



---

# SOFTWARE ENGINEERING

---

Final Project



## Phase 2: Design and Implementation

Name	ID
Riham Mohamed Elsayed	6121
Reem Abdel Haleem	6114
Khaled Gamal	6117

## 1) Requirements

### a) User Requirements:

Definition: Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

The Library Management System shall provide a digitalized documentation for a library. It shall help the librarian control borrowed and returned books. It shall help the User keep track of the books he has borrowed and the books available at the library and finally, it shall help the system Admin control all librarians who have access to the system.

### b) System Requirements:

Definition: A structured document setting out detailed descriptions of the system's functions, services, and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

The Library Management system shall be able to:

- Help the user to create an account and insert all his information in the system database.
- User shall be able to register safely into the system.
- User shall be able to view all his borrowed books to know when they should be returned.
- User shall also be able to view all available books at the library so that he knows what books he can borrow later on when he arrives at the library.
- Librarian shall be able to register safely into the system.

- Librarian shall be able to add books to the system to be stored on the database.
- Librarian shall also be able to view all the borrowed books and their corresponding borrowers.
- Librarians shall be able to register in the system all the books borrowed and their corresponding users.
- Librarians shall be able to register in the system the books returned by the borrowers to be added in the available books for people to borrow.
- There shall be only one Admin to register into the system.
- The admin shall be able to add new Librarians into the system to be stored in the database, he can also view all these Librarians who have access to the system and delete any librarian from the system.

## 2) Functional and non-Functional Requirements

### a) Functional Requirements:

- Only authentic user must have the access to the system.
- System must validate right credentials when a user logs into system.
- User can create a new account if they don't have one.
- Same Id's for 2 or more books shall not be allowed.
- Librarian must enter borrow and return date in database when registering a borrow.
- Librarian must enter the user-id and book ISBN when registering a return.
- System must update the quantity of a book when its borrowed or returned.

- System admin must be able to add a new librarian to the systems database.
- User must be able to view the books that he borrowed.
- The system must be able to provide the details of a book.
- Librarian must be able to add books to the system.
- The system must assign each user a unique id.

#### **b) Non-Functional Requirements:**

- **Performance:** system should be able to perform desired tasks in reasonable time.
- **Scalability:** The proposed system should be scalable to support extended number of users.
- **Ease of Use:** system should be user-friendly and would provide Graphical User Interface (GUI).
- **Accessibility:** system should be a desktop application.
- **Security:** The system would provide access to only legitimate users. It will be secure on network and only authorized person can use it.
- Librarian can specify the number of copies of the books for a single insert.
- User is warned if he submitted empty field in creating a new account.
- Librarian is warned if he submitted empty field in adding a new book or registering a borrow or registering a return.
- Admin is warned if he submitted empty field in creating a new librarian or deleting an existing librarian.
- The return date of the book must be greater than the borrowing date.

- Phone number must be 11 digits and starting with 01
- The user age must be above 14 years old.
- The librarian age must be above 21 years old.

### 3) Software Process

#### a) Suggested type of software Process:

Since the project was small and not overly detailed it was a better approach for us to use Agile method as we kept changing the requirements to make it as realistic as possible for a real digitalized documentation of a library.

We worked level by level and made sure it is functioning properly before moving on to the next level.

#### b) Division of phases:

**Phase 1:** we started by picking a theme for our project and decided upon the library management system concept. We provided a detailed description of the problem and an outline of how we plan to solve it. And finally listed our goals and objectives.

**Phase 2:** design and implementation

We started our design by deciding upon the functional and nonfunctional requirements we need to implement in the library. Then made basic use case diagram that helps us understand what is needed to be done and a simple class diagram, sequence diagram, activity diagram, state machine diagram with the all basics needed.

Then decided upon the architecture model that fits our project and created a simple diagram of how things can go.

We divided the project into phases:

1)Login system: we created a login system for Library user, Librarian and Admin that takes in the username and password and check if they exist correctly in the database and that if any error occurs our system sends a message to the user to enter correct username or password.

2)create an account: where user can create an account or Admin to register a new Librarian into the system, we handled all errors that could occur in the any of the input information and a specific error message should be sent for different information errors.

3)Add books: Librarian can add books to the Available books database any error during the process was handled.

4)register a borrow/register a return: Librarian can register in the database if a user has borrowed or returned a book. All errors were handled.

5)providing all the views and reports needed: we created a table view to display all the necessary reports and views needed.

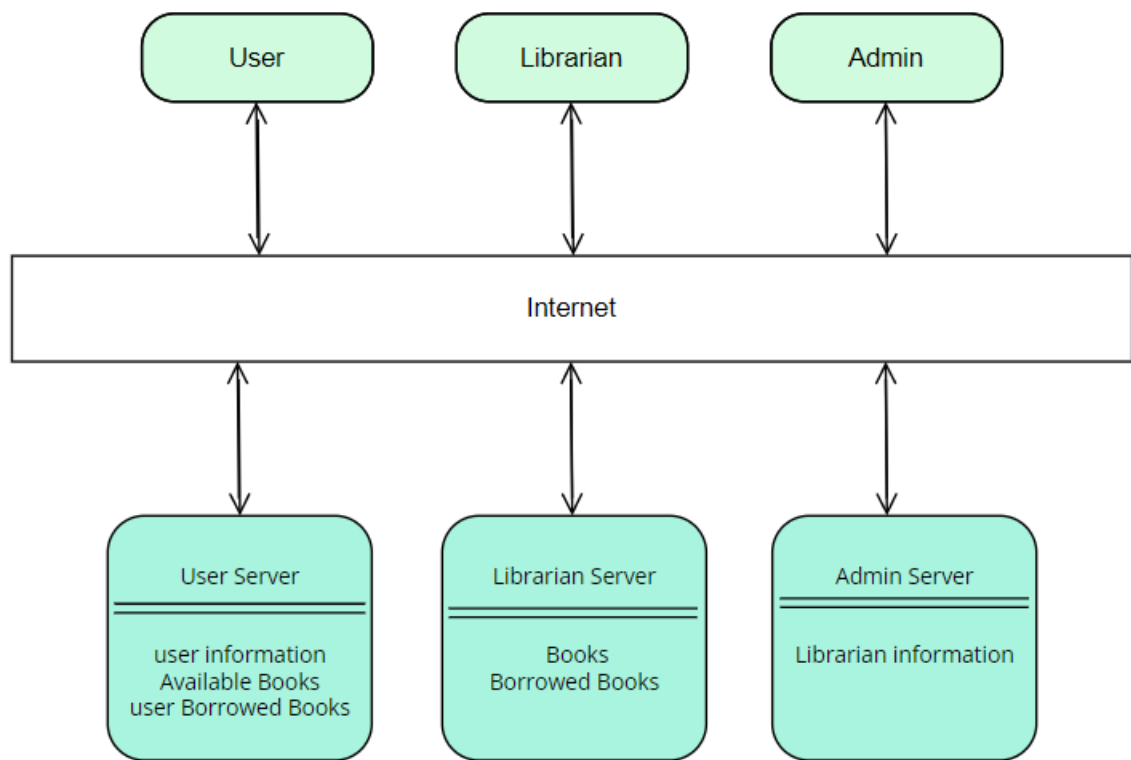
Of course, the requirements kept changing as we went along with the project and so diagrams have been altered multiple times.

At the end we applied a CSS file to the project to make it as clear and appealing as possible.

#### 4) Architectural design

As for the Architectural design we used a client server architecture pattern for our system and so our architecture model had the following:

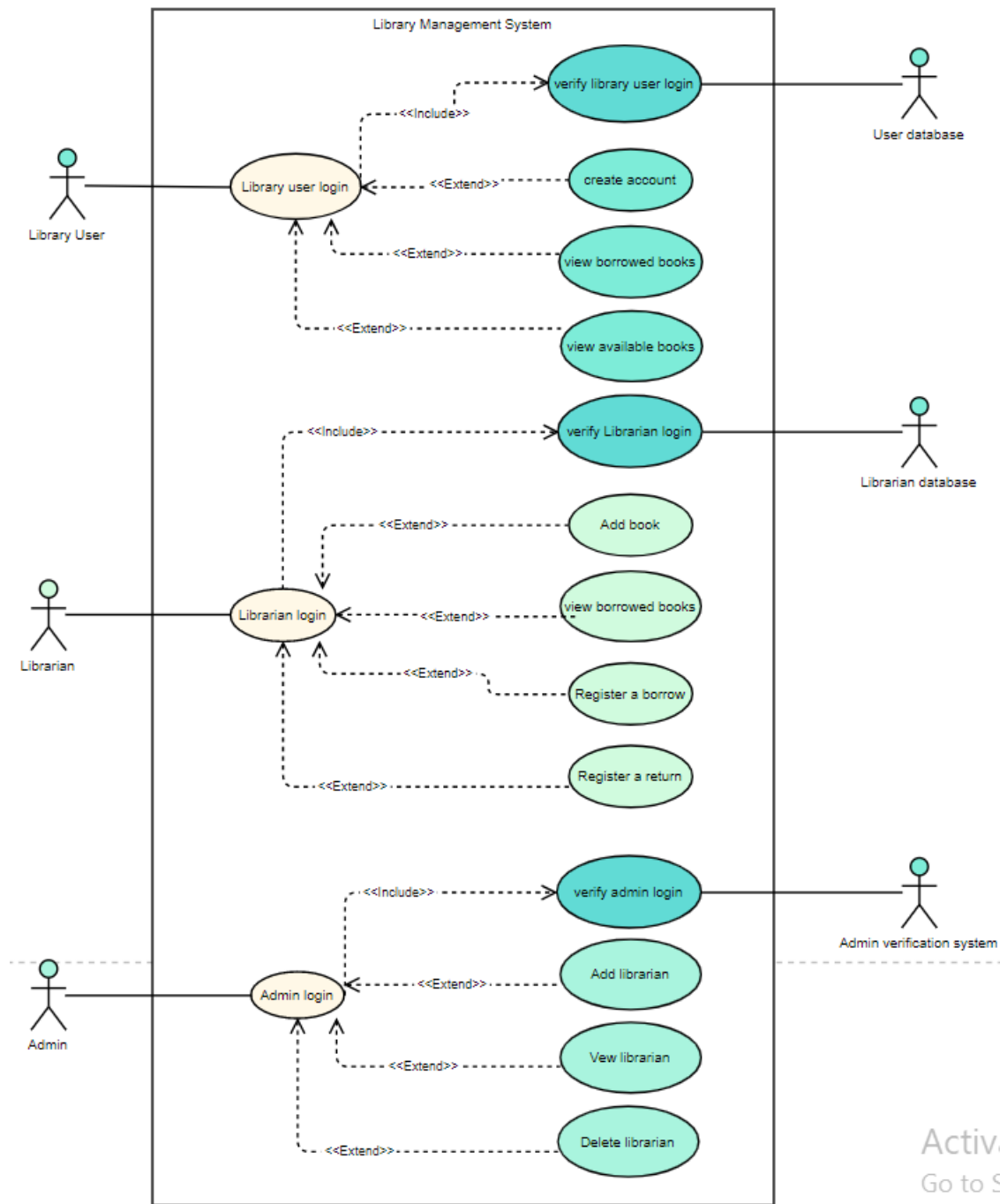
- Set of stand-alone servers which provide specific services (user server-Librarian Server-Admin server).
- Set of clients which call on these services (User-Librarian-Admin).
- Network which allows clients to access servers.



