

CS 521
Summer 2016
Object Oriented Analysis/Design

Team 9

Library Management System
Project Report

A20358054	Ramachandra, Tarakeswari	tramach1@hawk.iit.edu
A20355488	Masand, Roshni	rmasand@hawk.iit.edu
A20364489	More, Aditya	amore@hawk.iit.edu
A20359827	Prabhakaran, Pranisha	pprabha4@hawk.iit.edu

Table of contents

Sr. No.	Content	Page No.
1	Project Overview	3
2	Requirements and Feature List	4
3	Domain Dictionary	5
4	Actor Dictionary	6
5	Use Case Diagram	7
6	Analysis Model	9
7	Design Model	10
8	Sequence Diagram	11
9	Object-Relational Model	16
10	Documentation and class diagrams for Design Patterns used	17
11	Source Code	20
12	Screen Shots	76

1. Project Overview

This project aims to deliver an online Library Management System for managing and administering all the relevant operations in a library using advanced Object-Oriented Analysis and Design concepts. This system will support inventory management of the library books as well as manage multiple user roles such as – Member, Librarian and Administrator and each role will have a set of actions supported by the system.

Any visitor to the online Library Management portal (Guest user) will have the ability to register themselves in the system and become a Member. Once they are members, they can access or search the library for books on the basis of their Title, ISBN, Author Name or Domain and can Request, Return or Renew any book in the library. They can also view their account information, issued books, overdue books penalties and make appropriate payments.

The Member also has an additional option of subscribing to particular domains like “Computer Science”, “Robotics” etc. so that whenever a new book fitting this domain gets added to the system, the system will send out an alert email to the subscribed member.

The Librarian will have the ability to add, delete or modify the library inventory and also issue books for the member. The Library Administrator can manage the staff in the system and also add or delete books.

This Library Management system will be optimized for performance to ensure that all system actions and responses are executed in or under 2 seconds. The system will be easy to navigate without requiring any special training and it will not provide any incorrect details to its users. Additionally, controls will be in place to ensure that all users need to login before using the system with new users requiring signup/registration and Members with overdue late/damage charges cannot issue any new book till all payments are settled and they can only view/update their own account information.

The Library Management System will be developed using Swing (Java) and MySQL.

2. Requirements and Features List

Functional Requirements

- The system must allow the administrator to add new staff member or to remove staff member.
- The system must allow both the administrator and librarian to add and delete a book to/from the System.
- The system must allow the librarian to add or delete library member and to view the members list.
- The system should allow the manager just to view the library members list.
- The system must allow a guest to register for the system.
- The system must allow the library members to issue, return and reissue the book. Then the librarian must be able to take appropriate action to complete the order.
- The system must allow to search the books using either book title, ISBN author or domain.
- The system must allow users to register for the new book notification and the system must send the email notification to the registered users when the new book in the field of interest is added.
- The system must calculate the penalty for the overdue books and the members should be able to pay the amount.
- The system must allow the user to view the issued books and overdue books also.

Nonfunctional Requirements

- **Performance:**

The system should respond within 2 seconds. The system should not provide wrong details. The responses should be fast enough to avoid user response collisions.

- **Security:**

Users need to login before using the system.

Member can view only own account information and nothing about other user.

- **Usability:**

The system is easy to use. No special training is needed to use the system.

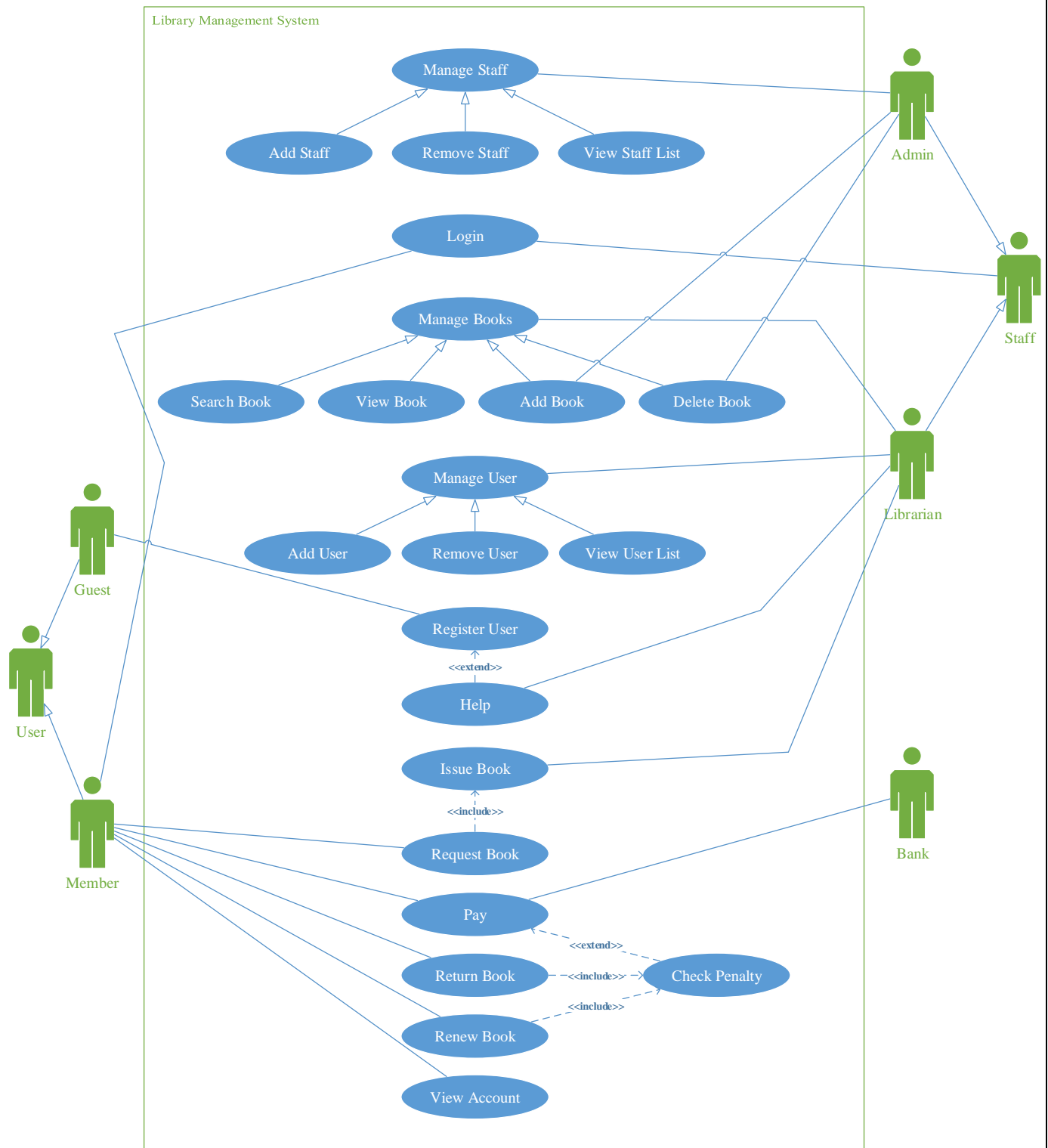
3. Domain Dictionary

Name	Type	Description
Staff	Role	Staff is the abstract super class for Administrator and Librarian.
Login	Function	The Member, Librarian and Administrator can login to the system to perform their respective operations.
Administrator	Role	The Administrator is responsible for managing staff, books and viewing user list.
Manage Books	Function	The Administrator and Librarian can search books, view books, add books and delete books.
View User List	Function	The Administrator and Librarian can view user list.
Manage Staff	Function	The Administrator can add staff, remove staff and view staff.
Librarian	Role	The Librarian is responsible for managing users and issuing the books.
Manage User	Function	The Librarian can add user, remove user and view user list.
Issue Book	Function	The Librarian issues the book requested by the Member.
Unregister User	Function	The Member can choose to be unregistered. The Librarian then checks the member records for penalties and then the user can be unregistered.
User	Role	User is an abstract super class which contains Member and Guest.
Member	Role	The Member can view book list, search book, request book, renew book, return book and pay fine if necessary.
Guest	Role	The Guest can register onto the library management system.
Register User	Function	Any user who wants to access the library system must be registered. The Guest can register onto the library management system. If they face any problem, the librarian can help them.
Pay	Function	The member pays a penalty if the book is returned later than the due date.
Return Book	Function	When the member returns a borrowed book, the dates are checked and accordingly the pending penalty is also calculated.
Check Penalty	Function	The penalty is checked when the user returns the borrowed book.
Request Book	Function	The member can request to issue a book from the library.
Renew Book	Function	The member can renew an issued book.
View Account	Function	The member can view details (like books issued, due dates, etc.) of her or his account
Bank	Role	Bank is the secondary actor which is responsible for verifying payments.

4. Actor Dictionary

Actor	Description	Abstract	Use Case(s)
Administrator	The Manager is responsible for- 1. All the Staff Management functionalities 2. View the list of users in the system 3. All the Book Management functionalities		Add Staff Remove Staff View Staff List Search Book View Book List Add New Book Delete Book View User List
Librarian	The Librarian is responsible for - 1. All the User Management functionalities 2. All the Book Management functionalities 3. Issuing the book 4. Helping the new user to register		Search Book View Book List Add New Book Delete Book View User List Add User Remove User Issue Book
Staff	This generalizes the Administrator and the Librarian of the Library Management online portal.	✓	Login
Guest	The Guest can register to become member of the library		Register User
Member	The Member is responsible for- 1. Searching for the book 2. Viewing the list of books 3. Viewing account details		Search Book View Book List View Account Pay Check Penalty Return Book Request Book Renew Book Unregister User Login
User	This generalizes the member and the guest users of the Library Management online portal.	✓	
Bank	The Bank will verify Member payment.		Pay

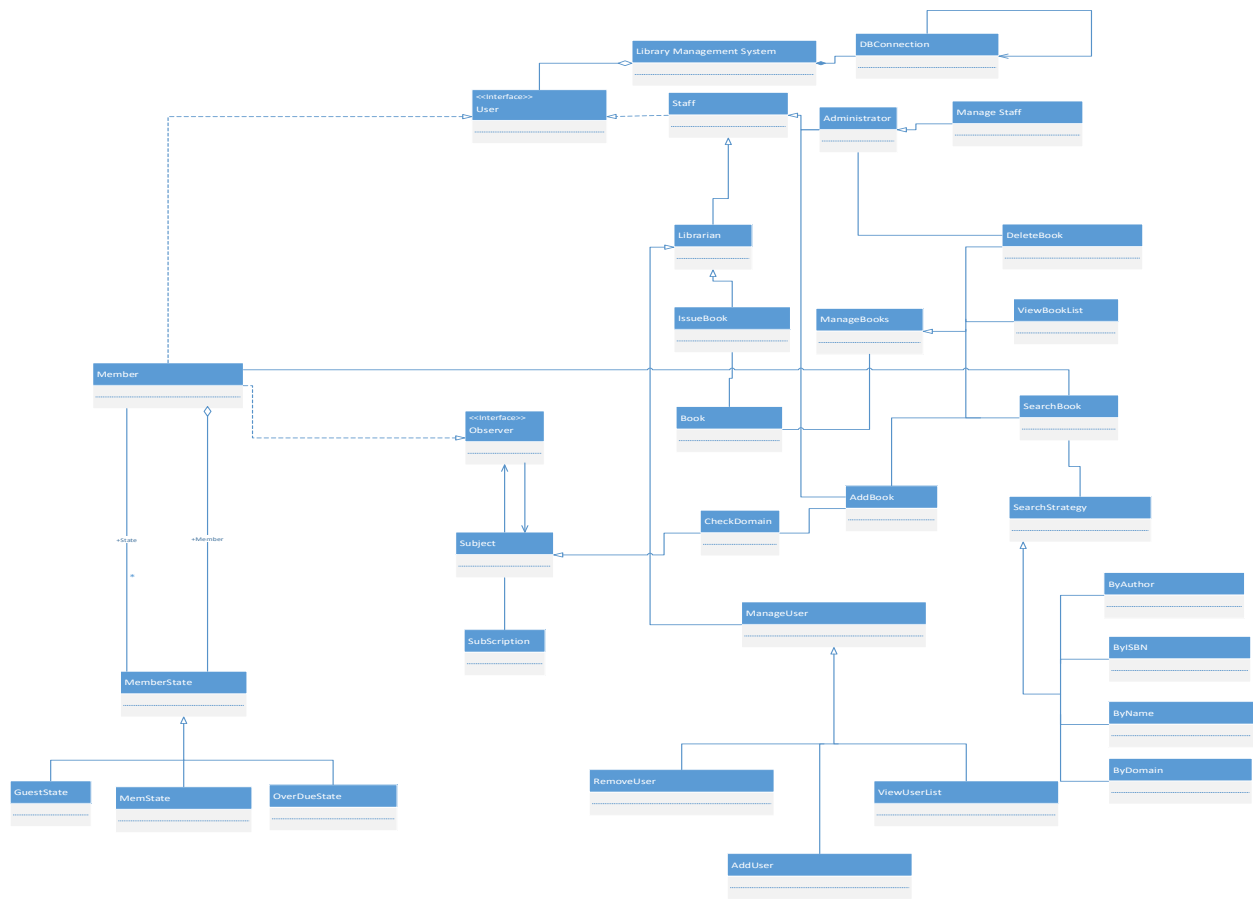
5. Use Case Diagram



Use Case Summary:

ID	Name	Description	Actors
100	Manage Staff	This allows Administrator to Add/Remove Staff or View Staff list.	Administrator
101	Add Staff	This allows Administrator to register new staff to the system.	Administrator
102	Remove Staff	This allows Administrator to remove staff from the system.	Administrator
103	View Staff List	This allows Administrator to View Staff List from the system.	Administrator
104	Login	Member, Administrator and Librarian must be login before using Library Management System.	Member, Administrator, Librarian
105	Manage Books	Administrator and Librarian can Search/View and Add/Delete Book from the system.	Librarian
106	Search Book	The User, Administrator and Librarian can search book in the library and can borrow the available book.	Member, Administrator, Librarian
107	View Book List	The User, Administrator and Librarian can View Book List in library system.	Librarian
108	Add Book	The Administrator and Librarian can Add a new book in library system.	Administrator, Librarian
109	Delete Book	The Administrator and Librarian can Delete a book from the library system.	Administrator, Librarian
110	Manage User	Librarian can Add/Remove User or View User List in the Library Management System.	Librarian
111	Add User	Librarian can Add User to the Library Management System	Librarian
112	Remove User	Librarian can Remove User from the Library Management System	Librarian
113	View User List	Librarian can View User List to the Library Management System.	Librarian
114	Register User	This is used when New user wants to register to the Library System.	Guest
115	Help	Librarian is called on when the user has issues with system.	Librarian
118	Request Book	Member can request Book	Member
119	Issue Book	The librarian would Issue the book requested by the user	Librarian
120	Pay	If any Book returned after due date the penalty is calculated and fine must be paid.	Member, Bank
121	Return Book	The Member must return book before the due date.	Member
122	Check Penalty	If the User returns books after due date penalty is calculated.	
123	Renew Book	The borrowed book can be renew again by the Member	Member
124	View Account	Member can View his or her account summary.	Member

6. Analysis Model



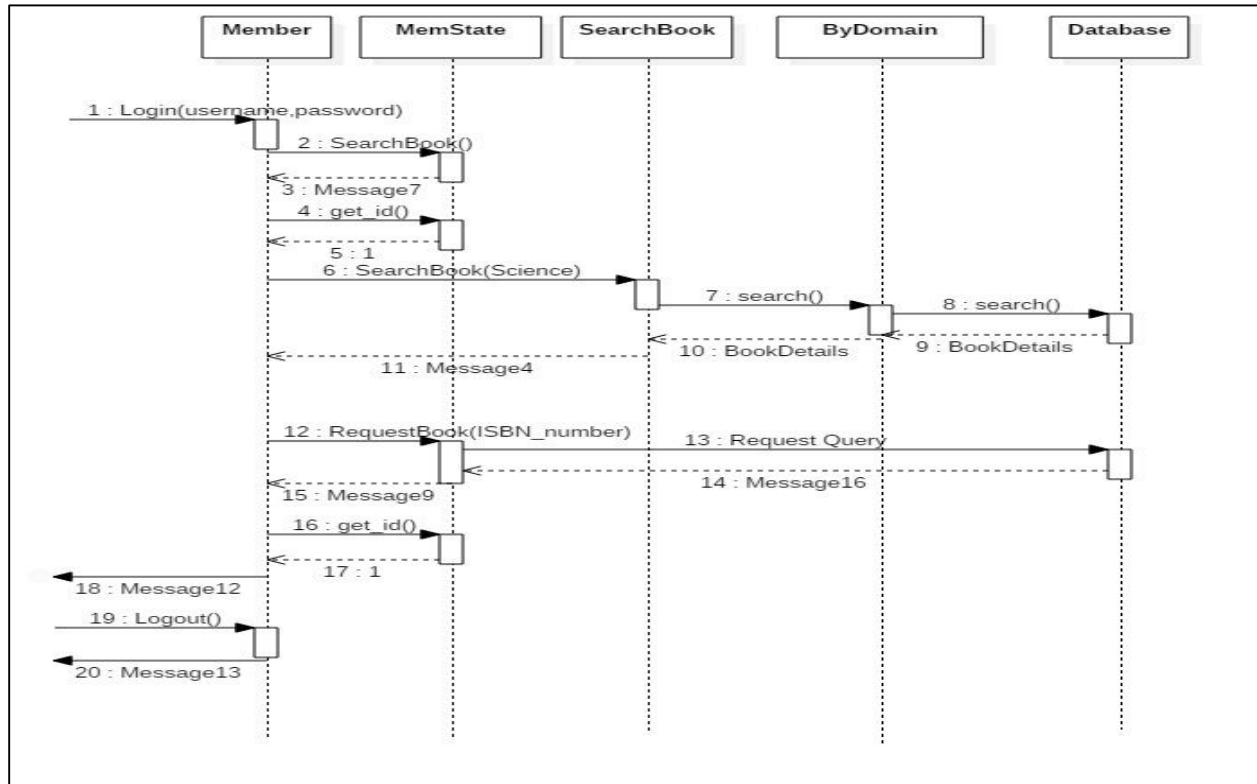
7. Design Class Diagram



8. Sequence Diagram

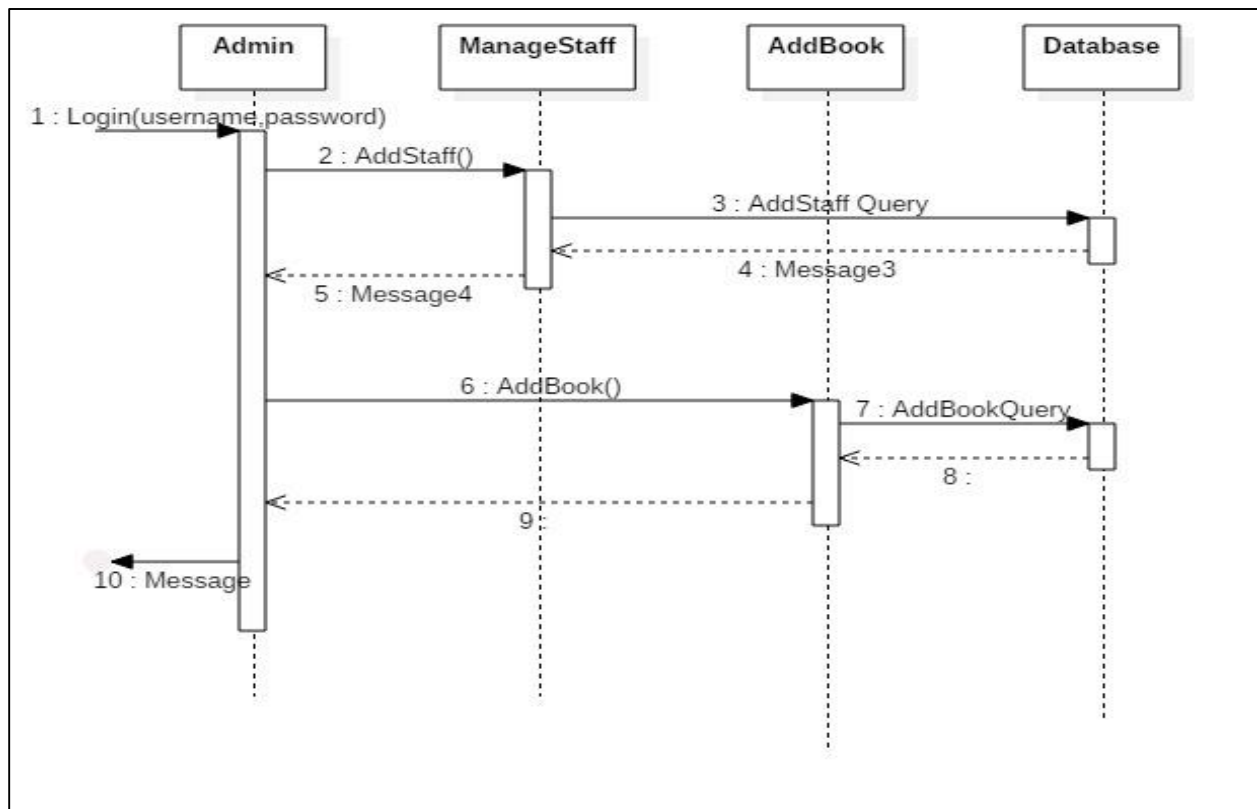
Sequence 1:

This sequence diagram shows how a Library Member can search for a book. The book can be searched on the basis of book name, author, ISBN or domain. Here the member is performing search using domain.



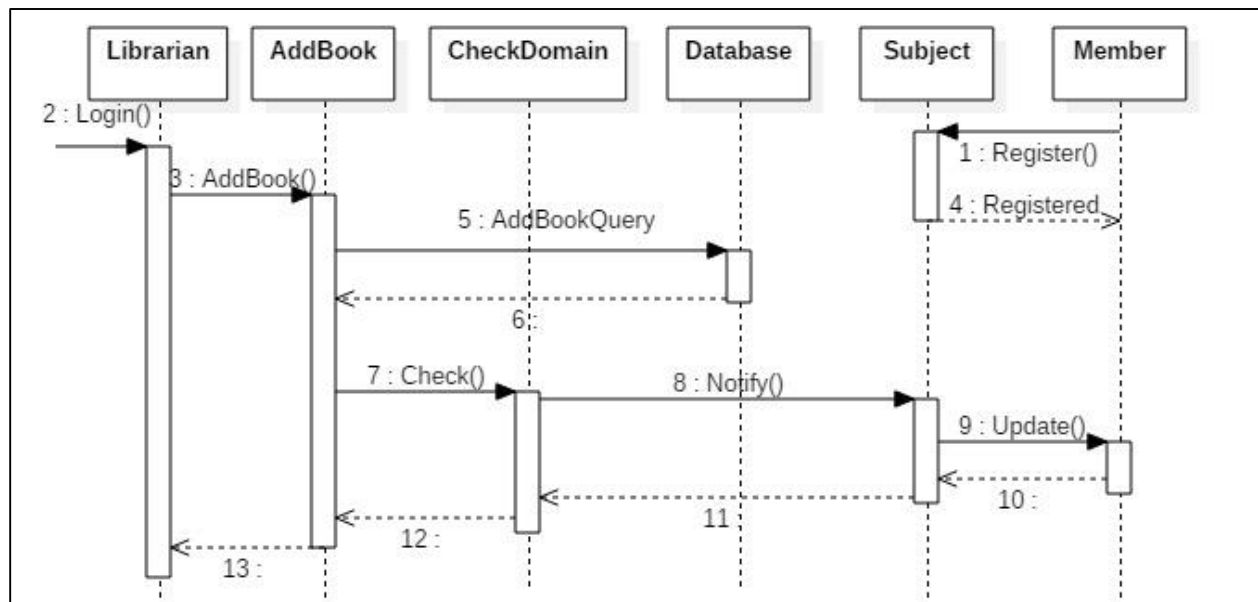
Sequence 2:

This sequence diagram shows how Librarian can add a new staff member and a new book to the system.



