OOP Mini-Project

DSE 2123

Student Name	KP Sai Praneeth
Registration Number	230968248
Assignment Number	IA-3
Subject Code	DSE2123
Subject	OOP with Java
Marks	10

Library Management System

Problem Statement

Create a GUI Library Management System

Requirements:

1. User Authentication:

- The system should start with a login window where users enter their credentials.
- If the login is successful, proceed to the library management window.

2. Book Inventory Management:

- The application should allow users to add, edit, or delete books from the inventory.
- Each book should have details such as Title, Author, ISBN, and AvailabilityStatus (Available/Not Available).

3. Search and Borrowing:

- Include a search feature that allows users to find books by title or author.
- Allow users to borrow a book, which updates the availability status.

Methodology

1. **User Authentication**

The storage of information dealt with by creating a class called `User` which stores the credentials of the user using a `Map` data structure. The authentication part is done using the `authenticate` method in the `LibraryManagement` class. This method fetched the `User` object with the username entered and compares the password with the password entered.

The login page also initializes 2 users with the following credentials:

- `user1`, `password1`
- `user2`, `password2`

There is a register button provided for registration of new users. However, the information created in every session is not stored on the secondary storage.

The login panel takes username (`String`) and password (`String`) as an input and calls the `authenticate` method. This methods returns a boolean value. If the authentication is successful, the library panel is opened, or else a message dialog is showed saying that the credentials are invalid.

2. **Book Inventory Management**

The inventory panel is initialized using the `createInventoryPanel` method in the `LibraryManagement` class. The panel consists of input fields for Title, Author, ISBN, and Status. The status is a combo box with 2 options: Available and Not Available. The Add Book button adds the book to the inventory list. The ISBN is validated using the ISBN-13 algorithm. If the ISBN is invalid, a `NumberFormatException` is thrown and caught in the inventory method.

The inventory panel also consists of a `JList` to display the list of books in the inventory. The `DefaultListModel` is used to store the list of books. The `addButton` action listener adds the book to the list and clears the input fields.

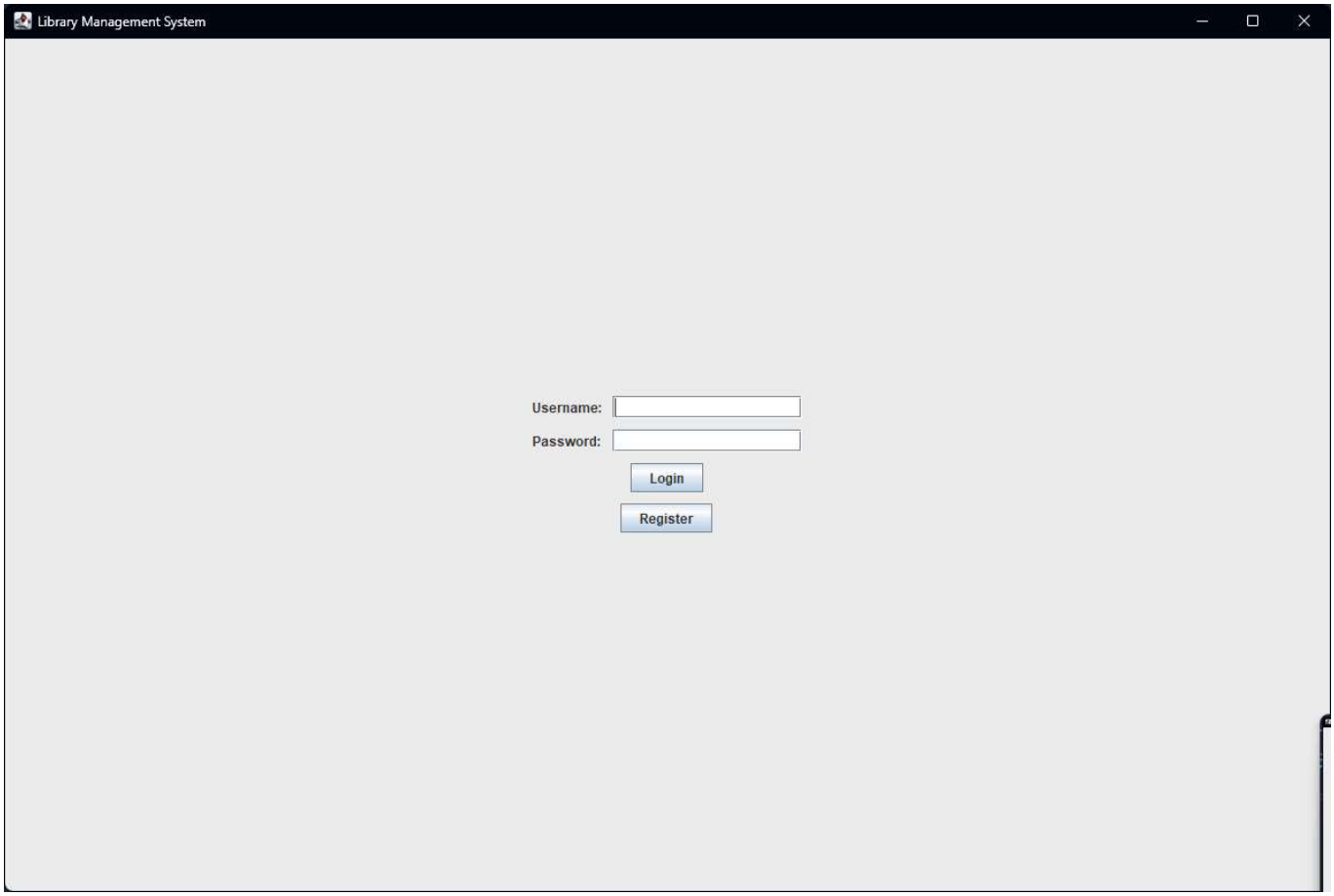
When a user borrows a book, the status of the book is changed to Not Available and the book is marked as borrowed by the user. The user can return the book by clicking the Return Selected Book button. The status of the book is changed to Available and the book is marked as not borrowed.

