COMP6699001: Object Oriented Programming

Final Project Report

"Library Management System"

MADE BY:

Ryan Patrick Komala (2702334853) of L2AC

LECTURER:

Mr. Jude Joseph Lamug Martinez MCS

School of Computing and Creative Arts

Computer Science Program

Bina Nusantara International University

Jakarta

2024

TABLES OF CONTENT

I. PROJECT SPECIFICATION

1.1 Project Description

For this final project, I made a "Library Management System" completely with the Java programming language as well as a relatively simple graphic user interface using the Java Swing implementation to go along with the system. As its name suggests, this is a data management system that is meant to be used inside a Library, it serves as a centralized platform designed to manage and streamline the operations of a library, ensuring the efficient organization and accessibility of its resources. The features of this program reduce the manual workload on the library staffs, whilst also benefiting the users from simplified processes for searching, borrowing, and returning materials, which enhances their overall library experience.

1.2 Project Link

The code and documentation of this final project can be accessed through this link:

https://github.com/RyanPK528/Library-Management-System

1.3 Features

1. Login or Sign-Up Page
    - The first thing users will see after running the program, it prompts the user to either login with an existing account or create a new one. Users are separated through the account they made; library staffs would have an "Admin" account while library visitors would have a "Normal User" account.

2. View Books
    - Displays all of the stored data regarding books inside the library, the data includes: name of the book, book author, book publisher, quantity of the book on sale, price of the book, and available copies of book to borrow. This feature is accessible by both Admins and Normal Users.

3. Search
    - Display the data regarding a specific book based on the searched name of the book, it displays the same information as view book but only for a specific book. This feature is accessible by both Admins and Normal Users.

4. Add Books
    - Add new data of a book into the library's database of books. This feature is only accessible to admins, to add a book admins must input all the necessary information regarding the book, name of the book, book author, book

publisher, quantity of the book on sale, price of the book, and available copies of book to borrow.

5. Delete Book
    - Deletes the data of a specified book based on the name of the book. This feature is only accessible to admins, and admins only have to input the name of the book to delete the information of that book.

6. Delete Data
    - Deletes the entire database of the system, this includes all information regarding the books in the library as well as the various user accounts on the systems. This feature is only accessible to admins.

7. View Orders
    - Displays the order history of a specific book, it shows the name of the user who ordered the book, the price of the ordered book as well as the quantity of books they ordered. This feature is only accessible to admins.

8. Place Order
    - Allow users to order the desired amount of a specific book, the user only has to input the name of the book as well as the quantity they want. This feature is only accessible to normal users.

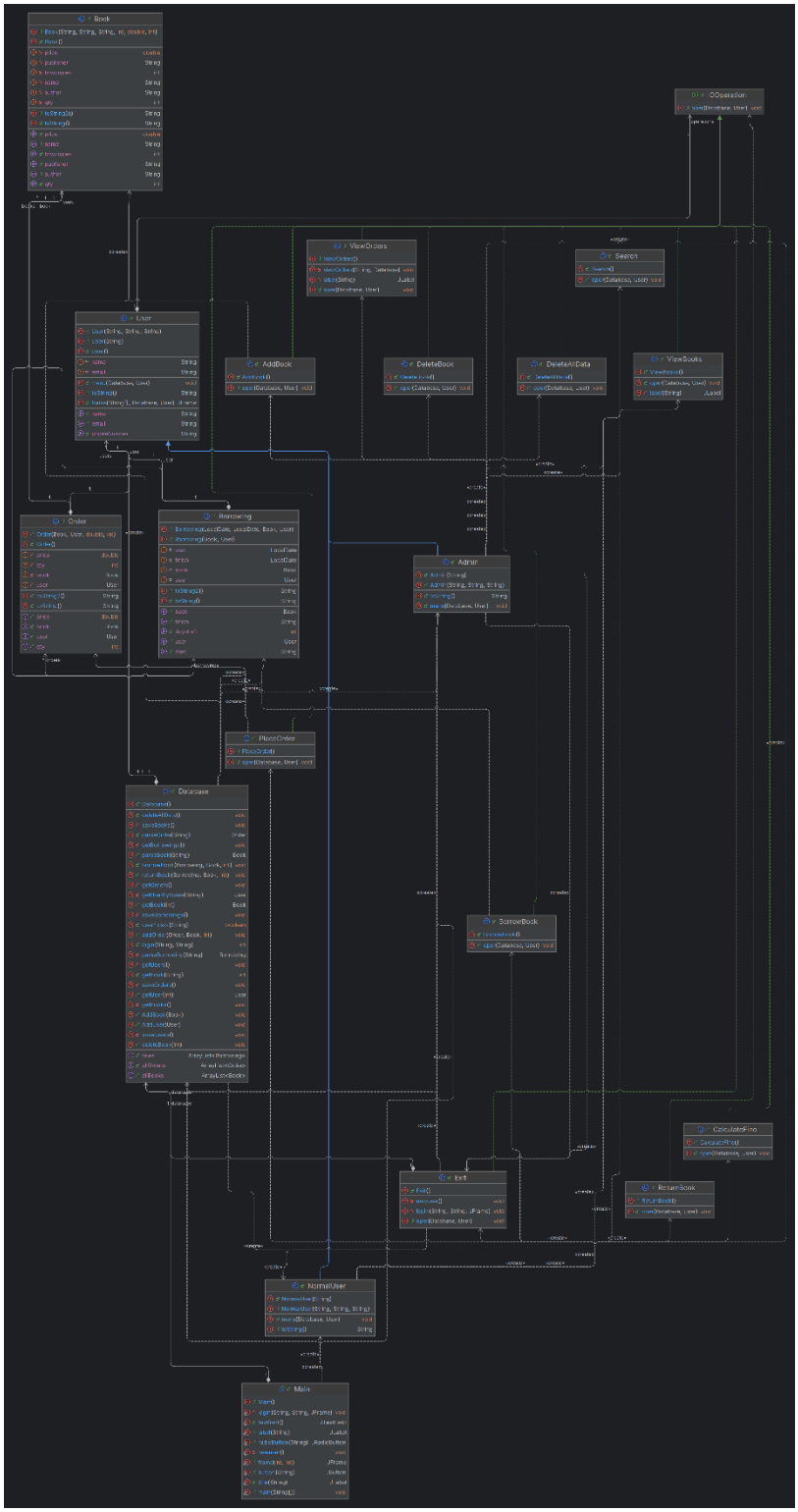
9. Borrow Book
    - Allow users to borrow a book from the available supply of copies lent for borrowing. Users can only borrow the book for a maximum of 14 days or in a 2-week span and they must return it before then. This feature is only accessible to normal users.