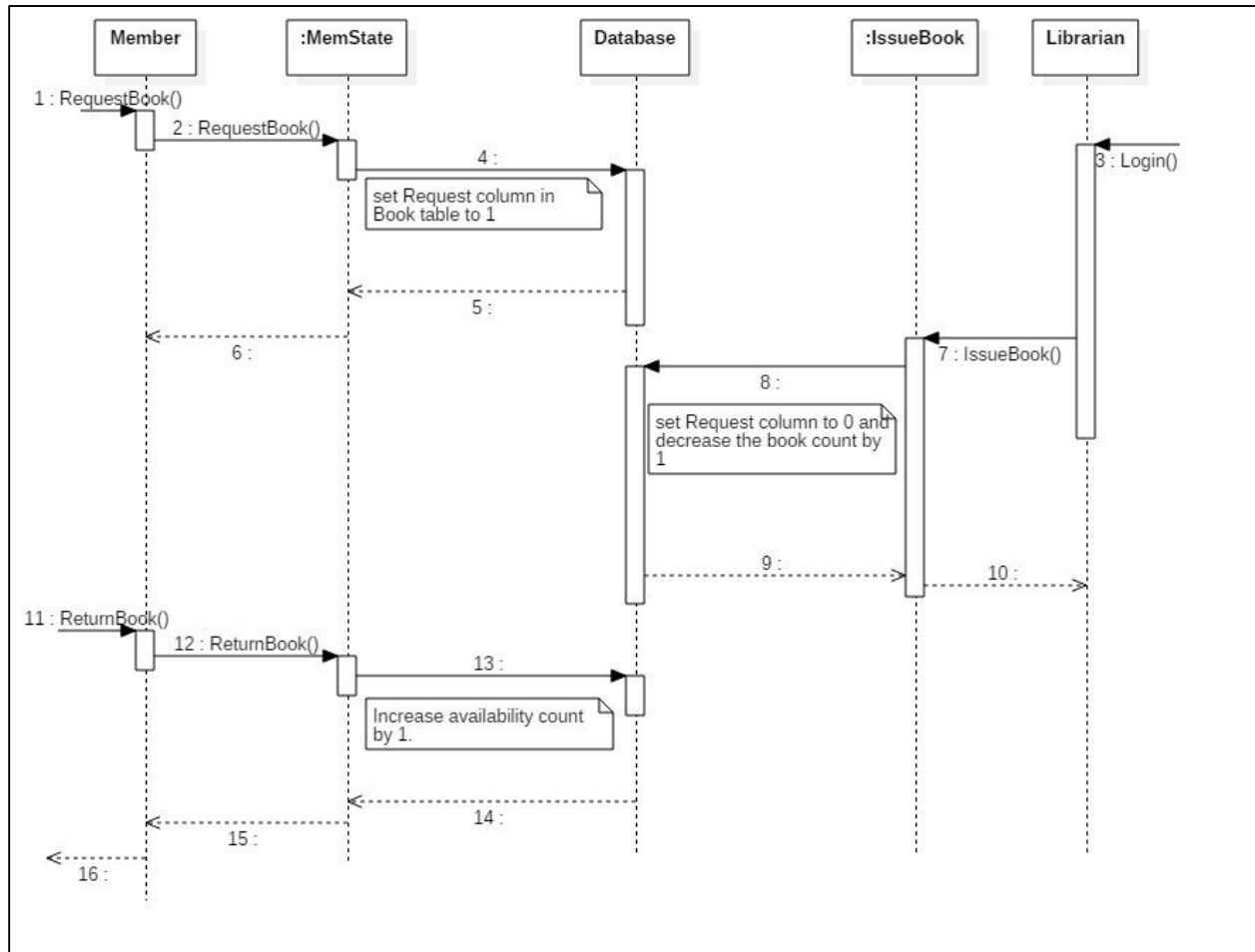
Sequence 3:

This sequence diagram shows how the subscribed member is notified for a new arrival of a book in the domain of interest.



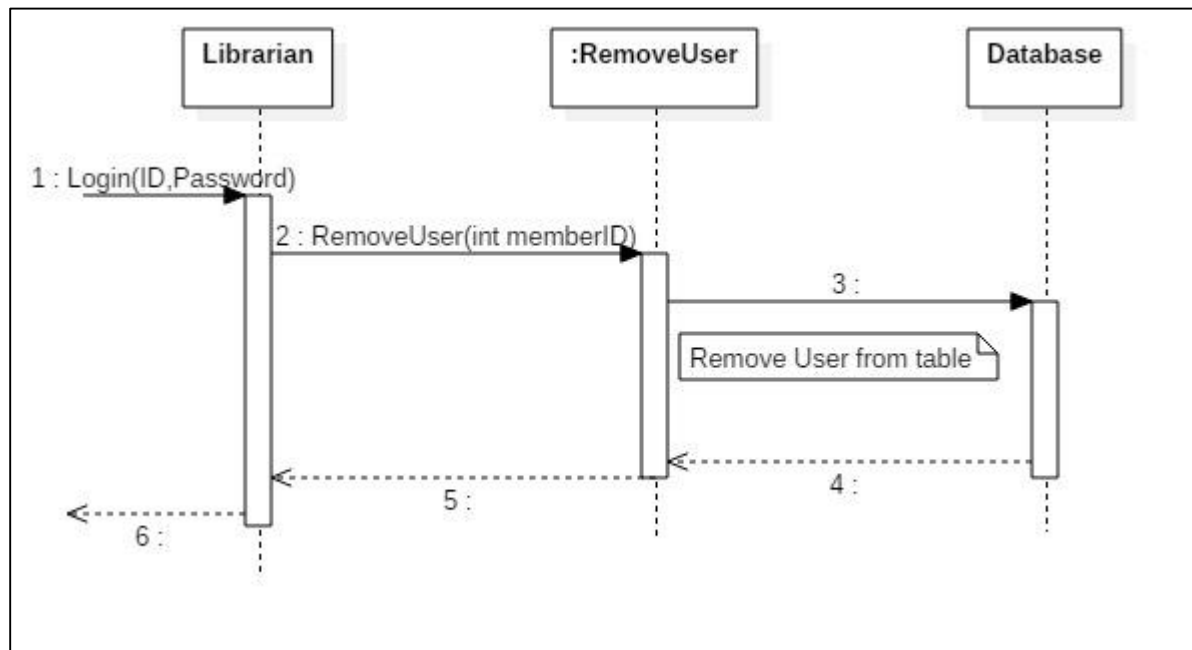
Sequence 4:

This sequence diagram shows how a member can request a book and how the librarian completes the order.

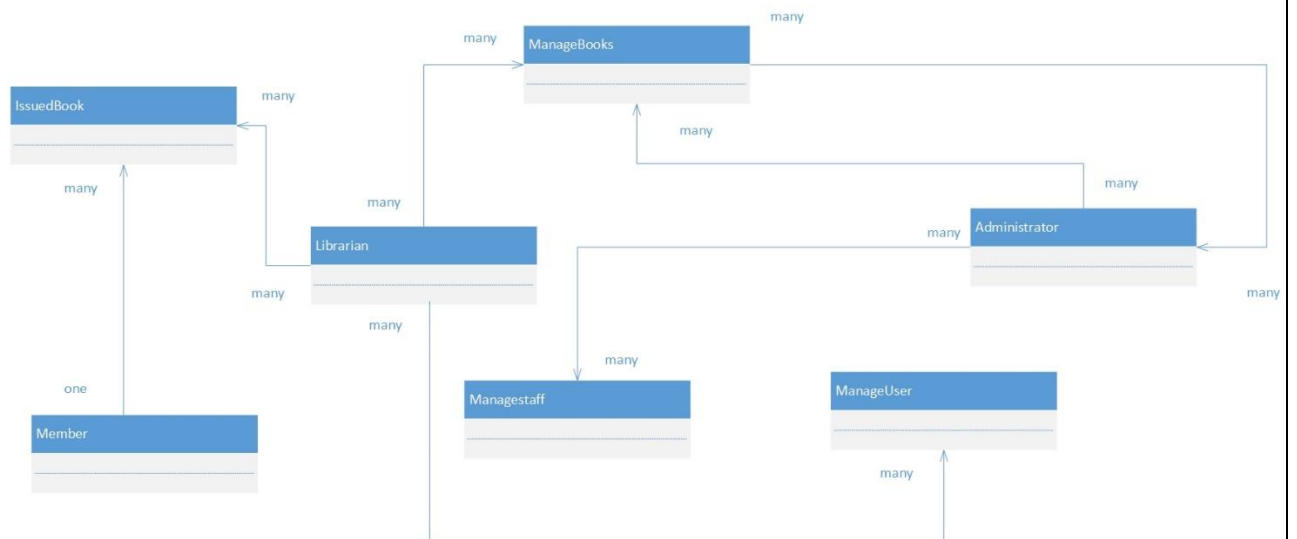


Sequence 5:

This sequence diagram shows how Librarian can remove a user from the system.



9. Object Relation Model

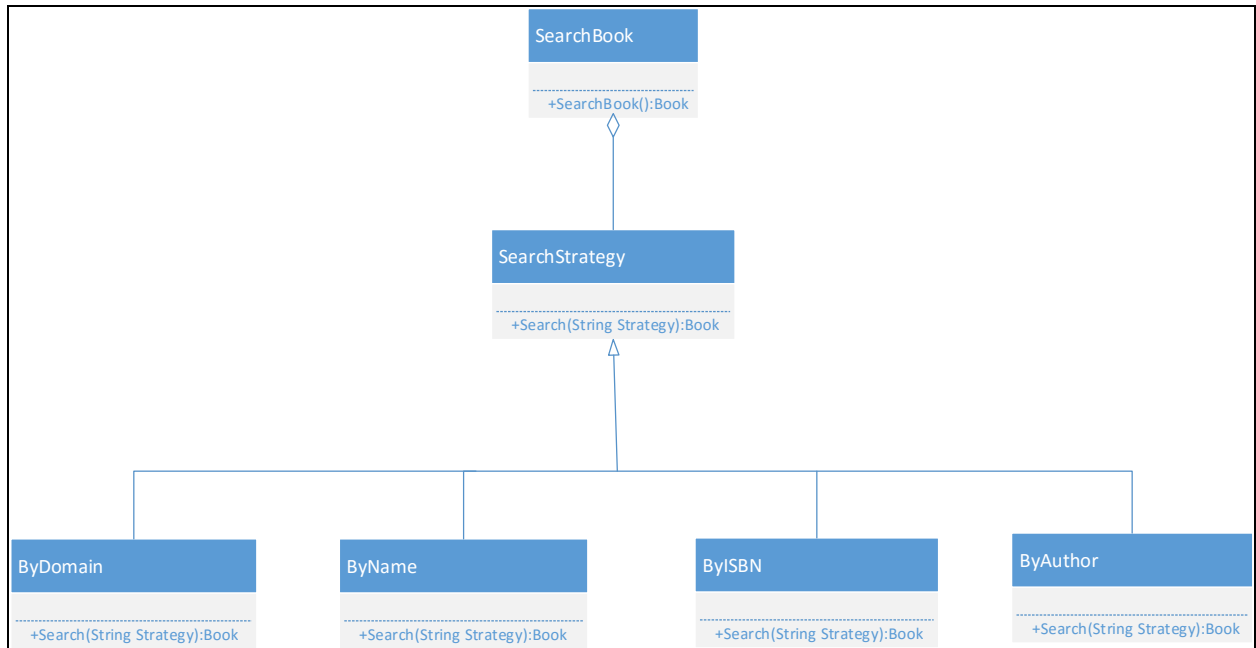


10. Documentation and class diagrams for Design Patterns used

1. Strategy pattern:

Strategy pattern allows the algorithm's behavior to be selected at runtime. It lets the algorithm to vary independently from clients that use it. The algorithm can be selected based on the type of data, the source of the data, user choice, or other discriminating factors. Strategy pattern is a behavioral pattern.

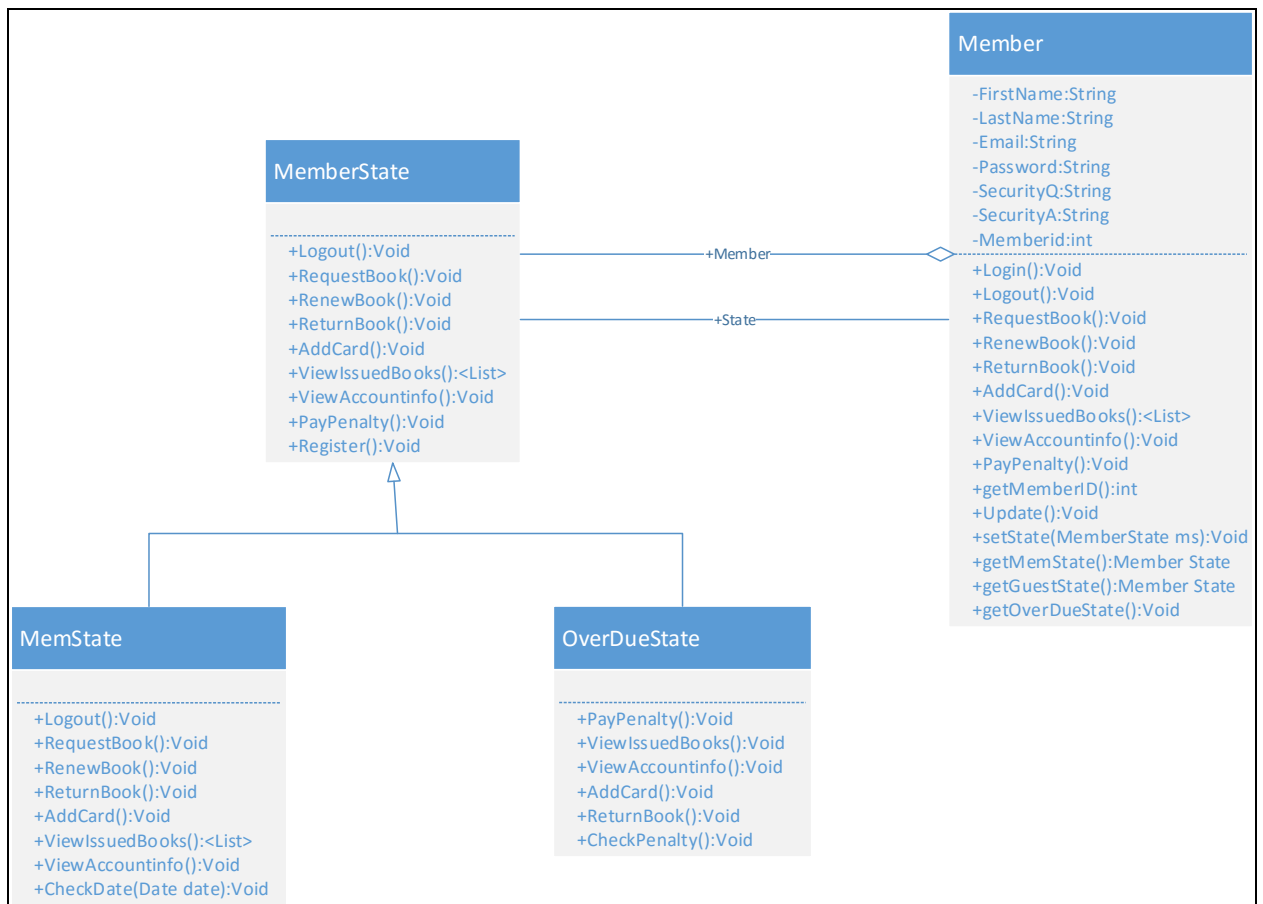
In this system we have used strategy pattern to search a book based on either of the following four factors: name of the book, author of the book, ISBN of the book or domain of the book. The search method is selected on the basis of user's choice at runtime.



2. State Pattern:

State Pattern is a behavioral design pattern that is used to implement the State machine. Each state is represented by a separate class that is derived from the State Pattern interface. These state classes implement the methods defined by the pattern's superclass.

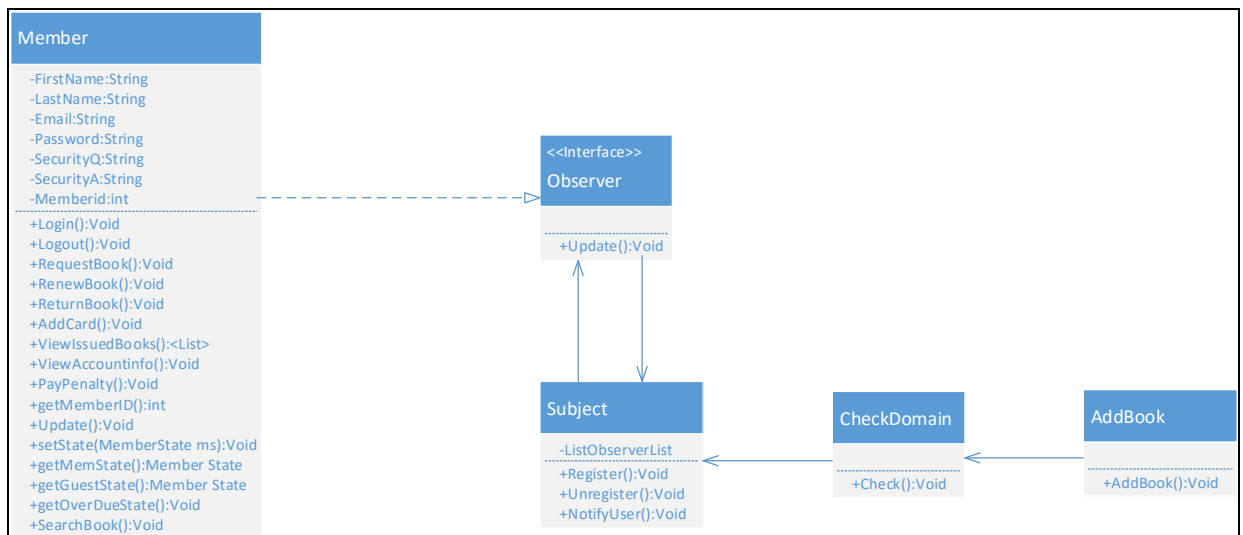
In this system there are two states. The first state is normal Member state where the member can request, renew and return a book. The second state is the overdue state that occurs when the book is overdue and the member in this state can pay the penalty and return the book.



3. Observer Pattern:

The Observer Pattern is a behavioral design pattern in which a subject called object maintains a list of its dependents. Dependents register themselves with the subject. The subject notifies the observers about any change in the state using one of their methods.

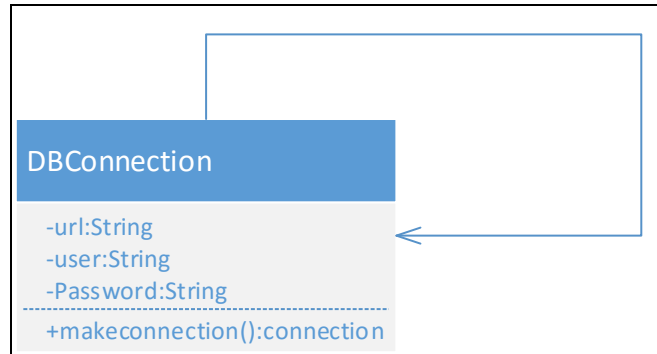
In this system the members who are interested in receive an email notification about arrival of a new book in the library, can register themselves for the same. When a new book is arrived in the library, the members registered for the field of interest are notified via email.



4. Singleton Pattern:

The Singleton pattern is a creational design pattern that restricts the instantiation of a class to only one object. This is useful when only one object is needed to carry out the operations throughout the system.

In this system the database connection needs only one object to be instantiated.