3. **Search and Borrowing**

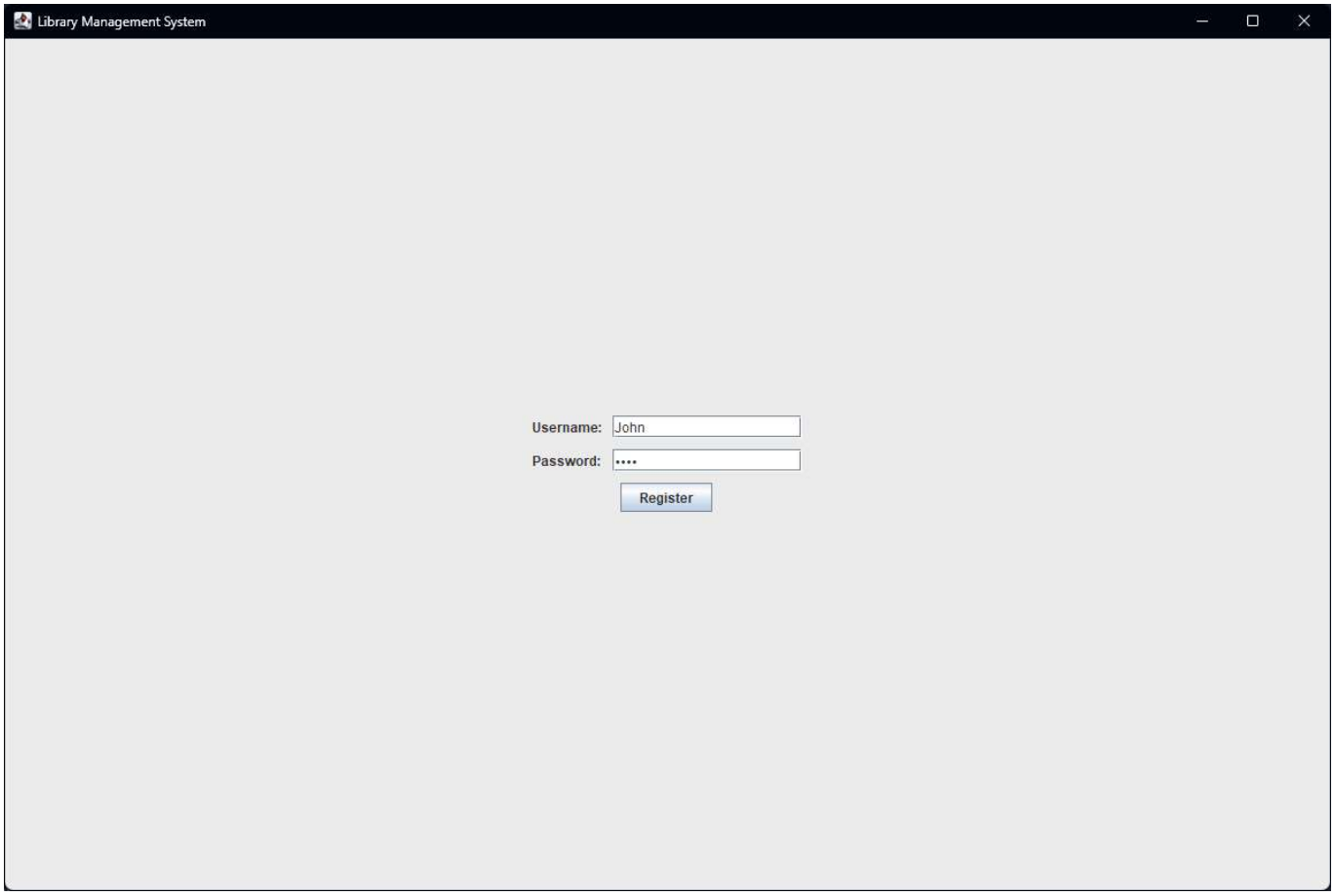
The search panel is initialized using the `createSearchPanel` method in the `LibraryManagement` class. The panel consists of a search field and a search button. The search button searches for books by title, author, or ISBN-13 number. The search results are displayed in a `JList`. The user can borrow a book by selecting the book and clicking the Borrow Selected Book button. The user can return a book by selecting the book and clicking the Return Selected Book button. Only the user who borrowed the book can return the book, if any other user tries to return the book, a message dialog is displayed saying that the user cannot return the book.

Results (w/ screenshots)