10. Calculate Fine
    - After the initial 14 days, if the book has not been returned by the user, the program will start calculating the fine that the user must pay, where each fine will increase by a constant rate per day. When user clicks on this feature, the amount of fine they have to pay will be displayed. This feature is only accessible to normal users.

11. Return Book
    - Allow users to return the book that they have previously borrowed, to return the user only has to input the name of the book and the book will be “returned”, as in the number of available borrowing copies will be adjusted and the countdown for the fine will stop. This feature is only accessible to normal users.

II. SOLUTION DESIGN

2.1 Class Diagram



2.2 Classes

No.	Java Class	Inheritance
1	AddBook	Implements IOOperation
2	Admin	Extends User
3	Book	None
4	BorrowBook	Implements IOOperation
5	Borrowing	None
6	CalculateFine	Implements IOOperation
7	Database	None
8	DeleteAllData	Implements IOOperation
9	DeleteBook	Implements IOOperation
10	Exit	Implements IOOperation
11	IOOperation	Implemented by AddBook, BorrowBook, CalculateFine, DeleteAllData, DeleteBook, Exit, PlaceOrder, ReturnBook, Search, ViewBooks and ViewOrders
12	Main	None
13	NormalUser	Extends User
14	Order	None
15	PlaceOrder	Implements IOOperation
16	ReturnBook	Implements IOOperation
17	Search	Implements IOOperation
18	User	Inherited by Admin and NormalUser
19	ViewBooks	Implements IOOperation
20	ViewOrders	Implements IOOperation

2.3 Implementation

1. Java AWT:
   Java AWT or Abstract Window Toolkit serves as one of the main API used for developing the GUI for this project. Java AWT is part of the Java Foundation Classes (JFC) that provides a way to build platform-independent graphical applications.

2. Java Swing:
   Java Swing is similar to AWT as it is also used in creating the GUI for this project. Java Swing acts as an extension of AWT and provides additional useful features.

3. Java I/O:
   Java I/O (Input and Output) is used to process the input and produce the output. This API helps in creating a specific desired output based on the user's input it

also performs other file handling tasks.

4. java.time:
   This API is used to keep track of the dates or day so that for the program could calculate the days between the borrowing date and returning date

5. java.util:
   Used to implement an Array List data structure into the program which is used in managing the various data in the system

2.4 Important Code

- **Login or create account**

```
/**
 * This method is used to login a user
 */
private static void login(String phonenumber, String email, JFrame frame) {  1 usage
    int n = database.login(phonenumber, email);
    if (n!= -1) {
        User user = database.getUser(n);
        user.menu(database, user);
        frame.dispose();
    } else {
        JOptionPane.showMessageDialog(new JFrame(),  message: "User doesn't exist");
    }
}
```

```
// Create a new user
User user;
if (admin.isSelected()) {
    user = new Admin(name.getText().toString(),
            email.getText().toString(), phonenumber.getText().toString());
} else {
    user = new NormalUser(name.getText().toString(),
            email.getText().toString(), phonenumber.getText().toString());
}
frame.dispose();
database.AddUser(user);
user.menu(database, user);
```

This is a code snippet taken from Main.java, these algorithms prompt the user to input their login credentials in order to access the other features. If the user has no account, the user can Sign up and create their account by inputting their name, phone number and

email.

- **Add Book Data**

```
// Create a new Book object and set its properties
Book book = new Book();
if (database.getBook(name.getText().toString())>-1) {
    JOptionPane.showMessageDialog(new JFrame(), message: "There is already a book with this name!");
    return;
} else {
    book.setName(name.getText().toString());
    book.setAuthor(author.getText().toString());
    book.setPublisher(pub.getText().toString());
    book.setQty(Integer.parseInt(qty.getText().toString()));
    book.setPrice(Double.parseDouble(price.getText().toString()));
    book.setBrwcopies(Integer.parseInt(brwcpis.getText().toString()));

    // Add the book to the database
    database.AddBook(book);
    JOptionPane.showMessageDialog(new JFrame(), message: "Book added successfully");
    frame.dispose();
}
```

This is a code snippet taken from AddBook.java, after going through some input validations, this is the algorithm that will create a new Book object with its corresponding properties based on the user's inputs and adding it into the database.

