



CSE231: ADVANCED PROGRAMMING

Major Task

Mohamed Omar adel elBialy 21P0068
Hussein Serageldin Adel 21P0280
Salma Youssef 21P0148
Abdelrahman Sherif Abdelaziz 21P0098
Nouran Magdy Mohamed 2100688

Contents

Contents	1
1. Project Description	2
2. UML Diagram	3
3. Classes Functionality	4
3.1. User Class	4
3.1.1. Class Attributes	4
3.1.2. Class Constructor and Functions	5
3.2. Librarian Class	6
3.2.1. Class Constructor	6
3.3. Reader Class	7
3.3.1. Class Attributes	7
3.3.2. Class Constructor and Functions	7
3.4. Book Class	8
3.4.1. Class Attributes	8
3.4.2. Class Constructor and Functions	9
3.5. Library Class	10
3.5.1. Class Attributes	10
3.5.2. Class Constructors and Functions	10
3.6. App Class	14
3.6.1. Login and Start Function	14
3.6.2. Librarian Dashboard	17
3.6.3. Add Book	20
3.6.4. Remove Book	23
3.6.5. Search Book	26
3.6.6. Add User	30
3.6.7. Remove User	37
3.6.8. Search User	39
3.6.9. Block User	45
3.6.10. Reader Dashboard	51
3.6.11. Rent Book	54
3.6.12. View Order list	58

1. Project Description

The library contains tens of thousands of books and members which must be organized to prevent chaos. This software system allows the performance of actions needed to manage the library in a simple and comfortable way like addition/removal of books, addition/removal of members, member or book searches, and much more.

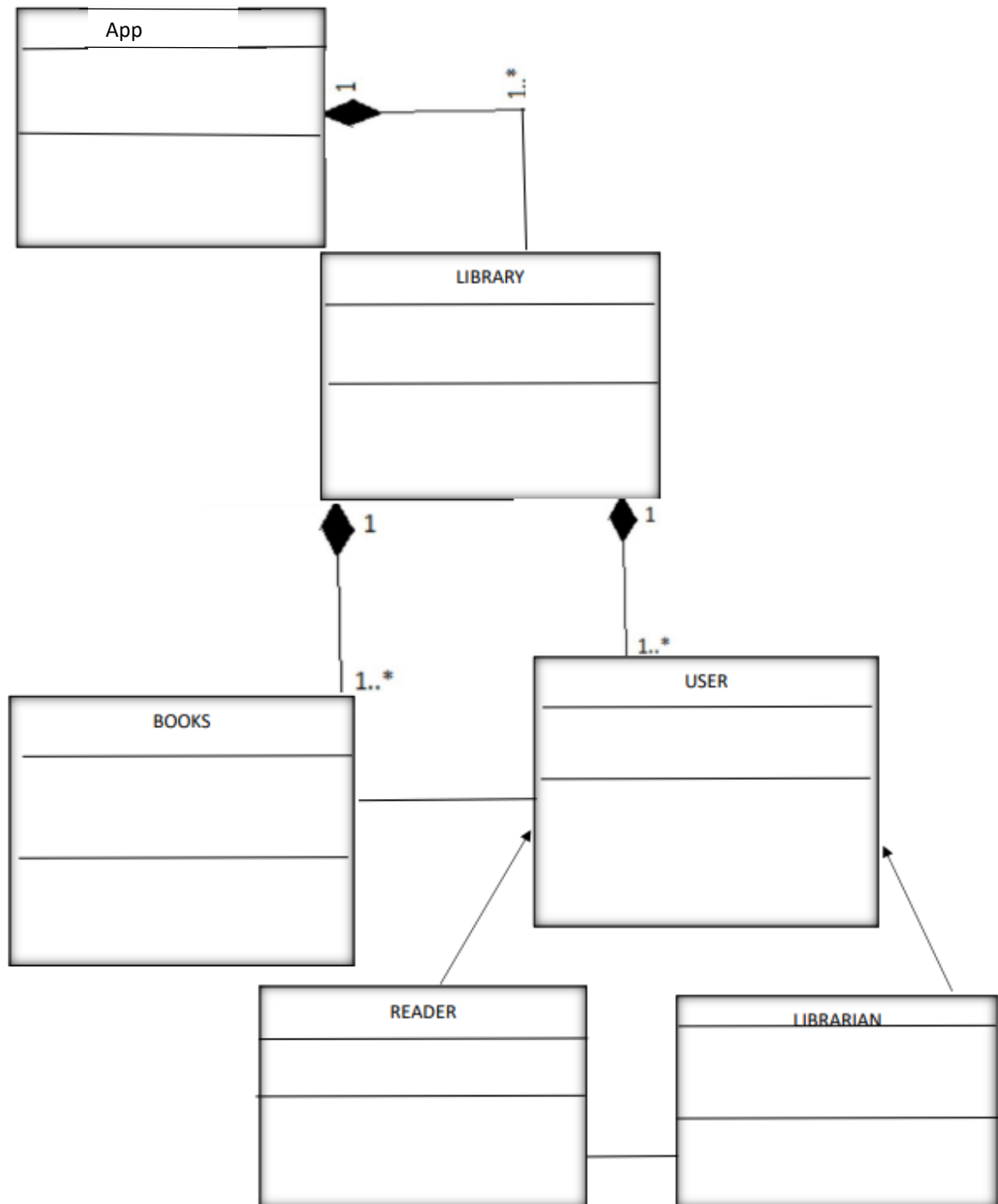
The system will support two different types of users: Librarians and Readers. For each user exists a unique ID and password which prevents users from accessing information which are not allowed to be accessed.