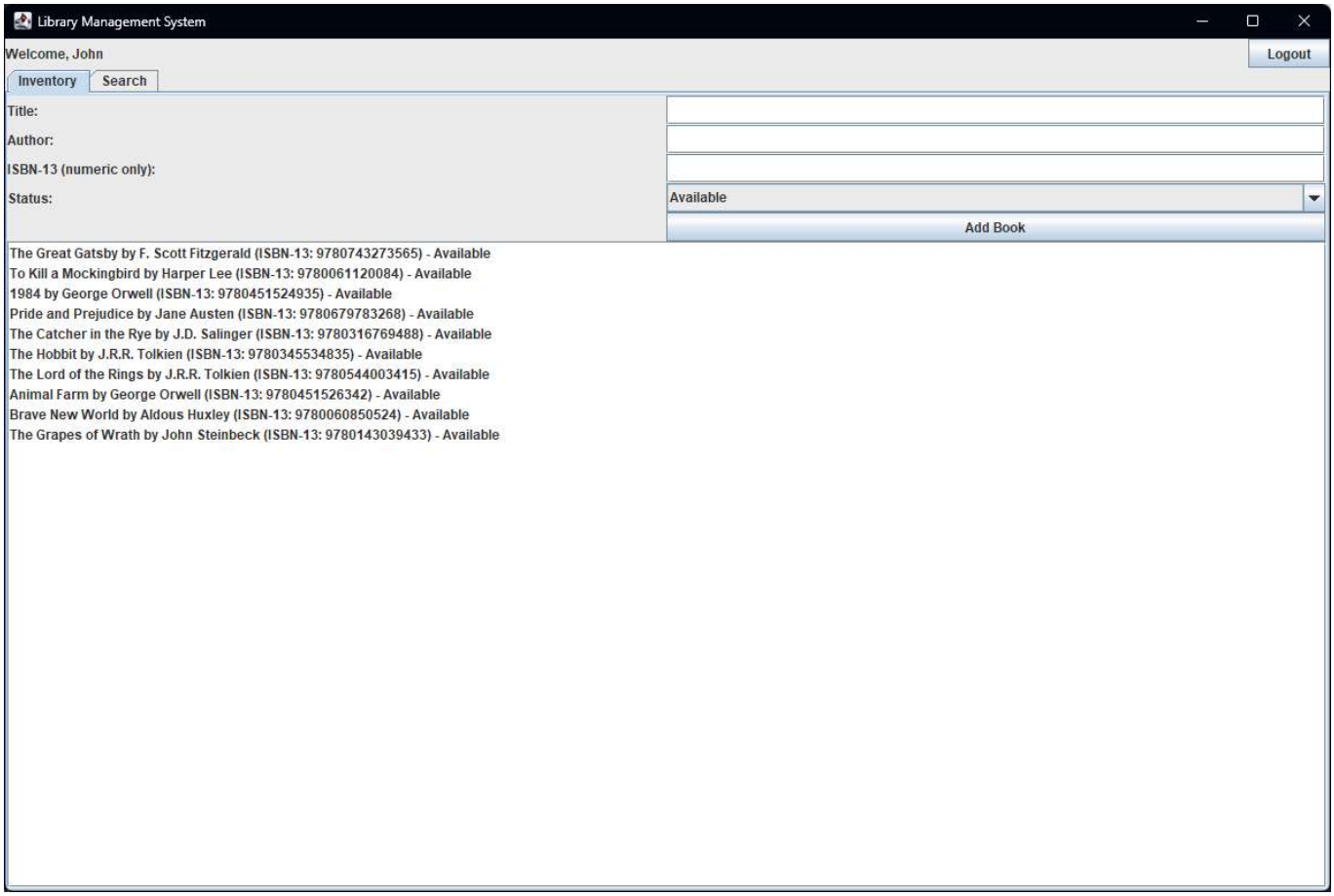
1. **Login Panel**



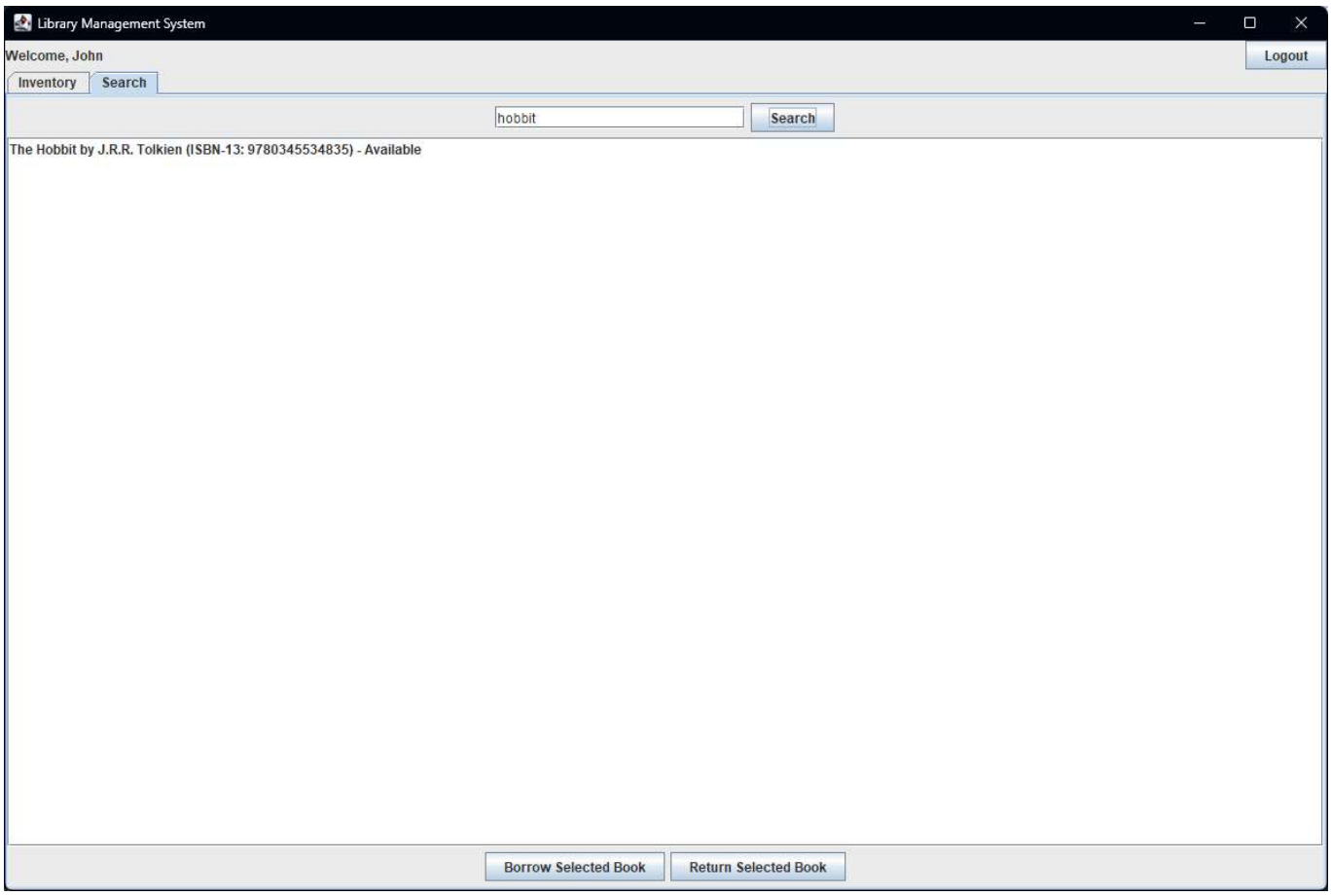
2. **Registration Panel**



