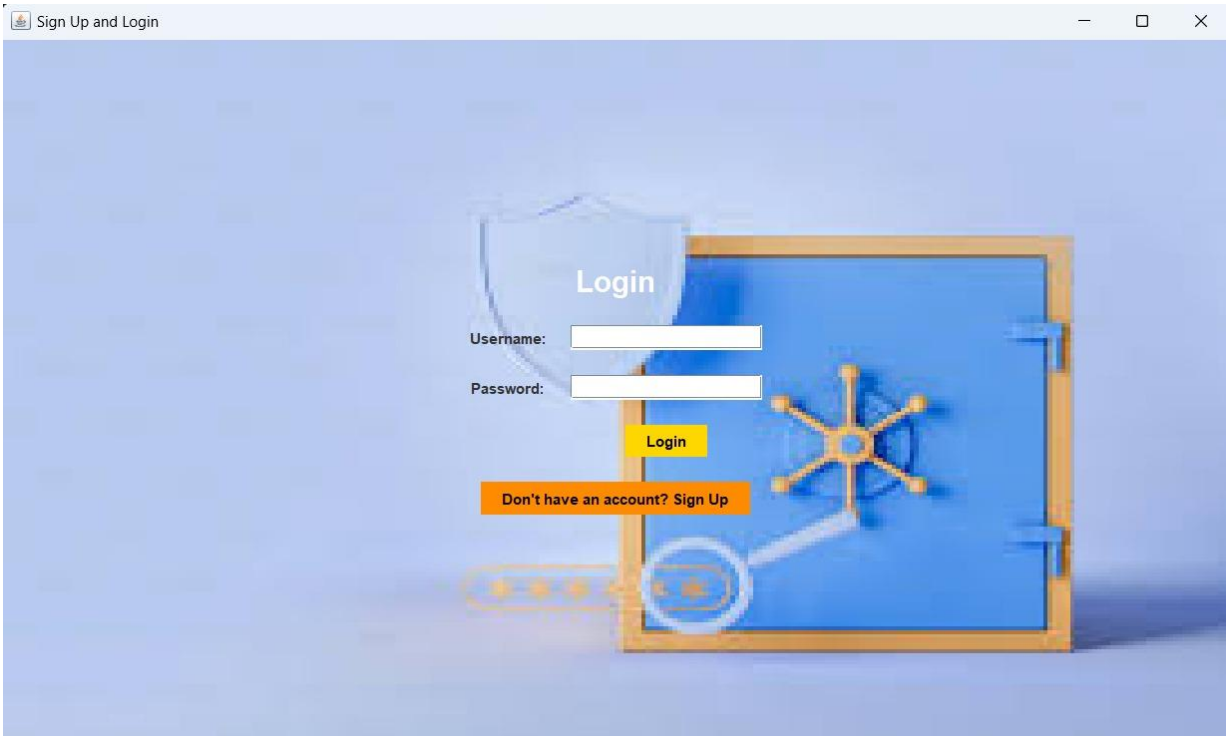
```
    private void openOrderDetails() {
        int selectedRow = bookTable.getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(this, "Please select a book to order.");
            return;
        }
        String bookId = model.getValueAt(selectedRow, 0).toString();
        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);
```