11.Source Code

AddBook.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.PreparedStatement;
import javax.swing.AbstractButton;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;

public class AddBook extends ManageBooks{
    public void AddBook(){
        super.AddBook();
    }
}
  
```

AddUser.java

```

import java.sql.Connection;
import java.util.*;

public class AddUser extends ManageUser {
    public void AddUser()
    {
        super.AddUser();
    }
}
  
```

Administrator.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
  
```

```

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;
import javax.swing.JTextField;
public class Administrator extends Staff implements ActionListener
{
    JFrame adminframe;
    JDialog addbookdialog,deletebookdialog;
    private JButton AddStaff;
    private JButton RemoveStaff;
    private JButton ViewStaffList,Logout;
    private JButton AddBooks;
    private JButton DeleteBooks;
    Connection conn=LibraryManagementSystem.connection;
    JTextField bookISBN,bookName,bookAuthor,bookCount,bookDomain;
    ManageStaff staffManage;
    //Staff Interface
    public Administrator()
    {
        adminframe=new JFrame("Admin Login");
        adminframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        adminframe.setSize(310,270);
        adminframe.setResizable(false);
        Logout=new JButton("Logout");
        JTabbedPane admintabs=new JTabbedPane();
        admintabs.setSize(310,230);
        JPanel managestaff=new JPanel();
        managestaff.setLayout(null);
        JPanel managebooks=new JPanel();
        managebooks.setLayout(null);
        admintabs.add("Manage Staff", managestaff);
        admintabs.add("Manage Books", managebooks);
        AddStaff=new JButton("Add Staff");
        RemoveStaff=new JButton("Remove Staff");
        ViewStaffList=new JButton("View Staff List");
        AddStaff.setBounds(20, 10, 250, 40);
        RemoveStaff.setBounds(20,60,250,40);
        ViewStaffList.setBounds(20, 110, 250, 40);
        managestaff.add(AddStaff);
        managestaff.add(RemoveStaff);
        managestaff.add(ViewStaffList);
        // Manage Books Interface
        AddBooks=new JButton("Add Book");
        DeleteBooks=new JButton("Delete Book");
        AddBooks.setBounds(20, 20, 250, 50);
        DeleteBooks.setBounds(20, 90, 250, 50);
        Logout.setBounds(60, 180, 170, 40);
    }
}

```

```

        adminframe.add(Login);
        managebooks.add(AddBooks);
        managebooks.add(DeleteBooks);
        adminframe.add(adminframe);
        adminframe.setVisible(true);
        AddStaff.addActionListener(this);
        RemoveStaff.addActionListener(this);
        ViewStaffList.addActionListener(this);
        AddBooks.addActionListener(this);
        Logout.addActionListener(this);
        DeleteBooks.addActionListener(this);
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        try{
            if(e.getSource()==AddBooks){
                AddBook addBook=new AddBook();
                addBook.AddBook();
            }
            if(e.getSource()==DeleteBooks){
                DeleteBook deleteBook=new DeleteBook();
                deleteBook.DeleteBook();
            }
            if(e.getSource()==AddStaff){
                staffManage=new ManageStaff();
                staffManage.AddStaff();
            }
            if(e.getSource()==RemoveStaff){
                staffManage=new ManageStaff();
                staffManage.RemoveStaff();
            }
            if(e.getSource()==ViewStaffList){
                staffManage=new ManageStaff();
                staffManage.ViewStaffList();
            }
            if(e.getSource()==Logout){
                adminframe.dispose();
                new LibraryManagementSystem();
            }
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
}

```

Book.java

```

public class Book {
    private String ISBN;

```

```

private String BookName;
private String Author;
private String domain;
public String getDomain() {
    return domain;
}
public void setDomain(String domain) {
    this.domain = domain;
}
private int TotalCount;
private int AvaibleCount;
public String getISBN() {
    return ISBN;
}
public void setISBN(String iISBN) {
    ISBN = iISBN;
}
public String getBookName() {
    return BookName;
}
public void setBookName(String bookName) {
    BookName = bookName;
}
public String getAuthor() {
    return Author;
}
public void setAuthor(String author) {
    Author = author;
}
public int getTotalCount() {
    return TotalCount;
}
public void setTotalCount(int totalCount) {
    TotalCount = totalCount;
}
public int getAvaibleCount() {
    return AvaibleCount;
}
public void setAvaibleCount(int avaibleCount) {
    AvaibleCount = avaibleCount;
}
}

```

ByAuthor.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

```

```

import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;

public class ByAuthor extends StrategySearch implements ActionListener
{
    Statement statement = null;
    ResultSet rs;
    Connection connection =LibraryManagementSystem.connection;
    private final JTextField searchtext;
    private final JButton searchButton;
    JDialog byAuthor;
    private JLabel searchlabel;
    JTable list;
    JDialog dialog;
    TableModel tbm;
    ByAuthor() {
        searchlabel=new JLabel("Enter Author:");
        byAuthor=new JDialog();
        byAuthor.setResizable(false);
        byAuthor.setLayout(null);
        byAuthor.setModal(true);
        byAuthor.setTitle("Search By Author");
        byAuthor.setSize(280,150);
        byAuthor.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        searchtext=new JTextField();
        searchButton=new JButton("Search Book");
        searchButton.addActionListener(this);
        byAuthor.add(searchButton);
        byAuthor.add(searchtext);
        byAuthor.add(searchlabel);
        this.searchlabel.setBounds(10,20,100,30);
        this.searchtext.setBounds(90,20,170,30);
        this.searchButton.setBounds(60, 70, 150, 30);
        byAuthor.setVisible(true);
    }
    public void search() throws SQLException {
        String authorname=this.searchtext.getText();
        String searchByAuthorQuery = "SELECT * FROM book WHERE Author=?";
        PreparedStatement ps = connection.prepareStatement(searchByAuthorQuery);
        ps.setString(1, authorname);
        rs=ps.executeQuery();
        tbm=new DefaultTableModel(new String[]{"ISBN","Book
Name","Author","Total Count","Available Count"},0);

```



```

        while(rs.next())
        {
            ((DefaultTableModel) this.tbm).addRow(new
Object[]{rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getString(5)});
        }
        list=new JTable(tbm);
        list.setAutoResizeMode(JTable.AUTO_RESIZE_SUBSEQUENT_COLUMNS);
        JScrollPane pane = new JScrollPane(list);
        dialog=new JDialog();
        dialog.add(pane);
        dialog.setTitle("Book List");
        list.setEnabled(false);
        dialog.setSize(500,500);
        list.setBounds(0,0,500,500);
        dialog.add(pane);
        dialog.setVisible(true);
    }
    @Override
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==searchButton)
        {
            byAuthor.dispose();
            try {
                search();
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
    }
    public static void main(String[] args) {
        new ByAuthor();
    }
}

```

ByDomain.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTable;

```

```

import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;

public class ByDomain extends StrategySearch implements ActionListener
{
    Statement statement = null;
    ResultSet rs;
    Connection connection = LibraryManagementSystem.connection;
    private final JTextField searchtext;
    private final JButton searchButton;
    JDialog byDomain;
    private JLabel searchlabel;
    JTable list;
    JDialog dialog;
    TableModel tbm;
    ByDomain() {
        searchlabel=new JLabel("Enter Domain:");
        byDomain=new JDialog();
        byDomain.setResizable(false);
        byDomain.setLayout(null);
        byDomain.setModal(true);
        byDomain.setTitle("Search By Domain");
        byDomain.setSize(280,150);
        byDomain.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        searchtext=new JTextField();
        searchButton=new JButton("Search Book");
        searchButton.addActionListener(this);
        byDomain.add(searchButton);
        byDomain.add(searchtext);
        byDomain.add(searchlabel);
        this.searchlabel.setBounds(10,20,100,30);
        this.searchtext.setBounds(90,20,170,30);
        this.searchButton.setBounds(60, 70, 150, 30);
        byDomain.setVisible(true);
    }
    public void search() {
        String domainname=this.searchtext.getText();
        try{
            String searchbyDomainQuery = "SELECT * FROM book WHERE domain=?";
            PreparedStatement ps = connection.prepareStatement(searchbyDomainQuery);
            ps.setString(1, domainname);
            rs=ps.executeQuery();
            dialog=new JDialog(byDomain, "Search Results");
            tbm=new DefaultTableModel(new String[]{"ISBN", "Book
Name", "Author", "Total Count", "Available Count"},0);
            while(rs.next())
            {
                ((DefaultTableModel) this.tbm).addRow(new
Object[]{rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getString(5)});
            }
        }
    }
}

```

```

        list=new JTable(tbm);
        list.setAutoResizeMode(JTable.AUTO_RESIZE_SUBSEQUENT_COLUMNS);
        JScrollPane pane=new JScrollPane(list);
        dialog.add(pane);
        dialog.setSize(500,500);
        dialog.setModal(true);
        dialog.setVisible(true);
    }
    catch (Exception e1) {
        e1.printStackTrace();
    }
}
@Override
public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==searchButton)
    {
        byDomain.dispose();
        search();
    }
}
}

```

ByISBN.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;

public class ByISBN extends StrategySearch implements ActionListener
{
    Statement statement = null;
    ResultSet rs;
    Connection connection = LibraryManagementSystem.connection;
    private final JTextField searchtext;
    private final JButton searchButton;
    JDialog byISBN;
    private JLabel searchlabel;
}

```

```

JTable list;
JDialog dialog;
TableModel tbm;
ByISBN() {
    searchlabel=new JLabel("Enter ISBN:");
    byISBN=new JDialog();
    byISBN.setResizable(false);
    byISBN.setLayout(null);
    byISBN.setModal(true);
    byISBN.setTitle("Search By ISBN");
    byISBN.setSize(280,150);
    byISBN.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
    searchtext=new JTextField();
    searchButton=new JButton("Search Book");
    searchButton.addActionListener(this);
    byISBN.add(searchButton);
    byISBN.add(searchtext);
    byISBN.add(searchlabel);
    this.searchlabel.setBounds(10,20,100,30);
    this.searchtext.setBounds(90,20,170,30);
    this.searchButton.setBounds(60, 70, 150, 30);
    byISBN.setVisible(true);
}
public void search() {
    String ISBN=this.searchtext.getText();
    try{
        String searchbyISBNQuery = "SELECT * FROM book WHERE ISBN=?";
        PreparedStatement ps = connection.prepareStatement(searchbyISBNQuery);
        ps.setString(1, ISBN);
        rs=ps.executeQuery();
        dialog=new JDialog(byISBN, "Search Results");
        tbm=new DefaultTableModel(new String[]{"ISBN","Book
Name","Author","Total Count","Available Count"},0);
        while(rs.next())
        {
            ((DefaultTableModel) this.tbm).addRow(new
Object[]{rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getString(5)});
        }
        list=new JTable(tbm);
        list.setAutoResizeMode(JTable.AUTO_RESIZE_SUBSEQUENT_COLUMNS);
        JScrollPane pane=new JScrollPane(list);
        dialog.add(pane);
        dialog.setSize(500,500);
        dialog.setModal(true);
        dialog.setVisible(true);
    }
    catch (Exception e1) {
        e1.printStackTrace();
    }
}
@Override

```

```

        public void actionPerformed(ActionEvent e)
        {
            if(e.getSource()==searchButton)
            {
                byISBN.dispose();
                search();
            }
        }
    }
}

```

ByName.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;

public class ByName extends StrategySearch implements ActionListener
{
    Statement statement = null;
    ResultSet rs;
    Connection connection = LibraryManagementSystem.connection;
    private final JTextField searchtext;
    private final JButton searchButton;
    JDialog byName;
    private JLabel searchlabel;
    JTable list;
    JDialog dialog;
    TableModel tbm;
    ByName() {
        searchlabel=new JLabel("Enter Name:");
        byName=new JDialog();
        byName.setResizable(false);
        byName.setLayout(null);
        byName.setModal(true);
        byName.setTitle("Search By Name");
        byName.setSize(280,150);
        byName.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        searchtext=new JTextField();
    }
}

```

```

        searchButton=new JButton("Search Book");
        searchButton.addActionListener(this);
        byName.add(searchButton);
        byName.add(searchtext);
        byName.add(searchlabel);
        this.searchlabel.setBounds(10,20,100,30);
        this.searchtext.setBounds(90,20,170,30);
        this.searchButton.setBounds(60, 70, 150, 30);
        byName.setVisible(true);
    }
    public void search() {
        String bookname=this.searchtext.getText();
        try{
            String searchbyNameQuery = "SELECT * FROM book WHERE bookname=?";
            PreparedStatement ps = connection.prepareStatement(searchbyNameQuery);
            ps.setString(1, bookname);
            rs=ps.executeQuery();
            dialog=new JDialog(byName, "Search Results");
            tbm=new DefaultTableModel(new String[]{"ISBN", "Book
Name", "Author", "Total Count", "Available Count"},0);
            while(rs.next())
            {
                ((DefaultTableModel) this.tbm).addRow(new
Object[]{rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getString(5)});
            }
            list=new JTable(tbm);
            list.setAutoResizeMode(JTable.AUTO_RESIZE_SUBSEQUENT_COLUMNS);
            JScrollPane pane=new JScrollPane(list);
            dialog.add(pane);
            dialog.setSize(500,500);
            dialog.setModal(true);
            dialog.setVisible(true);
        }
        catch (Exception e1) {
            e1.printStackTrace();
        }
    }
    @Override
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==searchButton)
        {
            byName.dispose();
            search();
        }
    }
}

```

CheckDomain.java

```
import java.util.*;
```

```

public class CheckDomain{
    public void Check(Book book) {
        Subject subj = new Subject(book.getDomain());
        subj.notifyUsers(book);
    }
}

```

DbConnection.java

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DbConnection {

    public static String url;
    public static String user;
    public static String password;
    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }
    public String getUser() {
        return user;
    }
    public void setUser(String user) {
        this.user = user;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public DbConnection(String u,String username,String pass) {
        this.url=u;
        this.user=username;
        this.password=pass;
    }
}

```

DeleteBook.java

```

import java.awt.event.ActionEvent;

```

```

import java.awt.event.ActionListener;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class DeleteBook extends ManageBooks {
    DeleteBook(){
        super.DeleteBook();
    }
}

```

Forgot.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

class Forgot
    extends JDialog
    implements ActionListener
{
    JTextField u;
    JTextField q;
    JTextField a;
    JTextField p;
    JButton get;
    JButton back;
    String dans;
    String uans;
    String pass;
    Connection localConnection = LibraryManagementSystem.connection;
    public Forgot() {
    }
    Forgot(String user)
    {
        setTitle("Forgot Password...");
        setSize(360, 330);
        setLayout(null);
    }
}

```



```

setVisible(true);
setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
JLabel localJLabel1 = new JLabel("Password Retrieval");
JLabel localJLabel2 = new JLabel("Username");
JLabel localJLabel3 = new JLabel("Question");
JLabel localJLabel4 = new JLabel("Answer");
JLabel localJLabel5 = new JLabel("Password");
this.u = new JTextField(user);
this.q = new JTextField();
this.a = new JTextField();
this.p = new JTextField();
this.u.setEditable(false);
this.p.setEditable(false);
this.q.setEditable(false);
this.get = new JButton("Get Password");
this.back = new JButton("Back to Login");
this.get.addActionListener(this);
this.back.addActionListener(this);
localJLabel1.setBounds(100, 10, 180, 30);
localJLabel2.setBounds(10, 50, 100, 30);
this.u.setBounds(150, 50, 180, 30);
localJLabel3.setBounds(10, 90, 100, 30);
this.q.setBounds(150, 90, 180, 30);
localJLabel4.setBounds(10, 130, 100, 30);
this.a.setBounds(150, 130, 180, 30);
this.get.setBounds(100, 170, 160, 30);
localJLabel5.setBounds(10, 210, 100, 30);
this.p.setBounds(150, 210, 180, 30);
this.back.setBounds(100, 260, 160, 30);
add(localJLabel1);
add(this.back);
add(localJLabel2);
add(localJLabel3);
add(localJLabel4);
add(this.u);
add(this.a);
add(this.q);
add(this.get);
add(this.p);
add(localJLabel5);
setModal(true);
setResizable(false);
try
{
    String str = "select * from member where memberid=?";
    PreparedStatement localPreparedStatement = localConnection.prepareStatement(str);
    localPreparedStatement.setString(1,user);
    ResultSet localResultSet = localPreparedStatement.executeQuery();
    while (localResultSet.next())
    {
        this.dans = localResultSet.getString("securityans");
    }
}

```

```

        this.q.setText(localResultSet.getString("securityques"));
        this.pass = localResultSet.getString("Password");
    }
}
catch (SQLException localSQLException)
{
    localSQLException.printStackTrace();
}
}

public void actionPerformed(ActionEvent paramActionEvent)
{
    Object localObject = paramActionEvent.getSource();
    this.uans = this.a.getText();
    if (localObject == this.get()) {
        if (this.uans.equalsIgnoreCase(this.dans)) {
            this.p.setText(this.pass);
        } else {
            JOptionPane.showMessageDialog(null, "Invalid Answer", "Project", 0);
        }
    }
    if (localObject == this.back)
    {
        new LibraryManagementSystem();
        dispose();
    }
}
}

```

Guest.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class Guest implements ActionListener
{
    JFrame guestFrame;
    private JLabel firstnamelabel;
    private JTextField firstnametext;
    private JLabel lastnamelabel;
    private JTextField lastnametext;

```

```

private JLabel emaillabel;
private JTextField emailtext;
private JLabel passwordlabel;
private JPasswordField passwordtext;
private JLabel sequelabel;
private JLabel secanslabel;
private JTextField secquetext;
private JTextField secanstext;
private JButton Submit;
Connection conn=LibraryManagementSystem.connection;
public Guest(){

```

```

    guestFrame= new JFrame("Register Guest");
    guestFrame.setSize(320,310);
    guestFrame.setResizable(false);
    guestFrame.setLayout(null);
    firstnamelabel=new JLabel("First Name");
    firstnamelabel.setBounds(10,10,150,25);
    guestFrame.add(firstnamelabel);
    firstnametext=new JTextField();
    firstnametext.setBounds(140, 10, 150, 25);
    guestFrame.add(firstnametext);
    lastnamelabel=new JLabel("Last Name");
    lastnamelabel.setBounds(10,45,150,25);
    guestFrame.add(lastnamelabel);
    lastnametext=new JTextField();
    lastnametext.setBounds(140, 45, 150, 25);
    guestFrame.add(lastnametext);
    emaillabel=new JLabel("Email Address");
    emaillabel.setBounds(10,80,150,25);
    guestFrame.add(emaillabel);
    emailtext=new JTextField();
    emailtext.setBounds(140, 80, 150, 25);
    guestFrame.add(emailtext);
    passwordlabel=new JLabel("Password");
    passwordlabel.setBounds(10,115,150,25);
    guestFrame.add(passwordlabel);
    passwordtext=new JPasswordField();
    passwordtext.setBounds(140,115,150,25);
    guestFrame.add(passwordtext);
    sequelabel=new JLabel("Security Question");
    sequelabel.setBounds(10,150,150,25);
    guestFrame.add(sequelabel);
    secquetext=new JTextField();
    secquetext.setBounds(140, 150, 150, 25);
    guestFrame.add(secquetext);
    secanslabel=new JLabel("Security Answer");
    secanslabel.setBounds(10,185,150,25);
    guestFrame.add(secanslabel);
    secanstext=new JTextField();
    secanstext.setBounds(140, 185, 150, 25);

```

```

        guestFrame.add(secanstext);
        guestFrame.setVisible(true);
        Submit=new JButton("Submit");
        Submit.addActionListener(this);
        Submit.setBounds(100,230,100,30);
        guestFrame.add(Submit);
    }
    public void Register()
    {
        String registerquery="Insert into member Values(?,?,?,?);";
        PreparedStatement ps;
        try {
            ps = conn.prepareStatement(registerquery);
            ps.setString(1, null);
            ps.setString(2, firstnametext.getText());
            ps.setString(3, lastnametext.getText());
            ps.setString(4, emailtext.getText());
            ps.setString(5, passwordtext.getText());
            ps.setString(6, secquetext.getText());
            ps.setString(7, secanstext.getText());
            ps.executeUpdate();
            JOptionPane.showMessageDialog(null, "User Registered", "User Registered", 1);
            guestFrame.dispose();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==Submit)
        {
            Register();
        }

    }

}

```

IssueBook.java

```

import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;

import javax.swing.JOptionPane;

public class IssueBook extends Librarian{
    private int OrderID;

```

```

private String ISBN;
private int memberID;
private Date DateOfIssue;
private Date DueDate;
Connection conn=LibraryManagementSystem.connection;
    public int getOrderID() {
        return OrderID;
    }
    public void setOrderID(int orderID) {
        OrderID = orderID;
    }
    public String getISBN() {
        return ISBN;
    }
    public void setISBN(String iISBN) {
        ISBN = iISBN;
    }
    public int getMemberID() {
        return memberID;
    }
    public void setMemberID(int memberID) {
        this.memberID = memberID;
    }
    public Date getDateOfIssue() {
        return DateOfIssue;
    }
    public void setDateOfIssue(Date dateOfIssue) {
        DateOfIssue = dateOfIssue;
    }
    public Date getDueDate() {
        return DueDate;
    }
    public void setDueDate(Date dueDate) {
        DueDate = dueDate;
    }
    void IssueBook(){
        try{
            String issuebookquery="update issuedbook set duedate= adddate(now(),14),
dateofissue=now(), pending=0 where pending=1;";
            PreparedStatement ps= conn.prepareStatement(issuebookquery);
            ps.executeUpdate(issuebookquery);
            JOptionPane.showMessageDialog(null, "Books Issued", "Book Issued",1);
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
    }
}

```

Librarian.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.sql.SQLException;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;
import javax.swing.JTextField;

public class Librarian implements ActionListener,ItemListener
{
    JFrame LibrarianFrame;
    JTabbedPane tabs;
    JPanel panel;
    JTextField Bookname;
    JButton
IssueBook,ViewAllrequestedBooks,AddMember,RemoveMember,ViewMemberList,AddBook,DeleteBo
ok,ViewBookList,SearchBook;
    JPanel tab1,tab2,tab3;
    JComboBox<String> list;
    JLabel SearchBy;
    String searchtype;
    public Librarian()
    {
        LibrarianFrame=new JFrame("Librarian Login");
        LibrarianFrame.setTitle("Librarian Login");
        LibrarianFrame.setVisible(true);
        LibrarianFrame.setSize(380,350);
        LibrarianFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        tabs=new JTabbedPane();
        tab1=new JPanel();
        tab2=new JPanel();
        tab3=new JPanel();
        tabs.addTab("Manage Orders",tab1);
        tabs.addTab("Manage Members", tab2);
        tabs.addTab("Manage Books", tab3);
        LibrarianFrame.add(tabs);
        //Tab1 all components
        IssueBook=new JButton("Issue Book");
        IssueBook.setToolTipText("Issue All Requested Books");
        ViewAllrequestedBooks=new JButton("View All Requested Books");
        ViewAllrequestedBooks.setToolTipText("View all requested book list and Issue
Books");
        tab1.setLayout(null);
        tab1.add(IssueBook);

```

```

tab1.add(ViewAllrequestedBooks);
IssueBook.setBounds(10, 50, 340, 80);
ViewAllrequestedBooks.setBounds(10,140,340,80);
IssueBook.addActionListener(this);
ViewAllrequestedBooks.addActionListener(this);
//Tab2 all components
AddMember=new JButton("Add Member");
RemoveMember=new JButton("Remove Member");
ViewMemberList=new JButton("View Member List");
tab2.setLayout(null);
tab2.add(AddMember);
tab2.add(ViewMemberList);
tab2.add(RemoveMember);
AddMember.setBounds(10, 10, 340, 80);
RemoveMember.setBounds(10,100, 340, 80);
ViewMemberList.setBounds(10, 190, 340, 80);
AddMember.addActionListener(this);
RemoveMember.addActionListener(this);
ViewMemberList.addActionListener(this);
//Tab3 all Components
AddBook=new JButton("Add Book");
DeleteBook=new JButton("DeleteBook");
SearchBook=new JButton("Search Book");
ViewBookList=new JButton("View Book List");
list=new JComboBox<>(new String[]{"(Select)","Name","Author","ISBN","Domain"});
SearchBy=new JLabel("Search by:");
tab3.setLayout(null);
tab3.add(AddBook);
tab3.add(DeleteBook);
tab3.add(SearchBook);
tab3.add(ViewBookList);
tab3.add(list);
tab3.add(list);
tab3.add(SearchBy);
AddBook.setBounds(10, 10, 340, 50);
DeleteBook.setBounds(10, 70, 340, 50);
ViewBookList.setBounds(10, 130, 340, 50);
SearchBook.setBounds(10, 190, 340, 50);
SearchBy.setBounds(50,250,150,30);
list.setBounds(150, 250, 150, 25);
list.addItemListener(this);
AddBook.addActionListener(this);
DeleteBook.addActionListener(this);
ViewBookList.addActionListener(this);
SearchBook.addActionListener(this);
}
@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==AddBook){
        ManageBooks manageBooks=new ManageBooks();
        manageBooks.AddBook();
    }
}

```

```

    }
    if(e.getSource()==DeleteBook){
        ManageBooks manageBooks=new ManageBooks();
        manageBooks.DeleteBook();
    }
    if(e.getSource()==ViewBookList){
        ManageBooks manageBooks=new ManageBooks();
        manageBooks.ViewBookList();
    }
    if(e.getSource()==SearchBook){
        ManageBooks manageBooks=new ManageBooks();
        manageBooks.SearchBook(searchtype);
    }
    if(e.getSource()==AddMember){
        ManageUser manageUser=new AddUser();
        manageUser.AddUser();
    }
    if(e.getSource()==RemoveMember){
        ManageUser manageUser=new RemoveUser();
        manageUser.RemoveUser();
    }
    if(e.getSource()==ViewMemberList){
        try {
            ManageUser manageUser=new ViewUserList();
            manageUser.ViewUserList();
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
    }
    if(e.getSource()==IssueBook){
        IssueBook issueBook=new IssueBook();
        issueBook.IssueBook();
    }
    if(e.getSource()==ViewAllrequestedBooks){
        ViewAllRequestedBooks viewallrequestedbooks = new
ViewAllRequestedBooks();
        viewallrequestedbooks.ViewAllRequestedBooks();
    }
}

@Override
public void itemStateChanged(ItemEvent e) {
    if(e.getSource()==list)
    {
        searchtype=list.getSelectedItemAt().toString();
    }
}
}