3. **Library Panel with Inventory Panel**

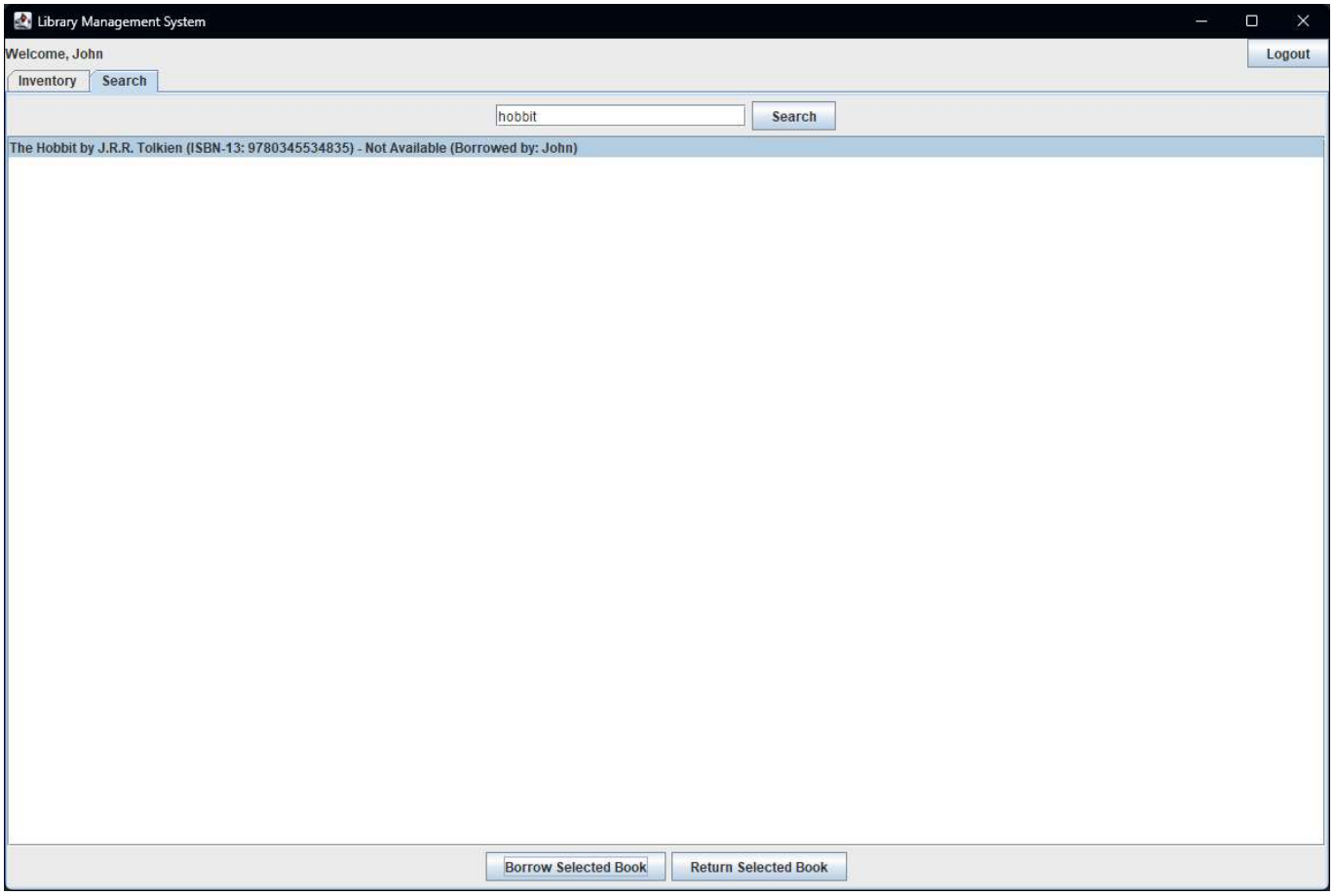


4. **Search Panel**

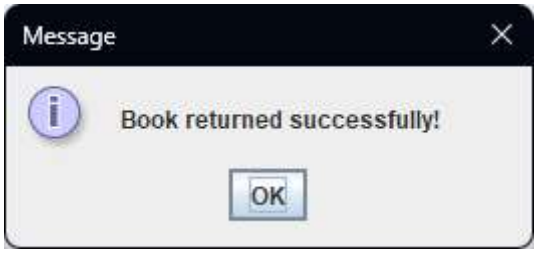