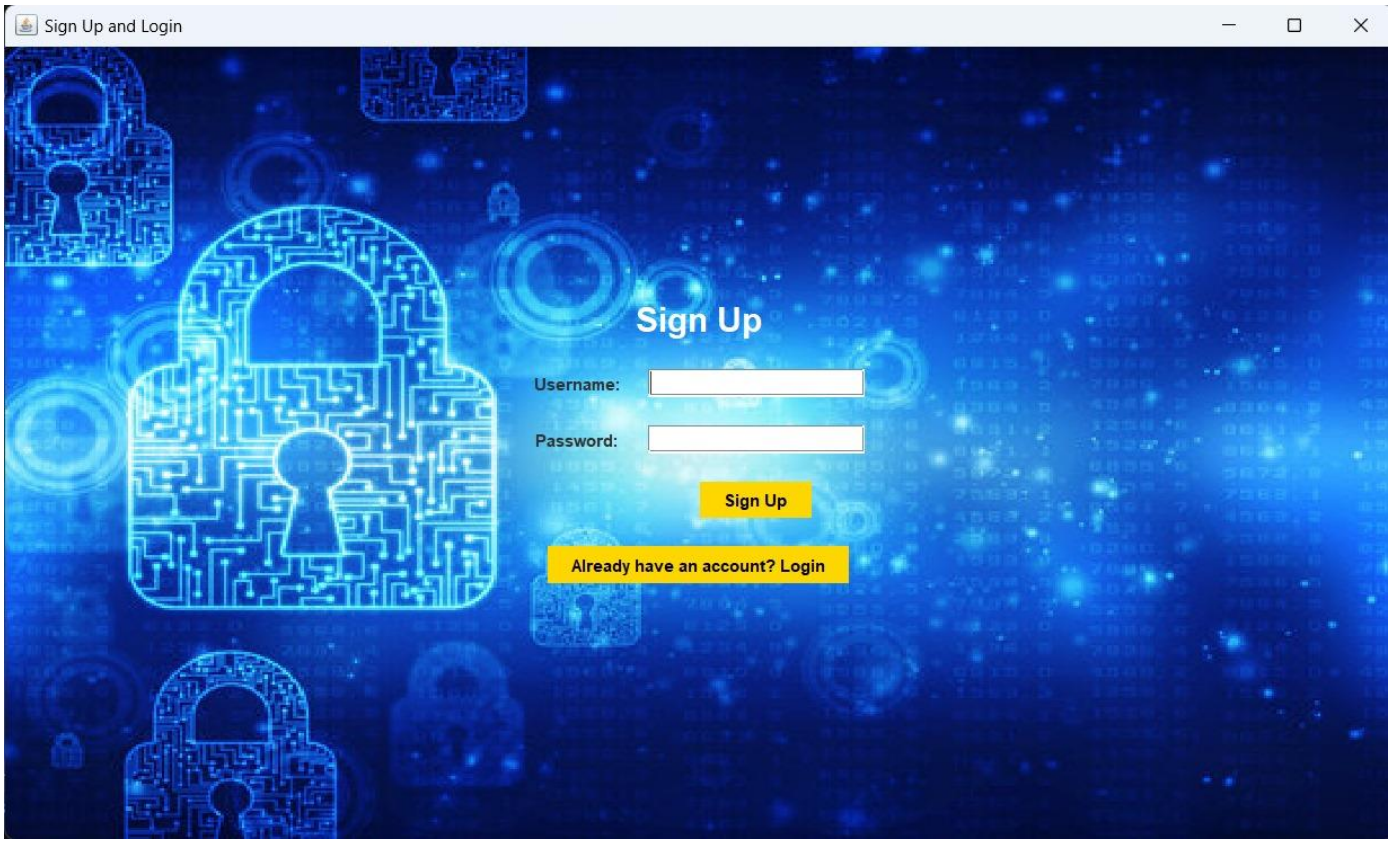
```
            pstmt.setString(4, address);
            pstmt.setString(5, phone);
            pstmt.setString(6, paymentType);
            int rowsInserted = pstmt.executeUpdate();
            if (rowsInserted > 0) {
                JOptionPane.showMessageDialog(this, "Order placed successfully!");
            } 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));
    }
}
```