```

LibraryManagementSystem.java

```
import java.awt.event.ActionEvent;
```



```

import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTabbedPane;
import javax.swing.JTextField;

public class LibraryManagementSystem extends JFrame implements ActionListener
{
    private JPanel User;
    private JPanel Staff;
    private JPasswordField password1,password2;
    private JTextField username1,username2;
    private JButton Login1;
    private JButton Register;
    private JButton forgot1;
    private JButton cancel1;
    private JButton Login2;
    private JButton cancel2;
    private String user;
    private String pass;
    private String role;
    public int memberID;
    Administrator admin;
    Librarian libr;
    public static Connection connection = null;
    public LibraryManagementSystem()
    {
        DbConnection dbConnection = new
        DbConnection("jdbc:mysql://localhost:3306/library", "root", "root");
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            connection = DriverManager.getConnection(dbConnection.url,
            dbConnection.user, dbConnection.password);
        } catch (SQLException e) {
            // handle any errors
            System.out.println("SQLException: " + e.getMessage());
            System.out.println("SQLState: " + e.getSQLState());
            System.out.println("VendorError: " + e.getErrorCode());
            e.printStackTrace();
        } catch (Exception ex) {
            // handle the error

```

```

        ex.printStackTrace();
    }
    setTitle("Library Management System");
    setSize(340,280);
    setResizable(false);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JTabbedPane BaseTabs=new JTabbedPane();
    User=new JPanel();
    Staff=new JPanel();
    BaseTabs.addTab("UserLogin", User);
    User.setLayout(null);
    BaseTabs.addTab("Staff Login", Staff);
    Staff.setLayout(null);
    JLabel usernameLabel1 = new JLabel("Username");
    JLabel passwordLabel1 = new JLabel("Password");
    this.Login1 = new JButton("Login");
    this.Register = new JButton("Register User");
    this.forgot1 = new JButton("Forgot Password");
    forgot1.setToolTipText("Input you ID here for reteriving password");
    this.cancel1 = new JButton("Cancel");
    this.username1 = new JTextField();
    this.password1 = new JPasswordField();
    usernameLabel1.setBounds(20, 10, 100, 30);
    passwordLabel1.setBounds(20, 50, 100, 30);
    this.username1.setBounds(130, 10, 150, 30);
    this.password1.setBounds(130, 50, 150, 30);
    this.Login1.setBounds(75, 100, 75, 30);
    this.cancel1.setBounds(180, 100, 75, 30);
    this.forgot1.setBounds(75, 140, 150, 30);
    this.Register.setBounds(75, 180, 150, 30);
    User.add(usernameLabel1);
    User.add(passwordLabel1);
    User.add(username1);
    User.add(password1);
    User.add(Login1);
    User.add(Register);
    User.add(cancel1);
    User.add(forgot1);
    Login1.addActionListener(this);
    Register.addActionListener(this);
    cancel1.addActionListener(this);
    forgot1.addActionListener(this);
    //Making staff tab
    JLabel usernameLabel2 = new JLabel("Username");
    JLabel passwordLabel2 = new JLabel("Password");
    this.username2 = new JTextField();
    this.password2 = new JPasswordField();
    this.Login2 = new JButton("Login");
    this.cancel2 = new JButton("Cancel");
    usernameLabel2.setBounds(20, 10, 100, 30);
    passwordLabel2.setBounds(20, 50, 100, 30);

```

```

        this.username2.setBounds(130, 10, 150, 30);
        this.password2.setBounds(130, 50, 150, 30);
        this.Login2.setBounds(75, 100, 75, 30);
        this.cancel2.setBounds(180, 100, 75, 30);
        Staff.add(usernameLabel2);
        Staff.add(passwordLabel2);
        Staff.add(username2);
        Staff.add(password2);
        Staff.add(Login2);
        Staff.add(cancel2);
        Login2.addActionListener(this);
        cancel2.addActionListener(this);
        add(BaseTabs);
        setVisible(true);
    }
    public static void main(String[] args) {
        new LibraryManagementSystem();
    }
    @Override
    public void actionPerformed(ActionEvent e)
    {
        try{
            if(e.getSource()==Login1)
            {
                user=username1.getText();
                pass=password1.getText();
                String loginquery = "select * from member where memberID=? and
password=?";
                PreparedStatement localPreparedStatement =
connection.prepareStatement(loginquery);
                localPreparedStatement.setString(1, user);
                localPreparedStatement.setString(2, pass);
                ResultSet rs = localPreparedStatement.executeQuery();
                if (rs.next())
                {
                    JOptionPane.showMessageDialog(null, "Login Ok", "Project",
1);
                    dispose();
                    new Member(rs.getInt(1));
                }
                else
                {
                    JOptionPane.showMessageDialog(null, "Invalid ID", "Project",
0);
                }
            }
            if(e.getSource()==Login2){
                user=username2.getText();
                pass=password2.getText();
                String loginquery = "select * from staff where staffID=? and
password=?";

```

```

        PreparedStatement localPreparedStatement =
connection.prepareStatement(loginquery);
        localPreparedStatement.setString(1, user);
        localPreparedStatement.setString(2, pass);
        ResultSet rs = localPreparedStatement.executeQuery();
        if (rs.next())
        {
            JOptionPane.showMessageDialog(null, "Login Ok", "Project",
1);

            role=rs.getString(8);
            if(role.equalsIgnoreCase("Admin")){
                admin = new Administrator();
                dispose();
            }
            if(role.equalsIgnoreCase("Librarian")){
                libr = new Librarian();
                dispose();
            }
        }
        else
        {
            JOptionPane.showMessageDialog(null, "Invalid ID", "Project",
0);

        }
    }
    if(e.getSource()==cancel1){
        dispose();
    }
    if(e.getSource()==cancel2){
        dispose();
    }
    if(e.getSource()==forgot1){
        new Forgot(username1.getText());
        dispose();
    }
    if(e.getSource()==Register){
        new Guest();
    }
} catch (Exception ex) {
    ex.printStackTrace();
}
}
}

```

Logout.java

```

import java.util.*;

public class Logout extends ManageBooks {

public Logout() {

```

```

    }

    public void Logout() {
        // TODO implement here
    }
}

```

ManageBooks.java

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;

public class ManageBooks extends Librarian implements ActionListener
{
    Connection conn=LibraryManagementSystem.connection;
    private DefaultTableModel tbm;
    private JTable jt;
    private JDialog viewbooklistdialog;
    private static JDialog addbookdialog;
    private static JLabel ISBN;
    private static JLabel count;
    private static JLabel Name;
    private static JLabel Author;
    private static JLabel domain;
    private static JTextField bookName;
    private static JTextField bookAuthor;
    private static JTextField bookCount;
    private static JTextField bookDomain;
    private static JTextField bookISBN;
    private static JButton Add;
    private static JDialog deletebookdialog;
    private static JButton delete;
    StrategySearch search;
    public void AddBook()
    {

```

```

        addbookdialog=new JDialog();
        addbookdialog.setTitle("Book Details");
        addbookdialog.setModal(true);
        addbookdialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        addbookdialog.setResizable(false);
        addbookdialog.setLayout(null);
        addbookdialog.setSize(300, 300);
        ISBN=new JLabel("Enter ISBN");
        Name=new JLabel("Enter Book Name");
        Author=new JLabel("Enter Author Name");
        count=new JLabel("Enter Book Count");
        domain=new JLabel("Enter Domain Name");
        bookISBN=new JTextField();
        bookName=new JTextField();
        bookAuthor=new JTextField();
        bookCount=new JTextField();
        bookDomain=new JTextField();
        ISBN.setBounds(10,10,100,25);
        Name.setBounds(10,45,100,25);
        Author.setBounds(10, 80, 120, 25);
        count.setBounds(10,115,120,25);
        domain.setBounds(10,150,120,25);
        bookISBN.setBounds(120, 10, 100, 25);
        bookName.setBounds(120, 45, 100, 25);
        bookAuthor.setBounds(120, 80, 100,25);
        bookCount.setBounds(120, 115, 50, 25);
        bookDomain.setBounds(130, 150, 100, 25);
        Add=new JButton("Add Book");
        Add.setBounds(60, 185,120, 30);
        Add.addActionListener(this);
        addbookdialog.add(domain);
        addbookdialog.add(bookDomain);
        addbookdialog.add(Add);
        addbookdialog.add(bookAuthor);
        addbookdialog.add(bookCount);
        addbookdialog.add(bookISBN);
        addbookdialog.add(bookName);
        addbookdialog.add(ISBN);
        addbookdialog.add(Name);
        addbookdialog.add(Author);
        addbookdialog.add(count);
        addbookdialog.setVisible(true);
    }
    public void DeleteBook() {
        deletebookdialog=new JDialog();
        deletebookdialog.setTitle("Book ISBN");
        deletebookdialog.setModal(true);
        deletebookdialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        deletebookdialog.setLayout(null);
        deletebookdialog.setResizable(false);
        deletebookdialog.setSize(300, 150);
    }

```

```

        ISBN=new JLabel("Enter ISBN");
        bookISBN=new JTextField();
        ISBN.setBounds(20,10,100,25);
        bookISBN.setBounds(120, 10, 150, 25);
        delete = new JButton("Delete Book");
        delete.addActionListener(this);
        delete.setBounds(50,45,120,30);
        deletebookdialog.add(ISBN);
        deletebookdialog.add(bookISBN);
        deletebookdialog.add(delete);
        deletebookdialog.setVisible(true);
    }
    public void SearchBook(String searchtype){
        if(searchtype.equalsIgnoreCase("Name"))
            search=new ByName();
        if(searchtype.equalsIgnoreCase("Domain"))
            search=new ByDomain();
        if(searchtype.equalsIgnoreCase("ISBN"))
            search=new ByISBN();
        if(searchtype.equalsIgnoreCase(" Author"))
            search=new ByAuthor();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==Add){
            try{
                Book book = new Book();
                book.setBookName(bookName.getText());
                book.setAuthor(bookAuthor.getText());
                book.setISBN(bookISBN.getText());
                book.setTotalCount(Integer.parseInt(bookCount.getText()));
                book.setAvaibleCount(book.getTotalCount());
                book.setDomain(bookDomain.getText());
                PreparedStatement pst = null;
                String insertIntoMemberQuery = "INSERT INTO Book VALUES "
                    + "(?, ?, ?, ?, ?, ?)";
                pst = conn.prepareStatement(insertIntoMemberQuery);
                pst.setString(1, book.getISBN());
                pst.setString(2, book.getBookName());
                pst.setString(3, book.getAuthor());
                pst.setInt(4, book.getTotalCount());
                pst.setInt(5, book.getAvaibleCount());
                pst.setString(6, book.getDomain());
                pst.executeUpdate();
                JOptionPane.showMessageDialog(null, "Book Added", "Book Added",
1);

                addbookdialog.dispose();
                CheckDomain checkDomain = new CheckDomain();
                checkDomain.Check(book);
            }

```

```

        catch(Exception ex){
            JOptionPane.showMessageDialog(null,"No fields can be empty",
"Error",0);
            ex.printStackTrace();
        }
    }
    if(e.getSource()==delete){
        System.out.println("delte");
        PreparedStatement pst = null;
        String deleteQuery = "DELETE FROM Book WHERE ISBN = ?;";
        try{
            pst = conn.prepareStatement(deleteQuery);
            pst.setString(1,bookISBN.getText());
            pst.executeUpdate();
            JOptionPane.showMessageDialog(null, "Book Deleted", "Book
Deleted", 1);

            deletebookdialog.dispose();
        }
        catch(SQLException ex){
            ex.printStackTrace();
        }
    }
}

public void ViewBookList() {
    try{
        Statement st=conn.createStatement();
        ResultSet rs=st.executeQuery("Select * from book");
        tbm=new DefaultTableModel(new String[]{"ISBN","Book Name","Author","Total
Count","Available Count","Domain"},0);
        while(rs.next())
        {
            tbm.addRow(new
Object[]{rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getString(5),rs.getString(6)});
        }
        jt=new JTable();
        jt.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
        jt.setModel(tbm);
        JScrollPane pane = new JScrollPane(jt);
        viewbooklistdialog=new JDialog();
        viewbooklistdialog.setTitle("Book List");
        jt.setEnabled(false);
        viewbooklistdialog.setSize(400,400);
        jt.setBounds(5,40,390,360);
        viewbooklistdialog.add(pane);
        viewbooklistdialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        viewbooklistdialog.setVisible(true);}
        catch(Exception ex){
            ex.printStackTrace();
        }
    }
}
}

```


ManageStaff.java

```
import java.awt.Component;
import java.awt.Dialog;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;

public class ManageStaff implements ItemListener,ActionListener
{
    private static JLabel firstnamelabel;
    private static JLabel lastnamelabel;
    private static JTextField firstnametext;
    private static JTextField lastnametext;
    private static JLabel emaillabel;
    private static JLabel passwordlabel;
    private static JPasswordField passwordtext;
    private static JLabel sequelabel;
    private static JTextField secquetext;
    private static Component secanslabel;
    private static JTextField secanstext;
    private static JLabel rolalabel;
    private static JButton AddStaff2;
    private static JComboBox<String> roletext;
    private static JDialog addstaffdialog,viewstafflistdialog,removestaffdialog;
    static Connection conn=LibraryManagementSystem.connection;
    private static DefaultTableModel tbm;
    private static JTable jt;
    private static JTextField emailtext;
    private static JButton RemoveStaff2;
```

```

String role=null;
public void AddStaff()
{
    addstaffdialog=new JDialog();
    addstaffdialog.setModal(true);
    addstaffdialog.setTitle("Add Staff");
    addstaffdialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
    addstaffdialog.setSize(350,330);
    addstaffdialog.setResizable(false);
    addstaffdialog.setLayout(null);
    firstnamelabel = new JLabel("First Name");
    firstnamelabel.setBounds(10, 10, 150,25);
    addstaffdialog.add(firstnamelabel);
    lastnamelabel = new JLabel("Last Name");
    lastnamelabel.setBounds(10,45,150,25);
    addstaffdialog.add(lastnamelabel);
    firstnametext=new JTextField();
    firstnametext.setBounds(170, 10, 150, 25);
    addstaffdialog.add(firstnametext);
    lastnametext=new JTextField();
    lastnametext.setBounds(170, 45, 150, 25);
    addstaffdialog.add(lastnametext);
    emaillabel=new JLabel("E-mail Address");
    emaillabel.setBounds(10,80,150,25);
    addstaffdialog.add(emaillabel);
    emailtext = new JTextField();
    emailtext.setBounds(170, 80, 150, 25);
    addstaffdialog.add(emailtext);
    passwordlabel=new JLabel("Password");
    passwordlabel.setBounds(10,115,150,25);
    addstaffdialog.add(passwordlabel);
    passwordtext=new JPasswordField();
    passwordtext.setBounds(170,115,150,25);
    addstaffdialog.add(passwordtext);
    sequelabel=new JLabel("Security Question");
    sequelabel.setBounds(10,150,150,25);
    addstaffdialog.add(sequelabel);
    secquetext=new JTextField();
    secquetext.setBounds(170, 150, 150, 25);
    addstaffdialog.add(secquetext);
    secanslabel=new JLabel("Security Answer");
    secanslabel.setBounds(10,185,150,25);
    addstaffdialog.add(secanslabel);
    secansextext=new JTextField();
    secansextext.setBounds(170,185,150,25);
    addstaffdialog.add(secansextext);
    rolalabel=new JLabel("Role");
    rolalabel.setBounds(10,220,100,25);
    addstaffdialog.add(rolalabel);
    roletext=new JComboBox<>(new String[]{"Admin","Librarian"});
    roletext.setBounds(170, 220, 100, 25);
}

```

```

        roletext.addItemListener(this);
        addstaffdialog.add(roletext);
        AddStaff2=new JButton("Add Staff");
        AddStaff2.setBounds(120, 255, 100, 30);
        addstaffdialog.add(AddStaff2);
        AddStaff2.addActionListener(this);
        addstaffdialog.setVisible(true);
    }
    public void RemoveStaff()
    {
        removestaffdialog=new JDialog();
        removestaffdialog.setTitle("Remove Staff");
        removestaffdialog.setModal(true);
        removestaffdialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        removestaffdialog.setSize(300, 150);
        removestaffdialog.setLayout(null);
        removestaffdialog.setResizable(false);
        emaillabel=new JLabel("Enter Email Address:");
        emailtext=new JTextField();
        emaillabel.setBounds(20,10,200,25);
        emailtext.setBounds(20,40, 200, 25);
        RemoveStaff2 = new JButton("Remove Staff");
        RemoveStaff2.setBounds(20,80,120,30);
        RemoveStaff2.addActionListener(this);
        removestaffdialog.add(emaillabel);
        removestaffdialog.add(emailtext);
        removestaffdialog.add(RemoveStaff2);
        removestaffdialog.setVisible(true);
    }
    public void ViewStaffList() throws SQLException
    {
        Statement st=conn.createStatement();
        ResultSet rs=st.executeQuery("Select * from staff");
        tbm=new DefaultTableModel(new String[]{"StaffID", "First Name", "Last
Name", "Email", "Role"},0);
        while(rs.next())
        {
            tbm.addRow(new
Object[]{rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getString(5)});
        }
        jt=new JTable();
        jt.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
        jt.setModel(tbm);
        JScrollPane pane = new JScrollPane(jt);
        viewstafflistdialog=new JDialog();
        viewstafflistdialog.setTitle("Staff List");
        jt.setEnabled(false);
        viewstafflistdialog.setSize(400,400);
        jt.setBounds(5,40,390,360);
        viewstafflistdialog.add(pane);
        viewstafflistdialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
    }

```

```

        viewstafflistdialog.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        try{
            if(e.getSource()==AddStaff2){
                PreparedStatement pst = null;
                String insertIntoMemberQuery = "INSERT INTO Staff VALUES "
                    + "(?, ?, ?, ?, ?,?,?,?)";
                pst = conn.prepareStatement(insertIntoMemberQuery);
                pst.setString(1, null);
                pst.setString(2, firstnametext.getText());
                pst.setString(3, lastnametext.getText());
                pst.setString(4, emailtext.getText());
                pst.setString(5, passwordtext.getText());
                pst.setString(6, secquetext.getText());
                pst.setString(7, secanstext.getText());
                pst.setString(8, role);
                pst.executeUpdate();
                JOptionPane.showMessageDialog(null, "Staff Added", "Staff Added", 1);
                addstaffdialog.dispose();
                new Administrator();
            }
            if(e.getSource()==RemoveStaff2){
                PreparedStatement pst = null;
                String deleteQuery = "DELETE FROM staff WHERE email = ?;";
                try{
                    pst = conn.prepareStatement(deleteQuery);
                    pst.setString(1,emailtext.getText());
                    pst.executeUpdate();
                    JOptionPane.showMessageDialog(null, "Staff Removed", "Staff
Removed", 1);

                    removestaffdialog.dispose();
                }
                catch(SQLException ex){
                    ex.printStackTrace();
                }
            }
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
    public void itemStateChanged(ItemEvent e) {
        if(e.getSource()==roletext){
            role=roletext.getSelectedItem().toString();
        }
    }
}

```

Observer.java

```

package lms;
import java.util.*;
public interface Observer {
    public void update(List<Subscription> subscribers,Book book);
}

```

OverdueState.java

```

package lms;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;

public class OverdueState extends MemberState implements ActionListener
{
    private JFrame Overdueframe;
    private JDialog addcarddialog;
    private JButton Pay;
    private JButton ViewIssuedBooks;
    private JButton ViewAccountInfo;
    private JButton AddCard,addcard1;
    JFrame AccountInfoframe;
    JLabel memberIDLabel;
    JLabel firstNameLabel;
    JLabel lastNameLabel;
    JLabel emailIDLabel;
    JLabel memberIDTextfield;
    JLabel firstNameText;
    JLabel lastNameText;
    JLabel emailIDText;
    private JTable jt;
    private JDialog issuedbookdialog;
}

```

```

private JLabel memberidlabel;
DefaultTableModel tbm;
Connection conn=LibraryManagementSystem.connection;
String ISBN;
String query,query2;
int checker=0;
int memberID;
int x;
Date d=null;
public OverdueState(){
    }
public OverdueState(int ID)
{
    memberID=ID;
    Overdueframe=new JFrame("Member Overdue");
    Overdueframe.setSize(280,240);
    Overdueframe.setResizable(false);
    Overdueframe.setLayout(null);
    Pay=new JButton("Pay");
    ViewIssuedBooks=new JButton("View Issued Book");
    ViewAccountInfo=new JButton("View Account Info");
    AddCard=new JButton("Add Card");
    Pay.setBounds(10,10,250,40);
    ViewIssuedBooks.setBounds(10, 60, 250, 40);
    ViewAccountInfo.setBounds(10, 110, 250, 40);
    AddCard.setBounds(10, 160, 250, 40);
    Pay.addActionListener(this);
    ViewIssuedBooks.addActionListener(this);
    ViewAccountInfo.addActionListener(this);
    AddCard.addActionListener(this);
    Overdueframe.add(Pay);
    Overdueframe.add(ViewAccountInfo);
    Overdueframe.add(ViewIssuedBooks);
    Overdueframe.add(AddCard);
    Overdueframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Overdueframe.setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==Pay){
        PayPenalty();
    }
    if(e.getSource()==ViewAccountInfo){
        ViewAccountInfo();
    }
    if(e.getSource()==ViewIssuedBooks){
        ViewIssuedBooks();
    }
}

```

```

        if(e.getSource()==AddCard){
            AddCard();
        }
        if(e.getSource()==addcard1){
            setCardInfo(cardInfo);
            JOptionPane.showMessageDialog(null,"Payment method Added", "Card
            Added",1);
            addcarddialog.dispose();
        }
    }
}

```

```

public void CheckPenalty(Date dueDate)
{
    Date dNow = new Date( );
    String dateStart =dueDate.toString();
    String dateStop = dNow.toString();
    SimpleDateFormat format = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");
    Date d1 = null;
    Date d2 = null;
    try {
        d1 = format.parse(dateStart);
        d2 = format.parse(dateStop);
        long diff = d2.getTime() - d1.getTime();
        long diffDays = diff / (24 * 60 * 60 * 1000);
        long penalty= diffDays*2;
        JOptionPane.showMessageDialog(null,("Amount = $" +penalty),"Payment
        Done",1);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

```

@Override
public void PayPenalty()
{
    try{
        query="SELECT DueDate FROM IssuedBook WHERE
        memberID="+memberID+"";
        PreparedStatement ps= conn.prepareStatement(query);
        ResultSet rs= ps.executeQuery(query);
        while(rs.next()){
            d= rs.getDate("DueDate");
            CheckPenalty(d);
        }
    }
    catch(SQLException e){
        e.printStackTrace();
    }
}