The principal advantage of this model is that servers can be distributed across a network. General functionality can be available to all clients and does not need to be implemented by all services. And since the load on the system is variable it's better for each client to be reaching a specific server for a specific functionality.

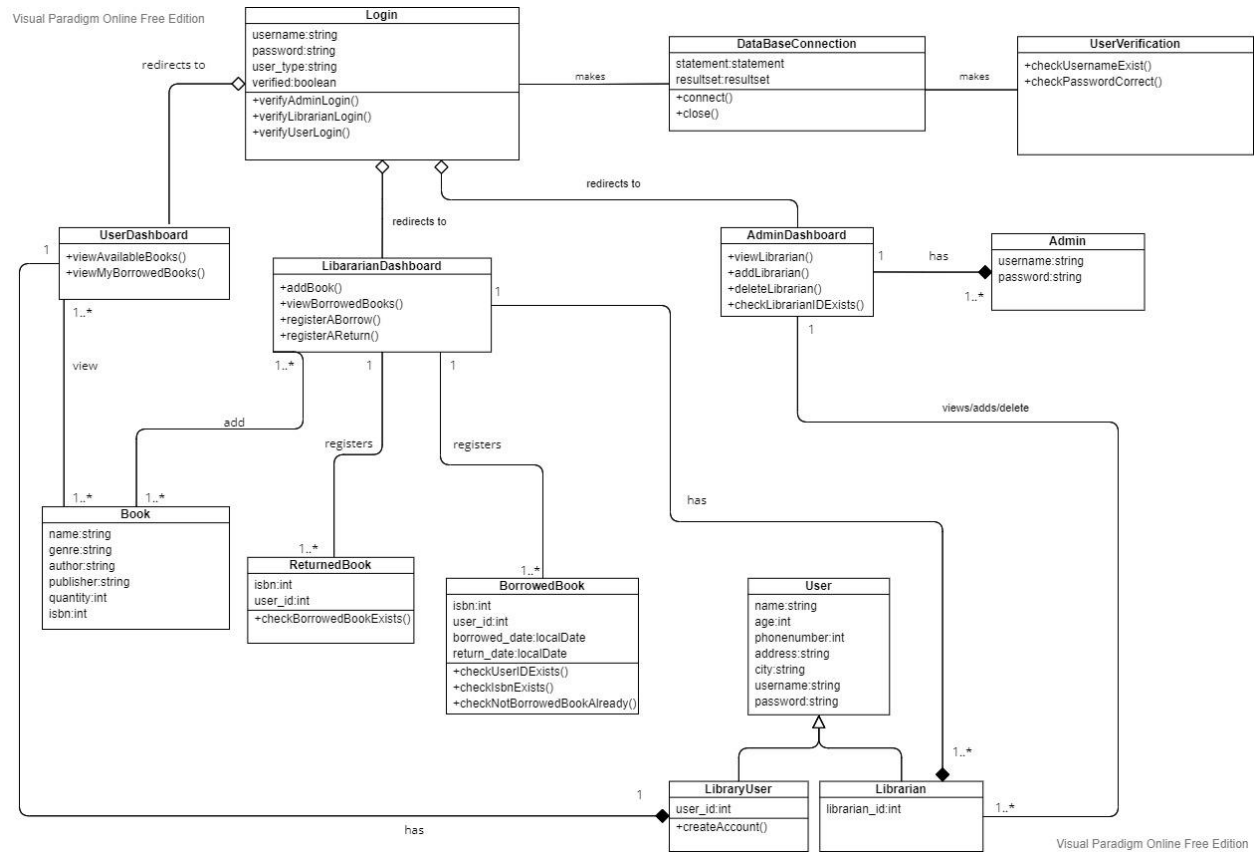
## 5) System Modeling

### a-Use case Diagram:



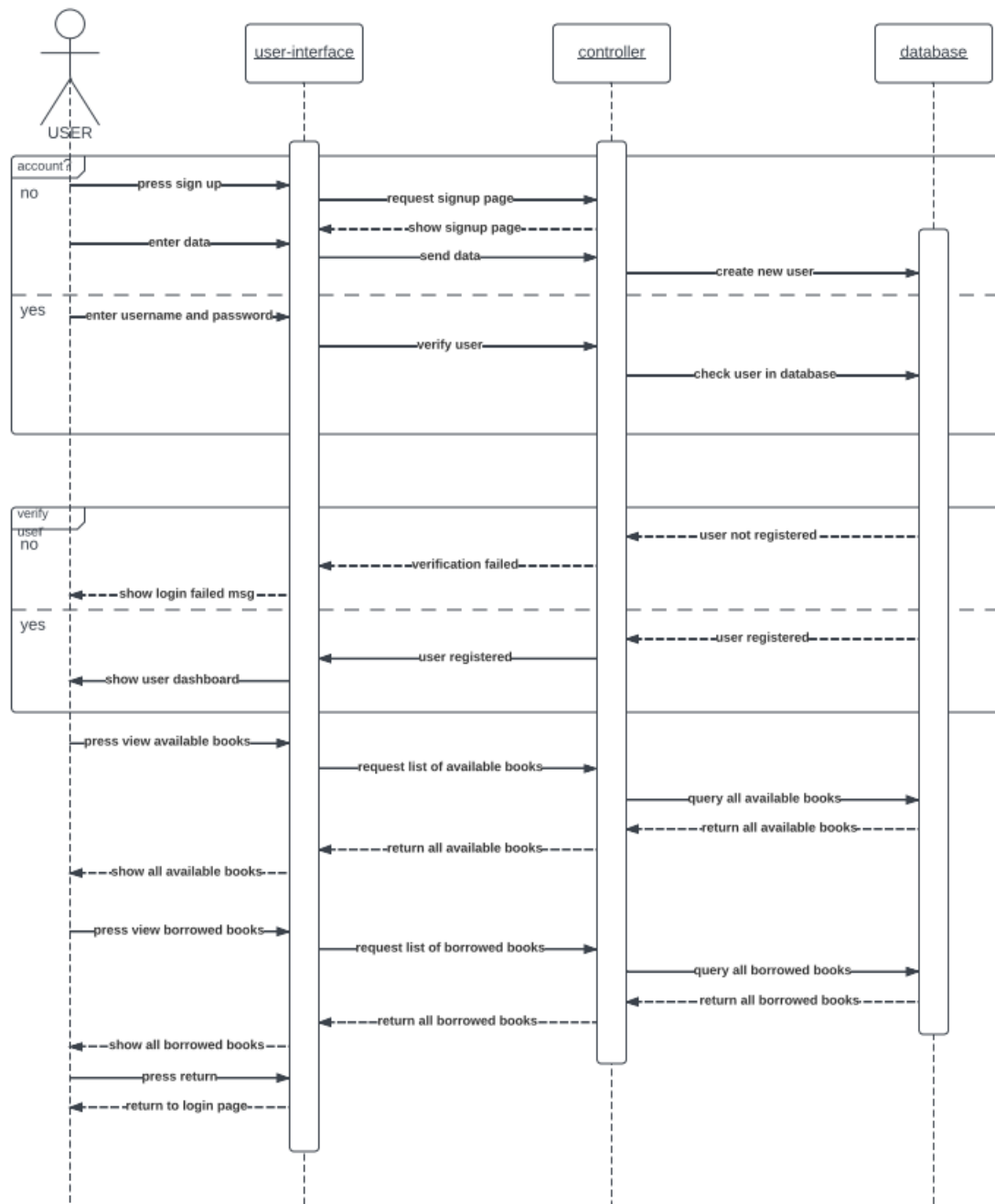


## b-Class Diagram:

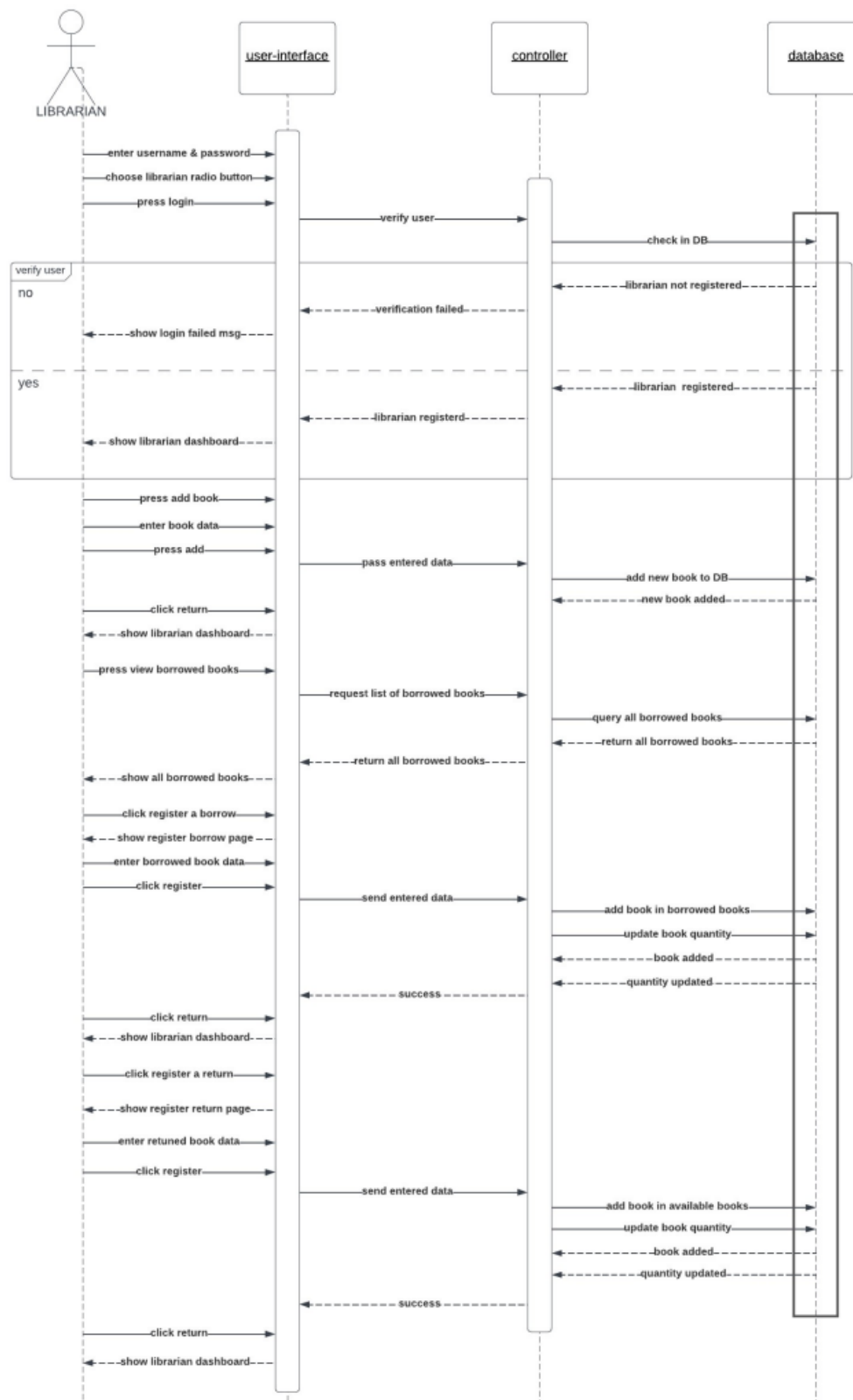


## c-Sequence Diagram:

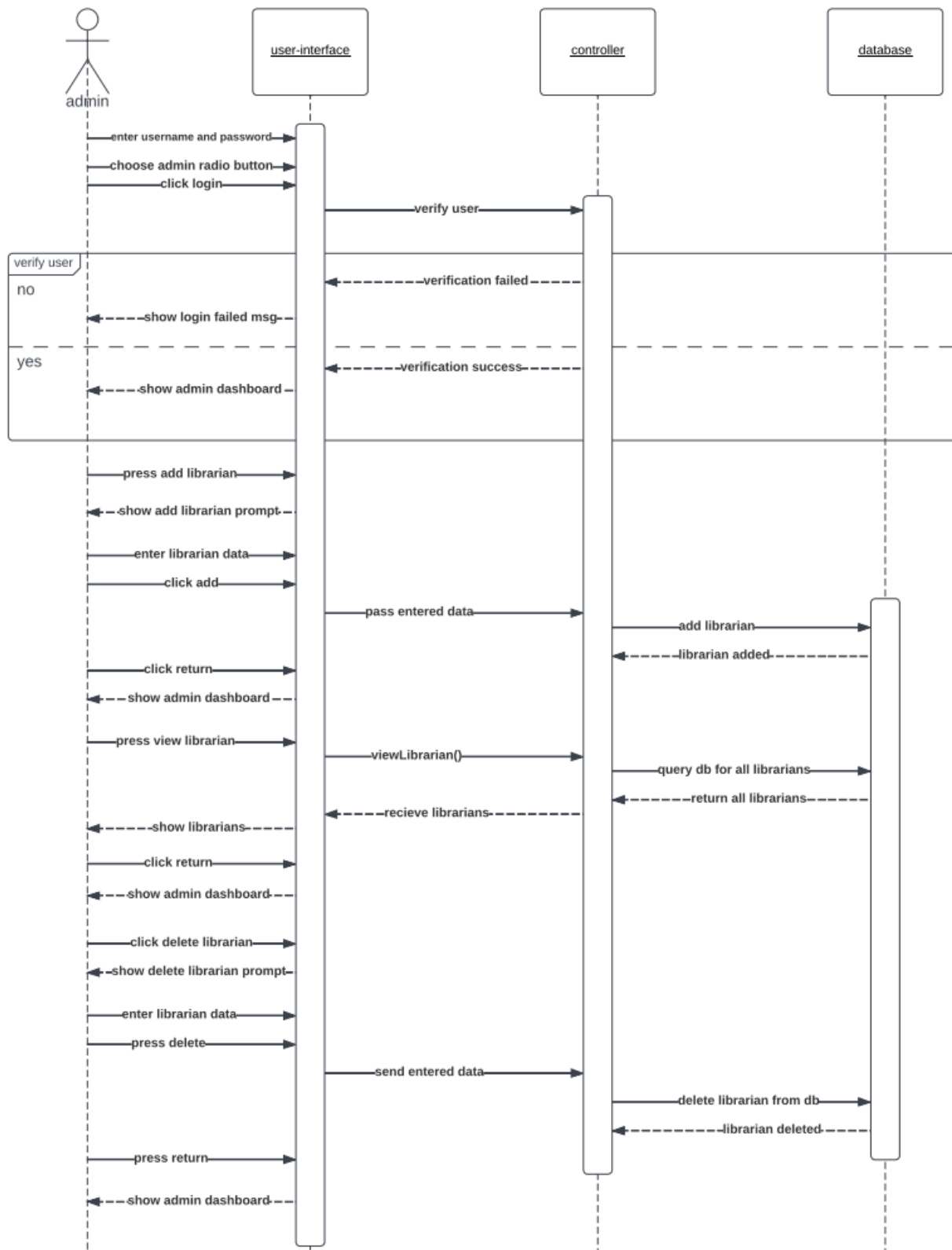
For library user:



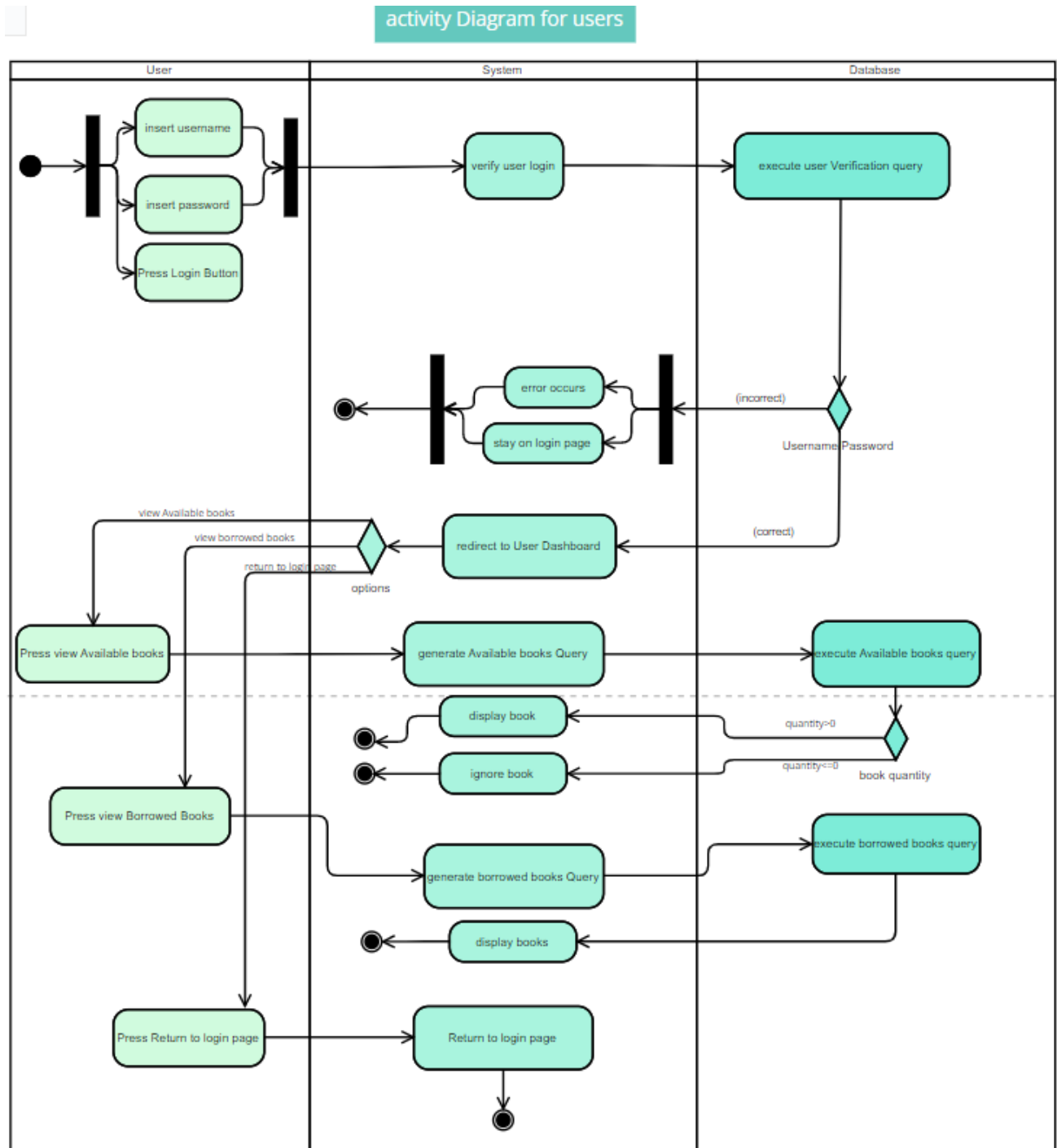
For Librarian:



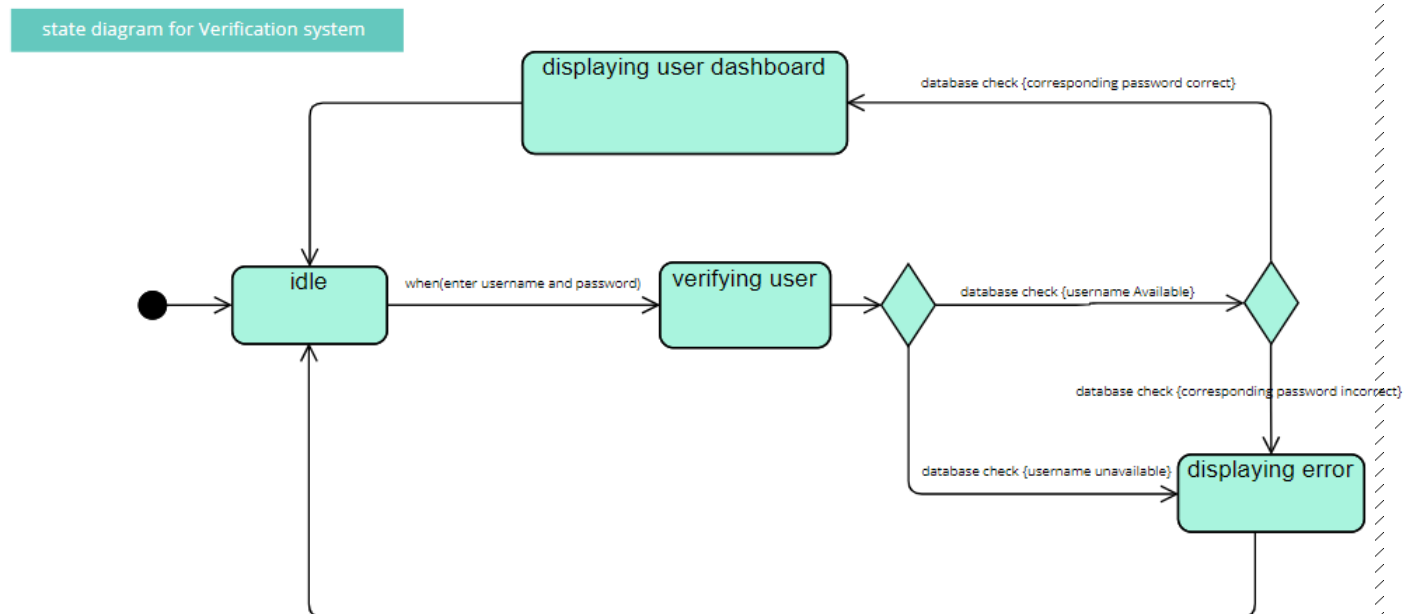
For Admin:



## d-Activity Diagram:



## e-State Machine Diagram:



## 6) Design & Implementation

### a-Design Description:

- system starts up with a login page where each user can login or create an account
- if login isn't successful the system shows an error message indicating what went wrong
- after successful login, each user is shown their assigned dashboard where they can then choose from the available options shown to them
- based on chosen option, user is shown a window corresponding to what they chose

-the system uses a client-server approach, having the user-interface work as the client and the database working as the server

## **b-Implementation (Development Environment & Coding):**

development environment: used IntelliJ IDE to develop the app

language: JAVA

database: MySQL

Frameworks: JavaFX

Code is divided into 3 packages:

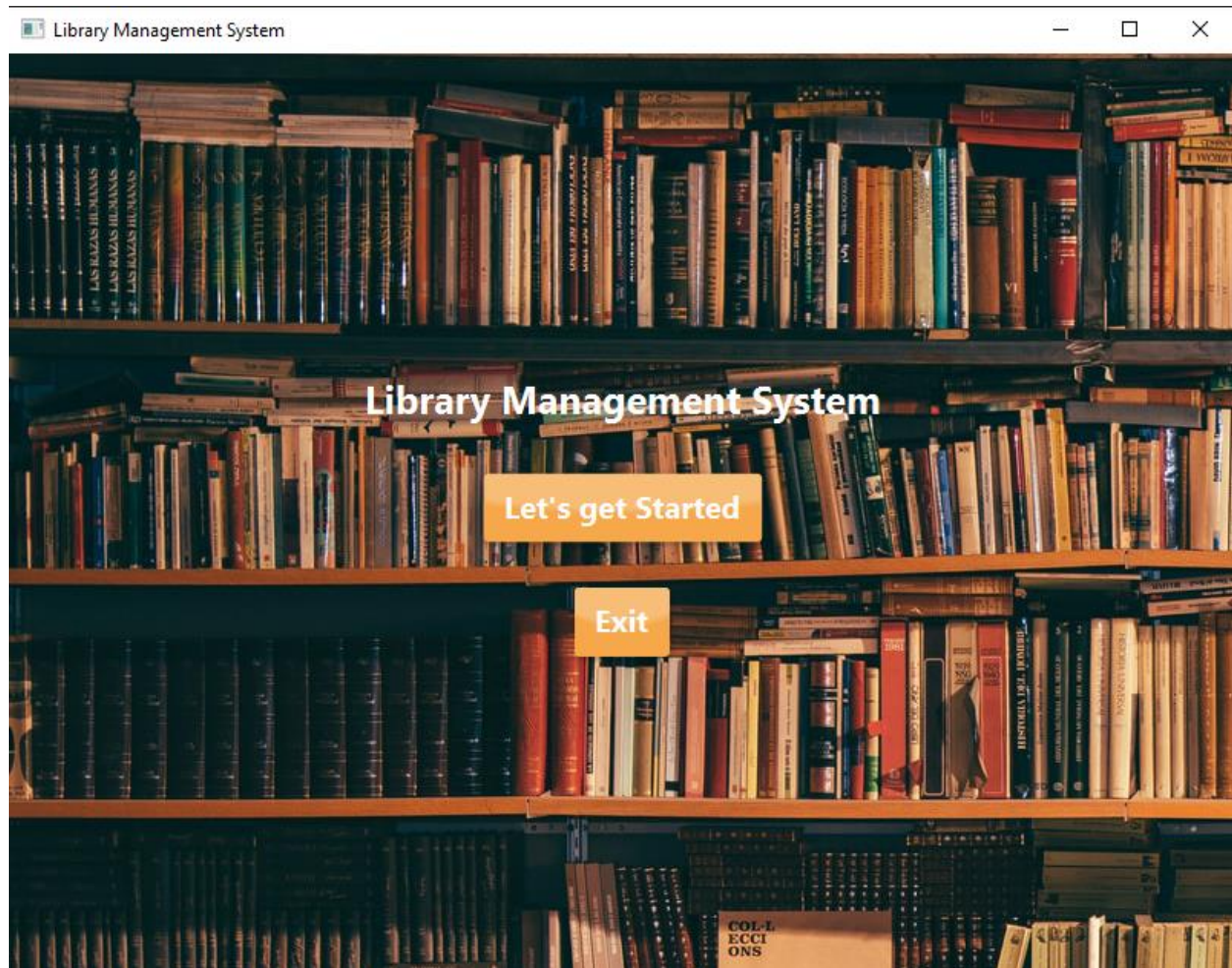
GUI: contains user-interface files

DB: contains DDL and dummy data, as well as the controllers that handle data

Tests: contains unit tests of systems functions

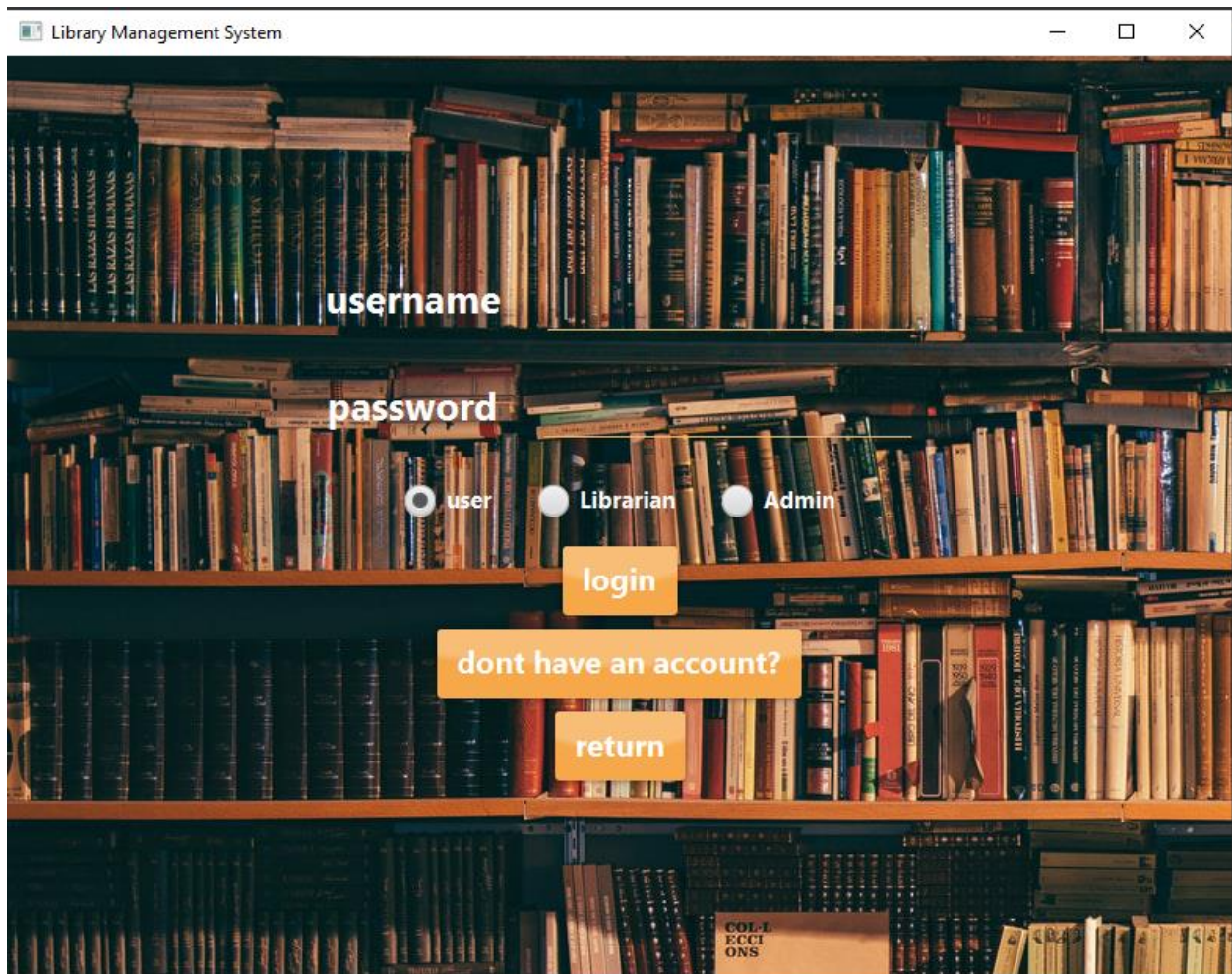
## Screenshots from the Program:

Welcoming Page:





## Login Page:



Library Management System

username

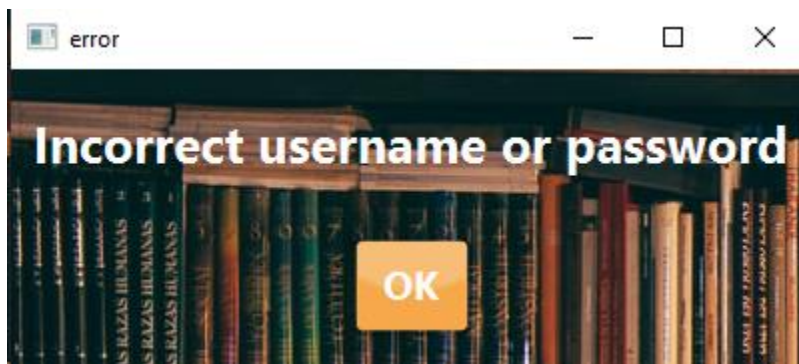
password

☐ user ☐ Librarian ☐ Admin

login

dont have an account?

return



error

Incorrect username or password

OK

Creating new account for user page:

create account

Name

Age

PhoneNumber

Address

city

username

password

Create

return

User Dashboard:

Available Books

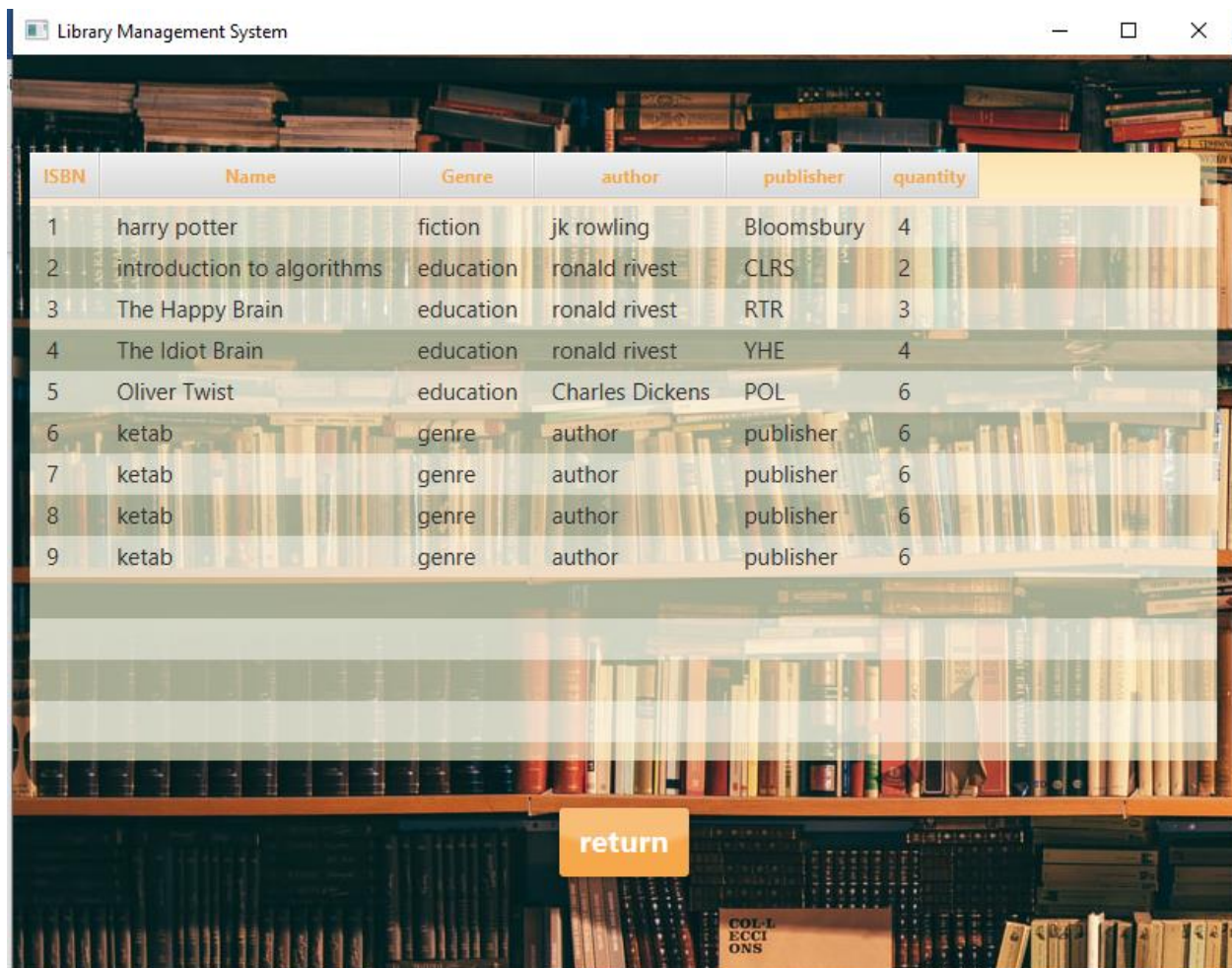
My Borrowed Books

return

COLLECTIONS



## Available Books page(user):

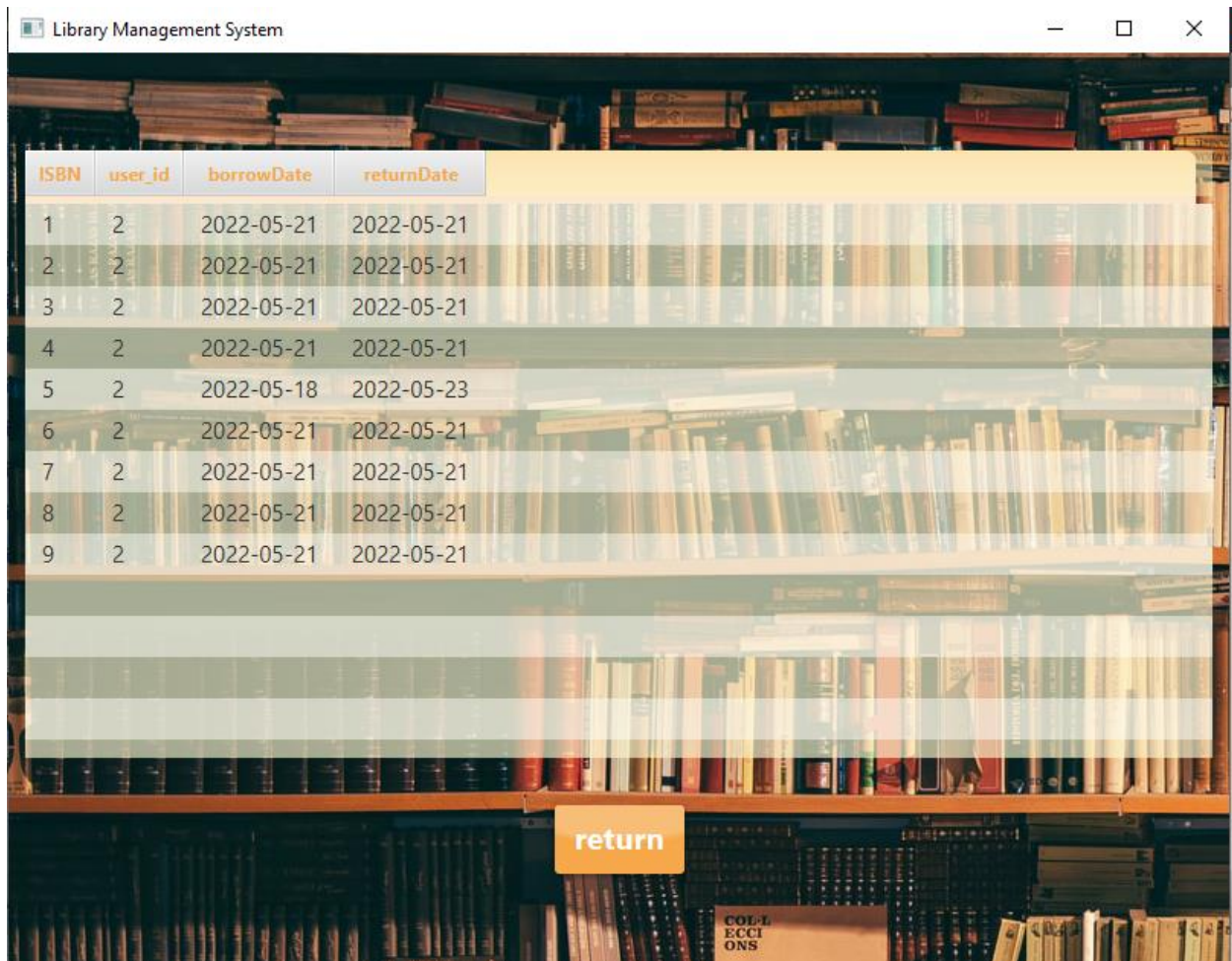


ISBN	Name	Genre	author	publisher	quantity
1	harry potter	fiction	jk rowling	Bloomsbury	4
2	introduction to algorithms	education	ronald rivest	CLRS	2
3	The Happy Brain	education	ronald rivest	RTR	3
4	The Idiot Brain	education	ronald rivest	YHE	4
5	Oliver Twist	education	Charles Dickens	POL	6
6	ketab	genre	author	publisher	6
7	ketab	genre	author	publisher	6
8	ketab	genre	author	publisher	6
9	ketab	genre	author	publisher	6

return

COLLECTIONS

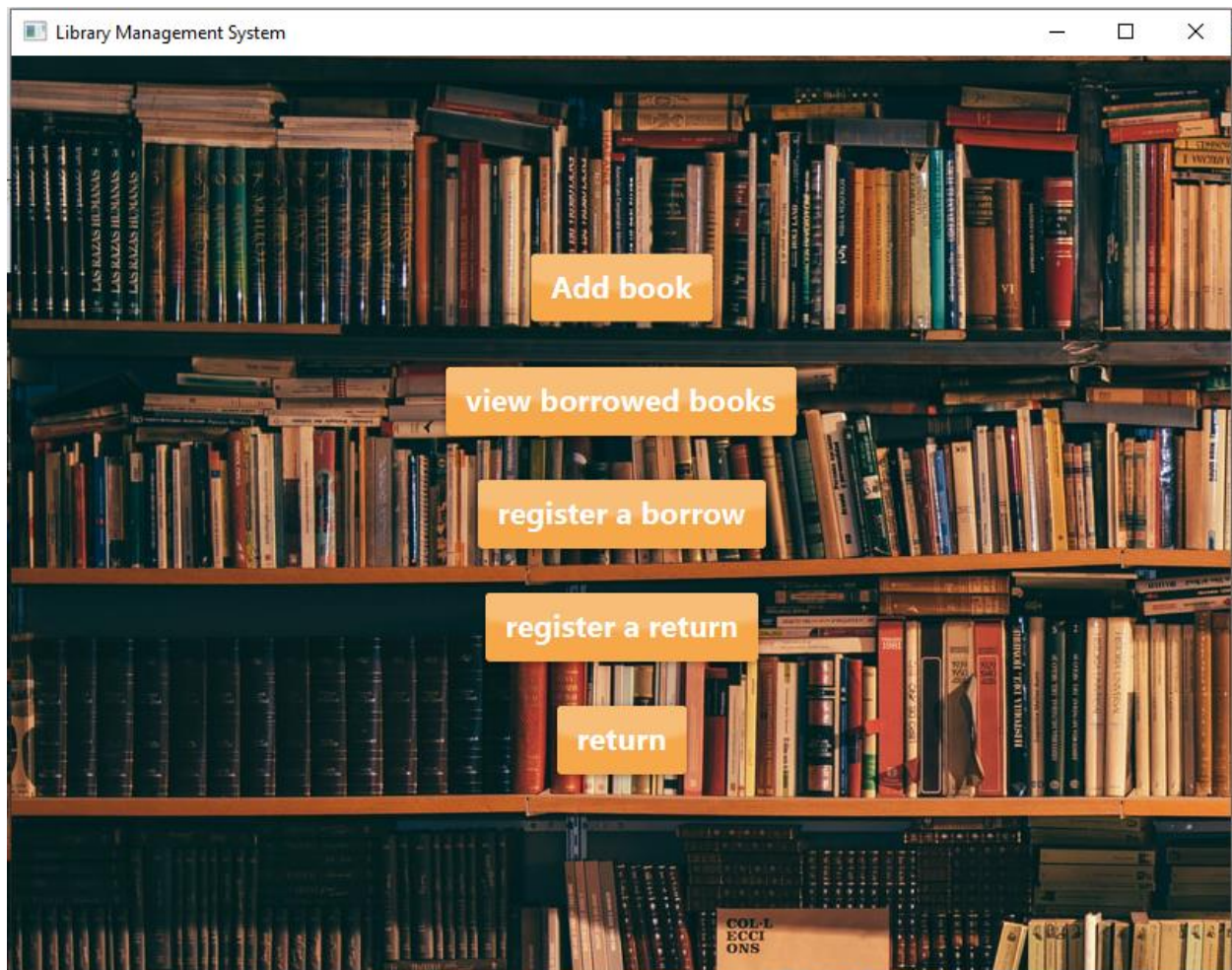
## My borrowed Books page(user):



ISBN	user_id	borrowDate	returnDate
1	2	2022-05-21	2022-05-21
2	2	2022-05-21	2022-05-21
3	2	2022-05-21	2022-05-21
4	2	2022-05-21	2022-05-21
5	2	2022-05-18	2022-05-23
6	2	2022-05-21	2022-05-21
7	2	2022-05-21	2022-05-21
8	2	2022-05-21	2022-05-21
9	2	2022-05-21	2022-05-21