SNAPSHOTS

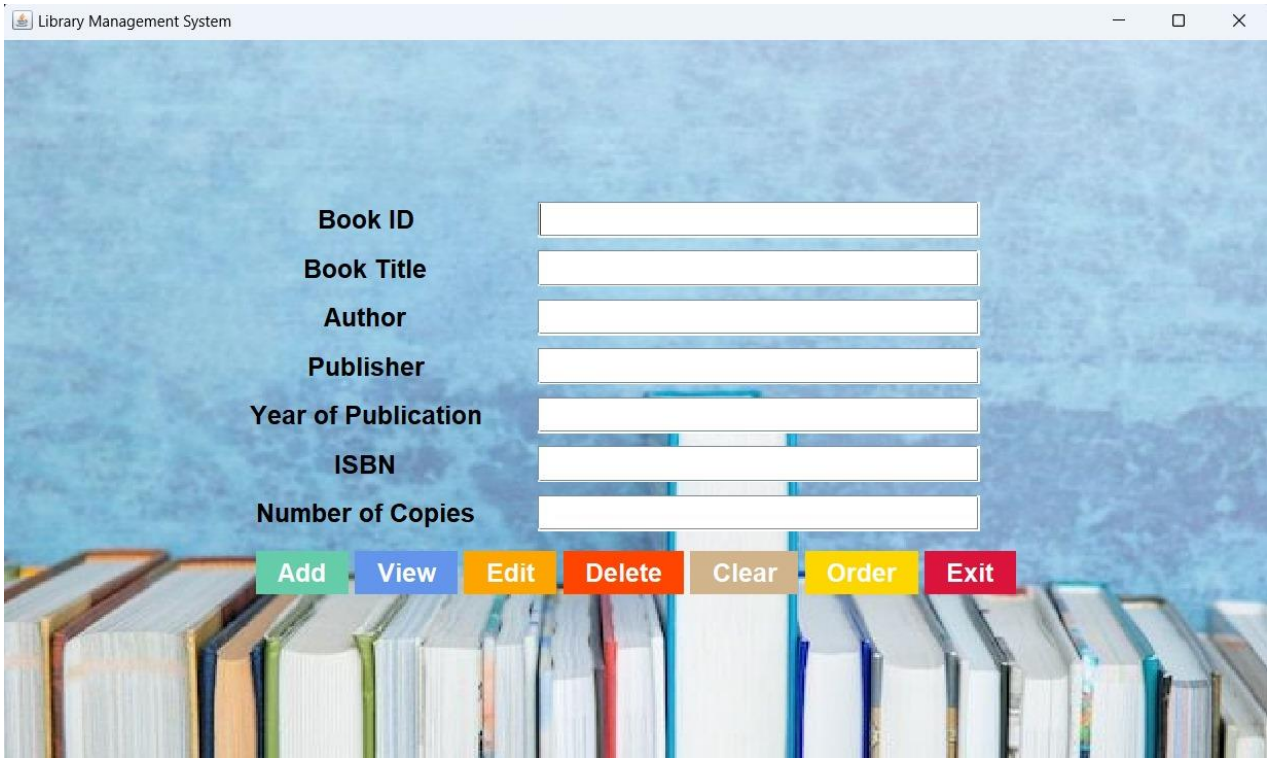
4.1 LOGIN PAGE



4.2 SIGN UP PAGE



4.3BOOK DETAILS PAGE

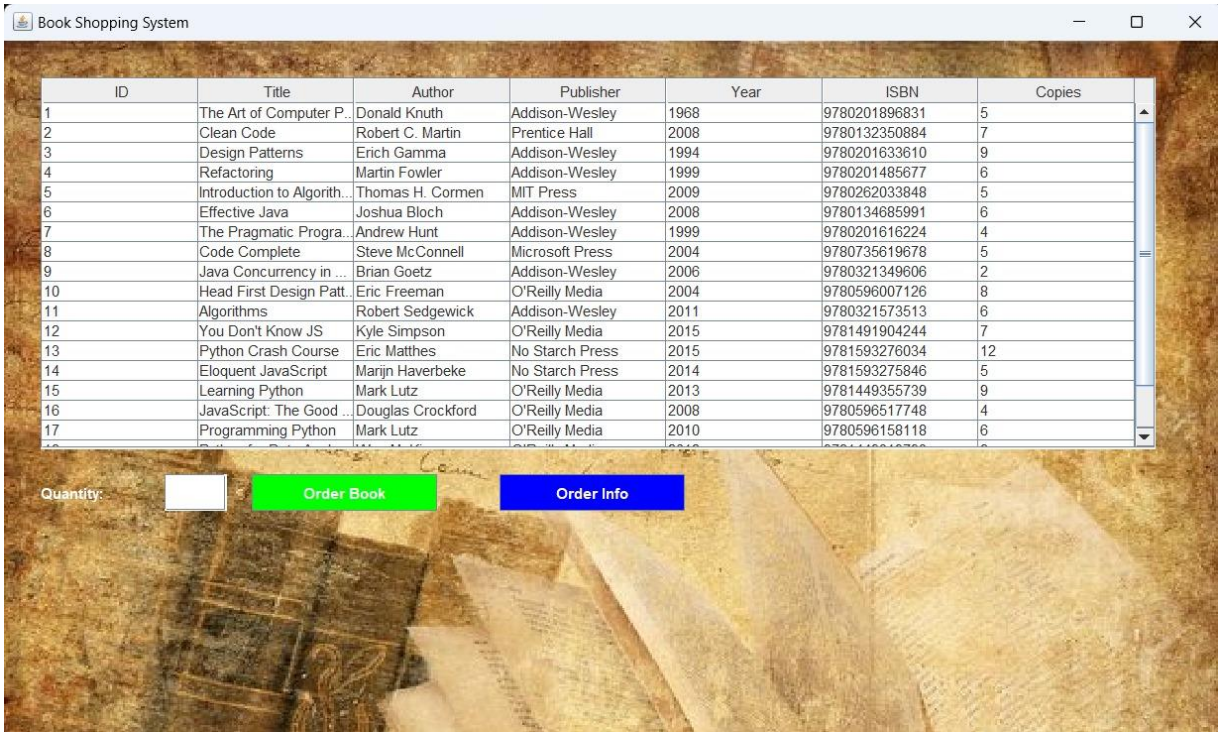


4.4 VIEW BOOK PAGE

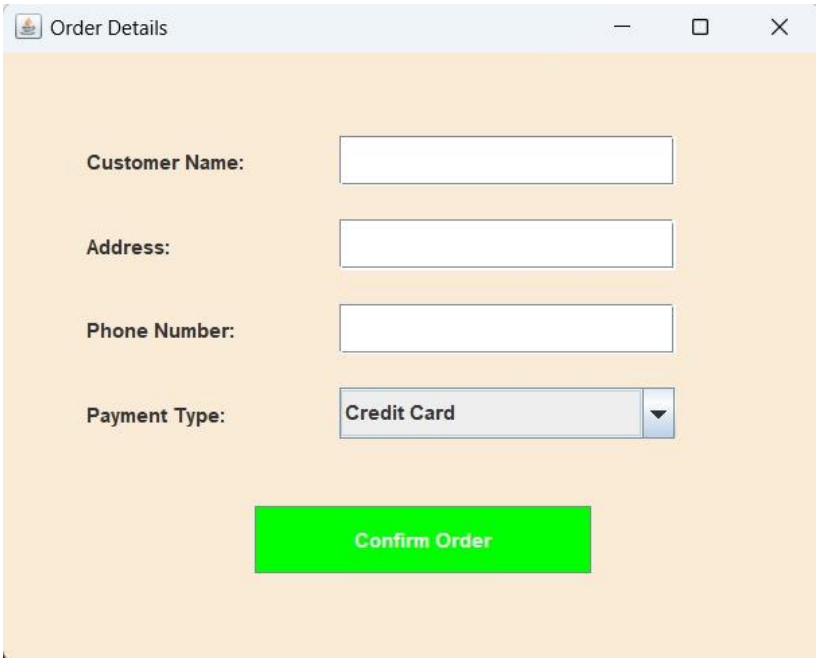
View Books

Book ID	Book Title	Author	Publisher	Year of Publication	ISBN	Number of Copies
1	The Art of Com...	Donald Knuth	Addison-Wesley	1968	9780201896831	5
2	Clean Code	Robert C. Martin	Prentice Hall	2008	9780132350884	7
3	Design Patterns	Erich Gamma	Addison-Wesley	1994	9780201633610	9
4	Refactoring	Martin Fowler	Addison-Wesley	1999	9780201485677	6
5	Introduction to ...	Thomas H. Co...	MIT Press	2009	9780262033848	5
6	Effective Java	Joshua Bloch	Addison-Wesley	2008	9780134685991	6
7	The Pragmatic ...	Andrew Hunt	Addison-Wesley	1999	9780201616224	4
8	Code Complete	Steve McConnell	Microsoft Press	2004	9780735619678	5
9	Java Concurre...	Brian Goetz	Addison-Wesley	2006	9780321349606	2
10	Head First Des...	Eric Freeman	O'Reilly Media	2004	9780596007126	8
11	Algorithms	Robert Sedge...	Addison-Wesley	2011	9780321573513	6
12	You Don't Kno...	Kyle Simpson	O'Reilly Media	2015	9781491904244	7
13	Python Crash ...	Eric Matthes	No Starch Press	2015	9781593276034	12
14	Eloquent Java...	Marijn Haverbe...	No Starch Press	2014	9781593275846	5
15	Learning Python	Mark Lutz	O'Reilly Media	2013	9781449355739	9
16	JavaScript: Th...	Douglas Crock...	O'Reilly Media	2008	9780596517748	4
17	Programming P...	Mark Lutz	O'Reilly Media	2010	9780596158118	6
18	Python for Dat...	Wes McKinney	O'Reilly Media	2012	9781449319793	8
19	Fluent Python	Luciano Ramal...	O'Reilly Media	2015	9781491946008	6
20	Automate the B...	Al Sweigart	No Starch Press	2015	9781593275990	7