5. **Book Borrowing**

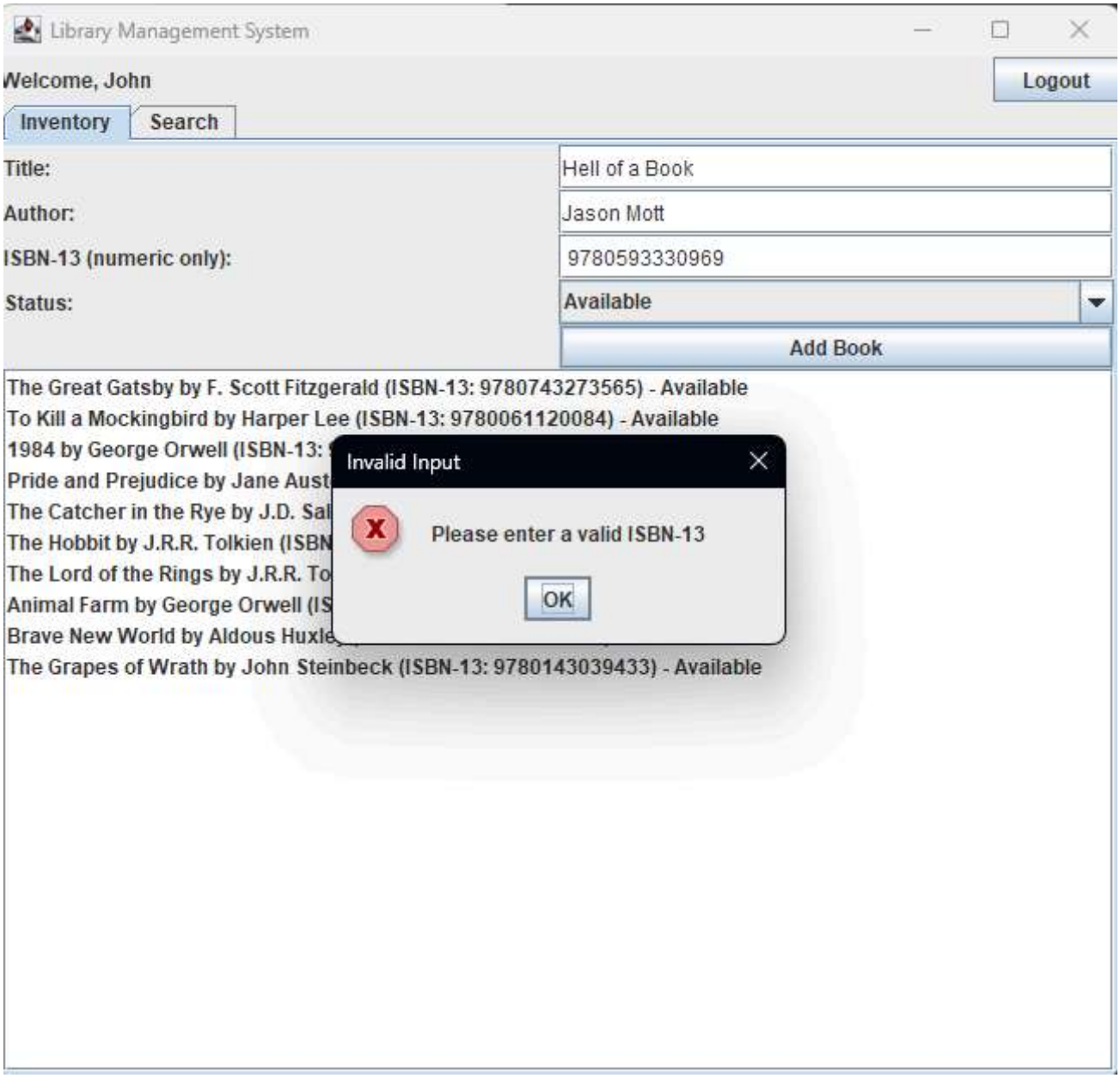
6. **Search Panel after borrowing**