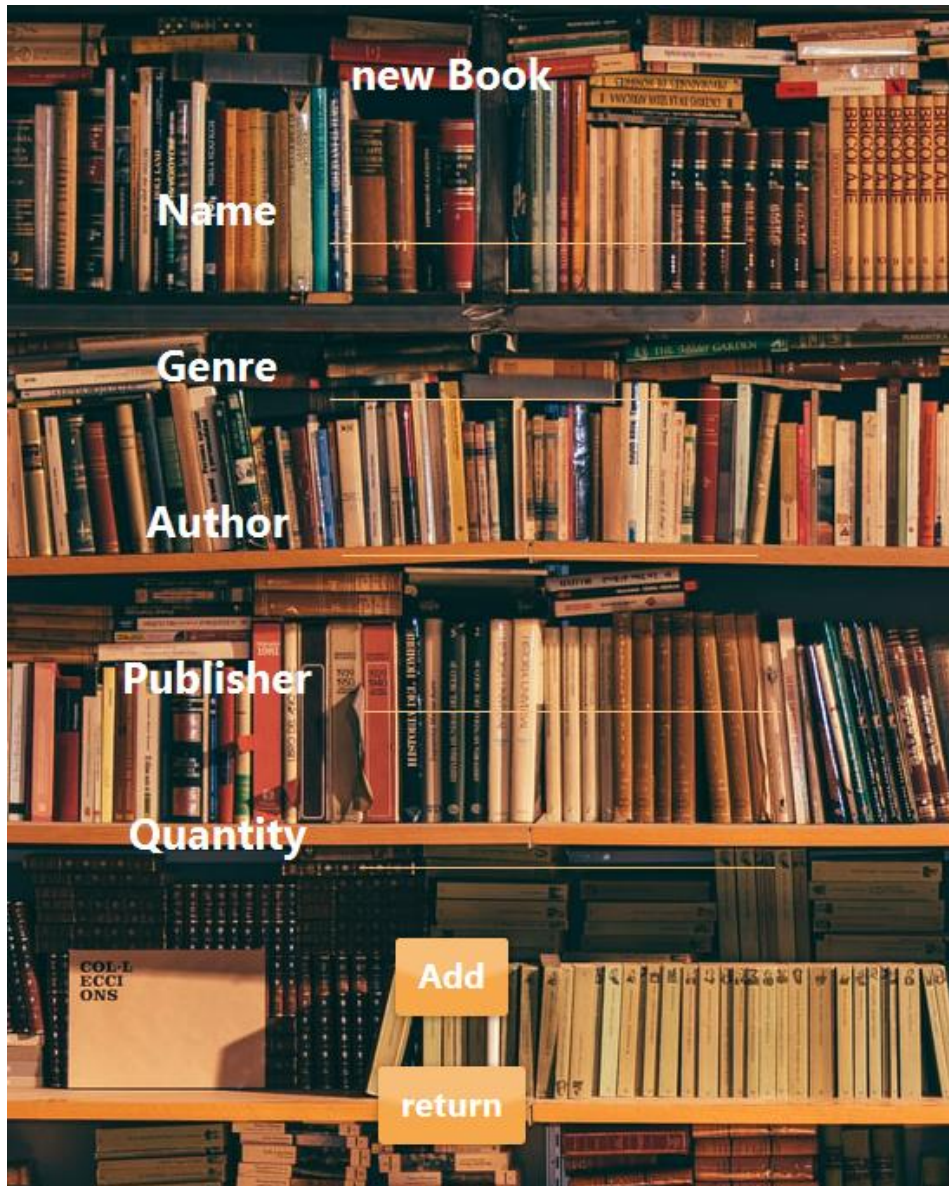
return

## Librarian Dashboard:





Add Book Page(librarian):



**new Book**

**Name**

**Genre**

**Author**

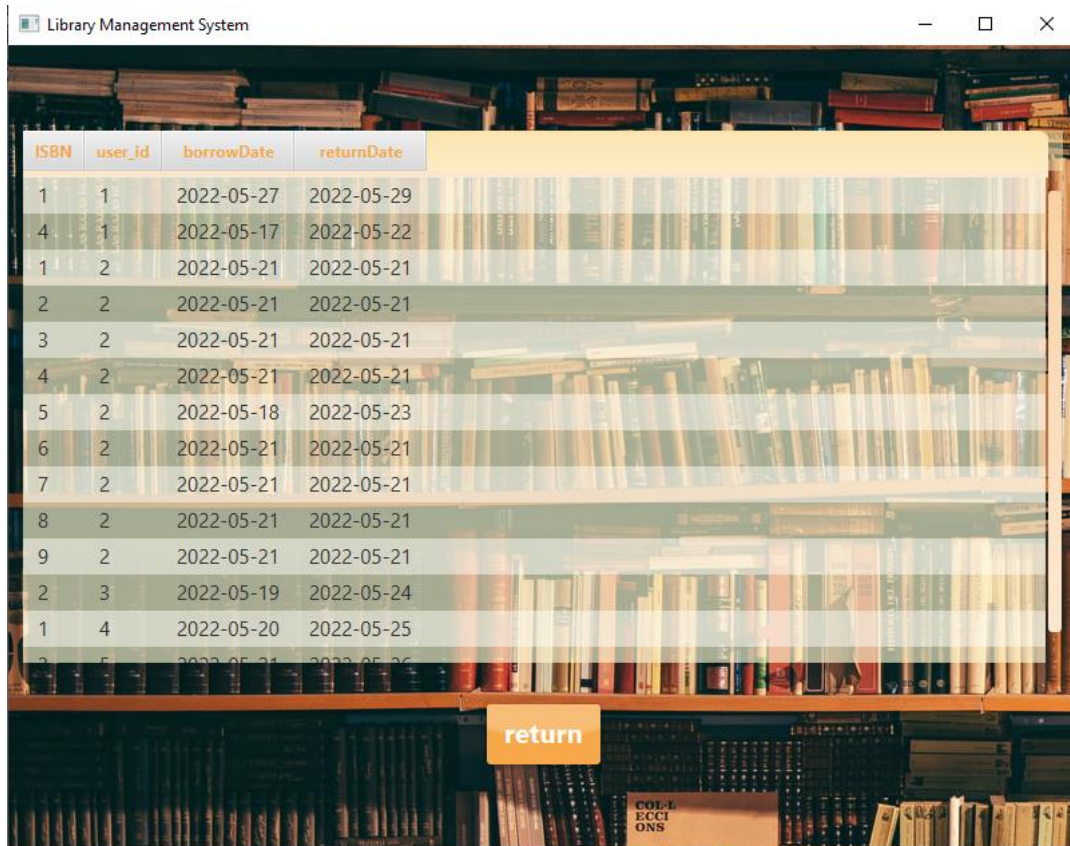
**Publisher**

**Quantity**

**Add**

**return**

## View Borrowed Books(librarian):



ISBN	user_id	borrowDate	returnDate
1	1	2022-05-27	2022-05-29
4	1	2022-05-17	2022-05-22
1	2	2022-05-21	2022-05-21
2	2	2022-05-21	2022-05-21
3	2	2022-05-21	2022-05-21
4	2	2022-05-21	2022-05-21
5	2	2022-05-18	2022-05-23
6	2	2022-05-21	2022-05-21
7	2	2022-05-21	2022-05-21
8	2	2022-05-21	2022-05-21
9	2	2022-05-21	2022-05-21
2	3	2022-05-19	2022-05-24
1	4	2022-05-20	2022-05-25
2	5	2022-05-21	2022-05-26

return



## Register a Borrow (librarian):

Library Management System

Register a borrow

user Id

book Id

borrow Date 5/21/2022

return Date 5/21/2022

register

return

May 2022

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

## Register a Return (librarian):

Register a return

user Id

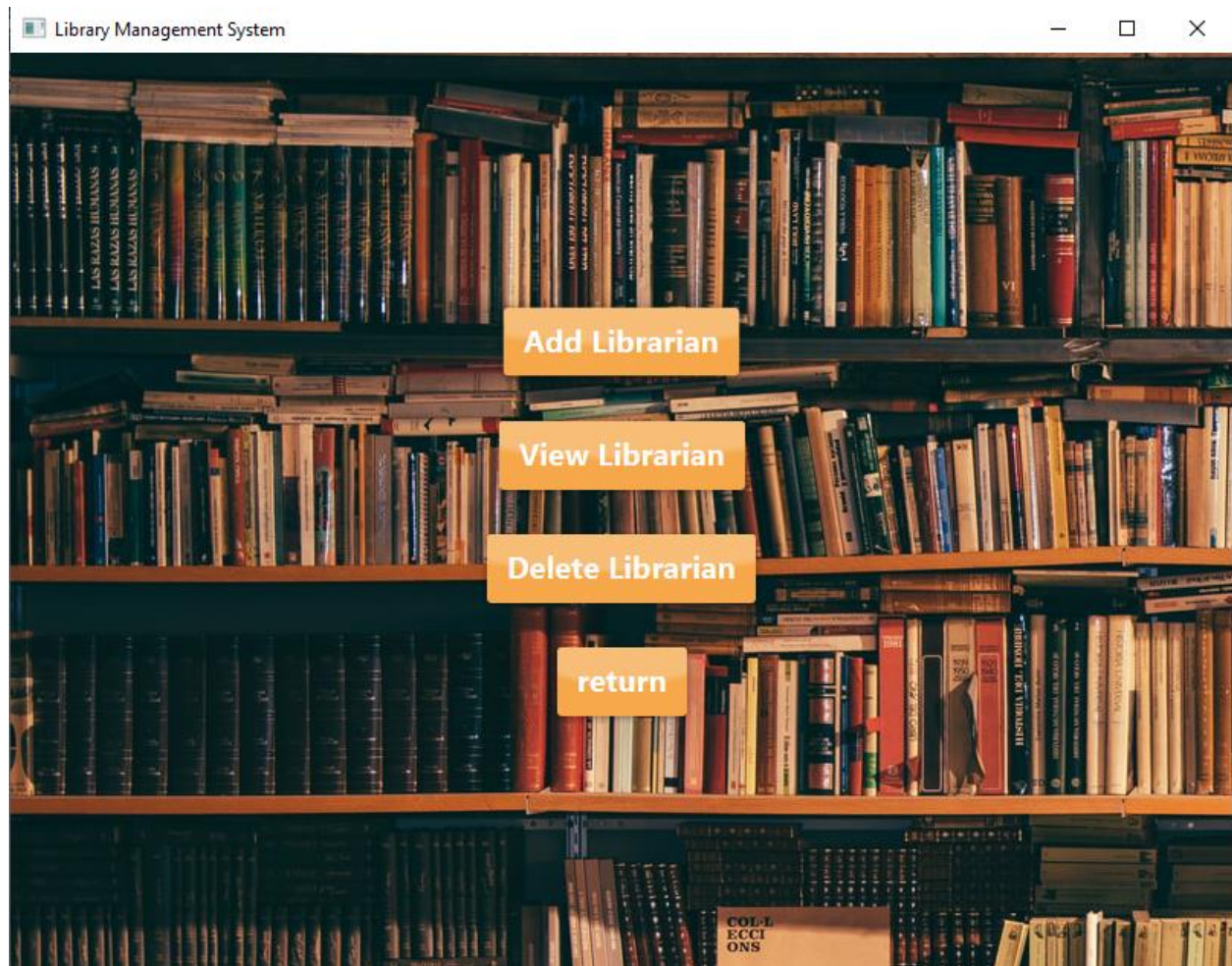
book Id

register

return



## Admin Dashboard:



Add Librarian (admin):

