CPP106: Library Management System GUI

Instructions:

Task: Design and construct a Java GUI application for managing library records. Each group member should contribute with skills in programming, design, and problem-solving. The project submission deadline is Sunday at 11:59 PM. Submit the completed project on time using any preferred Integrated Development Environment (IDE). Follow POLYA's problem-solving method as outlined below:

1. **Understand the Problem:**

   Begin by identifying the requirements of a Library Management System. Discuss essential features like adding books, managing member information, issuing and returning books, and tracking book availability. Consider additional functionalities like a search feature, overdue notifications, or report generation to improve the user experience.

2. **Plan a Solution:**

   Sketch the layout and functionality of your system, considering user interaction and data processing. Allocate tasks based on team members' strengths. Outline the logic using pseudocode and a flowchart.

3. **Execute the Plan:**

   Develop the system using Java Swing for the GUI. Divide work among team members and test each feature as it is developed. Include code comments to explain key components and logic, making it understandable for both your team and future developers.

4. **Assess the Solution:**

   Thoroughly test the Library Management System to ensure it meets requirements. Ensure the interface is user-friendly, and features function correctly. Document the system with a user manual and technical documentation for future maintenance and development.

A. Introduction:

A Library Management System (LMS) revolutionizes the way libraries function by automating and streamlining their processes. With the Group 8 Library Management System, librarians can efficiently catalog and organize books, track borrowed items, and oversee overdue returns. This system enhances user experience by providing quick access to a vast collection of resources and enabling seamless search capabilities. The significance of an LMS lies in its ability to save time, reduce errors, and improve the overall management of library operations. Its relevance is profound, as it caters to the evolving needs of modern libraries, ensuring they remain vibrant hubs of knowledge and learning.

B. System Overview:

Group 8 Library Management System (LMS) encompasses a range of core functionalities to effectively manage library operations. These include:

1. **Add Books:** The user can add books with a few clicks. The system will add the book after the user hits the add book button the user interface of the system.

2. **View Books:** The user will be able to view all books that are displayed in a table format making the desired book easy to find.

3. **Edit Book:** From the saved book in the table format, book information can be edited by selecting the row of the book that is needed to edit. After selecting, the user must click the edit button, then the new prompt will appear asking the user to input the new modified information for the selected book.

4. **Delete Book:** Similar to the Edit Book function, G8LMS also has a delete function. From the table, the user must select the row of the book to be deleted, then click the delete book button to redirect to a new prompt. The new prompt will ask for confirmation whether to delete the book or not. The user must click the yes button to confirm the book deletion.

5. **Issue Book:** As the library's purpose, which is to issue books, G8LMS can also issue books. The user must again select a row of the desired book in the Catalog Table to continue borrowing book. In the borrow book section, the system will only ask for the borrower's name that is

required for the recording of the system library. After filling in the name, the user must click the issue book button to borrow the book. The borrowed book record will be recorded by the system and will be displayed in the Book Borrowed Record.

6. **Return Book:** As the G8LMS has an issuing book function, it also has return book function. First the user must go to the Borrowed Book Record table to view the record of the borrowed books. After checking the borrowed book, the user must select the row of the book borrowed to be returned by the user to be able to continue. After selecting the row, the user must click the return book button to return the book successfully, which will update the Borrowed Book Record table.

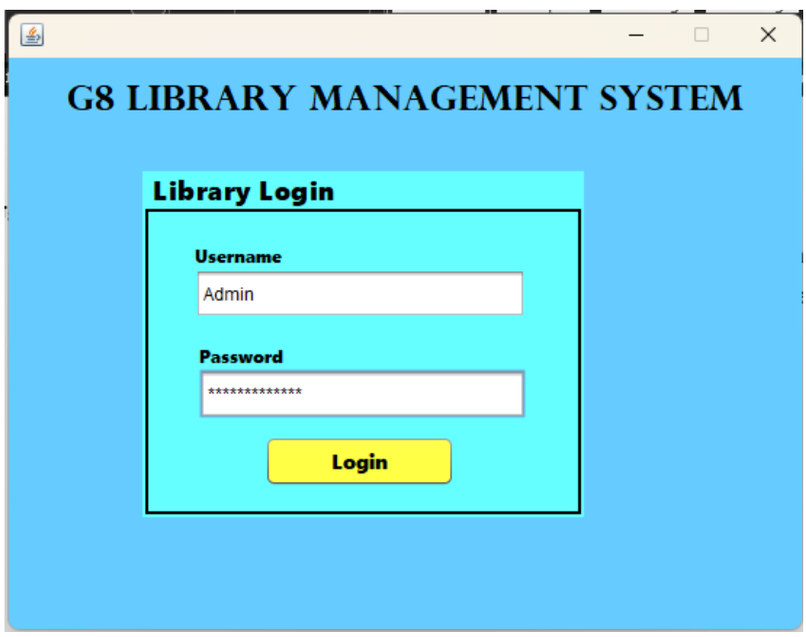
C. User Manual:

The G8 Library Management System is designed to simplify and manage library operations effectively. The main window is divided into three key sections: Issue Book, Borrowed Book Record, and Book Catalog. Each section offers specific functionalities to support library management tasks.

1. System Log In

Username: Admin

Password: LibrarySystem



2. Issue Book Section

This section enables issuing books to borrowers seamlessly.

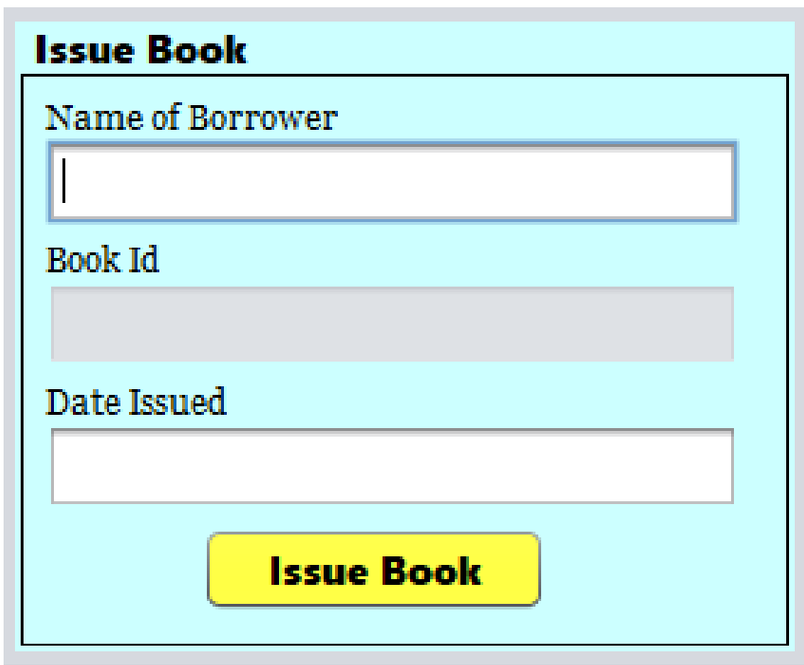
Features:

- Name of Borrower: Input the borrower's full name.

- Book ID: Enter the unique identifier for the book to be issued.

- Date Issued: Record the date when the book is issued.

- Issue Book Button: Click to finalize the issuing process. The book will be added to the Borrowed Book Record section.



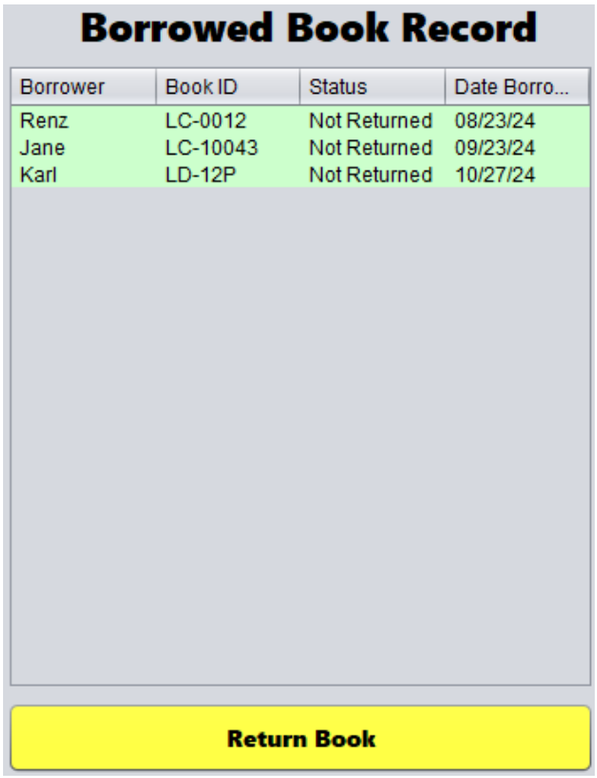
3. Borrowed Book Record Section

Displays details of books that have been issued but not yet returned.