- **Borrowing State**

```
// If the book is available, create a new borrowing and update the database
if (n) {
    if (book.getBrwcopies()>1) {
        Borrowing borrowing = new Borrowing(book, user);
        book.setBrwcopies(book.getBrwcopies()-1);
        database.borrowBook(borrowing, book, i);
        JOptionPane.showMessageDialog(new JFrame(),
                message: "You must return the book before 14 days from now\n"
                        + "Expiry date: "+borrowing.getFinish()+"\nEnjoy!\n");
        frame.dispose();
    } else {
        JOptionPane.showMessageDialog(new JFrame(),
                message: "This book isn't available for borrowing");
    }
}
```

```java
public Borrowing(Book book, User user) {  1 usage
    start = LocalDate.now();
    finish = start.plusDays( daysToAdd: 14);
    daysleft = Period.between(start, finish).getDays();
    this.book = book;
    this.user = user;
}
```

```java
// Method to get the start date of the borrowing as a string
public String getStart() { return formatter.format(start); }

//Method to get the finish date of the borrowing as a string
public String getFinish() { return formatter.format(finish); }

//Method to get the number of days left until the finish date
public int getDaysLeft() { return Period.between(finish, LocalDate.now()).getDays(); }
```

This is a code snippet taken from both BorrowBook.java and Borrowing.java, after going through some input validations, this is the algorithms used to update the state of a user to "borrowing" and decrease the number of available copies to lend whenever a Normal User request to borrow a certain book. When a user borrows a book, the program will check the date of borrowing and add another 14 days as the date of returning.

- **Calculating the Fine for late return**

```java
for (Borrowing b : database.getBrws()) {
    // Check if the user has borrowed the book
    if (b.getBook().getName().matches(name.getText().toString()) &&
            b.getUser().getName().matches(user.getName())) {
        // Check if the user is late
        if (b.getDaysLeft()>0) {
            JOptionPane.showMessageDialog(new JFrame(),
                    message: "You are late!\n"+"You have to pay "+b.getDaysLeft()*10+"k as fine");
            frame.dispose();
        } else {
            JOptionPane.showMessageDialog(new JFrame(),
                    message: "You don't have to pay fine");
            frame.dispose();
        }
        g = false;
        break;
    }
}
```

This is a code snippet taken from CalculateFine.java, after it validates the user's input, this algorithm checks if the user is past the returning date, if it is pass the returning date, the program will display to the user that they have to pay a fine of 10k multiplied by the

number of days after the returning date, anything before the returning date, then the program will say that the user don't have to pay fine.

- **Returning the borrowed book**

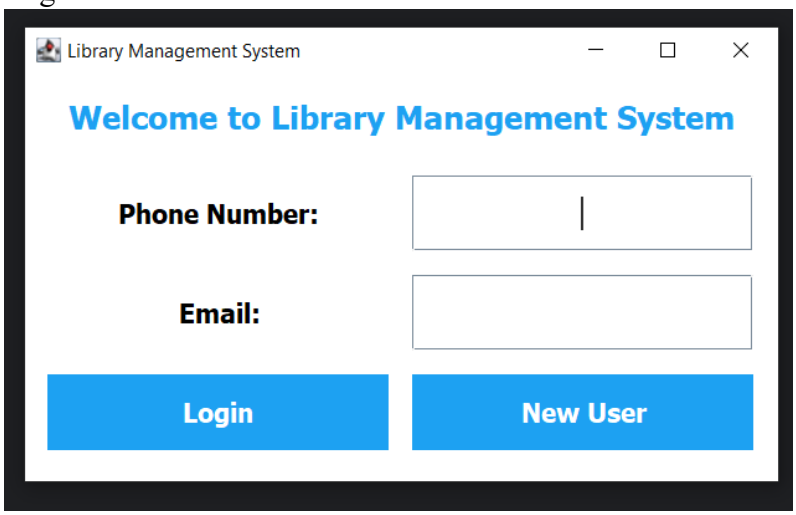
```
if (!database.getBrws().isEmpty()) {
    for (Borrowing b : database.getBrws()) {
        if (b.getBook().getName().matches(name.getText().toString()) &&
                b.getUser().getName().matches(user.getName())) {
            Book book = b.getBook();
            int i = database.getAllBooks().indexOf(book);
            if (b.getDaysLeft()>0) {
                JOptionPane.showMessageDialog(new JFrame(), message: "You are late!\n"
                        + "You have to pay "+Math.abs(b.getDaysLeft()*10)+"k as fine");
            }
            book.setBrwcopies(book.getBrwcopies()+1);
            database.returnBook(b, book, i);
            JOptionPane.showMessageDialog(new JFrame(), message: "Book has been returned\nThank you!");
            frame.dispose();
            break;
        } else {
            JOptionPane.showMessageDialog(new JFrame(), message: "You didn't borrow this book!");
        }
    }
```

This is a code snippet taken from ReturnBook.java, this algorithm will check if the user is late in returning the book and depending on the result, it will display a fine (similar to Calculate Fine) or it will display a normal message explaining the book has been returned. This also updates the database and add 1 to the available book copies to lend.