```

```

    }
    JOptionPane.showMessageDialog(null,"Amount Paid","Payment Done",1);
    Overdueframe.dispose();
    setState(getMemState());
}
public void ViewIssuedBooks()
{
    query="SELECT * FROM IssuedBook WHERE memberID="+memberID+"";
    try{
        PreparedStatement ps= conn.prepareStatement(query);
        ResultSet rs=ps.executeQuery(query);
        if(rs.next())
        {
            tbm=new DefaultTableModel(new String[]{"Order ID","ISBN","Date of
            Issue","Due Date"},0);
            rs.beforeFirst();
            while(rs.next()){
                tbm.addRow(newString[]{rs.getString(1),rs.getString(2),rs.getStr
                ing(4),rs.getString(5)});
            }
            jt=new JTable();
            jt.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
            jt.setModel(tbm);
            JScrollPane pane = new JScrollPane(jt);
            issuedbookdialog=new JDialog(Overdueframe, "Issued Book List");
            jt.setEnabled(false);
            issuedbookdialog.setSize(400,400);
            jt.setBounds(5,40,390,360);
            issuedbookdialog.add(pane);
            issuedbookdialog.setVisible(true);
        }
        else{
            JOptionPane.showMessageDialog(null,"No issued book","No issued
            book",0);
        }
    }
    catch(SQLException e){
    }
}
public void ViewAccountInfo()
{
    System.out.println("View Account info");
    AccountInfoframe = new JFrame("Member Information");
    AccountInfoframe.setSize(300,260);
    AccountInfoframe.setLayout(null);
    AccountInfoframe.setResizable(false);
    query="SELECT * FROM Member WHERE memberID="+memberID+"";
    try{

```



```

        PreparedStatement ps= conn.prepareStatement(query);
        ResultSet rs= ps.executeQuery();
        if(rs.next()){
            memberidlabel = new JLabel("Member ID : "+rs.getString(1));
            firstNameLabel=new JLabel("First Name : "+rs.getString(2));
            lastNameLabel=new JLabel("Last Name : "+rs.getString(3));
            emailIDLabel=new JLabel("Email ID : "+rs.getString(4));
            memberidlabel.setBounds(10, 10, 250, 40);
            AccountInfoframe.add(memberidlabel);
            firstNameLabel.setBounds(10, 60, 250, 40);
            AccountInfoframe.add(firstNameLabel);
            lastNameLabel.setBounds(10, 110, 250, 40);
            AccountInfoframe.add(lastNameLabel);
            emailIDLabel.setBounds(10, 160, 250, 40);
            AccountInfoframe.add(emailIDLabel);
            AccountInfoframe.setVisible(true);
        }
    }
    catch(SQLException e){
    }
}

@Override
public void AddCard(){
    addcarddialog=new JDialog();
    addcarddialog.setTitle("Add Card");
    addcarddialog.setSize(250,180);
    addcarddialog.setResizable(false);
    addcarddialog.setLayout(null);
    JLabel cardlabel=new JLabel("Enter Card Number");
    cardlabel.setBounds(20,10,150,30);
    addcarddialog.add(cardlabel);
    JTextField cardtext=new JTextField();
    cardtext.setBounds(20, 50, 200, 30);
    addcarddialog.add(cardtext);
    addcard1=new JButton("Add Card");
    addcard1.setBounds(20, 90, 120, 30);
    addcard1.addActionListener(this);
    addcarddialog.add(addcard1);
    addcarddialog.setVisible(true);
}
}

```

SearchBook.java

```

package lms;
import java.sql.SQLException;

```

```

public class SearchBook extends ManageBooks{
public void SearchBook(String searchtype) {
    super.SearchBook(searchtype);
}
public void search() throws SQLException{
    }
}

```

Staff.java

```

package lms;
import java.util.*;
public class Staff implements User {
    private String employeeId;
    private String role;
    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }

    public String getEmployeeId() {
        return employeeId;
    }

    public void setEmployeeId(String employeeId) {
        this.employeeId = employeeId;
    }

    @Override
    public String getFName() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void setFName(String fName) {
        // TODO Auto-generated method stub

    }

    @Override
    public String getLName() {
        // TODO Auto-generated method stub
        return null;
    }
}

```

```

@Override
    public void setLName(String lName) {
        // TODO Auto-generated method stub
    }

    @Override
    public String getEmail() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void setEmail(String email) {
        // TODO Auto-generated method stub
    }

    @Override
    public String getPassword() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void setPassword(String password) {
        // TODO Auto-generated method stub
    }

    @Override
    public String getSecQuestion() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void setSecQuestion(String secQuestion) {
        // TODO Auto-generated method stub
    }

    @Override
    public String getSecAnswers() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void setSecAnswers(String secAnswers) {

```

```

        // TODO Auto-generated method stub
    }

    @Override
    public void setState(MemberState s) {
        // TODO Auto-generated method stub
    }

    @Override
    public MemberState getMemState() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public MemberState getOverdueState() {
        // TODO Auto-generated method stub
        return null;}

    @Override
    public int getMemberID() {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public void setMemberID(int memberID) {
        // TODO Auto-generated method stub
    }

    @Override
    public int getCardInfo() {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public void setCardInfo(int cardInfo) {
        // TODO Auto-generated method stub
    }

    @Override
    public void RequestBook(String ISBN) {
        // TODO Auto-generated method stub
    }

    @Override

```

```

    public void RenewBook() {
        // TODO Auto-generated method stub
    }

    @Override
    public void ReturnBook(String ISBN) {
        // TODO Auto-generated method stub
    }

    @Override
    public void AddCard() {
        // TODO Auto-generated method stub
    }

    @Override
    public void ViewIssuedBooks() {
        // TODO Auto-generated method stub
    }

    @Override
    public void ViewAccountInfo(int memberID) {
        // TODO Auto-generated method stub
    }

    @Override
    public void getBookISBN() {
        // TODO Auto-generated method stub
    }

    @Override
    public void PayPenalty() {
        // TODO Auto-generated method stub
    }
}

```

State.java

```

package lms;
import java.util.*;
public class State {

    /**
     * Default constructor
     */
    public State() {
    }
}

```

StrategySearch.java

```
package lms;
import java.sql.SQLException;
import java.util.*;

abstract class StrategySearch extends SearchBook {
    public StrategySearch() {
    }
    public void search() throws SQLException {
        // TODO implement here
    }
}
```

Subject.java

```
package lms;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.*;

public class Subject {

    private List<Subscription> observerList;

    public Subject(String domain) {

        this.observerList = getSubscriberList(domain);
    }
    public List<Subscription> getSubscriberList(String domain){

        Connection connection = LibraryManagementSystem.connection;
        //PreparedStatement pst = null;
        String bookListQuery = "SELECT * FROM Subscriber WHERE domain =\"" + domain + "\"";
        ResultSet result = null;
        List<Subscription> subsList = new ArrayList<>();
        try{
            Statement statement = connection.createStatement();
            result = statement.executeQuery(bookListQuery);
            while(result.next()) {
                String userEmail = result.getString("emailID");
                String userDomain = result.getString("domain");
```

```

        Subscription subs = new Subscription();
        subs.setEmailId(userEmail);
        subs.setDomain(userDomain);
        subsList.add(subs);
    }
}
catch(SQLException e){
    e.printStackTrace();
}
return subsList;
}
public void Register(String emailId, String domain) {

    Connection connection = LibraryManagement.conn;
    PreparedStatement pst = null;
    String insertIntoSubscriptionQuery = "INSERT INTO Subscriber VALUES "+ "(NULL, ?, ?)";
    try{
        pst = connection.prepareStatement(insertIntoSubscriptionQuery);
        pst.setString(1, emailId);
        pst.setString(2, domain);
        pst.executeUpdate();
    }
    catch(SQLException e){
        e.printStackTrace();
    }
}
public void Unregister(String emailId, String domain) {
    Connection connection = LibraryManagement.conn;
    PreparedStatement pst = null;
    String deleteQuery = "DELETE FROM Subscriber WHERE emailID = "+ emailId +"AND
domain="+domain +"";
    try{
        pst = connection.prepareStatement(deleteQuery);
        pst.executeUpdate();
    }
    catch(SQLException e){
        e.printStackTrace();
    }
}
public void notifyUsers(Book book) {

    Observer obs = new Member();
    obs.update(observerList,book);

}
public List<Subscription> getObserverList() {

    return observerList;
}

```

```

    }
    public void setObserverList(List<Subscription> observerList) {
        this.observerList = observerList;
    }
}

```

Subscription.java

```

package lms;
public class Subscription {

    private int id;
    private String EmailId;
    private String Domain;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getEmailId() {
        return EmailId;
    }
    public void setEmailId(String emailId) {
        EmailId = emailId;
    }
    public String getDomain() {
        return Domain;
    }
    public void setDomain(String domain) {
        Domain = domain;
    }
}

```

User.java

```

package lms;
import java.util.*;
public interface User{
    public String getFName();
    public void setFName(String fName);
    public String getLName();
}

```



```

public void setLName(String lName);
public String getEmail();
public void setEmail(String email);
public String getPassword();
public void setPassword(String password);
public String getSecQuestion();
public void setSecQuestion(String secQuestion);
public String getSecAnswers();
public void setSecAnswers(String secAnswers);
public void setState(MemberState s);
public MemberState getMemState();
public MemberState getOverdueState();
public int getMemberID();
public void setMemberID(int memberID);
public int getCardInfo();
public void setCardInfo(int cardInfo);
public void RequestBook(String ISBN);
public void RenewBook();
public void ReturnBook(String ISBN);
public void AddCard();
public void ViewIssuedBooks();
public void ViewAccountInfo(int memberID);
public void getBookISBN();
void PayPenalty();}

```

UserLoginFactory.java

```

package lms;
import java.util.*;
public class UserLoginFactory {

    /**
     * Default constructor
     */
    public UserLoginFactory() {
    }
}

```

ViewAllRequestedBooks.java

```

package lms;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JDialog;
import javax.swing.JScrollPane;

```

```

import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class ViewAllRequestedBooks extends Librarian{
    private Connection conn=LibraryManagementSystem.connection;
    private DefaultTableModel tbm;
    private JTable jt;
    private JDialog viewissuebookdialog;

    void ViewAllRequestedBooks(){
        try {
            PreparedStatement ps=conn.prepareStatement("Select * from IssuedBook
            where pending=1;");
            ResultSet rs=ps.executeQuery();
            tbm=new DefaultTableModel(new String[]{"Order ID","ISBN","Member
            ID","Date of Issue","Due Date"},0);
            while(rs.next())
            {
                tbm.addRow(new Object[]{rs.getString(1),rs.getString(2),rs.getString(3),rs.getStri
                ng(4),rs.getString(5)});
            }
            jt=new JTable();
            jt.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
            jt.setModel(tbm);
            JScrollPane pane = new JScrollPane(jt);
            viewissuebookdialog=new JDialog();
            viewissuebookdialog.setTitle("Book List");
            jt.setEnabled(false);
            viewissuebookdialog.setSize(400,400);
            jt.setBounds(5,40,390,360);
            viewissuebookdialog.add(pane);
            viewissuebookdialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
            viewissuebookdialog.setVisible(true);
        }catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

ViewUserList.java

```

package lms;
import java.sql.SQLException;
import java.util.*;
public class ViewUserList extends ManageUser

```

```

{
    public ViewUserList() throws SQLException {
        super.ViewUserList();
    }
}

```

CreateDatabase.java

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
//package Main;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
/*
import dbConnection.makeConnection;
import Member;
import Staff;
import Book;
import IssuedBook;

*/

/**
 *
 * @author 5CB4125SDX0
 */
public class MainClass {
    public static void main(String args[]){
        Connection conn = null;
        conn = makeConnection.getDafaultConnection();
        if (conn != null) {

                MainClass mainClass = new MainClass();

                // create database and tablespace

                mainClass.createTablespace(conn);

```

```

        conn = makeConnection.getConnection("library");

        // create tables

        // if you dont wanaa create users again dont uncomment createUsers
        // function and grantPermission function.

        mainClass.createTables(conn);
        // mainClass.createUsers(conn);

        // create users

        // grant permissions

        // mainClass.grantPermission(conn);
        mainClass.insertValuesToTables(conn);
        // insert into tables

    } else {
        System.out.println("Driver is not connected");
    }
}

public void createTablespace(Connection conn) {

    // query
    String createDatabaseQuery = "CREATE DATABASE library";
    String useDB = "USE library";
    String createTablespaceCommand = "CREATE TABLESPACE library ADD DATAFILE
'library.ibd'";
    Statement statement = null;
    try {
        statement = conn.createStatement();
        statement.executeUpdate(createDatabaseQuery);
        statement.executeUpdate(useDB);
        statement.executeUpdate(createTablespaceCommand);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // statement
}

```

```

public void createTables(Connection conn) {
    Statement statement = null;
    try {
        statement = conn.createStatement();

        String createTableMemberQuery = "CREATE TABLE Member (memberID int(8) NOT NULL
        AUTO_INCREMENT, firstName VARCHAR(16), lastName VARCHAR(16),"
        + "email VARCHAR (50) UNIQUE, password VARCHAR(16), securityQues
        VARCHAR(50), securityAns VARCHAR(50),"
        + "CONSTRAINT MemberPK PRIMARY KEY (memberID));";

        statement.executeUpdate(createTableMemberQuery) ;

        String createTableStaffQuery = "CREATE TABLE Staff (staffID int(8) NOT NULL
        AUTO_INCREMENT, firstName VARCHAR(16), lastName VARCHAR(16),"
        + "email VARCHAR (50) UNIQUE, password VARCHAR(16), securityQues
        VARCHAR(50), securityAns VARCHAR(50), Role VARCHAR(15),"
        + "CONSTRAINT StaffPK PRIMARY KEY (staffID));";

        statement.executeUpdate(createTableStaffQuery) ;

        String createTableBookQuery = "CREATE TABLE Book (ISBN VARCHAR(25) NOT NULL,
        BookName VARCHAR(50), Author VARCHAR(50), TotalCount int(3),"
        + "AvailableCount int(4), Domain VARCHAR (25), CONSTRAINT BookPK PRIMARY
        KEY(ISBN));";

        statement.executeUpdate(createTableBookQuery) ;

        String createTableIssuedBookQuery = "CREATE TABLE IssuedBook (OrderID int (8) NOT NULL
        AUTO_INCREMENT, ISBN VARCHAR(25), memberID int(8),"
        + "DateOfIssue DATE, DueDate DATE, Pending int (8), CONSTRAINT IssuedBookPK
        PRIMARY KEY (OrderID));";
        /*
        + " FOREIGN KEY (memberID) REFERENCES Member (memberID),"
        + " FOREIGN KEY (ISBN) REFERENCES Book (ISBN));";
        */
        statement.executeUpdate(createTableIssuedBookQuery) ;

        String createTableSubscriber="CREATE TABLE Subscriber (Sr_Number int(8) NOT NULL
        AUTO_INCREMENT, EmailID VARCHAR(50), Domain VARCHAR (25), CONSTRAINT SubscriberPK PRIMARY
        KEY (Sr_Number));";
        statement.executeUpdate(createTableSubscriber) ;

    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

// Function to create Member objects and return list of those objects

```
private List<Member> getMemberObjects() {
    List<Member> MemberObjectList = new ArrayList<Member>();

    Member MemberObject1 = new Member();
    //MemberObject1.setmemberID(123);
    MemberObject1.setFirstName("ABC");
    MemberObject1.setLastName("XYZ");
    MemberObject1.setEmail("abc@gmail.com");
    MemberObject1.setPassword("ABC123");
    MemberObject1.setSecurityQues("Which is the your favourite animal?");
    MemberObject1.setSecurityAns("Cat");

    Member MemberObject2 = new Member();
    //MemberObject1.setmemberID(123);
    MemberObject2.setFirstName("PQR");
    MemberObject2.setLastName("LMN");
    MemberObject2.setEmail("pqr@gmail.com");
    MemberObject2.setPassword("PQR123");
    MemberObject2.setSecurityQues("Which is the your favourite car?");
    MemberObject2.setSecurityAns("Honda");

    MemberObjectList.add(MemberObject1);
    MemberObjectList.add(MemberObject2);

    return MemberObjectList;
}
```

```
private List<Staff> getStaffObjects() {
    List<Staff> StaffObjectList = new ArrayList<Staff>();

    Staff StaffObject1 = new Staff();
    StaffObject1.setFirstName("Roshni");
    StaffObject1.setLastName("Masand");
    StaffObject1.setEmail("rmasand@gmail.com");
    StaffObject1.setPassword("roshni123");
    StaffObject1.setSecurityQues("Which is the your favourite animal?");
    StaffObject1.setSecurityAns("Mouse");
    StaffObject1.setRole("Admin");

    Staff StaffObject2 = new Staff();
    StaffObject2.setFirstName("Harshit");
```

```

        StaffObject2.setLastName("Singh");
        StaffObject2.setEmail("hsingh@gmail.com");
        StaffObject2.setPassword("harshit123");
        StaffObject2.setSecurityQues("Which is the your favourite car?");
        StaffObject2.setSecurityAns("BMW");
        StaffObject2.setRole("Librarian");

        StaffObjectList.add(StaffObject1);
        StaffObjectList.add(StaffObject2);

        return StaffObjectList;
    }

    private List<Book> getBookObjects() {
        List<Book> BookObjectList = new ArrayList<Book>();

        Book BookObject1 = new Book();
        BookObject1.setISBN("9780201038019");
        BookObject1.setBookName("The Art of Computer Programming");
        BookObject1.setAuthor("Donald Knuth");
        BookObject1.setAvaibleCount(4);
        BookObject1.setTotalCount(5);
        BookObject1.setDomain("Computer Science");

        Book BookObject2 = new Book();
        BookObject2.setISBN("9780201485370");
        BookObject2.setBookName("Design Patterns");
        BookObject2.setAuthor("Erich Gamma");
        BookObject2.setAvaibleCount(4);
        BookObject2.setTotalCount(5);
        BookObject2.setDomain("Computer Science");

        Book BookObject3 = new Book();
        BookObject3.setISBN("9780262032933");
        BookObject3.setBookName("Introduction of Algorithms");
        BookObject3.setAuthor("CLRS");
        BookObject3.setAvaibleCount(10);
        BookObject3.setTotalCount(10);
        BookObject3.setDomain("Computer Science");

        Book BookObject4 = new Book();
        BookObject4.setISBN("97465871235735");
        BookObject4.setBookName("Matilda");
        BookObject4.setAuthor("Roald Dahn");
        BookObject4.setAvaibleCount(7);
        BookObject4.setTotalCount(8);
    }

```

```

        BookObject4.setDomain("Fiction");

        BookObjectList.add(BookObject1);
        BookObjectList.add(BookObject2);
        BookObjectList.add(BookObject3);
        BookObjectList.add(BookObject4);

        return BookObjectList;
    }

    private List<IssuedBook> getIssuedBookObjects() {
        List<IssuedBook> IssuedBookObjectList = new ArrayList<IssuedBook>();

        IssuedBook IssuedBookObject1 = new IssuedBook();
        IssuedBookObject1.setISBN("9780201038019");
        IssuedBookObject1.setMemberID(1);
        IssuedBookObject1.setPending(0);

        IssuedBook IssuedBookObject2 = new IssuedBook();
        IssuedBookObject2.setISBN("9780201485370");
        IssuedBookObject2.setMemberID(2);
        IssuedBookObject2.setPending(1);

        IssuedBook IssuedBookObject3 = new IssuedBook();
        IssuedBookObject3.setISBN("97465871235735");
        IssuedBookObject3.setMemberID(1);
        IssuedBookObject3.setPending(1);

        IssuedBookObjectList.add(IssuedBookObject1);
        IssuedBookObjectList.add(IssuedBookObject2);
        IssuedBookObjectList.add(IssuedBookObject3);

        return IssuedBookObjectList;
    }

    private List<Subscriber> getSubscriberObjects() {
        List<Subscriber> SubscriberObjectList = new ArrayList<Subscriber>();

        Subscriber SubscriberObject1 = new Subscriber();
        SubscriberObject1.setEmail("amore@hawk.iit.edu");
        SubscriberObject1.setDomain("Computer Science");

        Subscriber SubscriberObject2 = new Subscriber();
        SubscriberObject2.setEmail("rmasand@hawk.iit.edu");
        SubscriberObject2.setDomain("Computer Science");
    }

```



```

        SubscriberObjectList.add(SubscriberObject1);
        SubscriberObjectList.add(SubscriberObject2);

    return SubscriberObjectList;

}

public void insertValuesToTables(Connection conn) {
    this.insertIntoMemberTable(conn);
    this.insertIntoStaffTable(conn);
    this.insertIntoBookTable(conn);
    this.insertIntoIssuedBookTable(conn);
    this.insertIntoSubscriberTable(conn);
}

private void insertIntoMemberTable(Connection conn){

    PreparedStatement pst = null;

    String insertIntoMemberQuery = "INSERT INTO Member VALUES "
        + "(NULL, ?, ?, ?, ?, ?, ?)";

    try{

        for(Member member : this.getMemberObjects()){
            pst = conn.prepareStatement(insertIntoMemberQuery);
            pst.setString(1, member.getFirstName());
            pst.setString(2, member.getLastName());
            pst.setString(3, member.getEmail());
            pst.setString(4, member.getPassword());
            pst.setString(5, member.getSecurityQues());
            pst.setString(6, member.getSecurityAns());

            pst.executeUpdate();

        }
    }
    catch(SQLException e){
        e.printStackTrace();
    }
}

private void insertIntoStaffTable(Connection conn){

    PreparedStatement pst = null;

    String insertIntoStaffQuery = "INSERT INTO Staff VALUES "

```

```

        + "(NULL, ?, ?, ?, ?, ?, ?)";

    try{

        for(Staff staff : this.getStaffObjects()){
            pst = conn.prepareStatement(insertIntoStaffQuery);
            pst.setString(1, staff.getFirstName());
            pst.setString(2, staff.getLastName());
            pst.setString(3, staff.getEmail());
            pst.setString(4, staff.getPassword());
            pst.setString(5, staff.getSecurityQues());
            pst.setString(6, staff.getSecurityAns());
            pst.setString(7, staff.getRole());

            pst.executeUpdate();

        }
    }
    catch(SQLException e){
        e.printStackTrace();
    }
}

private void insertIntoBookTable(Connection conn){

    PreparedStatement pst = null;

    String insertIntoBookQuery = "INSERT INTO Book VALUES "
        + "(?, ?, ?, ?, ?, ?)";

    try{

        for(Book book : this.getBookObjects()){
            pst = conn.prepareStatement(insertIntoBookQuery);
            pst.setString(1, book.getISBN());
            pst.setString(2, book.getBookName());
            pst.setString(3, book.getAuthor());
            pst.setInt(4, book.getTotalCount());
            pst.setInt(5, book.getAvailableCount());
            pst.setString(6, book.getDomain());

            pst.executeUpdate();

        }
    }
    catch(SQLException e){
        e.printStackTrace();
    }
}

```

```

    }
}

private void insertIntoIssuedBookTable(Connection conn){

    PreparedStatement pst = null;