Features:

- Information Displayed: Borrower's name, Book ID, status, and date borrowed.

- Actions Available: Return books (see "Return Book" functionality below).



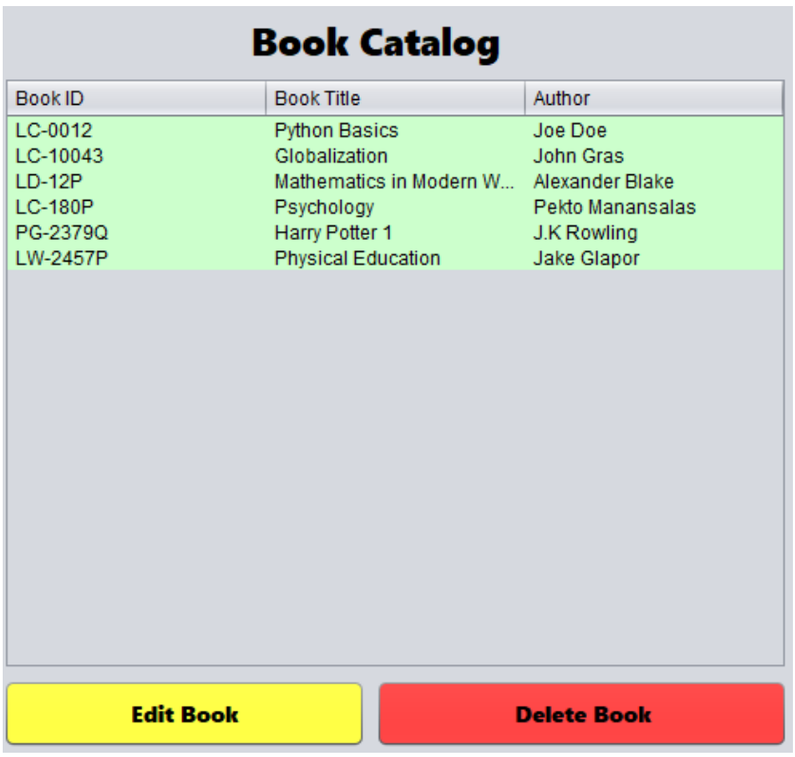
4. Book Catalog Section

A comprehensive list of all books in the library.

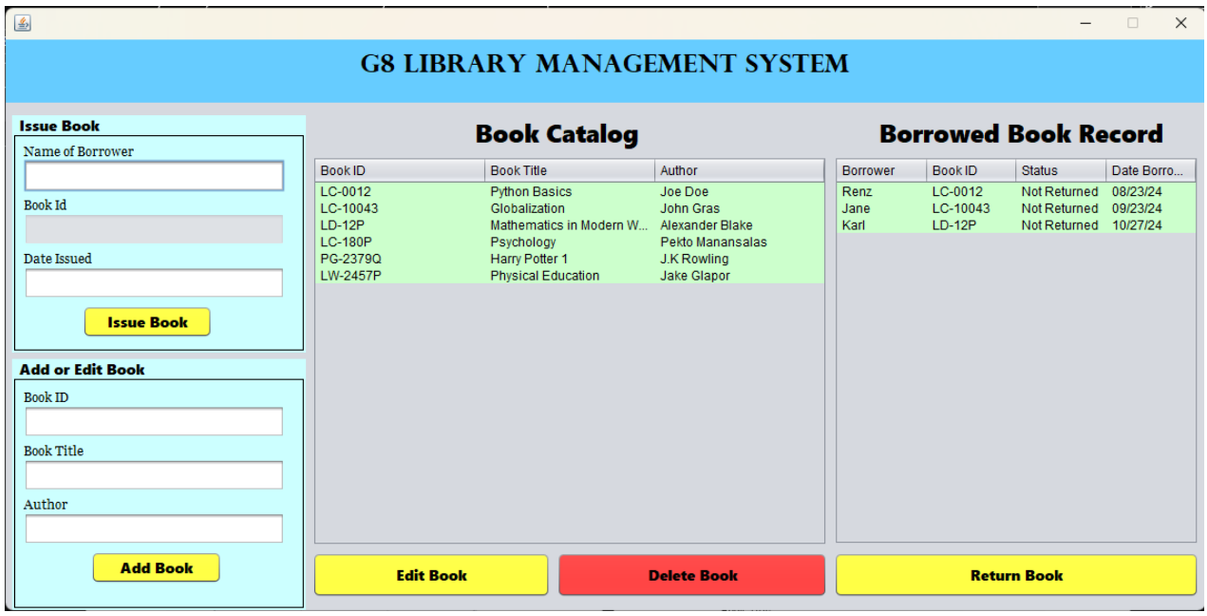
Features:

- Details Displayed: Book ID, Title, and Author.

- Actions Available: Add, Edit, or Delete books in the catalog.



Main Dashboard Functionalities



For a comprehensive and active demonstration of the system, you may click the link below for a simple video demonstration of G8LMS.

https://drive.google.com/file/d/1MTk8XDVmaVnCJURLQR8-_Wrvb4ta1ttS/view?usp=sharing

1. Adding a Book

Add new books to the library's catalog.

Steps:

1. Click the Add or Edit Book button.
2. Enter the following information:
   - Book ID: Unique identifier for the book.
   - Book Title: Title of the book.
   - Author: Author's name.
3. Click the Add Book button to save the new book in the catalog.

2. Issuing a Book

Issue a book to a borrower.

Steps:

1. Navigate to the Issue Book section.

2. Enter the following details:

- Name of Borrower
- Book ID
- Book Title

3. Click the Issue Book button to issue the book.

3. Editing a Book

Update book information in the catalog.

Steps:

1. Locate the desired book in the Book Catalog section.
2. Click the Edit Book button.
3. Modify the book's details as needed.
4. Click the Edit Book button again to save changes.

4. Deleting a Book

Remove books from the library catalog.

Steps:

1. Find the book(s) in the Book Catalog section.
2. Click the Delete Books button.
3. Confirm the action in the dialog box by selecting Yes.

5. Returning a Book

Mark a book as returned and update records.

Steps:

1. Locate the book in the Borrowed Book Record section.
2. Click the Return Book button.
3. The book will be removed from the Borrowed Book Record section and marked as available in the Book Catalog.

D. Technical Documentation:

START

Create Log In Window Frame

Add JPanel and JLabel for Title (“G8 Library Management System”)

Add JPanel:

Add JLabel and JTextField for Username input

Add JLabel and JPasswordField for Password input

Add JButton for Log In

On Log In Button

If Username and Password input are filed and matched:

Proceed to Main Dashboard:

Else:

Display message box (“Invalid Username or Password”)

Create main Dashboard Window Frame:

Add JPanel and JLabel for Title (“Library Management Dashboard”)

Add JPanel for “Issue Book Section”:

Add JLabel and JTextField for Borrower’s Name input

Add JLabel and JTextField for Book ID input

Add JLabel and JTextField for Issue Date input

add JButton for “Issue Book”

Add JPanel for “Borrowed Book Record Section”:

Add JLabel for Section Title

Add JTable:

Set Column 1 Title to Borrower

Set Column 2 Title to Book ID

Set Column 3 Title to Status

Set Column 4 Title to Date Borrowed

Add JButton for “Return Book”

Add JPanel for “Book Catalog Section”:

Add JLabel for Section Title

Add JTable:

Set Column 1 Title to Book ID

Set Column 2 Title to Title

Set Column 3 Title to Author

Set Column 4 Title to Date Borrowed

Add JButton for “Add Book”

Add JButton for “Edit Book”

Add JButton for “Delete Book”

On “Issue Book” JButton:

If Borrower’s Name, Book ID, and Issue Date are filled:

Append record to Borrowed Book JTable with:

Borrower’s Name, Book ID, “Not Returned” as Status, and Issue Date

Display message box (“Book Issued Successfully”)

Else:

Display message box (“All fields must be filled!”)