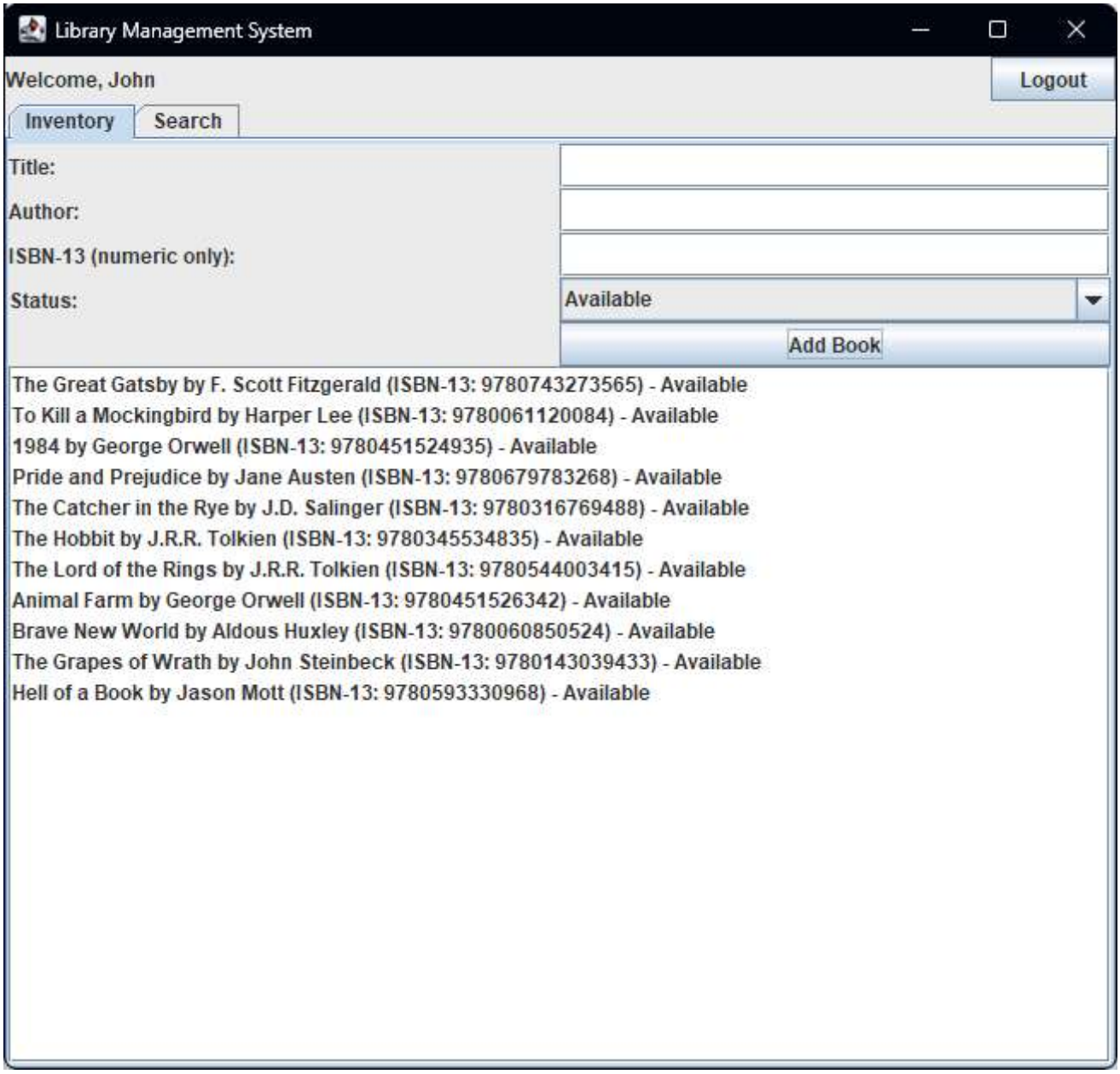
7. **Book Returning**



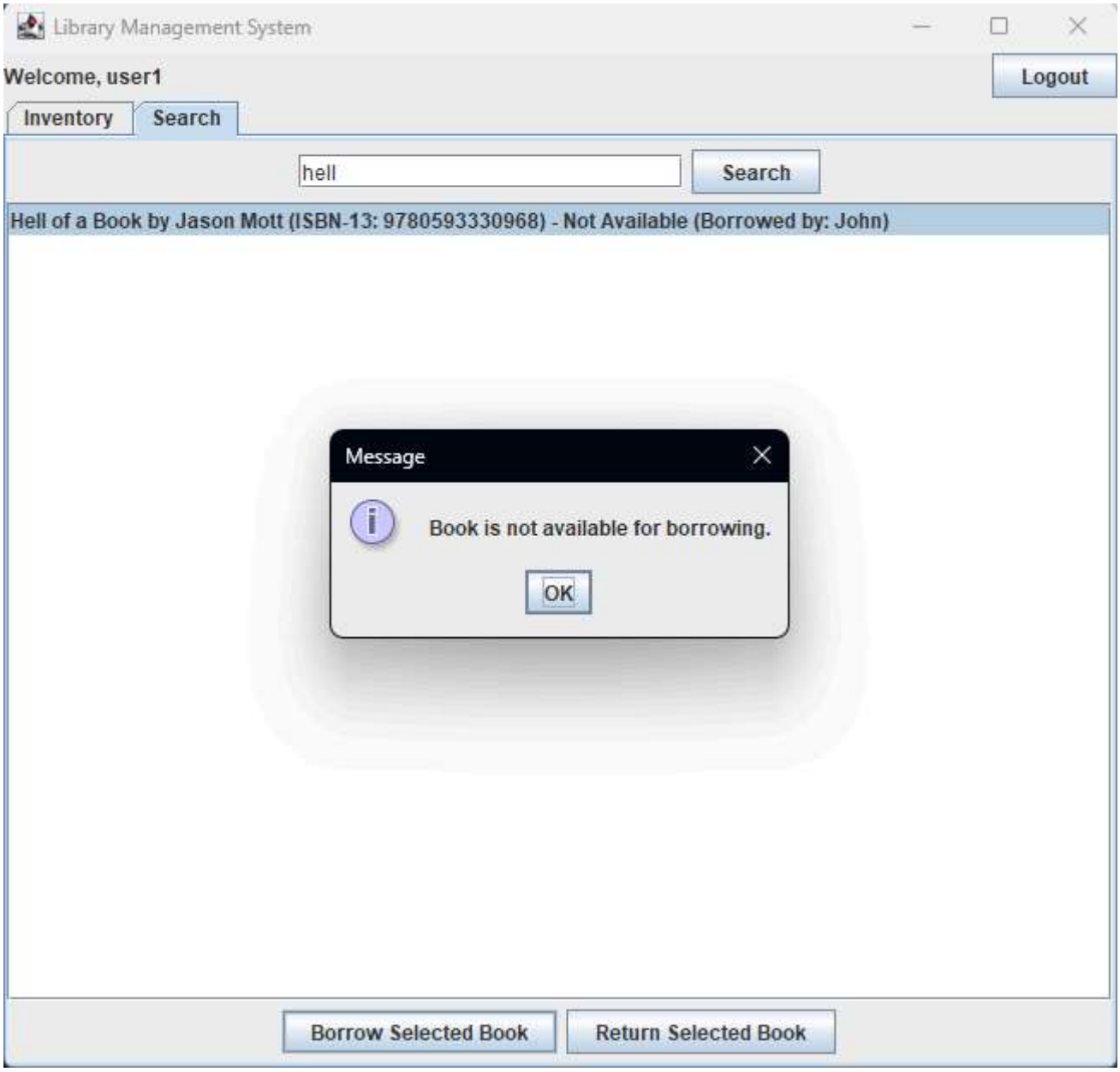
8. **Adding a Book with wrong ISBN number**