    String insertIntoIssuedBookQuery = "INSERT INTO IssuedBook VALUES "
        + "(NULL, ?, ?, NOW(), ADDDATE(CURDATE(),14, ?));";

    try{

        for(IssuedBook issuedBook : this.getIssuedBookObjects()){
            pst = conn.prepareStatement(insertIntoIssuedBookQuery);
            pst.setString(1, issuedBook.getISBN());

            pst.setInt(2, issuedBook.getMemberID());
            pst.setInt(3, issuedBook.getPending());

            pst.executeUpdate();

        }
    }
    catch(SQLException e){
        e.printStackTrace();
    }
}

private void insertIntoSubscriberTable(Connection conn){

    PreparedStatement pst = null;

    String insertIntoSubscriberQuery = "INSERT INTO Subscriber VALUES "
        + "(NULL, ?, ?);";

    try{

        for(Subscriber subscriber : this.getSubscriberObjects()){
            pst = conn.prepareStatement(insertIntoSubscriberQuery);
            pst.setString(1, subscriber.getEmail());

            pst.setString(2, subscriber.getDomain());

            pst.executeUpdate();

```

```

    }
}
catch(SQLException e){
    e.printStackTrace();
}
}

```

12.Screenshots

Screen Shots of Database

Book Table:

This table has details of the Books present in the Library Management System

The attributes in the table are ISBN, BookName, Author, TotalCount, AvailableCount and Domain

Connection Information

Name: abc
Host: 127.0.0.1:3306
Server: MySQL
Version: 5.1.52-community
User: root

ISBN	BookName	Author	TotalCount	AvailableCount	domain
12345678	Let us C	E Balaguru Swamy	10	9	Computer Science
123456780987	Gang of Four	Erich Gamma	10	10	Computer Science
2345678	Gangs of four	Erich Gamma	10	10	Computers
555555	book4	auth	10	10	Computer Science
654345645	Laws of universe	Erich Bachman	12	32	Astronomy
6564353567	book1	auth	4	4	Computer Science
718238917	Computer Science	Me	3	3	Computer Science
7654323	Palmistry	David Bryan	34	34	Astrology
7777777	book3	auth	10	10	Computer Science
8888	book6	auth	10	10	Computer Science
9780201485370	Design Patterns	Erich Gamma	5	3	Computer Science
987654323456	ABCDEFGHJI	qwerty	10	10	Computer Science
*					

Member Table:

This table has details of the Members registered to the Library Management System

The attributes in the table are memberID, firstName, lastName, email, password, security questions and security answers.

Object Browser

Default:

- emp_details
- jdbc1
- library

SQL Editor (abc) x

SQL Query* SQL Query SQL Query SQL Query* SQL Query x

1 • EDIT 'library`.`member`;

Overview Output Snippets member (1) x

Fetches 6 records

	memberID	firstName	lastName	email	password	securityQues	securityAns
▶	1	ABC	XYZ	abc@gmail.com	1	Which is the your favourite animal?	Cat
	3	ABC	XYZ	abc@gmail.com	ABC123	Which is the your favourite animal?	Cat
	5	ABC	XYZ	abc@gmail.com	ABC123	Which is the your favourite animal?	Cat
	7	Aditya	More	adimore@gmail.com	adi123	Why?	Abc
	8	Tara	Ramchandran	tara@gmail.com	tara123	why?	abc
	9	Harshit	Singh	qwert@gmail.com	123	car?	BMW
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Staff Table:

This table has details of the Staff in the Library Management System

The attributes in the table are staffID, firstName, lastName, email, password, security questions, security answers and role.

The screenshot shows a database management interface. On the left is the 'Object Browser' with a tree view containing 'emp_details', 'jdbc1', 'library', 'Tables' (with sub-items 'book', 'issuedbook', 'member', 'staff', 'subscriber'), 'Views', and 'Routines'. The 'staff' table is selected. The main area is the 'SQL Editor' with a tab for 'staff (1)*'. Below the editor is a table view showing 3 records. The table has columns: staffID, firstName, lastName, email, password, securityQues, securityAns, and Role. The data rows are: (1, Roshni, Masand, rmasand@gmail.com, roshni123, 'Which is the your favourite animal?', 'Mouse', 'Admin'), (3, Aditya, More, Aditya@gmail.com, Aditya123, 'Who?', 'abc', 'Librarian'), and (4, harshit, Singh, harshit@gmail.com, 123, 'Favorite car?', 'BMW', 'Librarian'). A fourth row with all NULL values is also visible.

staffID	firstName	lastName	email	password	securityQues	securityAns	Role
1	Roshni	Masand	rmasand@gmail.com	roshni123	Which is the your favourite animal?	Mouse	Admin
3	Aditya	More	Aditya@gmail.com	Aditya123	Who?	abc	Librarian
4	harshit	Singh	harshit@gmail.com	123	Favorite car?	BMW	Librarian
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

IssuedBook Table:

This table has the details of the books requested by the member.

The attributes of this table are OrderID, ISBN, memberID, DateOfIssue, DueDate and Pending