A librarian who can add, remove, or search for books or users, and block users who fail to return rented books after a specific time.

A Readers who can search for users or books and rent books that will be added to his order list.

2. UML Diagram

- Detailed UML Diagram is in the zip file



3. Classes Functionality

3.1. User Class

3.1.1. Class Attributes

- This is the Base Abstract User Class which contains all necessary information needed to define our user: ID, Password, Type, First Name ,Last Name ,Address, Cell Phone , Email , isBlocked.

```
abstract public class User {  
    4 usages  
    private String id;  
    2 usages  
    private String password;  
    4 usages  
    private String firstName;  
    4 usages  
    private String lastName;  
    4 usages  
    private String address;  
    3 usages  
    private String type;  
    4 usages  
    private String cellPhone;  
    5 usages  
    private String email;  
    5 usages  
    private boolean isBlocked;  
    2 usages  
    private static int count=0;
```

3.1.2. Class Constructor and Functions

3.1.2.1. Class Constructor

```
public User() {  
}  
2 usages  
public User(String id, String password, String firstName, String lastName, String address,  
            String cellPhone, String email, boolean isBlocked, String type) {  
    this.id = id;  
    this.password = password;  
    this.firstName = firstName;  
    this.lastName = lastName;  
    this.address = address;  
    this.cellPhone = cellPhone;  
    this.email = email;  
    this.isBlocked = isBlocked;  
    this.type = type;  
    count++;  
}
```

3.1.2.2. Class Methods

```
public String getId() { return id; }  
  
no usages  
public void setId(String id) { this.id = id; }  
  
2 usages  
public String getPassword() { return password; }  
  
4 usages  
public String getFirstName() { return firstName; }  
  
no usages  
public void setFirstName(String firstName) { this.firstName = firstName; }  
  
4 usages  
public String getLastName() { return lastName; }  
  
no usages  
public void setLastName(String lastName) { this.lastName = lastName; }  
  
2 usages  
public String getAddress() { return address; }  
  
no usages  
public void setAddress(String address) { this.address = address; }
```

2 usages

```
public String getCellPhone() { return cellPhone; }
```

no usages

```
public void setCellPhone(String cellPhone ) { this.cellPhone = cellPhone; }
```

2 usages

```
public String getEmail() { return email; }
```

no usages

```
public void setEmail() { this.email = email; }
```

4 usages

```
public boolean isBlocked() { return isBlocked; }
```

1 usage

```
public void setBlocked(boolean blocked) { isBlocked = blocked; }
```

9 usages

```
public String getType() { return type; }
```

no usages

```
public void setType(String type) { this.type = type; }
```

no usages

```
private static int getCount() { return count; }
```

3.2. Librarian Class

- This class extends the User Class

3.2.1. Class Constructor

- Super is call the attributes in Users, No functions needed to manipulate attributes.