On “Return Book” JButton:

If a row is selected in the Borrowed Book JTable:

Update the Status column of the selected row to “Returned”

Display message box (“Book Returned Successfully”)

Else:

Display message box (“Please select a book to return!”)

On “Add Book” JButton:

Open a dialog window:

Add JTextField for Book ID, Title, and Author input

Add JButton for “Save”

If all field are filled:

Append record to Book Catalog JTable

Display message box (“Book Added Successfully”

Else:

Display message box (“All fields must be filled!”)

On “Edit Book” JButton:

If a row is selected in the Book Catalog JTable:

Open a dialog window:

Pre-fill current Book ID, TItle, and Author values

Allow editing of values

Add JButton for “Save”

If new values are valid:

Update the selected row in the Book Catalog JTable

Display message box (“Book Edited Successfully”)

Else:

Display message box (“Please Select a book to edit!”)

On “Delete Book” JButton:

If a row is selected in the Book Catalog JTable:

Display confirmation dialog (“Are you sure you want to delete this book?”)

If confirmed:

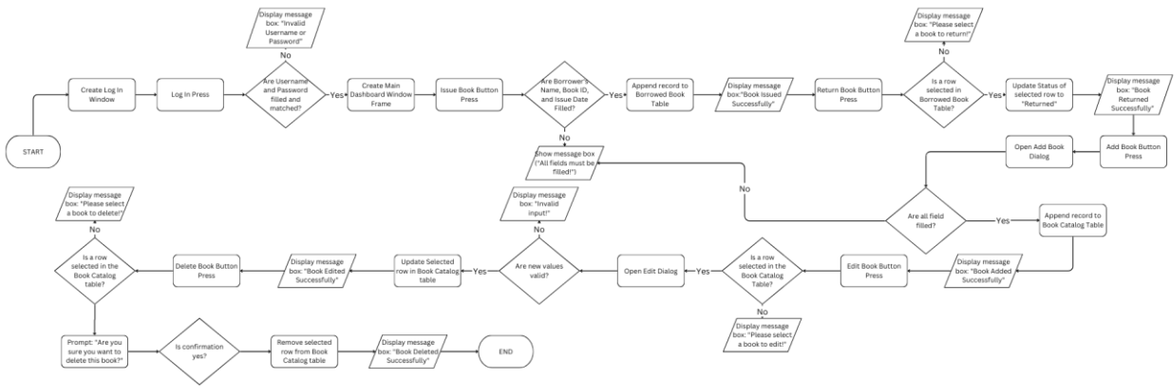
Remove the selected row from the Book Catalog JTable

Display message box (“Book Deleted Successfully”)

Else:

Display message box (“Please select a book to delete!”)

END



E. Testing and Validation:

1. Unit Testing

To ensure each feature worked flawlessly, we tested them one by one. In the Issue Book section, we confirmed that entering valid details like the borrower's name, book ID, and issue date successfully issued the book, while incorrect or missing information prompted clear error messages. The Borrowed Book Record was put to the test to make sure it accurately displayed issued books and updated the records correctly when a book was returned. Over in the Book Catalog, we verified that adding a book required all fields to be filled, editing reflected changes instantly, and deleting removed books entirely from the system. We also ran validation tests to catch errors—like blank entries or duplicate book IDs—ensuring the system handled them gracefully by preventing invalid actions and providing helpful feedback.

2. Integration Testing

Once the individual components were tested, we brought them together to check how they worked as a whole. Moving between the Issue Book, Borrowed Book Record, and the Book Catalog sections was smooth, with no loss of data. For instance, when a book was issued, it immediately updated its availability status in the catalog and appeared in the borrowed records. Buttons like "Add Book" and "Return Book" were tested repeatedly to confirm they performed their intended actions consistently across all sections. The goal was to ensure

everything—from navigating the system to performing tasks—felt seamless and interconnected, and we succeeded.

3. User Acceptance Testing (UAT)

The real test came when users got their hands on the system. They explored workflows such as adding new books, issuing them to borrowers, and returning borrowed items. Their feedback was invaluable in confirming that the interface was simple and intuitive, making library management tasks easy to perform. To challenge the system, users also tried entering incomplete or incorrect data, like leaving fields blank or inputting invalid book IDs. The system rose to the occasion, displaying clear, helpful error messages and blocking invalid actions. By the end of testing, it was clear that the system wasn't just functional—it was easy to use.