9. **Adding a Book with correct ISBN number**



10.; **Borrowing a book that is not available**



The Code

LibraryManagement.java

```
package src;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.lang.NumberFormatException;

// Main class for the Library Management System
class LibraryManagement {
    private JFrame mainFrame; // Main application window
    private JPanel currentPanel; // Current panel being displayed
    private List<Book> books; // List of books in the library
    private Map<String, User> users; // Map of users with their credentials
```

```
    private User currentUser; // Currently logged-in user

    private DefaultListModel<Book> listModel = new DefaultListModel<>(); // List
model for the book list
    private JList<Book> bookList = new JList<>(listModel); // JList to display
the book list

    // Constructor to initialize the library management system
    public LibraryManagement() {
        books = new ArrayList<>();
        users = new HashMap<>();
        initializeUsers(); // Initialize default users
        initializeGUI(); // Initialize the graphical user interface
    }

    // Method to initialize default users
    public void initializeUsers() {
        users.put("user1", new User("user1", "password1"));
        users.put("user2", new User("user2", "password2"));
    }

    // Method to initialize the graphical user interface
    public void initializeGUI() {
        mainFrame = new JFrame("Library Management System");
        mainFrame.setSize(1200, 800);
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        showLoginPanel(); // Show the login panel initially
        mainFrame.setVisible(true);
    }

    // Method to display the login panel
    public void showLoginPanel() {
        JPanel loginPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);

        JLabel userLabel = new JLabel("Username:");
        JLabel passLabel = new JLabel("Password:");
        JTextField userField = new JTextField(15);
        JPasswordField passField = new JPasswordField(15);
        JButton loginButton = new JButton("Login");
        JButton registerButton = new JButton("Register");

        // Action listener for the login button
        loginButton.addActionListener(e -> {
            String username = userField.getText();
            String password = new String(passField.getPassword());
            if (authenticate(username, password)) { // Authenticate user
                currentUser = users.get(username);
                showLibraryPanel(); // Show library panel on successful login
            } else {
                JOptionPane.showMessageDialog(mainFrame, "Invalid credentials",
"Login Failed", JOptionPane.ERROR_MESSAGE);
            }
        });
```

```
        // Action listener for the register button
        registerButton.addActionListener(e -> showRegistrationPanel());

        // Adding components to the login panel
        gbc.gridx = 0;
        gbc.gridy = 0;
        loginPanel.add(userLabel, gbc);

        gbc.gridx = 1;
        loginPanel.add(userField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 1;
        loginPanel.add(passLabel, gbc);

        gbc.gridx = 1;
        loginPanel.add(passField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        loginPanel.add(loginButton, gbc);

        gbc.gridy = 3;
        loginPanel.add(registerButton, gbc);

        setCurrentPanel(loginPanel); // Set the current panel to login panel
    }

    // Method to display the registration panel
    public void showRegistrationPanel() {
        JPanel registerPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);

        JLabel userLabel = new JLabel("Username:");
        JLabel passLabel = new JLabel("Password:");
        JTextField userField = new JTextField(15);
        JPasswordField passField = new JPasswordField(15);
        JButton registerButton = new JButton("Register");

        // Action listener for the register button
        registerButton.addActionListener(e -> {
            String username = userField.getText();
            String password = new String(passField.getPassword());
            if (users.containsKey(username)) {
                JOptionPane.showMessageDialog(mainFrame, "Username already
exists", "Registration Failed", JOptionPane.ERROR_MESSAGE);
            } else {
                users.put(username, new User(username, password));
                JOptionPane.showMessageDialog(mainFrame, "Registration
successful", "Registration", JOptionPane.INFORMATION_MESSAGE);
                showLoginPanel(); // Show login panel after successful
registration
```

```
            }
        });

        // Adding components to the registration panel
        gbc.gridx = 0;
        gbc.gridy = 0;
        registerPanel.add(userLabel, gbc);

        gbc.gridx = 1;
        registerPanel.add(userField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 1;
        registerPanel.add(passLabel, gbc);

        gbc.gridx = 1;
        registerPanel.add(passField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        registerPanel.add(registerButton, gbc);

        setCurrentPanel(registerPanel); // Set the current panel to registration
panel
    }

    // Method to authenticate user credentials
    public boolean authenticate(String username, String password) {
        User user = users.get(username);
        return user != null && user.getPassword().equals(password);
    }

    // Method to display the library panel
    public void showLibraryPanel() {
        JPanel libraryPanel = new JPanel(new BorderLayout());

        JTabbedPane tabbedPane = new JTabbedPane();
        tabbedPane.addTab("Inventory", createInventoryPanel()); // Tab for
inventory management
        tabbedPane.addTab("Search", createSearchPanel()); // Tab for searching
books

        JButton logoutButton = new JButton("Logout");
        logoutButton.addActionListener(e -> {
            currentUser = null;
            showLoginPanel(); // Show login panel on logout
        });

        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.add(new JLabel("Welcome, " + currentUser.getUsername()),
BorderLayout.WEST);
        topPanel.add(logoutButton, BorderLayout.EAST);

        libraryPanel.add(topPanel, BorderLayout.NORTH);
```

```
        libraryPanel.add(tabbedPane, BorderLayout.CENTER);

        setCurrentPanel(libraryPanel); // Set the current panel to library panel
    }

    // Method to create the inventory panel
    public JPanel createInventoryPanel() {
        JPanel panel = new JPanel(new BorderLayout());

        JPanel inputPanel = new JPanel(new GridLayout(5, 2));
        JTextField titleField = new JTextField();
        JTextField authorField = new JTextField();
        JTextField isbnField = new JTextField();
        JComboBox<String> statusCombo = new JComboBox<>(new String[]{"Available",
"Not Available"});
        JButton addButton = new JButton("Add Book");

        // Adding components to the input panel
        inputPanel.add(new JLabel("Title:"));
        inputPanel.add(titleField);
        inputPanel.add(new JLabel("Author:"));
        inputPanel.add(authorField);
        inputPanel.add(new JLabel("ISBN-13 (numeric only):"));
        inputPanel.add(isbnField);
        inputPanel.add(new JLabel("Status:"));
        inputPanel.add(statusCombo);
        inputPanel.add(new JLabel());
        inputPanel.add(addButton);

        JScrollPane scrollPane = new JScrollPane(bookList);

        // Action listener for the add button
        addButton.addActionListener(e -> {
            try {
                long isbn = Long.parseLong(isbnField.getText());
                Book book = new Book(titleField.getText(), authorField.getText(),
isbn, statusCombo.getSelectedItem().toString());
                books.add(book);
                listModel.addElement(book);
                clearFields(titleField, authorField, isbnField); // Clear input
fields after adding book
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(mainFrame, "Please enter a valid
ISBN-13", "Invalid Input", JOptionPane.ERROR_MESSAGE);
            }
        });

        panel.add(inputPanel, BorderLayout.NORTH);
        panel.add(scrollPane, BorderLayout.CENTER);

        return panel;
    }

    // Method to create the search panel
    public JPanel createSearchPanel() {
```

```
        JPanel panel = new JPanel(new BorderLayout());

        JPanel searchPanel = new JPanel(new FlowLayout());
        JTextField searchField = new JTextField(20);
        JButton searchButton = new JButton("Search");
        searchPanel.add(searchField);
        searchPanel.add(searchButton);

        DefaultListModel<Book> listModel = new DefaultListModel<>();
        JList<Book> resultList = new JList<>(listModel);
        JScrollPane scrollPane = new JScrollPane(resultList);

        // Action listener for the search button
        searchButton.addActionListener(e -> {
            String query = searchField.getText().toLowerCase();
            listModel.clear();
            for (Book book : books) {
                if (book.getTitle().toLowerCase().contains(query) ||
                    book.getAuthor().toLowerCase().contains(query) ||
                    String.valueOf(book.getIsbn()).contains(query)) {
                    listModel.addElement(book);
                }
            }
        });

        JPanel buttonPanel = new JPanel(new FlowLayout());
        JButton borrowButton = new JButton("Borrow Selected Book");
        JButton returnButton = new JButton("Return Selected Book");
        buttonPanel.add(borrowButton);
        buttonPanel.add(returnButton);

        // Action listener for the borrow button
        borrowButton.addActionListener(e -> {
            Book selectedBook = resultList.getSelectedValue();
            if (selectedBook != null) {
                if ("Available".equals(selectedBook.getStatus())) {
                    selectedBook.setStatus("Not Available");
                    selectedBook.setBorrowedBy(currentUser.getUsername());
                    JOptionPane.showMessageDialog(mainFrame, "Book borrowed
successfully!");
                } else {
                    JOptionPane.showMessageDialog(mainFrame, "Book is not
available for borrowing.");
                }
                resultList.repaint();
            }
        });

        // Action listener for the return button
        returnButton.addActionListener(e -> {
            Book selectedBook = resultList.getSelectedValue();
            if (selectedBook != null) {
                if ("Not Available".equals(selectedBook.getStatus()) &&

currentUser.getUsername().equals(selectedBook.getBorrowedBy())) {
```

```
                    selectedBook.setStatus("Available");
                    selectedBook.setBorrowedBy(null);
                    JOptionPane.showMessageDialog(mainFrame, "Book returned
successfully!");
                } else {
                    JOptionPane.showMessageDialog(mainFrame, "You cannot return
this book.");
                }
                resultList.repaint();
            }
        });

        panel.add(searchPanel, BorderLayout.NORTH);
        panel.add(scrollPane, BorderLayout.CENTER);
        panel.add(buttonPanel, BorderLayout.SOUTH);

        return panel;
    }

    // Method to set the current panel being displayed
    public void setCurrentPanel(JPanel panel) {
        if (currentPanel != null) {
            mainFrame.remove(currentPanel);
        }
        currentPanel = panel;
        mainFrame.add(currentPanel);
        mainFrame.revalidate();
        mainFrame.repaint();
    }

    // Method to clear input fields
    public void clearFields(JTextField... fields) {
        for (JTextField field : fields) {
            field.setText("");
        }
    }

    // Method to add books to the library while initializing
    public void initBooks() {
        books.add(new Book("The Great Gatsby", "F. Scott Fitzgerald",
9780743273565L, "Available"));
        books.add(new Book("To Kill a Mockingbird", "Harper Lee", 9780061120084L,
"Available"));
        books.add(new Book("1984", "George Orwell", 9780451524935L,
"Available"));
        books.add(new Book("Pride and Prejudice", "Jane Austen", 9780679783268L,
"Available"));
        books.add(new Book("The Catcher in the Rye", "J.D. Salinger",
9780316769488L, "Available"));
        books.add(new Book("The Hobbit", "J.R.R. Tolkien", 9780345534835L,
"Available"));
        books.add(new Book("The Lord of the Rings", "J.R.R. Tolkien",
9780544003415L, "Available"));
        books.add(new Book("Animal Farm", "George Orwell", 9780451526342L,
"Available"));
```

```
        books.add(new Book("Brave New World", "Aldous Huxley", 9780060850524L,
"Available"));
        books.add(new Book("The Grapes of Wrath", "John Steinbeck",
9780143039433L, "Available"));
        listModel.addAll(books);
    }
}
```