create account

Name

Age

PhoneNumber

Address

city

username

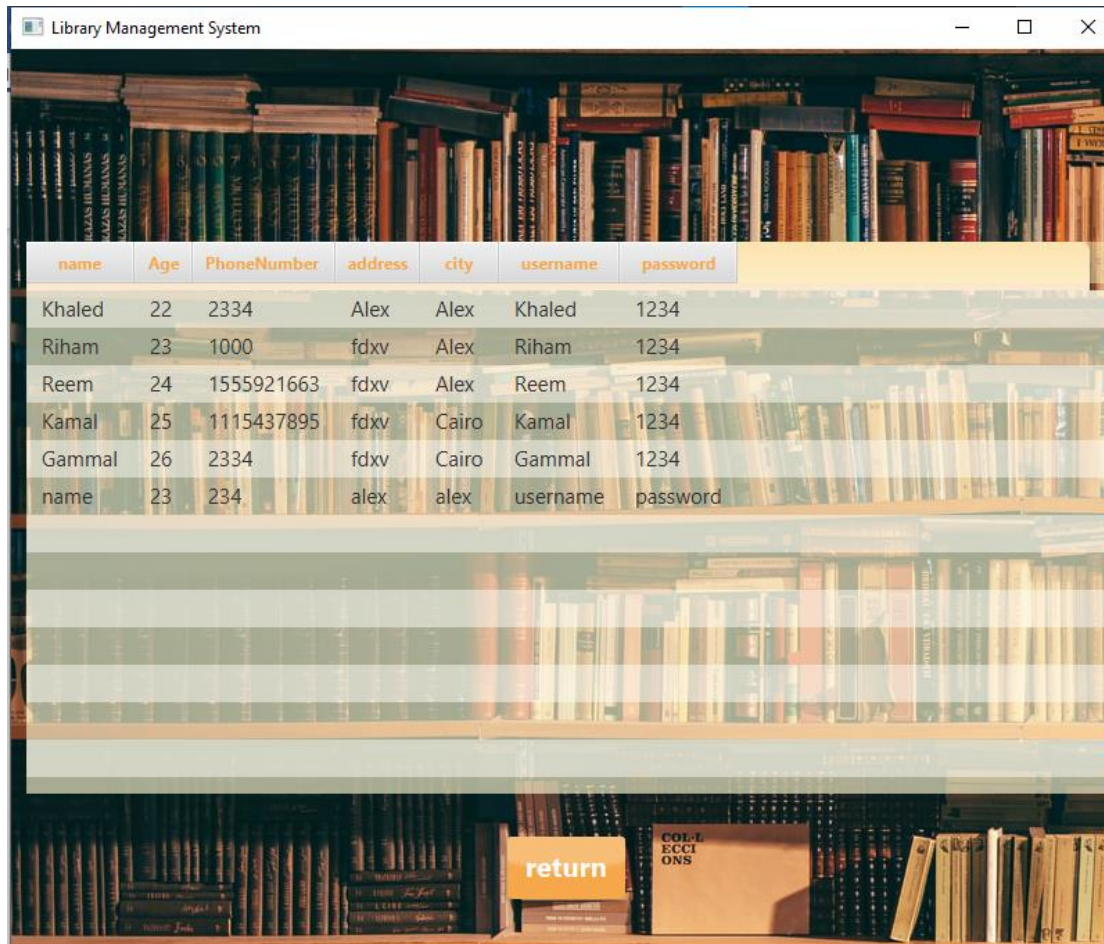
password

Create

return



view Librarian (admin):



The screenshot shows a window titled "Library Management System" with a background image of a library. Overlaid on the image is a table with the following data:

name	Age	PhoneNumber	address	city	username	password
Khaled	22	2334	Alex	Alex	Khaled	1234
Riham	23	1000	fdxv	Alex	Riham	1234
Reem	24	1555921663	fdxv	Alex	Reem	1234
Kamal	25	1115437895	fdxv	Cairo	Kamal	1234
Gammal	26	2334	fdxv	Cairo	Gammal	1234
name	23	234	alex	alex	username	password

Delete Librarian (admin):



The screenshot shows a form titled "Delete Librarian" with the text "enter Librarian username you want to delete:" followed by a text input field. Below the input field are two buttons: "Delete" and "return".

## 7) Tests:

Testing was done using JUNIT5 library

### 1-add user tests:

```
@Test
void verifyUserAge() throws SQLException {

    databaseConnection.connect();

    addUser.addUser( first_name: "name", age: 19, phone_number: 234, address: "alex", city: "alex", user_name: "username", password: "password", cases: 1);

    String testQuery= "SELECT * FROM user WHERE first_name='name'";

    ResultSet rs = databaseConnection.statement.executeQuery(testQuery);
    while(rs.next()){
        age=rs.getInt( columnLabel: "age");
    }
    assertTrue( condition: age>=14);

    databaseConnection.close();
}
```

```
@Test
void verifyLibrarianAge() throws SQLException {

    databaseConnection.connect();

    addUser.addUser( first_name: "name", age: 23, phone_number: 234, address: "alex", city: "alex", user_name: "username", password: "password", cases: 2);

    String testQuery= "SELECT * FROM librarian WHERE first_name='name'";

    ResultSet rs = databaseConnection.statement.executeQuery(testQuery);
    while(rs.next()){
        age=rs.getInt( columnLabel: "age");
    }
    assertTrue( condition: age>=21);

    databaseConnection.close();
}

@Test
void verifyUserPassword() throws SQLException{

    databaseConnection.connect();
    addUser.addUser( first_name: "name", age: 23, phone_number: 234, address: "alex", city: "alex", user_name: "username", password: "password", cases: 2);
    String testQuery= "SELECT * FROM user WHERE first_name='name'";

    ResultSet rs = databaseConnection.statement.executeQuery(testQuery);
    while(rs.next()){
        password=rs.getString( columnLabel: "password");
    }
    assertTrue( condition: password.length()>=8);

    databaseConnection.close();
}
```

✓ Test Results	1 s 714 ms
✓ AddUserTest	1 s 714 ms
✓ verifyUserPassword()	1 s 415 ms
✓ verifyLibrarianAge()	167 ms
✓ verifyUserAge()	132 ms

## 2- add book tests:

```

@Test
void testQueryCorrect() throws SQLException {
    DatabaseConnection databaseConnection = new DatabaseConnection();
    databaseConnection.connect();

    AddBook addBook = new AddBook();
    addBook.addbook( name: "ketab", genre: "genre", author: "author", publisher: "publisher", quantity: 6);
    String testQuery= "SELECT * FROM available_books WHERE book_name='ketab'";

    ResultSet rs = databaseConnection.statement.executeQuery(testQuery);
    while(rs.next()){
        name = rs.getString( columnLabel: "book_name");
        quantity=rs.getInt( columnLabel: "quantity");
    }
    assertEquals( expected: "ketab",name);
    assertTrue( condition: quantity>0);

    databaseConnection.close();
}

@Test
void testQuantityAboveZero() throws SQLException{
    DatabaseConnection databaseConnection = new DatabaseConnection();
    databaseConnection.connect();

    AddBook addBook = new AddBook();
    addBook.addbook( name: "ketab", genre: "genre", author: "author", publisher: "publisher", quantity: 6);
    String testQuery= "SELECT * FROM available_books WHERE book_name='ketab'";

    ResultSet rs = databaseConnection.statement.executeQuery(testQuery);
    while(rs.next()){
        quantity=rs.getInt( columnLabel: "quantity");
    }
    assertTrue( condition: quantity>0);
    databaseConnection.close();
}

```

✓ Test Results	772 ms
✓ AddBookTest	772 ms
✓ testQuantityAboveZero()	700 ms
✓ testQueryCorrect()	72 ms

### 3-delete librarian tests:

```

@Test
void deleteLibrarianTest() throws SQLException {
    databaseConnection.connect();
    deleteLibrarian.deleteLibrarian(username);

    String testQuery="SELECT * FROM librarian WHERE user_name='farah'";

    ResultSet rs = databaseConnection.statement.executeQuery(testQuery);
    while(rs.next()){
        username = rs.getString(columnLabel: "user_name");
        assertNull(username);
    }
}

```

✓ Test Results	664 ms
✓ DeleteLibrarianTest	664 ms
✓ deleteLibrarianTest()	664 ms

## EXTRA WORK:

- CSS file with an attractive appearance.
- good UX design that is easy for user to use and understand.
- animation of buttons are provided for attractive transitions.

You can find all diagrams used, mp4 video in the following link:

<https://drive.google.com/drive/folders/1Gh57fQq7k5T7eRPNj-5H9IKP3tXjCZwD?usp=sharing>