4. Stress Testing

We pushed the system to its limits to see how it performed under pressure. We simulated adding hundreds of books to the catalog and issuing dozens at a time. Even with a large dataset, the system handled operations like updating records and returning books without breaking a sweat. We also monitored how quickly it could read from and write to the files, and even with heavy loads, the performance remained impressive. Whether it was juggling multiple tasks or processing a backlog of operations, the system proved its resilience and reliability.

F. Known Issues and Limitations:

Limitations

1. The system does not use a database; instead, a simple TXT file stores all book and borrower information, which may limit scalability and data management efficiency.
2. There is no user account management system. Only basic access to the system is provided, and no file handling is implemented for login credentials.
3. Borrowed book records and catalog data are not connected to a real-time system, requiring manual updates for changes.
4. Features like advanced search, filters, or reporting tools are not included, limiting the functionality for large libraries.

Recommendations

1. Integrate a basic file handling system for login credentials to improve security and allow for user-specific access.
2. Implement a sign-up feature for new users with file handling or basic database integration to store user profiles.
3. Upgrade the storage system by replacing the TXT file with a relational database (e.g., MySQL or SQLite) to manage book and borrower data more effectively.
4. Add more functionality to the system, such as advanced search options, filters, and reporting tools, to cater to larger and more complex library setups.
5. Include an automated backup system to prevent data loss and ensure the integrity of the catalog and records.

G. Appendices:

Login Button Action code:

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String Username = User.getText();
        String Password = Pass.getText();


        if(Username.equals("") || Password.equals("")){
            JOptionPane.showMessageDialog(null, "Fill the missing field", "Error",
JOptionPane.ERROR_MESSAGE);
            User.requestFocus();
        }
        else if (Username.equals("Admin") && Password.equals("LibrarySystem")){
            JOptionPane.showMessageDialog(this, "Login SuccessFully");
            Dashboard h = new Dashboard();
            this.hide();
            h.setVisible(true);
        }
        else{
            JOptionPane.showMessageDialog(null, "Credential login did not
matched", "Error", JOptionPane.ERROR_MESSAGE);
            User.setText("");
            Pass.setText("");
            User.requestFocus();
```

Add Book Button Action code (Dashboard.java {Line 513 – 529}):

```
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        String id = Id.getText();
        String tle = title.getText();
        String otor = Author.getText();
        if(!id.equals("") || !tle.equals("") || !otor.equals("")){
            DefaultTableModel model = (DefaultTableModel)jTable1.getModel();
            model.addRow(new Object[]{Id.getText(), title.getText(), Author.getText()});
            JOptionPane.showMessageDialog(null, "Added Book Successfully");
```

```
            Id.setText("");
            title.setText("");
            Author.setText("");
        }
        else{
            JOptionPane.showMessageDialog(null, "Fill the missing field", "Error",
JOptionPane.ERROR_MESSAGE);
        }

    }
```

Issue Book Button Action Code (Dashboard.java {Line 436 – 452}):

```
    private void IssueActionPerformed(java.awt.event.ActionEvent evt) {
        String name = BName.getText();
        String id = bookId.getText();
        String sts = "Not Returned";
        String due = dateIssued.getText();
        if(!name.equals("") || !id.equals("") || !due.equals("")){
            DefaultTableModel model = (DefaultTableModel)jTable2.getModel();
            model.addRow(new      Object[]{BName.getText(),      bookId.getText(),sts,
dateIssued.getText()} );
            JOptionPane.showMessageDialog(null, "Issued Book Successfully");
            BName.setText("");
            bookId.setText("");
            dateIssued.setText("");
        }
        else{
            JOptionPane.showMessageDialog(null, "Fill the missing field", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