21 usages

```
public class Librarian extends User {
```

2 usages

```
public Librarian(String id, String password, String firstName, String lastName,
```

```
String address, String cellPhone, String email, boolean isBlocked, String type) {
```

```
super(id, password, firstName, lastName, address, cellPhone, email, isBlocked, type: "Librarian");
```

```
}
```

```
}
```

3.3. Reader Class

- This class extends the User class.

3.3.1. Class Attributes

- In addition to the inherited attributes, each reader has their own order list, which will contain all rented books.

```
30 usages
public class Reader extends User {
    4 usages
    private ArrayList<Book> orderList;
```

3.3.2. Class Constructor and Functions

3.3.2.1. Class Constructor

- In addition to the inherited attributes, an order List is created for each Reader

```
4 usages
public Reader(String id, String password, String firstName, String lastName, String address,
               String cellPhone, String email, boolean isBlocked, String type) {
    super(id, password, firstName, lastName, address, cellPhone, email, isBlocked, type: "Reader");
    orderList = new ArrayList<Book>();
}
```


3.3.2.2. Class Methods

```
public ArrayList<Book> getOrderList() {  
    return orderList;  
}  
  
1 usage  
public void addToOrderList(Book book) {  
    orderList.add(book);  
}  
  
no usages  
public void setOrderList(ArrayList<Book> orderList) { this.orderList = orderList; }  
}
```

3.4. Book Class

3.4.1. Class Attributes

```
public class Book {  
    3 usages  
    private String title;  
    3 usages  
    private String Author;  
  
    3 usages  
    private String Serial;  
    3 usages  
    private boolean is_available;  
    2 usages  
    private static int count = 0;  
    3 usages  
    private Date date;  
    2 usages  
    private Date Rentdate;  
}
```

3.4.2. Class Constructor and Functions

3.4.2.1. Class Constructor

```
public Book(String title , String Author , String Serial) {  
    this.title = title;  
    this.Author = Author;  
    this.Serial = Serial;  
    count++;  
    this.is_available = true;  
    this.date = new Date();  
}
```

3.4.2.2. Class Methods

```
public void setSerial(String serial) { Serial = serial; }  
  
no usages  
public void setAuthor(String author) { Author = author; }  
  
4 usages  
public String getTitle() { return title; }  
  
2 usages  
public boolean Is_available() { return is_available; }  
  
2 usages  
public void setIs_available(boolean is_available) { this.is_available = is_available; }  
  
no usages  
public Date getDate() { return date; }  
  
no usages  
public void setDate(Date date) { this.date = date; }  
  
2 usages  
public Date getRentdate() { return Rentdate; }  
  
1 usage  
public void setRentdate(Date rentdate) { Rentdate = rentdate; }
```

```
@Override
public String toString() { return "Book{" + "title=" + title + '}'; }
```

3.5. Library Class

- In this class , we combine the User , Reader , Librarian , and Book classes to perform several library functionalities

3.5.1. Class Attributes

- Contains all the classes created.

```
public class Library {
    5 usages
    ArrayList<User> users = new ArrayList<User>();
    3 usages
    ArrayList<Book> books = new ArrayList<Book>();
    3 usages
    ArrayList<Reader> readers = new ArrayList<Reader>();
    3 usages
    ArrayList<Librarian> librarians = new ArrayList<Librarian>();
}
```