The screenshot shows a database management interface. On the left, the 'Object Browser' displays a tree structure with 'emp_details', 'jdbc1', and 'library'. The main area is the 'SQL Editor' with a tab for 'SQL Query' containing the text: `EDIT `library`.`issuedbook`;`. Below the editor, the 'Overview' tab is active, showing a table view of the 'issuedbook' table. The table has 7 columns: OrderID, ISBN, memberID, DateOfIssue, DueDate, and Pending. It displays 4 records. A status bar at the bottom right indicates 'Fetched 4 records'.

	OrderID	ISBN	memberID	DateOfIssue	DueDate	Pending
▶	2	9780201485370	2	2016-06-30	2016-07-14	0
	3	3245678	3	2016-06-30	2016-07-14	0
	7	12345678	1	2016-06-30	2016-07-14	0
	8	9780201485370	1	2016-06-30	2016-07-14	0
*	NULL	NULL	NULL	NULL	NULL	NULL

Subscriber Table:

This table consists of details of the members who have subscribed for notification to a particular domain.

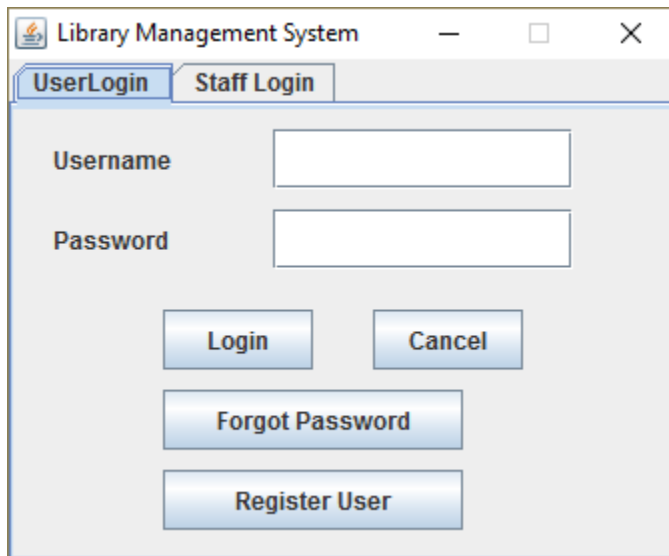
The attributes of this table are emailID and domain

The screenshot displays a database management interface. On the left, the 'Object Browser' shows a tree structure with 'emp_details', 'jdbc1', and 'library'. Under 'library', there is a 'Tables' folder containing 'book', 'issuedbook', 'member', 'staff', and 'subscriber'. The 'subscriber' table is selected. The main window shows an 'SQL Query' editor with the text: `EDIT `library`.`subscriber`;`. Below the editor, there are tabs for 'Overview', 'Output', 'Snippets', 'staff (1)*', and 'subscriber (1)*'. The 'subscriber (1)*' tab is active, showing a table with the following data:

Sr_Number	emailID	domain
1	pranishap1288@gmail.com	Computer Science
2	adityamore950@gmail.com	Fiction
5	hrshtsngh98@gmail.com	wqertyui
▶*	NULL	NULL

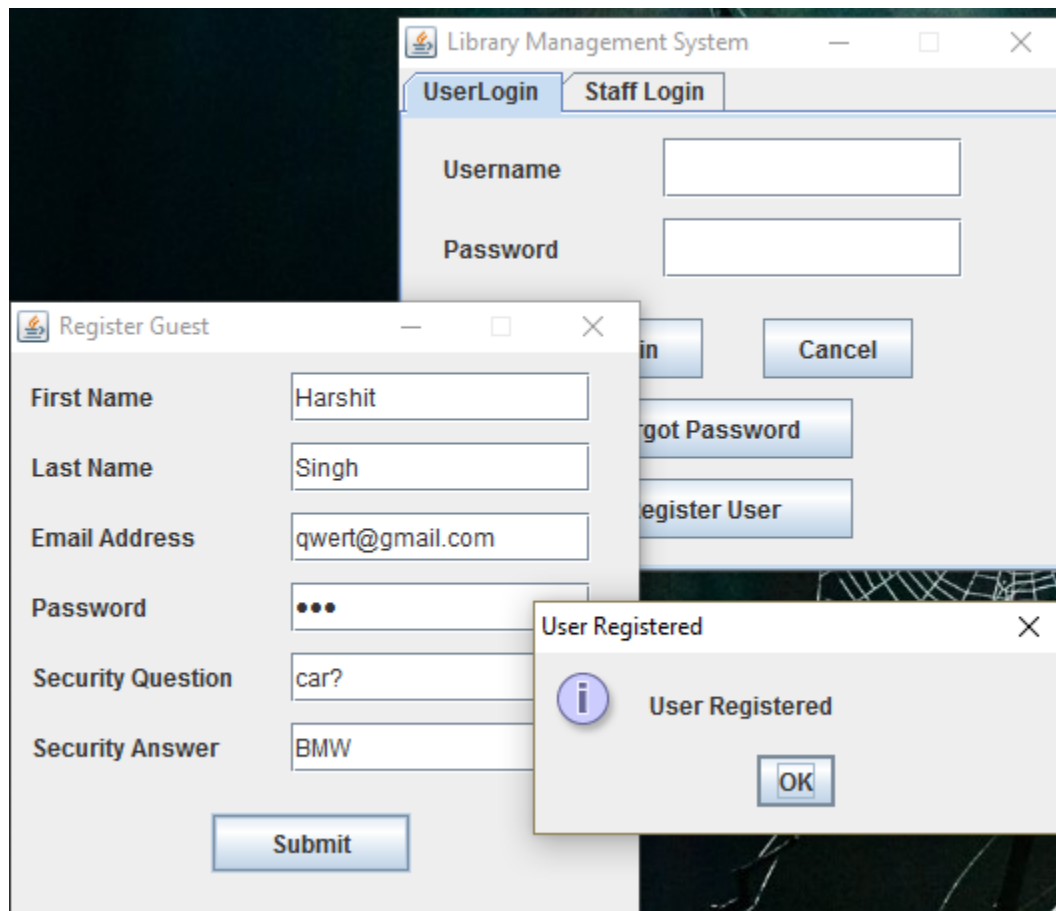
Screen Shots of GUI

Login Page: The first page that opens whenever the system is to be used.

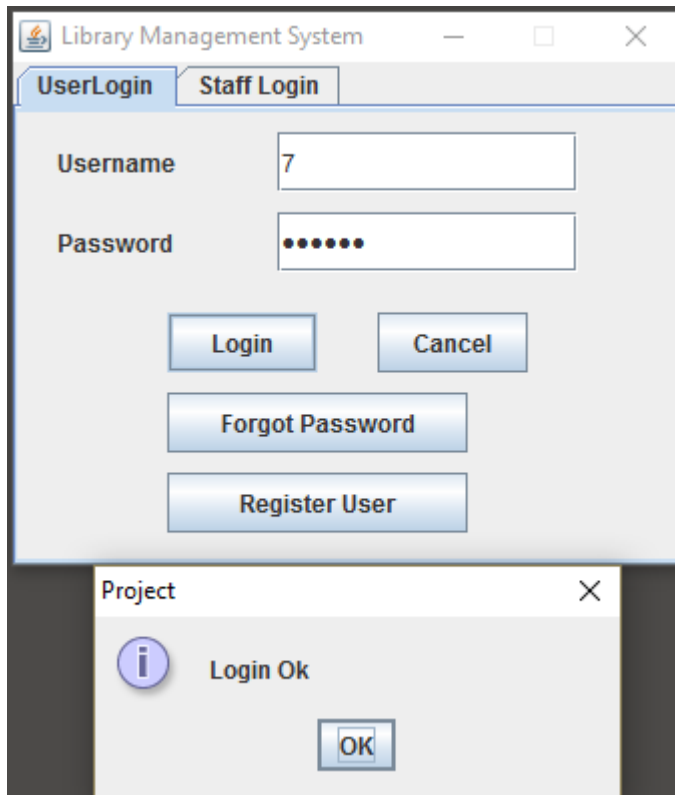


The screenshot shows a window titled "Library Management System" with standard Windows window controls (minimize, maximize, close). Inside the window, there are two tabs: "UserLogin" (which is selected) and "Staff Login". Below the tabs, there are two input fields: "Username" and "Password". Below these fields are four buttons: "Login", "Cancel", "Forgot Password", and "Register User". The "Register User" button is at the bottom of the form.

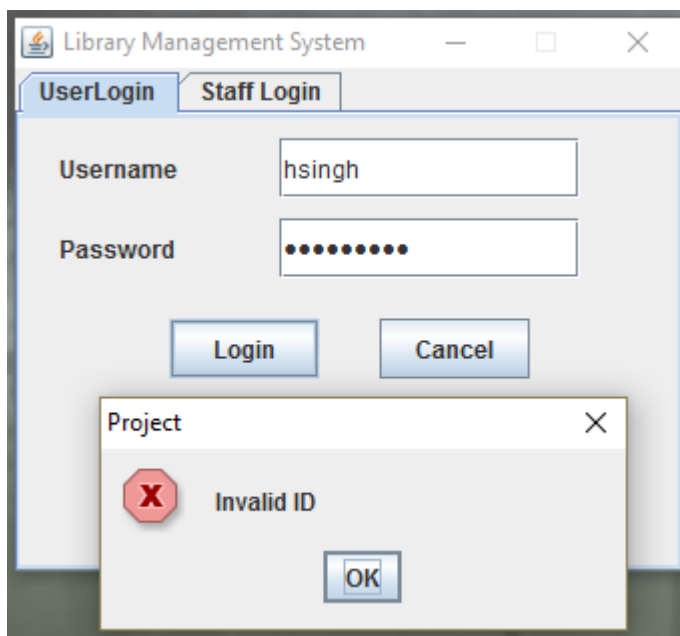
Register User: A new/guest user can register onto the Library Management System.



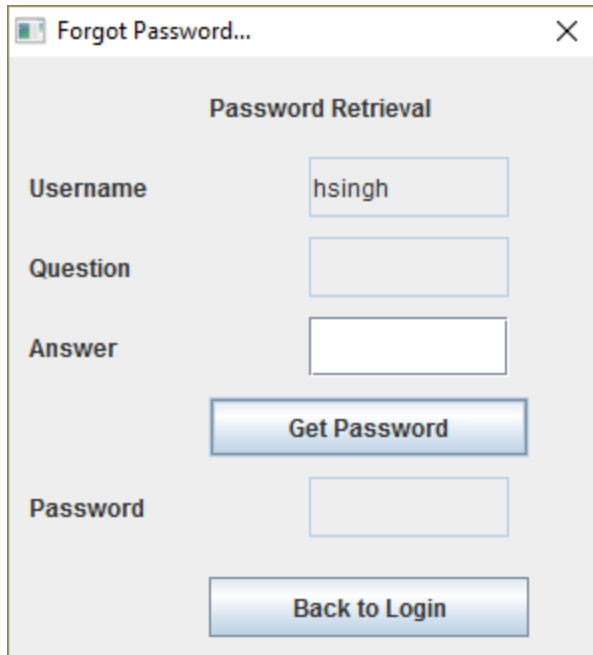
Member Login Confirmed: Once an account is created, the member can login to the system.



Wrong Login: If the username and password does not match the details in the database, invalid login is displayed.



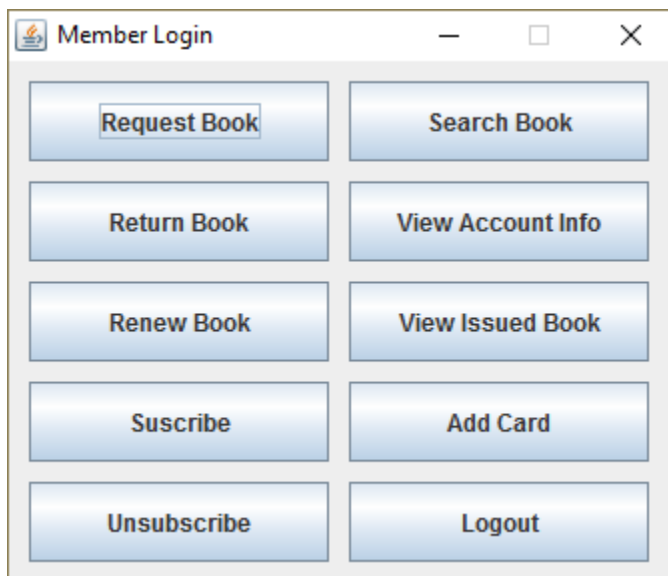
Forgot Password: If the Member forgets the password, he/she can restore the password using the Forgot Password Label.



A dialog box titled "Forgot Password..." with a close button (X) in the top right corner. The dialog has a title bar and a main content area. The content area is titled "Password Retrieval". It contains four input fields: "Username" (with the text "hsingh"), "Question", "Answer", and "Password". Below the "Answer" field is a blue button labeled "Get Password". Below the "Password" field is a blue button labeled "Back to Login".

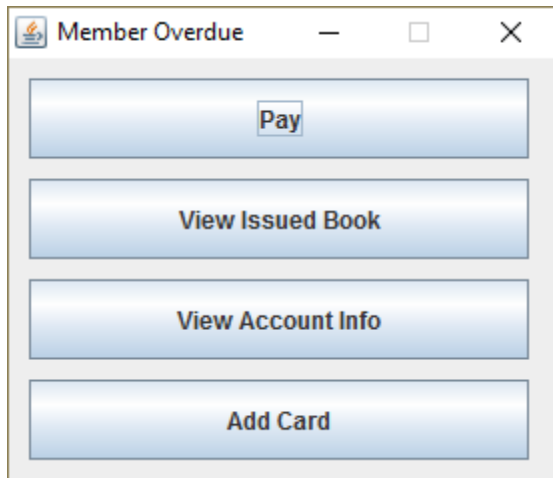
After login, the system will go into either Normal(MemState) or Overdue State depending on the books issued and due.

Member Login MemState: If the Member logs in and has no dues to pay, he/she is directed to the member login page.

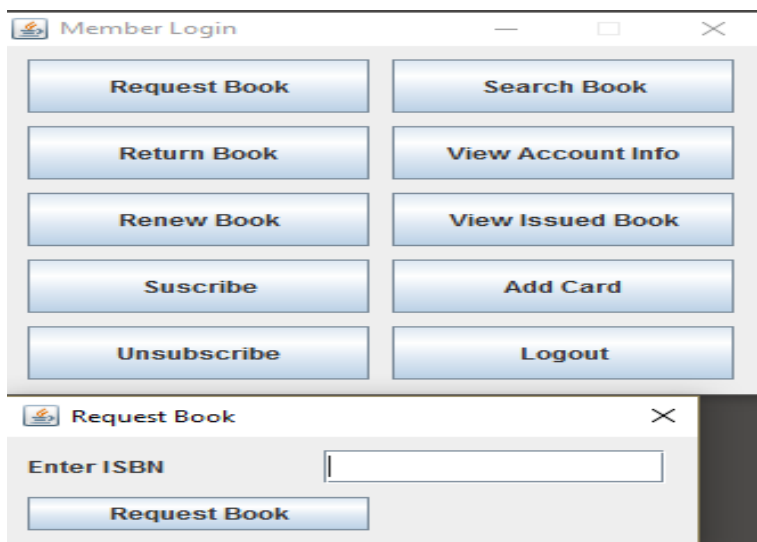


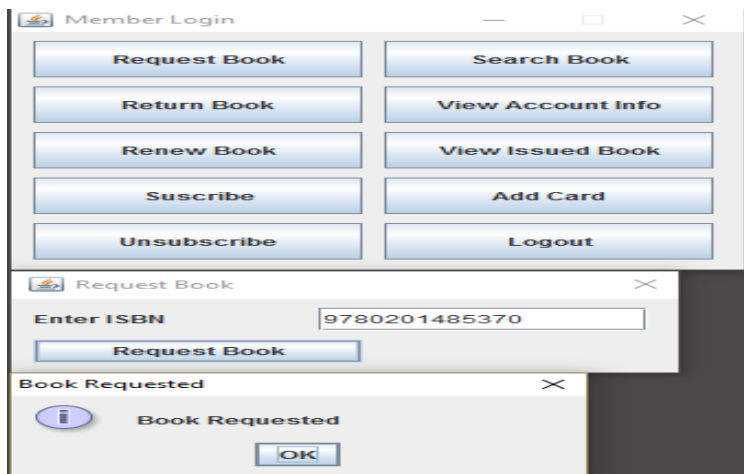
A dialog box titled "Member Login" with standard window controls (minimize, maximize, close) in the top right corner. The dialog contains a grid of ten blue buttons arranged in two columns and five rows. The buttons are labeled: "Request Book", "Search Book", "Return Book", "View Account Info", "Renew Book", "View Issued Book", "Suscribe", "Add Card", "Unsubscribe", and "Logout".

Member Login Overdue State: If the Member logs in and has dues to pay, he/she is directed to the overdue page.

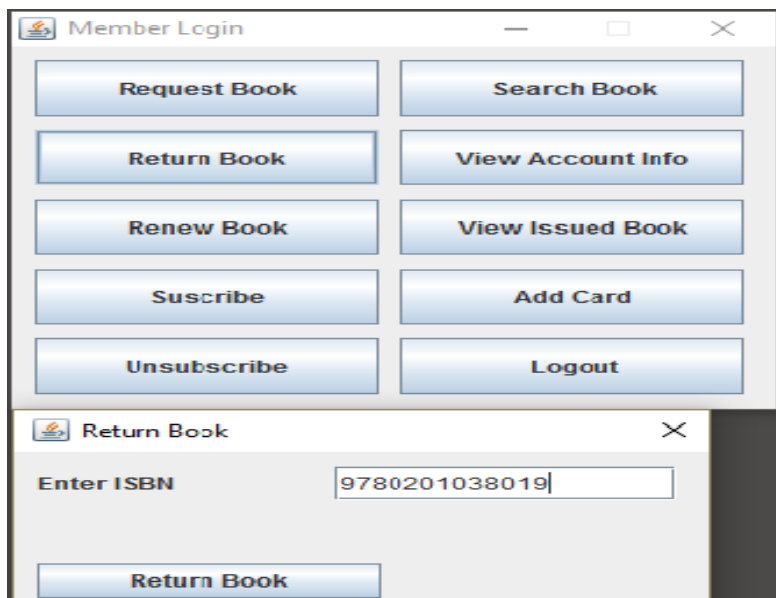


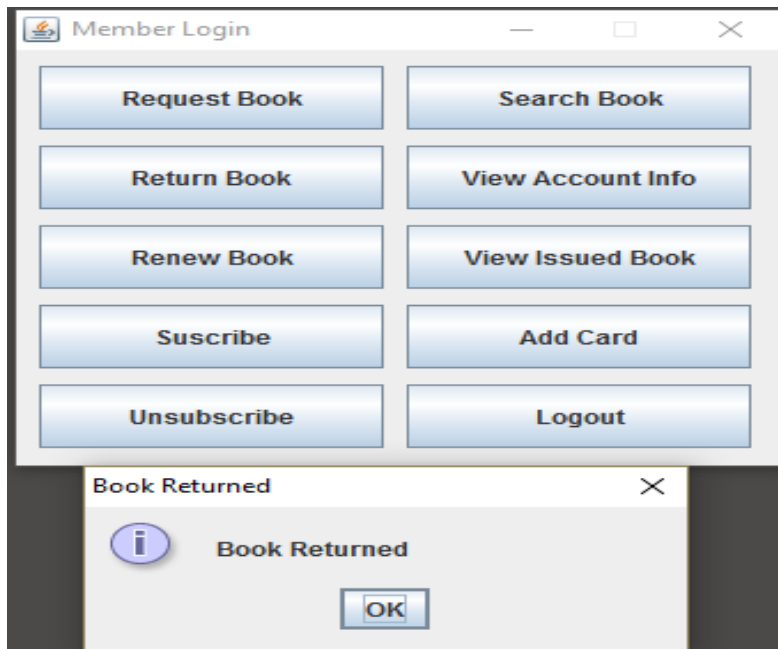
Request Book: The Member can request a book from the library on the basis of ISBN.



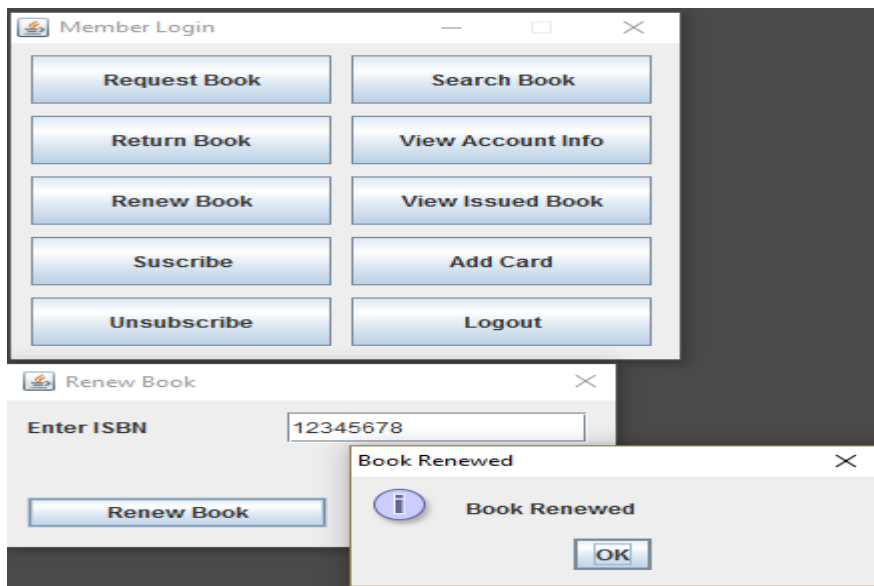


Return Book: The member can choose to return an issued book on the basis of ISBN.

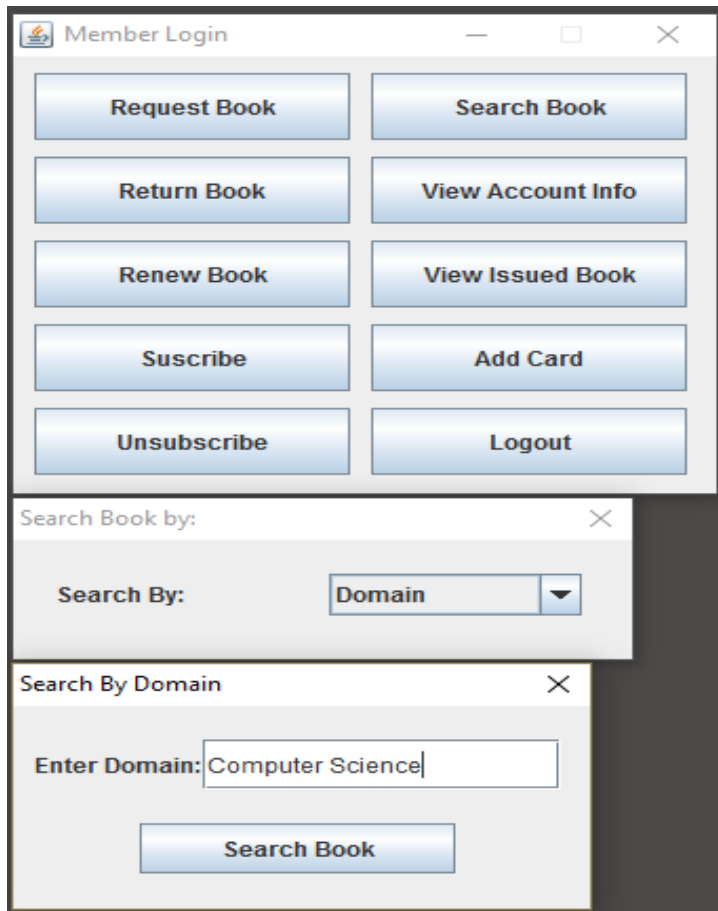




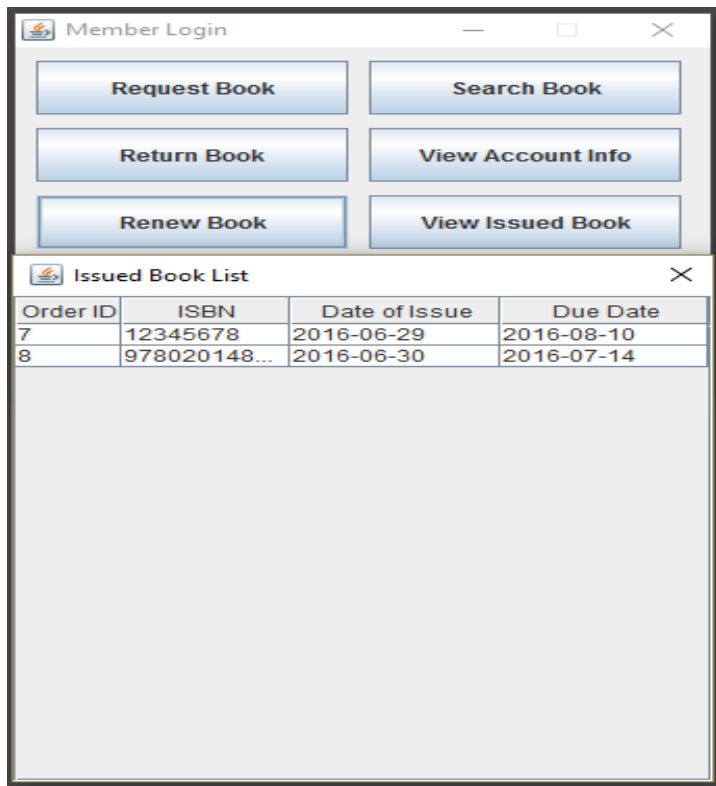
Renew Book: The member can renew an already issued book on or before its due date by 14 more days.