Return Book Action code ((Dashboard.java {Line 474 – 489}):

```
    private void ReturnActionPerformed(java.awt.event.ActionEvent evt) {
        DefaultTableModel model = (DefaultTableModel)jTable2.getModel();
        int selectedRowIndex = jTable2.getSelectedRow();
        if (selectedRowIndex != -1) {
            String status = model.getValueAt(selectedRowIndex, 2).toString();
            if(!status.equals("Returned")){
            String newStatus = "Returned";
            model.setValueAt(newStatus, selectedRowIndex, 2);
            JOptionPane.showMessageDialog(null, "Returned Book Successfully");
        } else{
            JOptionPane.showMessageDialog(this, "Book already Returned", "Error",
JOptionPane.WARNING_MESSAGE);
        }
    }
        else{
            JOptionPane.showMessageDialog(this, "Please select a book to Return.",
"No Selection", JOptionPane.WARNING_MESSAGE);
        }
```

Delete Book Button Action Code (Dashboard.java {Line 454 – 472}):

```
    private void DeleteActionPerformed(java.awt.event.ActionEvent evt) {
        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        int selectedRow = jTable1.getSelectedRow();

        // Check if a row is selected
        if (selectedRow != -1) {
            // Confirm deletion
            int confirmation = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this book", "Confirm Deletion", JOptionPane.YES_NO_OPTION);

            if (confirmation == JOptionPane.YES_OPTION) {
                // Remove the selected row from the table
                JOptionPane.showMessageDialog(null, "Delete Book Successfully");
                model.removeRow(selectedRow);
            }
        } else {
            JOptionPane.showMessageDialog(this, "Please select a book to delete.", "No Selection", JOptionPane.WARNING_MESSAGE);
        }
    }
```

Edit Book Button Action code (Dashboard.java {Line 538 – 563}):

```
    private void EditActionPerformed(java.awt.event.ActionEvent evt) {
        DefaultTableModel update =(DefaultTableModel)jTable1.getModel();
        int selectedRowIndex = jTable1.getSelectedRow();

        if (selectedRowIndex != -1) {

            String id = update.getValueAt(selectedRowIndex,0).toString();
            String b_title = update.getValueAt(selectedRowIndex,1).toString();
            String b_author = update.getValueAt(selectedRowIndex,2).toString();

            String NewID = JOptionPane.showInputDialog(null, "Enter new Book ID", id);
            String NewTitle = JOptionPane.showInputDialog(null, "Enter new Book Title", b_title);
            String NewAuthor = JOptionPane.showInputDialog(null, "Enter Author Name", b_author);

            JOptionPane.showMessageDialog(null, "Edit Book Details Successfully");

            update.setValueAt(NewID, selectedRowIndex, 0);
            update.setValueAt(NewTitle, selectedRowIndex, 1);
            update.setValueAt(NewAuthor, selectedRowIndex, 2);
            }
        else {
            JOptionPane.showMessageDialog(this, "Please select a book to edit.", "No Selection", JOptionPane.WARNING_MESSAGE);
        }
```