3.5.2. Class Constructors and Functions

3.5.2.1. Class Constructor

- Librarian ID: 12345
- Librarian Password: 12345
- Reader ID: 21345
- Reader Password: 21345

```

public Library() {

    Librarian librarian1 = new Librarian( id: "12345", password: "12345", firstName: "Anthony", lastName: "Davies",
        address: "23rd Wall Street", cellPhone: "+201011155234", email: "anthony@gmail.com",
        isBlocked: false, type: "librarian");

    Reader reader1 = new Reader( id: "21345", password: "21345", firstName: "Luka",
        lastName: "Modric", address: "22nd Wall Street", cellPhone: "+01011477682",
        email: "Modric@gmail.com", isBlocked: false, type: "reader");

    Book book1 = new Book( title: "Harry Potter", Author: "JK Rowling", Serial: "12345");

    librarians.add(librarian1);

    readers.add(reader1);

    books.add(book1);

    users.add(librarian1);

    users.add(reader1);

}

```

3.5.2.2. Class Methods

```

public User getUserType(String id, String password, ArrayList<User> users) {
    Iterator<User> iterator = users.iterator();
    while (iterator.hasNext()) {
        User user = iterator.next();
        if (user.getId().equals(id) && user.getPassword().equals(password)) {
            if (user.getType().equalsIgnoreCase( anotherString: "Librarian")
                || user.getType().equalsIgnoreCase( anotherString: "Reader"))
                return user;
        }
    }
    return null;
}

```

1 usage

```

public Boolean CompareDataFields(String DATA, ArrayList<User> users) {
    Iterator<User> iterator = users.iterator();
    while (iterator.hasNext()) {
        User user1 = iterator.next();
        if (user1.getId().equals(DATA)) {
            return true;
        }
    }
    return false;
}

```

```

public Boolean CompareDataFieldsPassword(String DATA, ArrayList<User> users) {
    Iterator<User> iterator = users.iterator();
    while (iterator.hasNext()) {
        User user1 = iterator.next();
        if (user1.getPassword().equals(DATA)) {
            return true;
        }
    }
    return false;
}
1 usage

```

```

public Book Block(Reader reader) {

    if (reader.isBlocked() == false) {
        ArrayList<Book> OrderList = reader.getOrderList();
        if (OrderList.isEmpty()) {
            System.out.println("Order list is empty");
        } else {
            for (int i = 0; i < OrderList.size(); i++) {
                Book book = OrderList.get(i);
                //if user has the book for 5 days
                if (book.getRentdate().getDay() >= 5) {
                    return book;
                }
            }
        }
    }
    return null;
}

```

```

public Book SearchBook(String Serial, ArrayList<Book> books) {
    Iterator<Book> iterator = books.iterator();
    while (iterator.hasNext()) {
        Book book = iterator.next();
        if (book.getSerial().equals(Serial)) {
            return book;
        }
    }
    return null;
}

1 usage
public Book BookRemoval(String Title, String isbn, ArrayList<Book> books) {
    Iterator<Book> iterator = books.iterator();
    while (iterator.hasNext()) {
        Book book = iterator.next();
        if (book.getTitle().equals(Title) && book.getSerial().equals(isbn)) {
            return book;
        }
    }
    return null;
}

```

```

public User SearchU(String ID, ArrayList<User> users) {
    Iterator<User> iterator = users.iterator();
    while (iterator.hasNext()) {
        User user = iterator.next();
        if (user.getId().equals(ID)) {
            return user;
        }
    }
    return null;
}

```

```

1 usage
public boolean EmailCheck(String Email) {
    if (Email.contains("@gmail.com"))
        return true;
    if (Email.contains("@hotmail.com"))
        return true;
    if (Email.contains("@yahoo.com"))
        return true;
    return false;
}

```

3.6. App Class

- This is where our app launches.
- It has all the functionality defined .