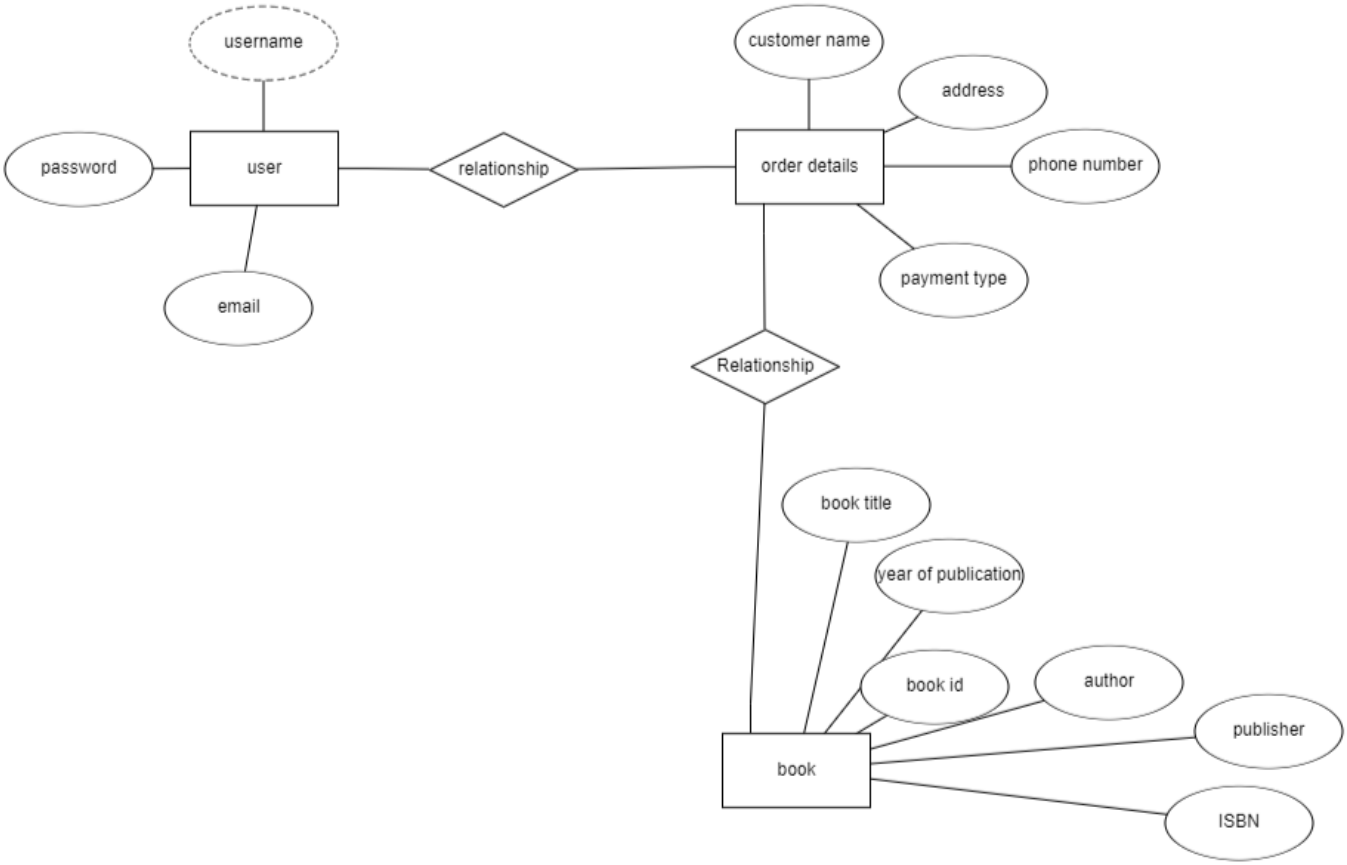
4.5 ORDER PAGE



4.6 ORDER DATAILS PAGE



ER DIAGRAM:



This system tracks users, books, and orders in an online bookstore or library management system. The **User** entity stores details about customers and administrators, including personal information like name, email, and role. The **Book** entity contains information about the books available, such as title, author, genre, and price.

The **Order** entity tracks the orders placed by users, linking them to a specific user via **UserID**. Each **Order** can contain multiple books, stored in the **OrderDetails** entity, which also tracks the quantity, price, and total cost for each book in the order. The **OrderDetails** links both the **Order** and **Book** entities, representing the specific books that are part of the order.

CONCLUSION

The Library Management and Book Exchange System provides an efficient and user-friendly solution for managing library operations while promoting a community-driven book exchange platform. The system features secure user authentication, seamless book management, and an intuitive interface for browsing, borrowing, and exchanging books. Admins can easily manage the catalog, user activities, and transactions. The inclusion of real-time notifications, search filters, and responsive design ensures a smooth user experience across devices.

A key highlight of this project is the Book Exchange feature, which encourages sustainability by allowing users to exchange books rather than buying new ones, fostering a sense of community. Overall, this system improves library operations, enhances user engagement, and offers a modern, scalable solution for managing books and orders.

REFERENCES

1. **https://www.javatpoint.com/java-tutorial**
2. https://www.wikipedia.org/
3. https://www.w3schools.com/sql/
4. SQL | Codecademy