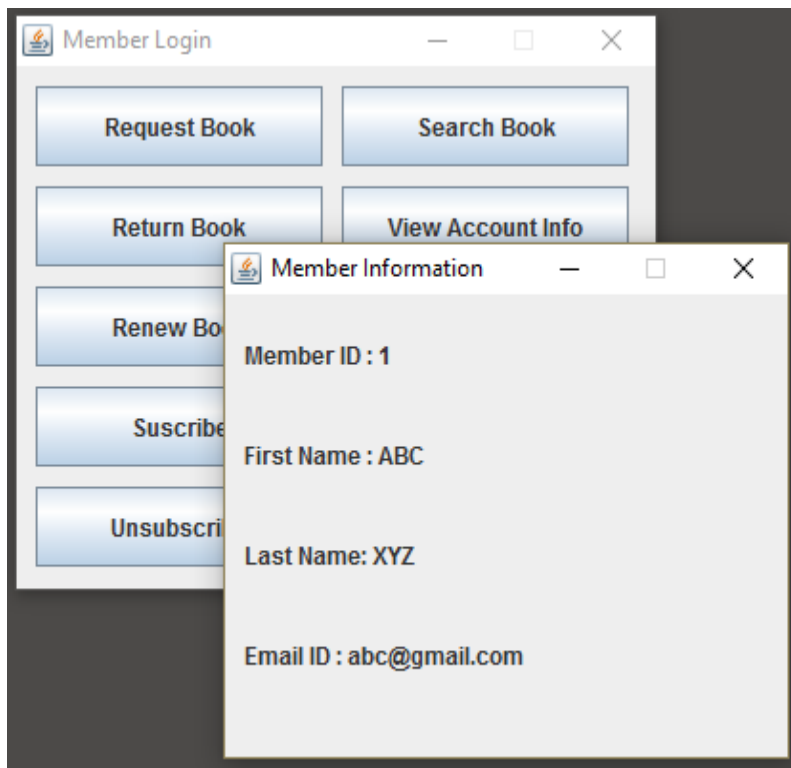
Search Book: The member can search book by author, name, domain or ISBN. The screenshot below searches by domain.



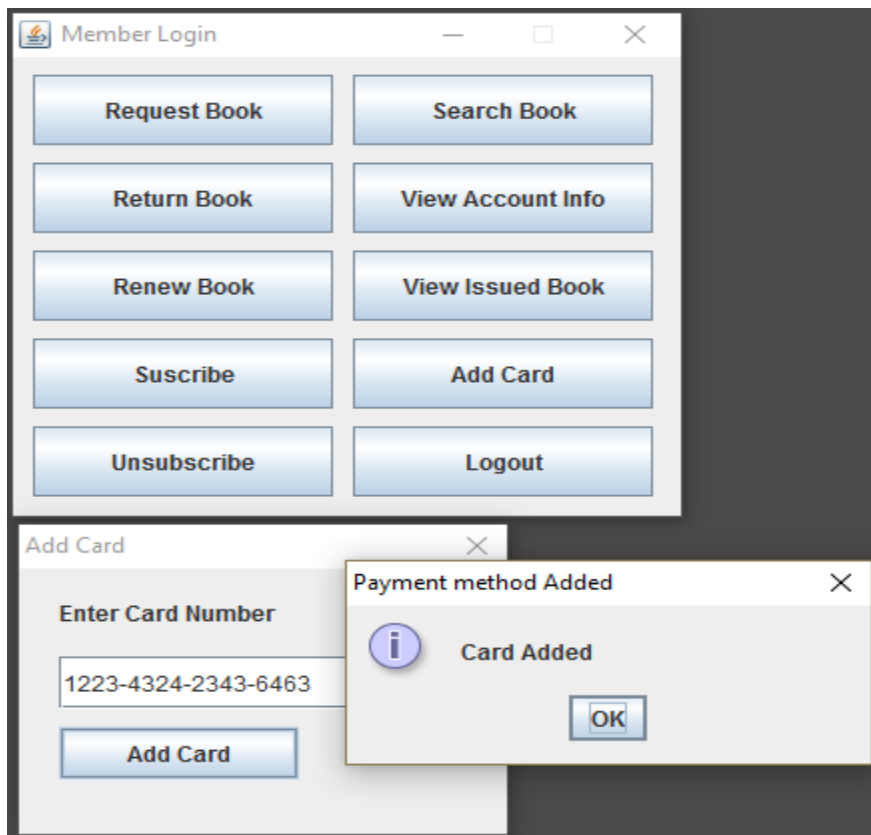
View Issued Book: The member can view all the books issued by him/her.



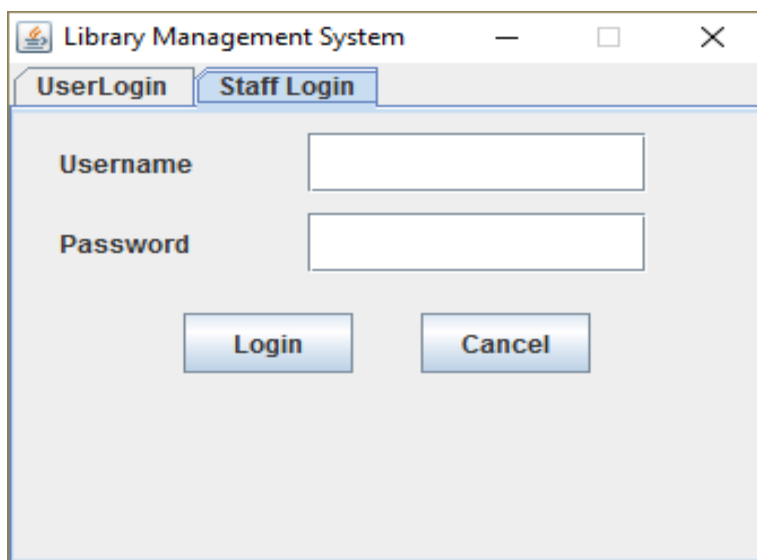
View Account Info: The member can view his/her account details.



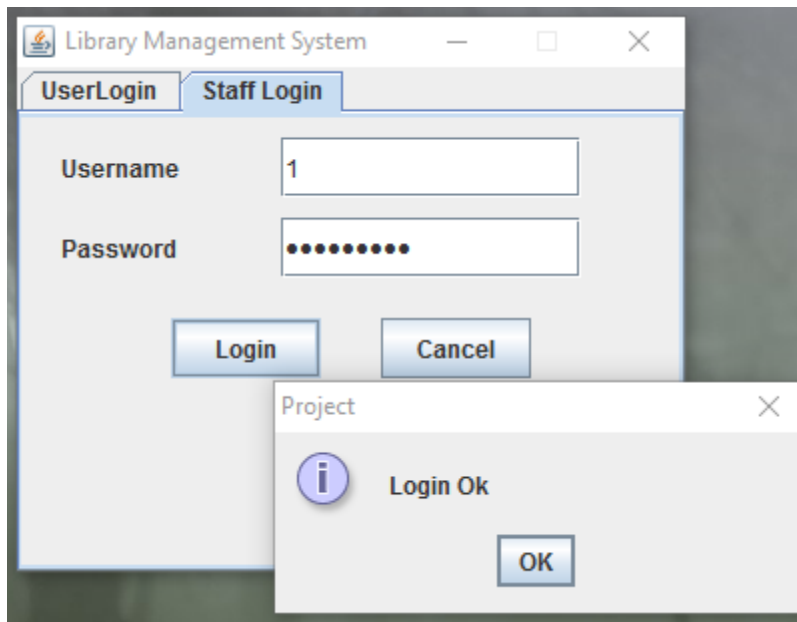
Add Card: The member can add their card details for payment purposes.



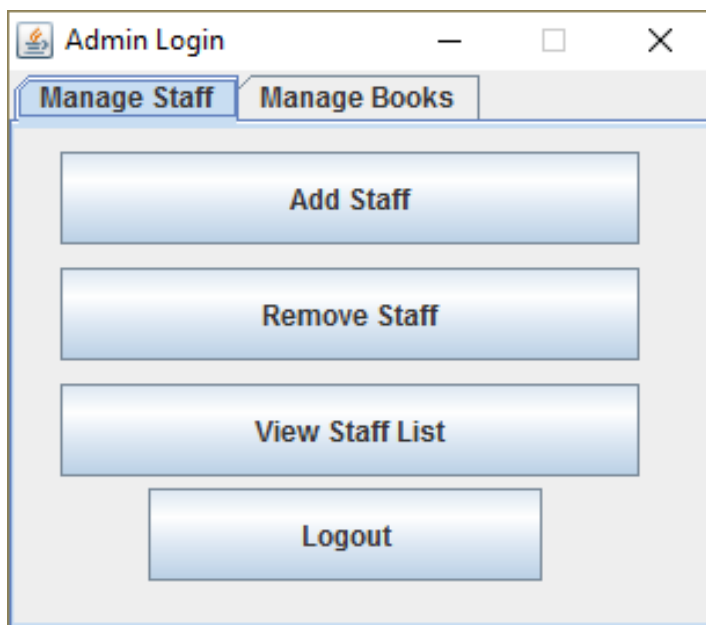
Staff Login: This page is for the Staff Login, i.e. either the Librarian or the Administrator.



Confirm Login: This page shows the confirmation on successful login by the staff.

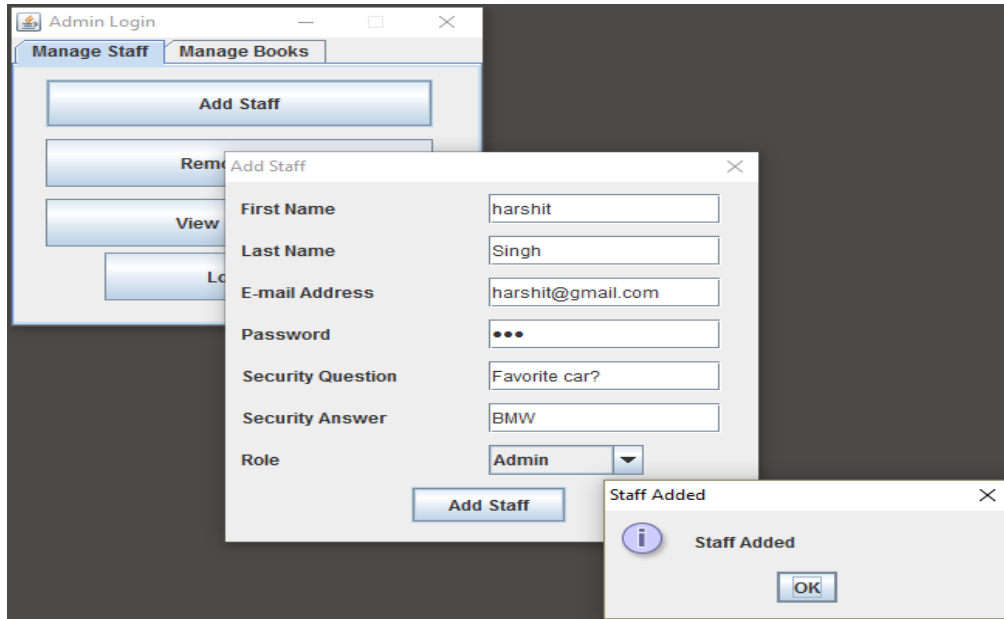


Admin Login View: When the Administrator logs in to the system, this page open and these are the options available to him/her.

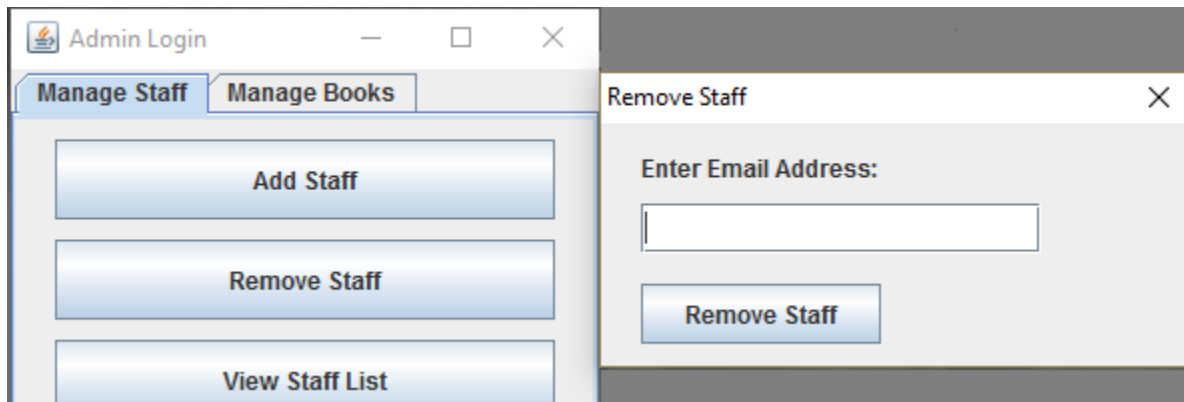


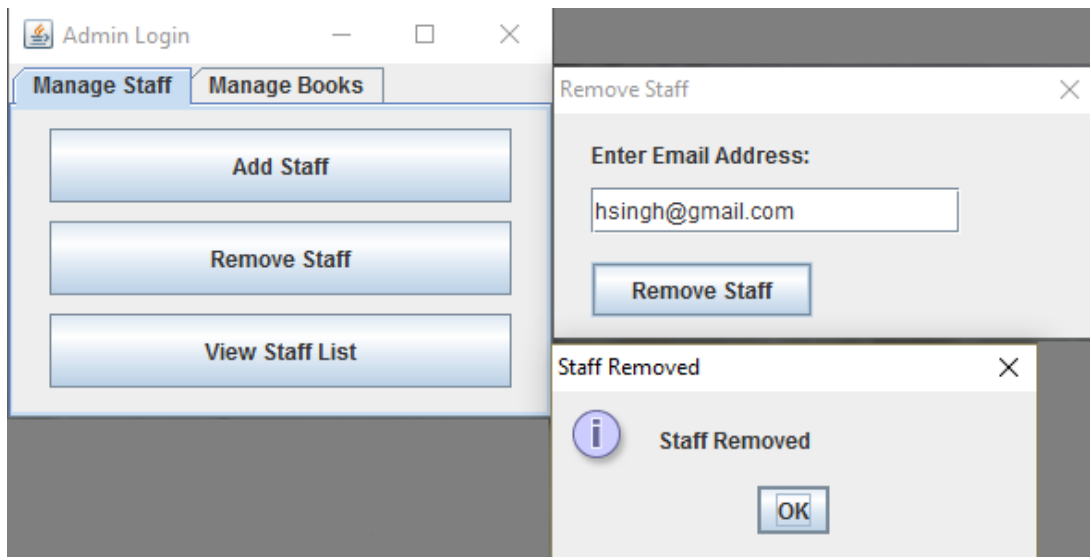
Manage Staff: The Manage Staff Tab has the following options.

Add Staff: Administrator can add new librarian or administrator.

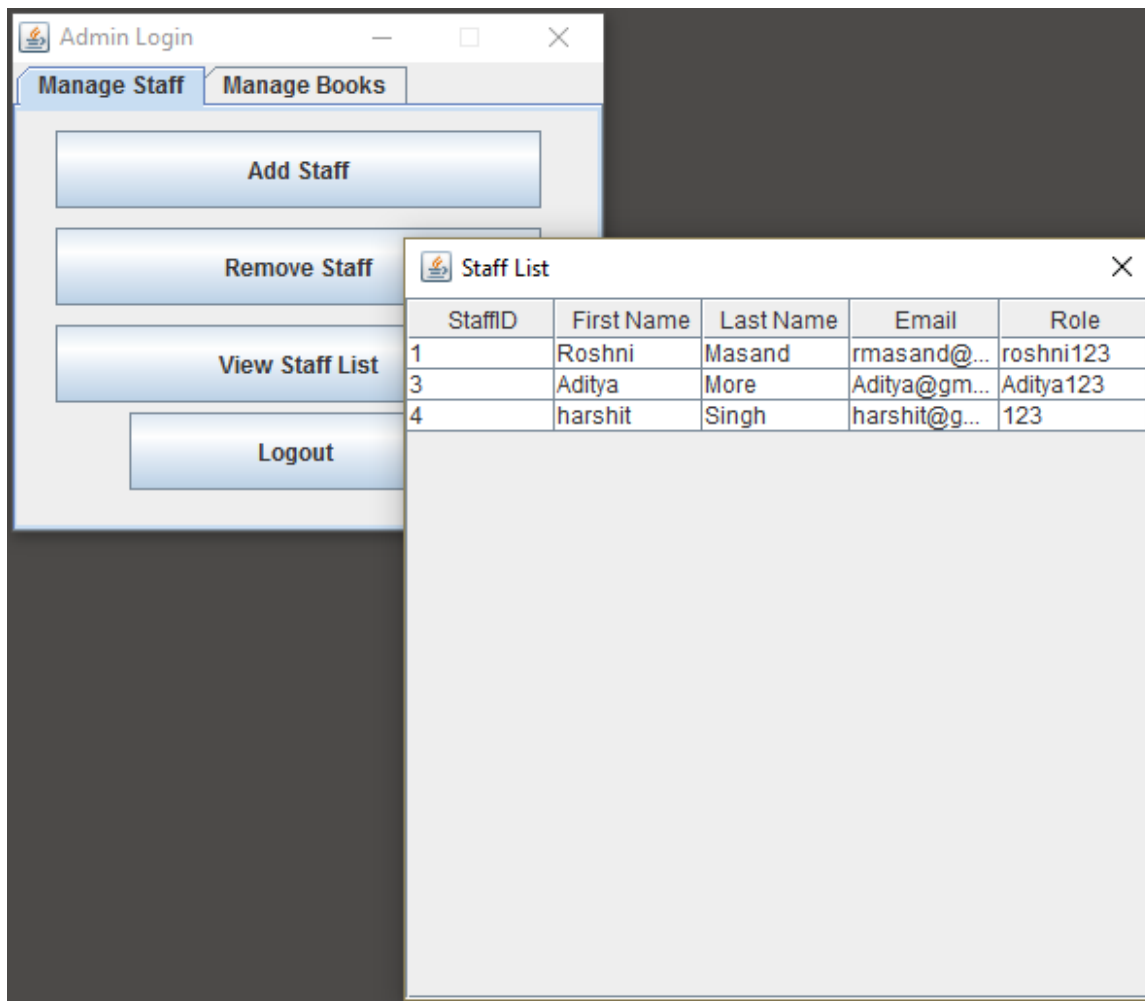


Remove Staff: The administrator can remove any staff from the Library Management System.

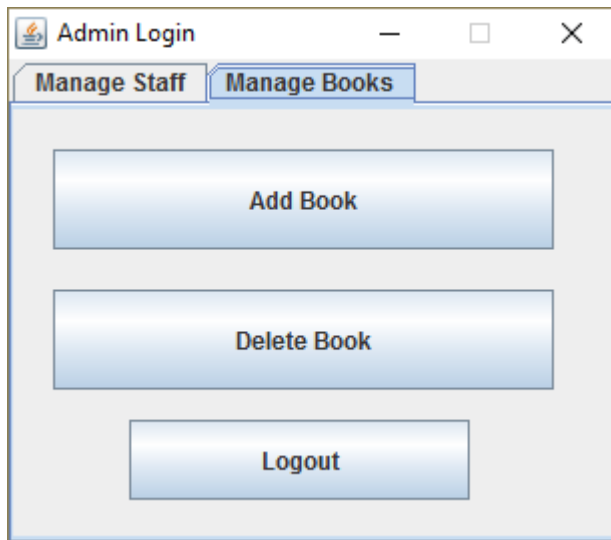




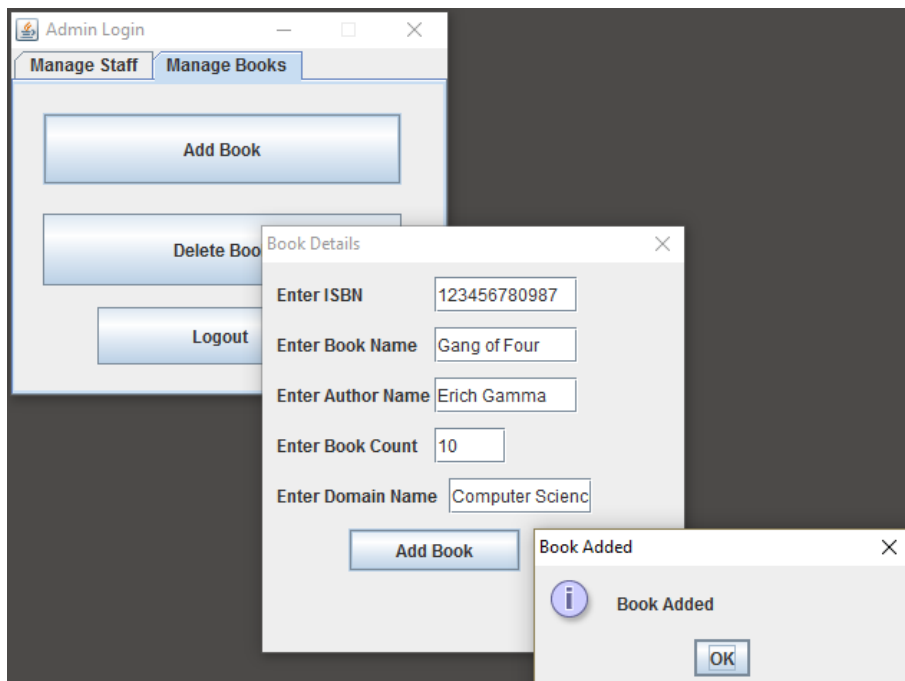
View Staff List: The administrator can view the list of staff members.



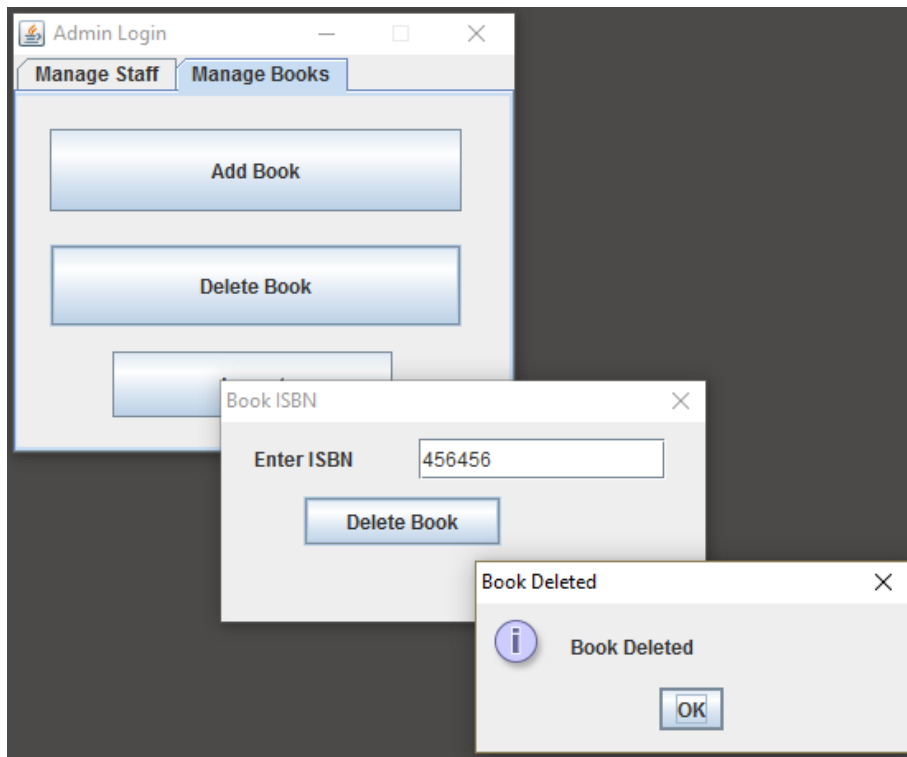
Admin Login Screen, Manage Books Tab:



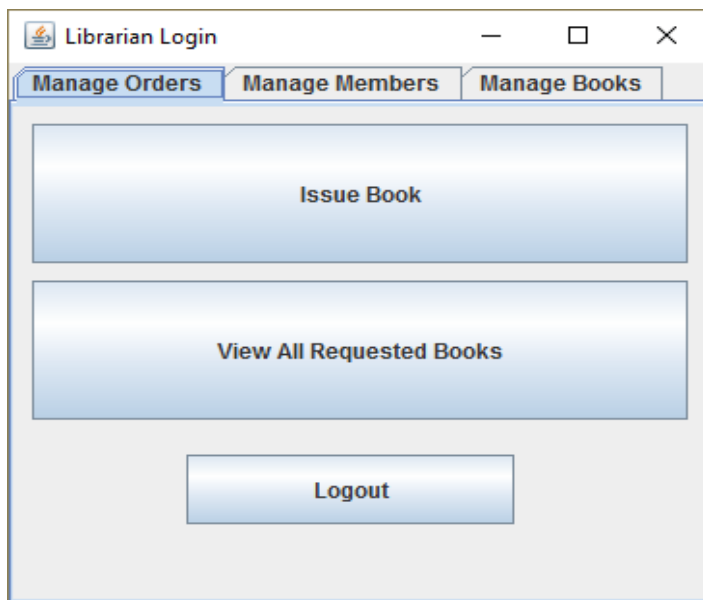
Add Book: The Administrator can add a new book to the system.



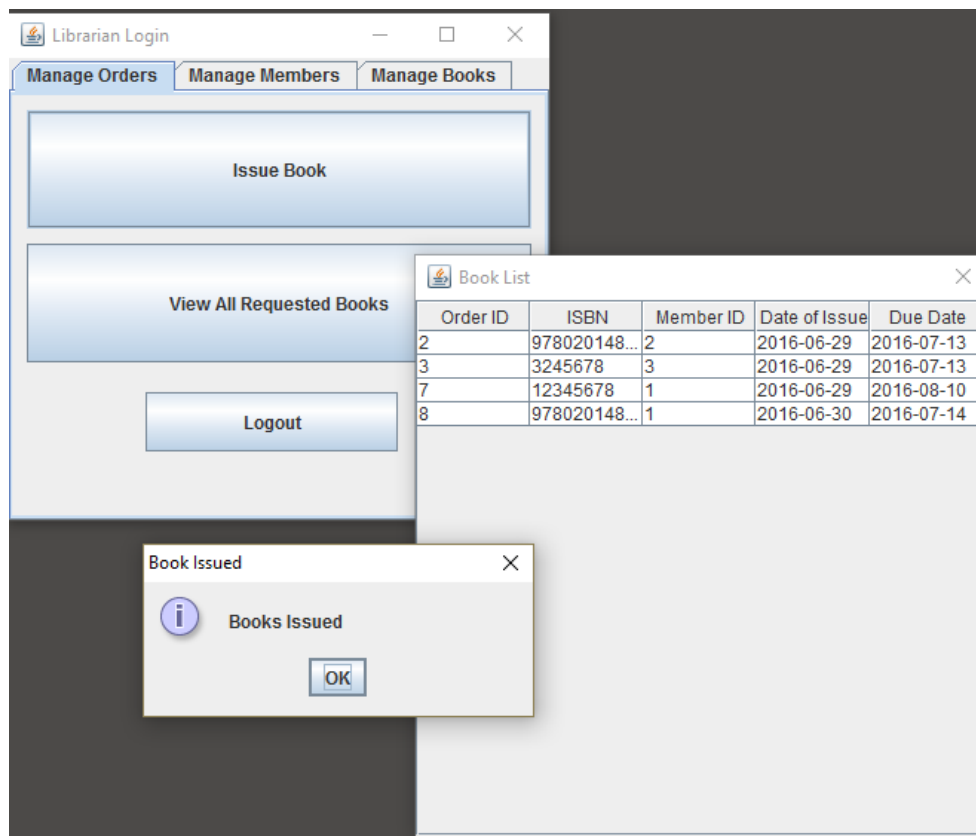
Delete Book: The administrator can delete a book from the system.



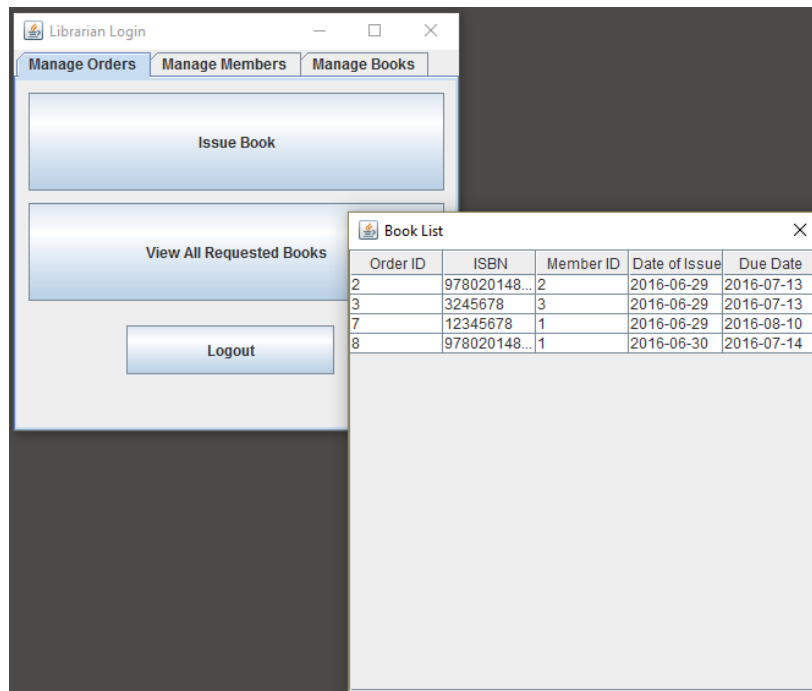
Librarian Login: On the login by Librarian, this frame opens. There are three tabs: Manage Orders, Manage Members and Manage Books



Issue Book: Librarian can issue all the books requested by the member.



View All Requested Book: The Librarian can view all the books requested by the member.



Add Member: The Librarian can add a new member to the system.

The screenshot shows a web application window titled "Librarian Login". It has three tabs: "Manage Orders", "Manage Members", and "Manage Books". The "Manage Members" tab is active. Below the tabs, there are three buttons: "Add Member", "Remove Member", and "View Member". The "Add Member" button is highlighted. An "Add user" dialog box is open over the "Add Member" button. The dialog box contains the following fields:

Field	Value
First Name	qwerty
Last Name	qwerty
E-mail Address	sdfg@gmail.com
Password	••••••
Security Question	why?
Security Answer	abc

At the bottom of the dialog box is an "Add user" button.

Remove Member: The Librarian can remove a particular member from the system.

The screenshot shows the same "Librarian Login" window. The "Remove Member" button is highlighted. Two dialog boxes are open. The first is a "Remove user" dialog box with the following fields:

Field	Value
Enter Email Address:	pqr@gmail.com

At the bottom of the "Remove user" dialog box is a "Remove user" button. The second dialog box is titled "Member Removed" and contains an information icon and the text "Member Removed". At the bottom of this dialog box is an "OK" button.

View Member List: The Librarian can view a list of members in the system.

The screenshot shows the 'Librarian Login' application with the 'Manage Members' tab selected. The 'View Member List' button is highlighted. A 'user List' window is open, displaying a table of members.

Member ID	First Name	Last Name	Email
1	ABC	XYZ	abc@gmail.com
2	PQR	LMN	pqr@gmail.com
3	ABC	XYZ	abc@gmail.com
4	PQR	LMN	pqr@gmail.com
5	ABC	XYZ	abc@gmail.com
6	PQR	LMN	pqr@gmail.com
7	Aditya	More	adimore@gma...
8	Tara	Ramchanchran	tara@gmail.com

Add Book: The librarian can add a new book to the system.

The screenshot shows the 'Librarian Login' application with the 'Manage Books' tab selected. The 'Add Book' button is highlighted. A 'Book Details' window is open, displaying input fields for book information.

Book Details

Enter ISBN

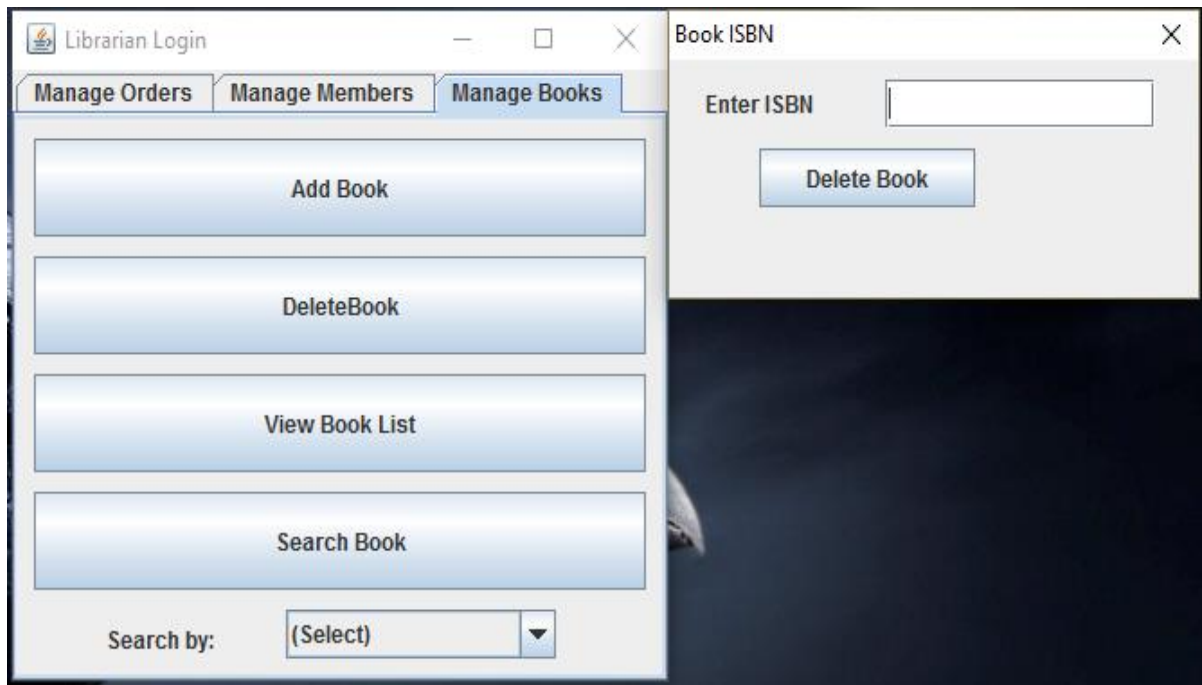
Enter Book Name

Enter Author Name

Enter Book Count

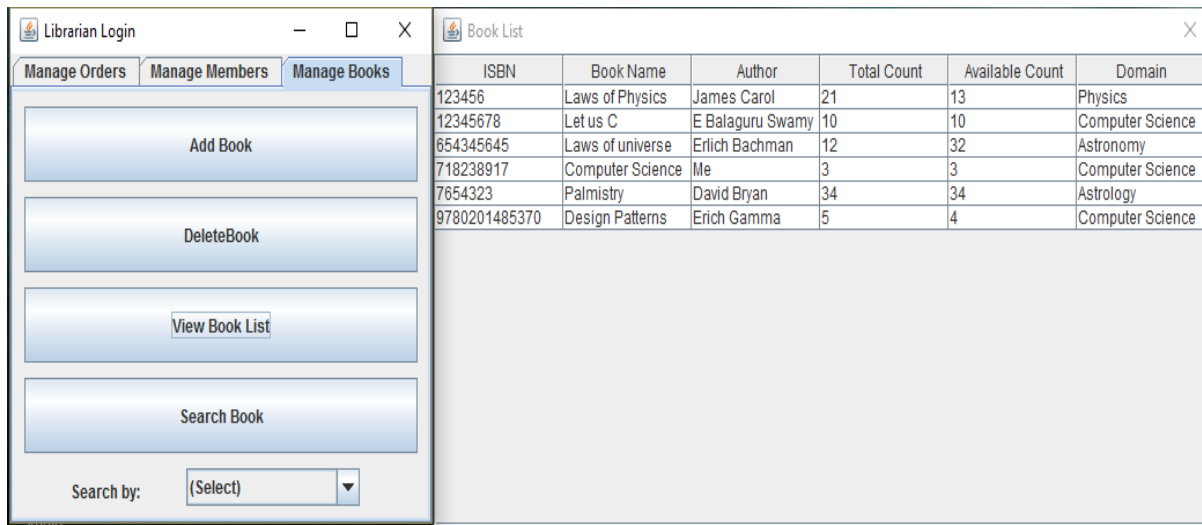
Enter Domain Name

Delete Book: The librarian can delete a book from the system based on ISBN.



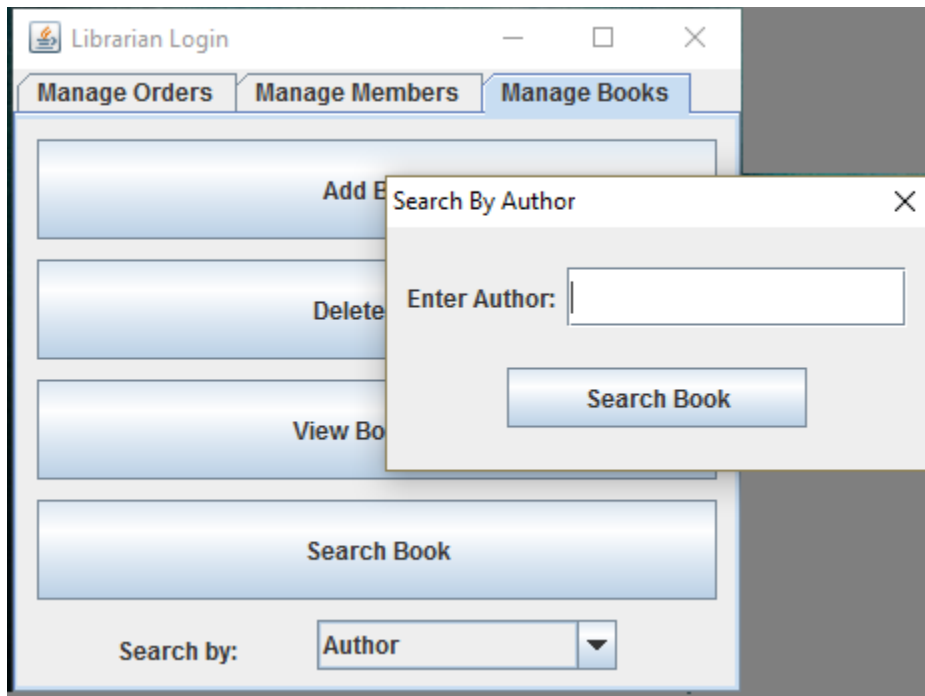
View Book List

View Book List: The librarian can view the list of books present in the system.



Search: The librarian can search book by Name, Author, ISBN or Domain. Below, search by author and ISBN is shown.

Search by Author:



Search by ISBN