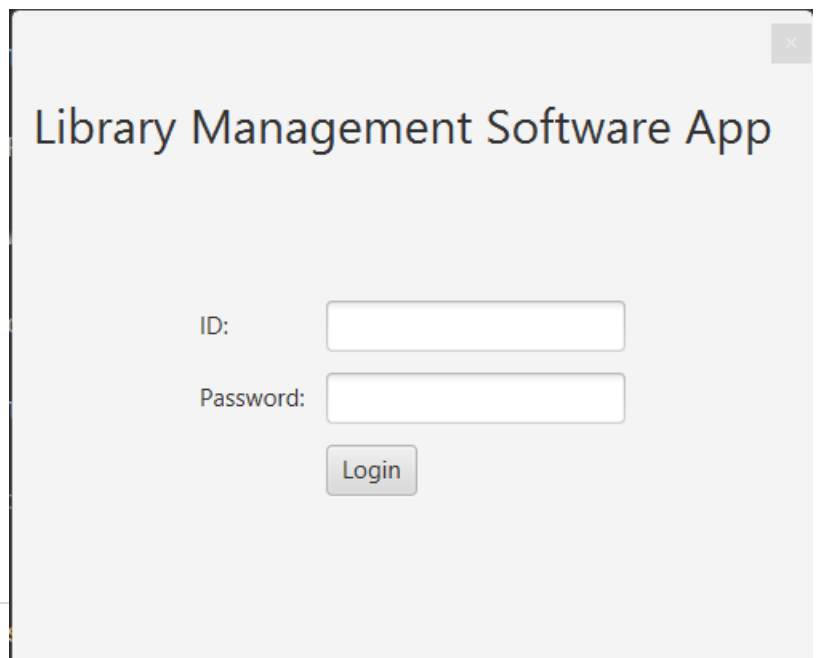
3.6.1. Login and Start Function

- Our Desktop Application opens up with this login page

As stated before:

Librarian ID is 12345 and pass is 12345

Reader is 21345 and pass is 21345



Library Management Software App

ID:

Password:

Login

- To prevent ambiguous access

By invalidating any wrong passwords or ID.

Library Management Software App

ID:

Password:

Invalid ID or password

```
public void start(Stage primaryStage) throws Exception
{
    // Initialize stage and root pane
    stage = primaryStage;
    root = new BorderPane();

    stage.initStyle(StageStyle.UTILITY);

    // Initialize UI components for login scene
    loginLabel = new Label( s: "Library Management Software App");
    loginLabel.setStyle("-fx-font-size: 24px; -fx-padding: 10px;");
    idLabel = new Label( s: "ID:");
    passwordLabel = new Label( s: "Password:");
    idField = new TextField();
    passwordField = new PasswordField();
    loginButton = new Button( s: "Login");
    messageLabel = new Label();

    // Initialize event handler for login button
    loginButton.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            messageLabel.setText("");
            String id = idField.getText();
            String password = passwordField.getText();
            User userType = library.getUserType(id, password, library.users);
            if (userType != null) {
```



```

        if (userType != null) {
            if (userType.getType().equals("Librarian")) {
                showLibrarianDashboard(userType, library.users, library.readers,
                    library.books, library.librarians);
            } else if (userType.getType().equals("Reader")) {
                showReaderDashboard(userType, library.users, library.readers,
                    library.books, library.librarians);
            }
        } else {
            messageLabel.setText("Invalid ID or password");
        }
    }
});

```

```

// Initialize UI components for login form
GridPane loginForm = new GridPane();
loginForm.setHgap(10);
loginForm.setVgap(10);
loginForm.setPadding(new Insets(20));
loginForm.setAlignment(Pos.CENTER);
loginForm.add(idLabel, 0, 0);
loginForm.add(idField, 1, 0);
loginForm.add(passwordLabel, 0, 1);
loginForm.add(passwordField, 1, 1);
loginForm.add(loginButton, 1, 2);
loginForm.add(messageLabel, 1, 3);