Book.java

```
package src;

import java.lang.NumberFormatException;

public class Book {
    private String title;
    private String author;
    private long isbn;
    private String status;
    private String borrowedBy;

    public Book(String title, String author, long isbn, String status) throws
NumberFormatException {
        // Validate the ISBN number to the ISBN-13 algorithm
        if (validISBN(isbn)) {
            this.title = title;
            this.author = author;
            this.isbn = isbn;
            this.status = status;
            this.borrowedBy = null;
        } else {
            System.out.println("Invalid ISBN: " + isbn);
            throw new NumberFormatException(); // Throws NumberFormatException
which is caught in the inventory method
        }
    }

    public String getTitle() { return title; }
    public String getAuthor() { return author; }
    public long getIsbn() { return isbn; }
    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }
    public String getBorrowedBy() { return borrowedBy; }
    public void setBorrowedBy(String borrowedBy) { this.borrowedBy = borrowedBy;
}

    // Return a string with book details for display
    @Override
    public String toString() {
        return title + " by " + author + " (ISBN-13: " + isbn + ") - " + status +
               (borrowedBy != null ? " (Borrowed by: " + borrowedBy + ")" : "");
    }
```

```
    // Method to validate the ISBN-13 number
    private boolean validISBN(long isbn) {
        int last = (int) (isbn % 10);
        isbn /= 10;
        int sum = 0;
        for (int i = 1; i <= 12; i++) {
            int digit = (int) (isbn % 10);
            sum += (i % 2 == 0) ? digit : digit * 3;
            isbn /= 10;
        }
        int check = 10 - (sum % 10);
        if (check == 10) check = 0;
        return check == last;
    }
}
```

User.java

```
package src;

class User {
    private String username; // Username of the user, displayed on the top-right
corner
    private String password; // Password of the user

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() { return username; } // Return username for
display and/or authentication
    public String getPassword() { return password; } // Return password for
authentication
}
```

App.java

```
package src;

import javax.swing.SwingUtilities;

// Main class used for running the application
public class App {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new LibraryManagement().initBooks(); // Start the GUI while
initialising the library with some books
        });
```

```
        }
    }
```

References

1. ISBN-13 Algorithm
2. Java Swing Tutorial
3. Java Documentation
4. Stack Overflow (for debugging and problem-solving)
5. Coursera Swing course
6. Quick references for Swing on YouTube