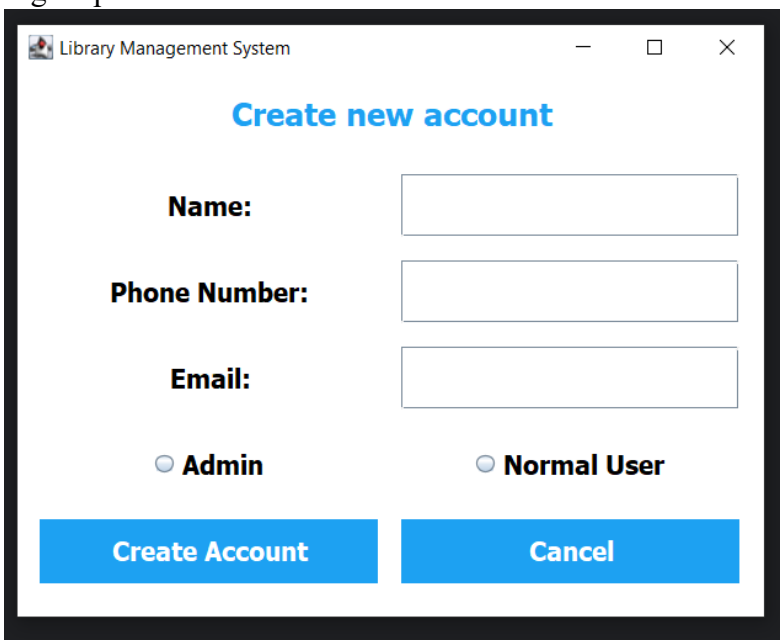
III. PROGRAM DEMO

3.1 Screenshots

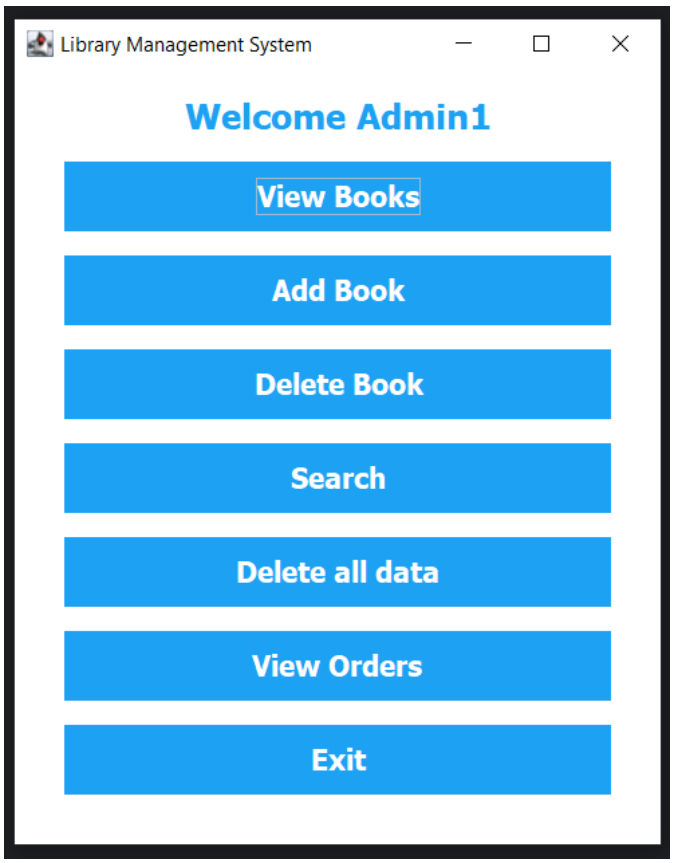
- Login Screen:



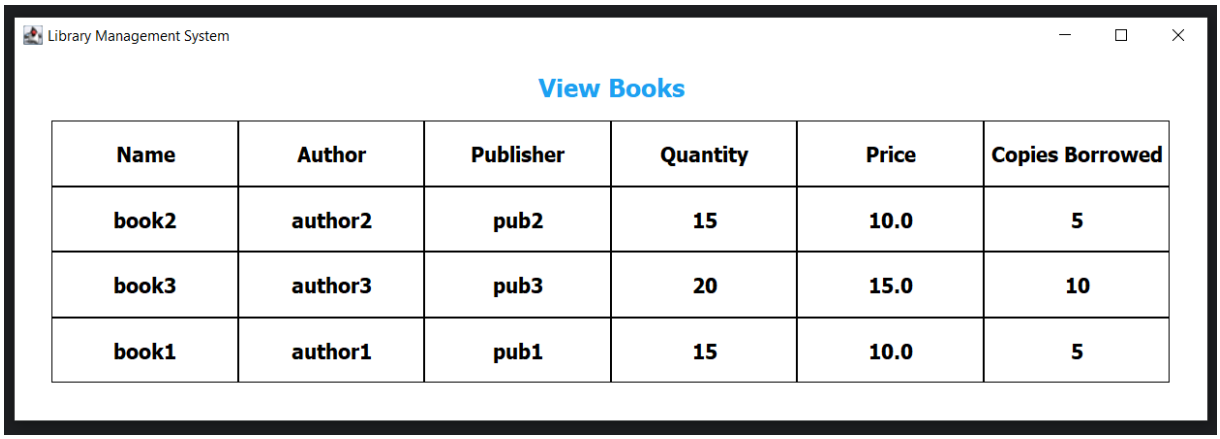
- Sign Up Screen:



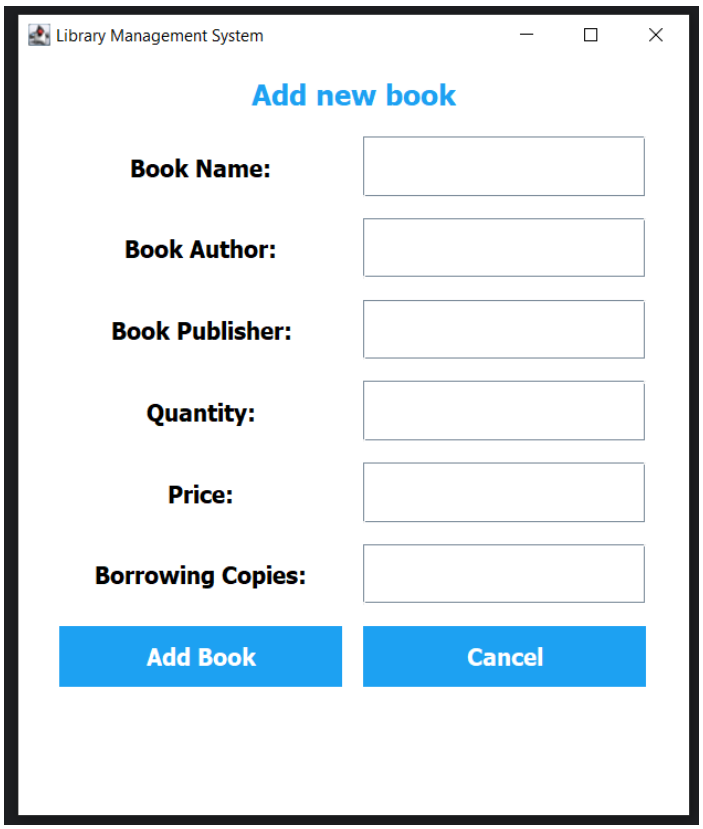
- Main Menu for Admins:



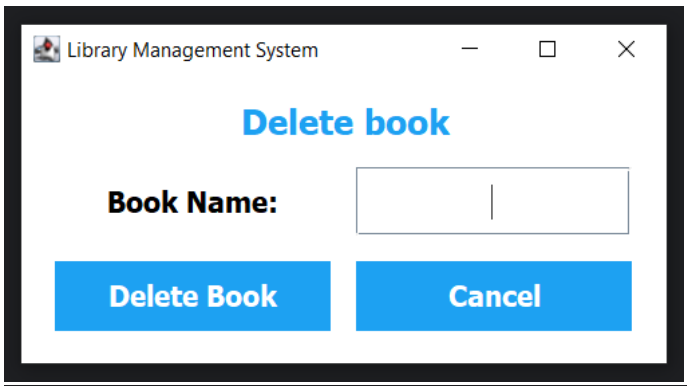
- View Books Screen:

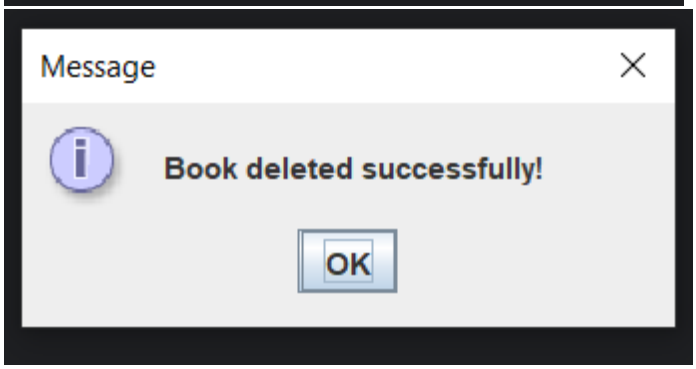


- Add Book Screen:

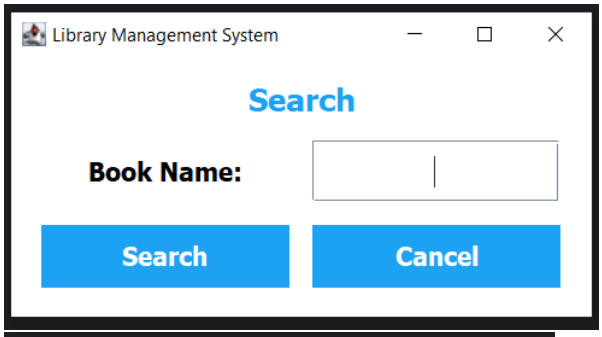


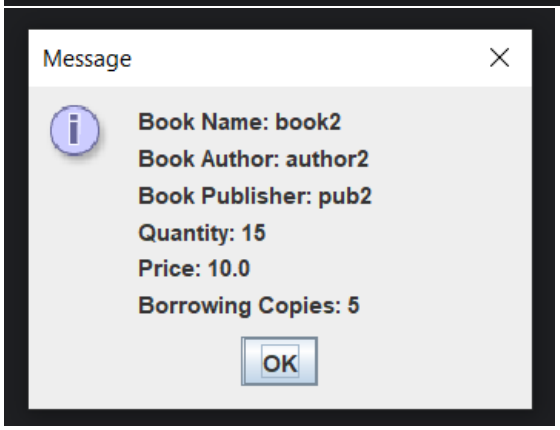
- Delete Book Screen:



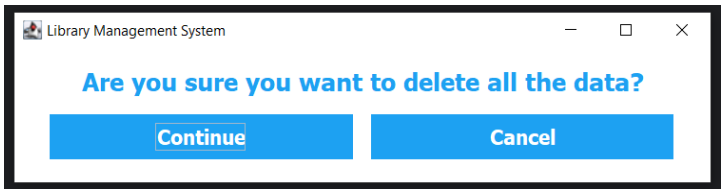


- Search Screen:

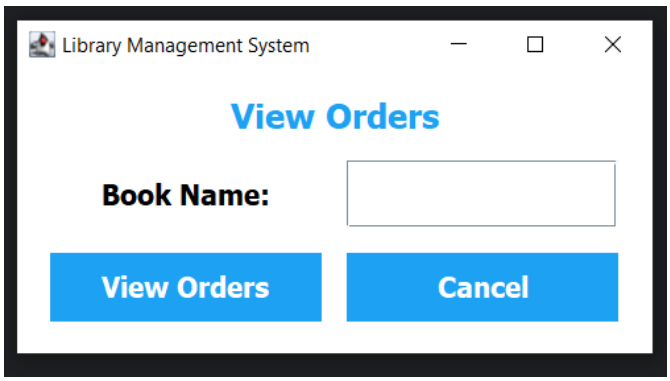


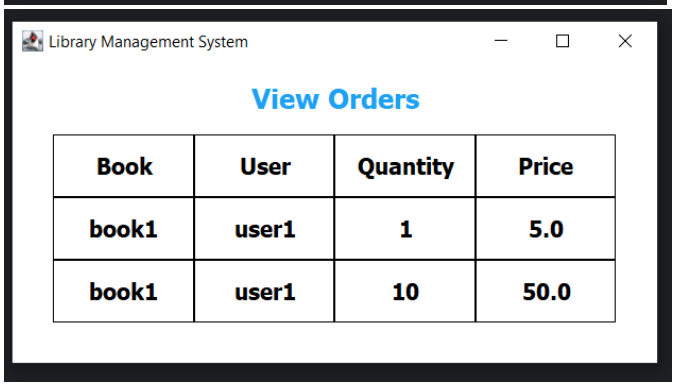


- Delete All Data Screen:

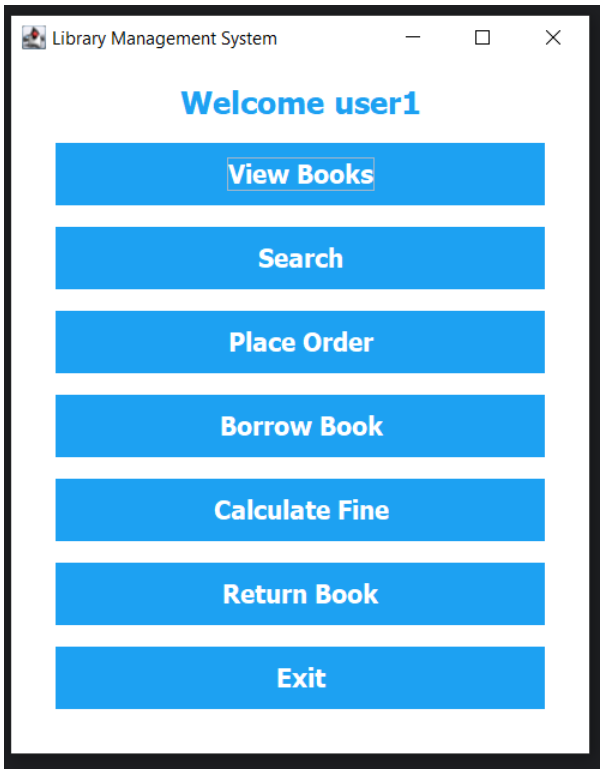


- View Orders Screen:

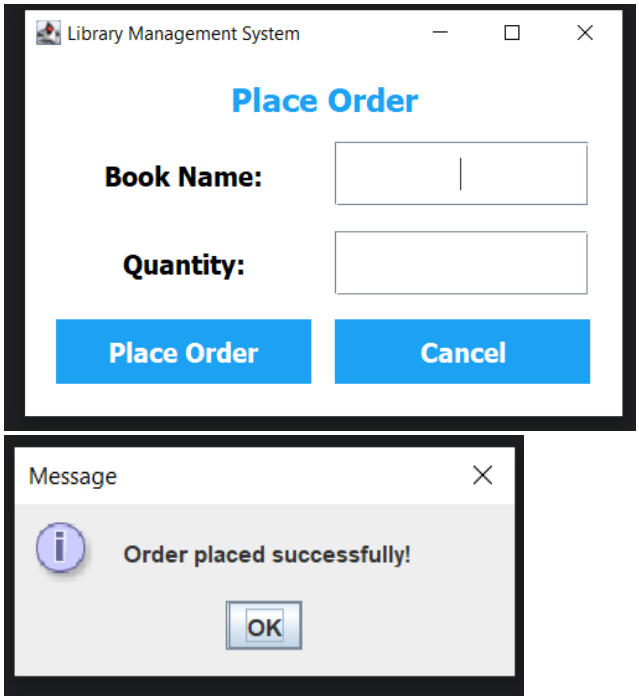




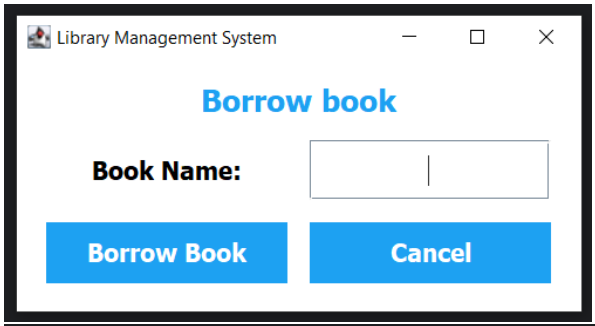
- Main Menu for Normal Users:

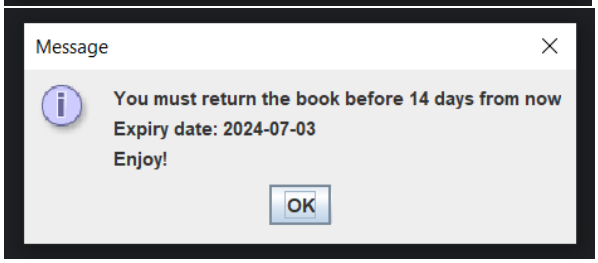


- Place Order Screen:

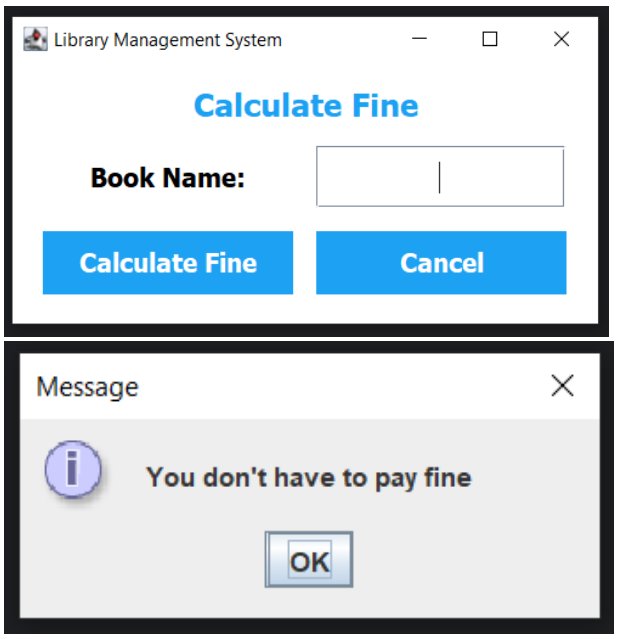


- Borrow Book Screen:

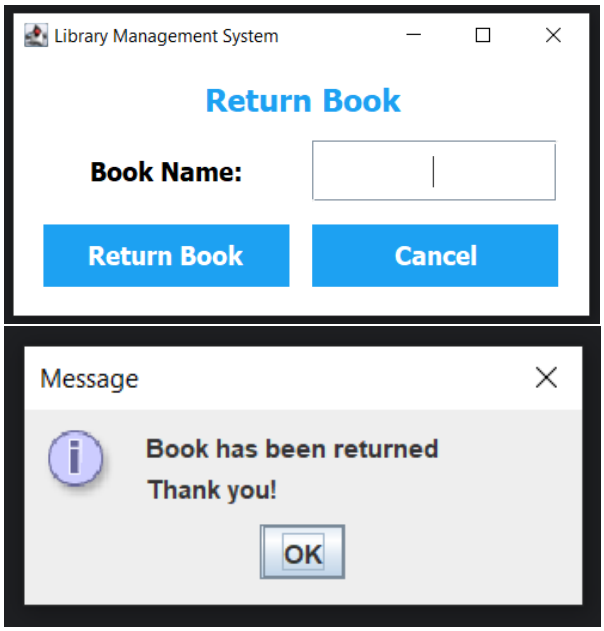




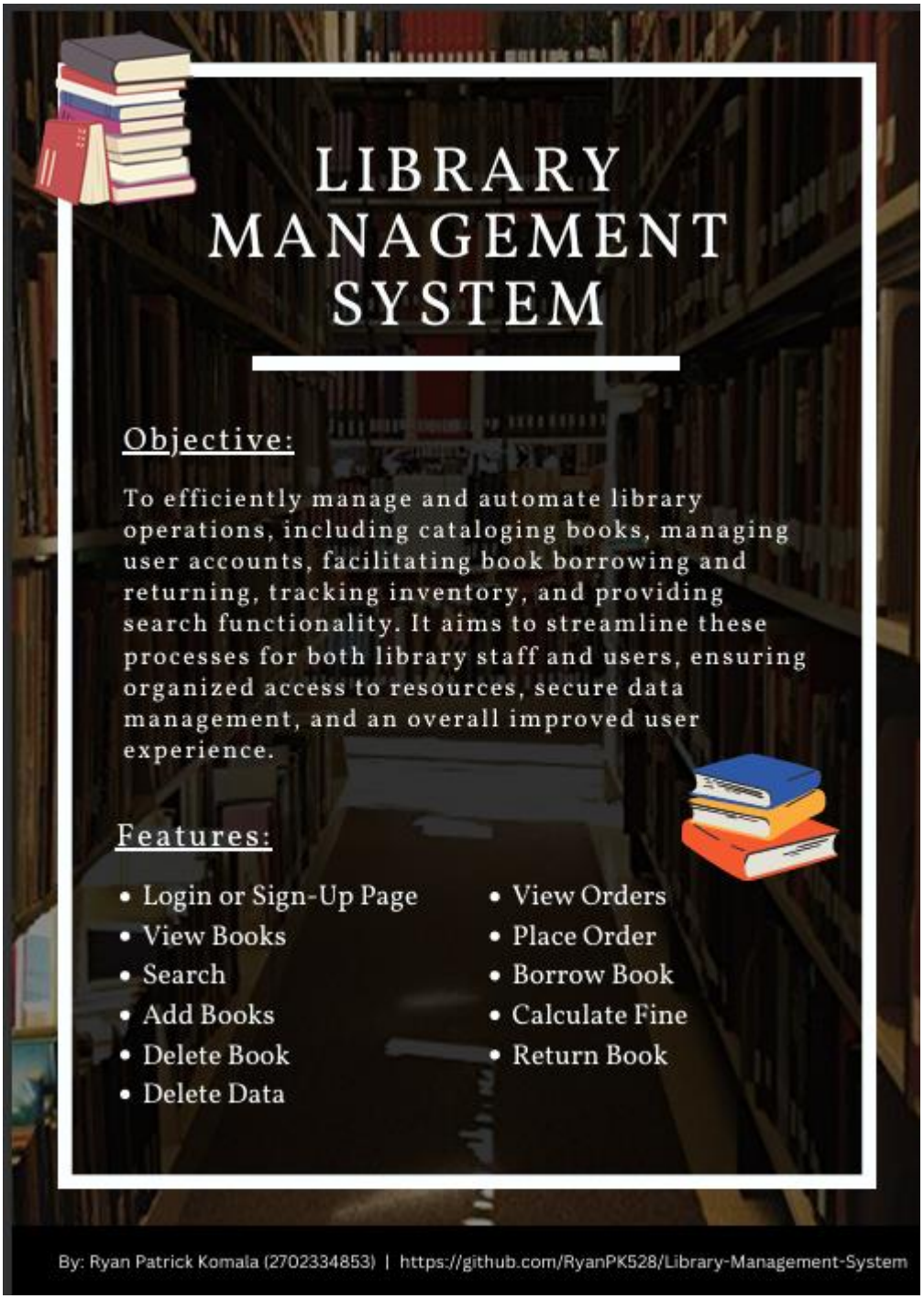
- Calculate Fine Screen:



- Return Book Screen:



3.2 Poster


LIBRARY MANAGEMENT SYSTEM
Objective:
To efficiently manage and automate library operations, including cataloging books, managing user accounts, facilitating book borrowing and returning, tracking inventory, and providing search functionality. It aims to streamline these processes for both library staff and users, ensuring organized access to resources, secure data management, and an overall improved user experience.
Features:
• Login or Sign-Up Page
• View Books
• Search
• Add Books
• Delete Book
• Delete Data
• View Orders
• Place Order
• Borrow Book
• Calculate Fine
• Return Book
By: Ryan Patrick Komala (2702334853) | https://github.com/RyanPK528/Library-Management-System

IV. EVALUATION AND REFLECTION

4.1 Reflection

In all honesty, I would say that my programming skills with the Java language is quite lackluster, this could be due to the unfamiliar format of how the code is written when compared to Python, it could also be because of how Java is heavily object-oriented, or it could be a completely different issue that I am unaware of. In the process of making this final project, I was met with multiple obstacles, to the point where I once reached a "roadblock" as the project would not perform the way I wanted it to. I had to familiarize myself to so many different things, like teaching myself how to create a GUI using APIs such as Java AWT and Swing or how to create a functioning database, where in my case, I ended up using local .txt files to save all the data.

The program in itself is incomplete and could use further improvements in order to be a completely functioning Library Management System that is applicable in real life. As of right now, there are some bugs with the code and the program lacks some crucial features for the Normal Users, such as displaying their own history of ordering and borrowing books. If there is one thing I would have done differently, it would be to manage my time better and start working on the project day by day. I regretfully chose to procrastinate and hold off working on the project, which in the end led to me rushing the work which was not only stressful but also physically straining. But with all of this said, this final project has without a doubt, allow me to fully understand my capabilities and limitation in Object-Oriented Programming and become a stepping stone in understanding the concepts and ideas we have learnt in class.

V. RESOURCES

- YouTube tutorial by "Tech with Bob"
  https://www.youtube.com/playlist?list=PL-cxzMmn1xXG87ak7josmprqsp3da_Ovk

- Introduction to Java Swing
  https://www.geeksforgeeks.org/introduction-to-java-swing/

- Java AWT Tutorial
  https://www.geeksforgeeks.org/java-awt-tutorial/