```

```

// Add login label and form to login scene
loginScene = new Scene(root, 400, 300);
root.setTop(loginLabel);
root.setCenter(loginForm);

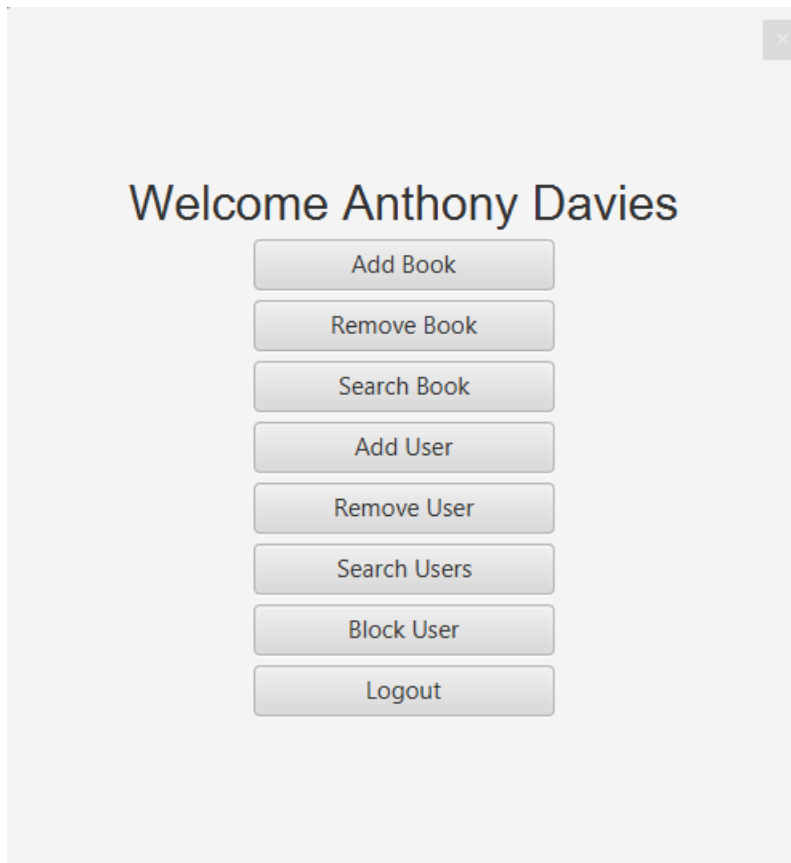
// Set the scene for the stage and show it

stage.setScene(loginScene);

stage.show();
}

```

3.6.2. Librarian Dashboard

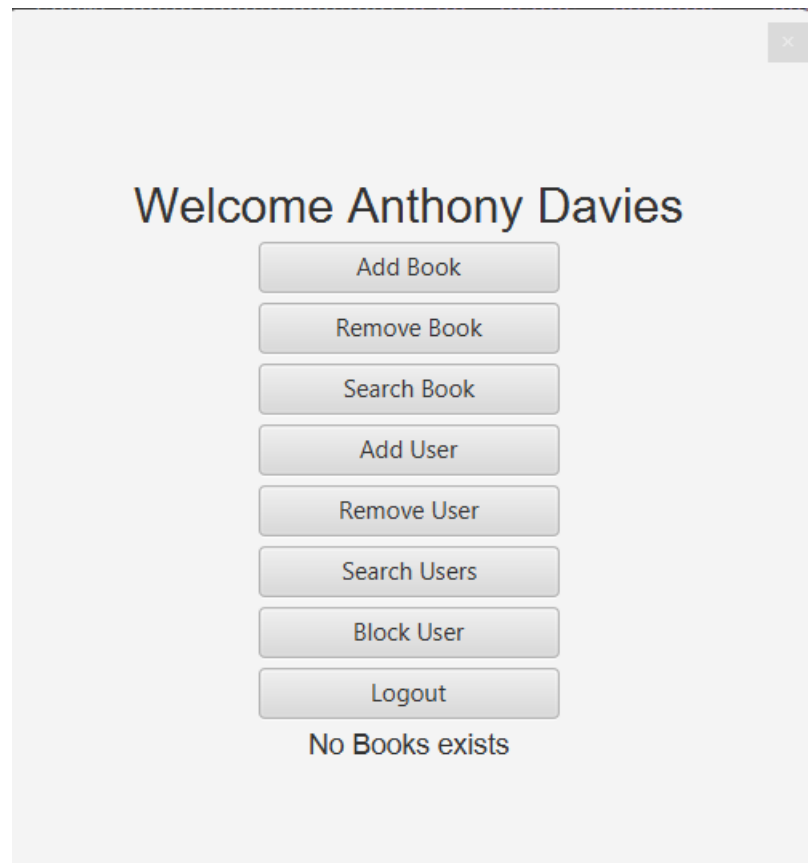


Buttons:-

- Add Book
- Remove Book
- Search Book
- Add User
- Remove User
- Search User
- Block User
- Logout

If 0 Books exists in the library.