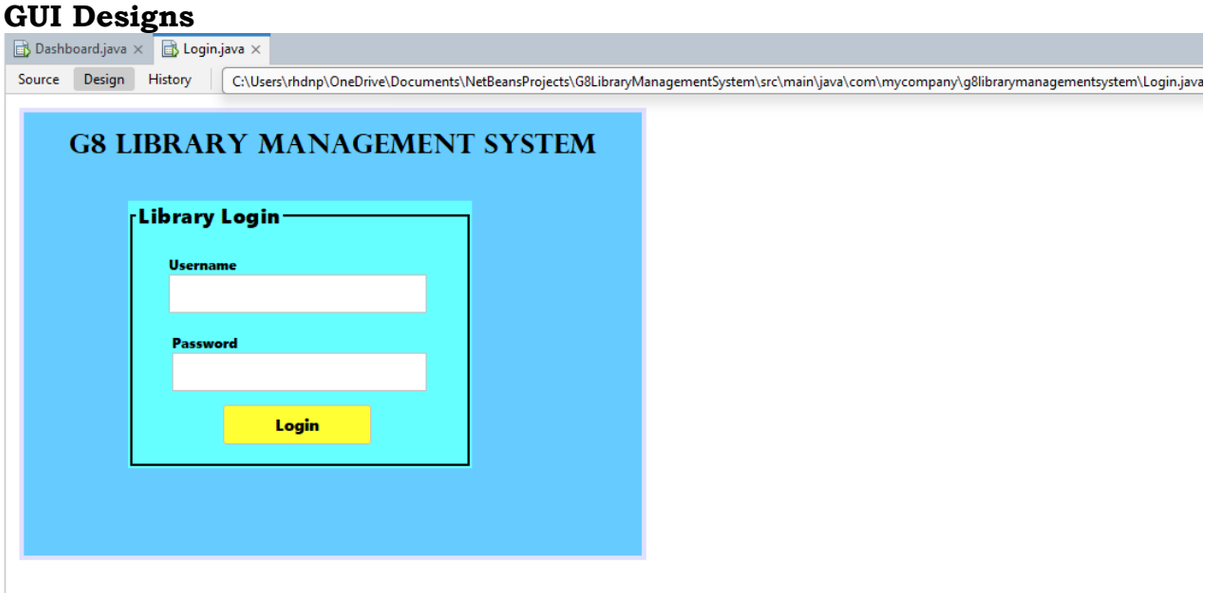
```
}
```

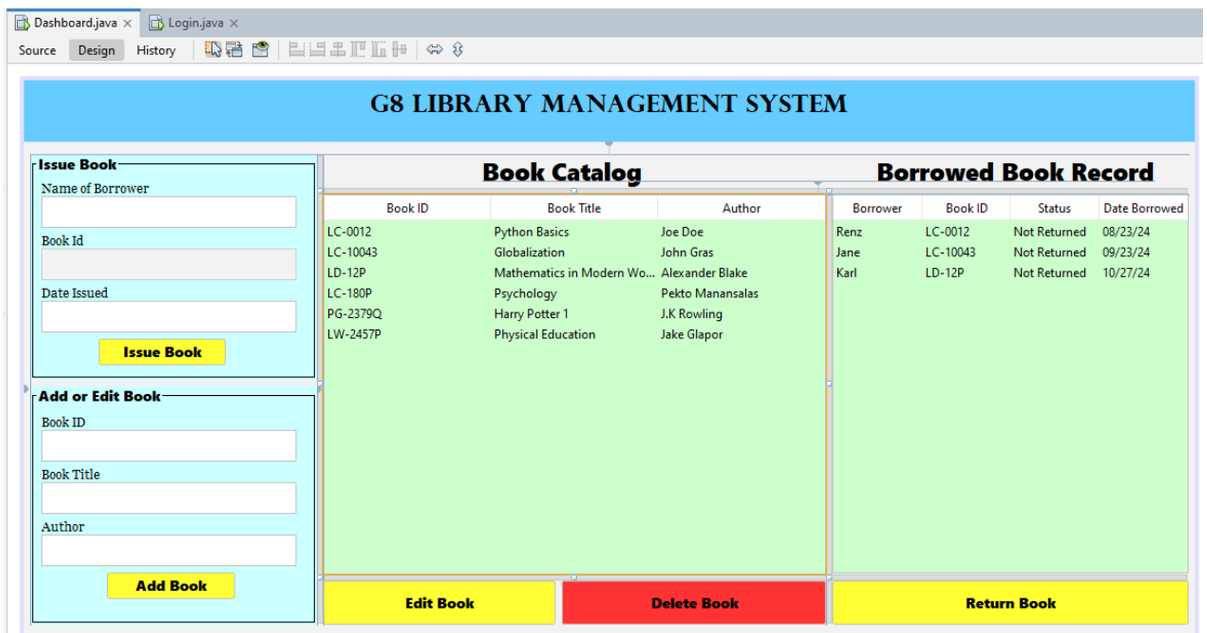
Table Clicked Action Code ((Dashboard.java {Line 531 – 536}):

```
private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
    DefaultTableModel click = (DefaultTableModel)jTable1.getModel();
    int selectedRowIndex = jTable1.getSelectedRow();

    bookId.setText(click.getValueAt(selectedRowIndex,0).toString());
}
```

GUI Designs





H. Group 8 Members:

Alipio, Christian Jericho T.

Cruz, Jazzen Paul V.

Padilla, Rheden N.

Punongbayan, Pamela C.

Rivera, Ivy Jean P.

Delegation of Activities:

Programmer and Designer – Padilla Rheden N.

Head and Consultant for Technical Documentation – Punongbayan, Pamela C.

Associate for Technical Documentation - Alipio, Christian Jericho T.

- Cruz, Jazzen Paul V.

- Rivera, Ivy Jean P.