Code :-



```
private void showLibrarianDashboard(User user, ArrayList<User> users, ArrayList<Reader> readers,
                                    ArrayList<Book> books, ArrayList<Librarian> librarians)
{
    // Initialize UI components for librarian dashboard
    Label librarianLabel = new Label( s: "Welcome " + user.getFirstName() + " " + user.getLastName());
    Button addButton = new Button( s: "Add Book");
    Button removeButton = new Button( s: "Remove Book");
    Button SearchBook = new Button( s: "Search Book");
    Button SearchUsers = new Button( s: "Search Users");
    Button RemoveUser = new Button( s: "Remove User");
    Button AddUser = new Button( s: "Add User");
    Button BlockUser = new Button( s: "Block User");
    Label lbl = new Label();

    Button logoutButton = new Button( s: "Logout");
    // Set UI component properties
    librarianLabel.setFont(new Font( s: "Arial", v: 24));
    addButton.setPrefWidth(150);
    removeButton.setPrefWidth(150);
    SearchBook.setPrefWidth(150);
    SearchUsers.setPrefWidth(150);
    RemoveUser.setPrefWidth(150);
    AddUser.setPrefWidth(150);
    BlockUser.setPrefWidth(150);
}
```

```

lbl.setFont(new Font(s: "Arial", v: 14));

logoutButton.setPrefWidth(150);

// Create layout for librarian dashboard
VBox librarianBox = new VBox(v: 5, librarianLabel, addButton, removeButton, SearchBook,
    AddUser, RemoveUser, SearchUsers,
    BlockUser, logoutButton, lbl);
librarianBox.setAlignment(Pos.CENTER);

// Set event handlers for buttons
addButton.setOnAction(e -> {
    // Show add book scene
    showAddBookScene(user, users, readers, books, librarians);
});

removeButton.setOnAction(e -> {
    // Show remove book scene
    showRemoveBookScene(user, users, readers, books, librarians);
});
SearchBook.setOnAction(e -> {
    if(books.size() == 0)
    {
        lbl.setText("No Books exists");
    }
    else
        ShowSearchBook(user, users, readers, books, librarians);
});

```

```

        ShowSearchBook(user, users, readers, books, librarians);
    });

SearchUsers.setOnAction(e -> {
    showSearchUserScene(user, users, readers, books, librarians);
});
RemoveUser.setOnAction(e -> {
    ShowRemoveUserScene(user, users, readers, books, librarians);
});
BlockUser.setOnAction(e -> {
    showBlockUserScene(user, users, readers, books, librarians);
});
AddUser.setOnAction(e -> {
    ShowAddUser(user, users, readers, books, librarians);
});

logoutButton.setOnAction(e -> {
    // Show login scene
    stage.setScene(loginScene);
    stage.show();
});

// Set librarian dashboard as the root of the scene
Scene librarianScene = new Scene(librarianBox, 400, 400);

stage.setScene(librarianScene);

```

3.6.3. Add Book

×

Add Book

Title:

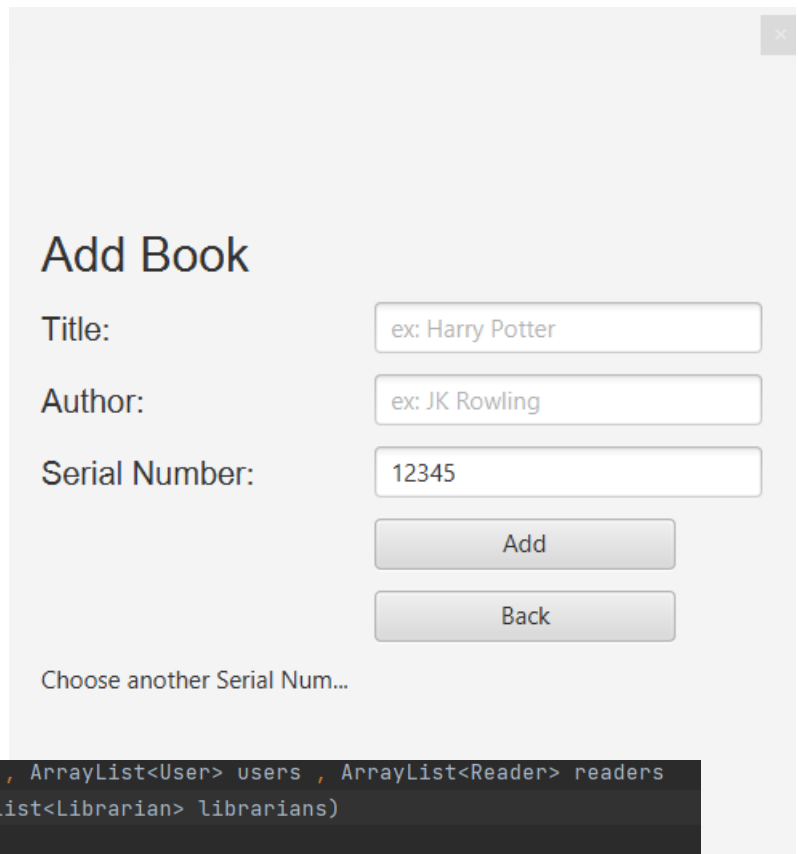
Author:

Serial Number:

Add

Back

One cannot add a book with the same
Serial number.



Add Book

Title:

Author:

Serial Number:

[Choose another Serial Num...](#)

```
private void showAddBookScene(User user , ArrayList<User> users , ArrayList<Reader> readers
    , ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    // Initialize UI components for add book scene
    Label addBookLabel = new Label( s: "Add Book");
    Label titleLabel = new Label( s: "Title:");
    Label authorLabel = new Label( s: "Author:");
    Label isbnLabel = new Label( s: "Serial Number:");
    Label messageLabel = new Label();
    TextField titleField = new TextField();
    titleField.setPromptText("ex: Harry Potter");
    TextField authorField = new TextField();
    authorField.setPromptText("ex: JK Rowling");
    TextField isbnField = new TextField();
    isbnField.setPromptText("ex: 123456789");
    Button addButton = new Button( s: "Add");
    Button backButton = new Button( s: "Back");

    // Set UI component properties
    addBookLabel.setFont(new Font( s: "Arial", v: 24));
    titleLabel.setFont(new Font( s: "Arial", v: 16));
    authorLabel.setFont(new Font( s: "Arial", v: 16));
    isbnLabel.setFont(new Font( s: "Arial", v: 16));
    titleField.setPrefWidth(200);
    authorField.setPrefWidth(200);
    isbnField.setPrefWidth(200);
    addButton.setPrefWidth(150);
    backButton.setPrefWidth(150);
}
```

```

GridPane addBookForm = new GridPane();
addBookForm.setHgap(10);
addBookForm.setVgap(10);
addBookForm.setPadding(new Insets( 20));
addBookForm.setAlignment(Pos.CENTER);
addBookForm.add(addBookLabel, 0, 0);
addBookForm.add(titleLabel, 0, 1);
addBookForm.add(titleField, 1, 1);
addBookForm.add(authorLabel, 0, 2);
addBookForm.add(authorField, 1, 2);
addBookForm.add(isbnLabel, 0, 3);
addBookForm.add(isbnField, 1, 3);
addBookForm.add(addButton, 1, 4);
addBookForm.add(backButton, 1, 5);
addBookForm.add(messageLabel, 0, 6);

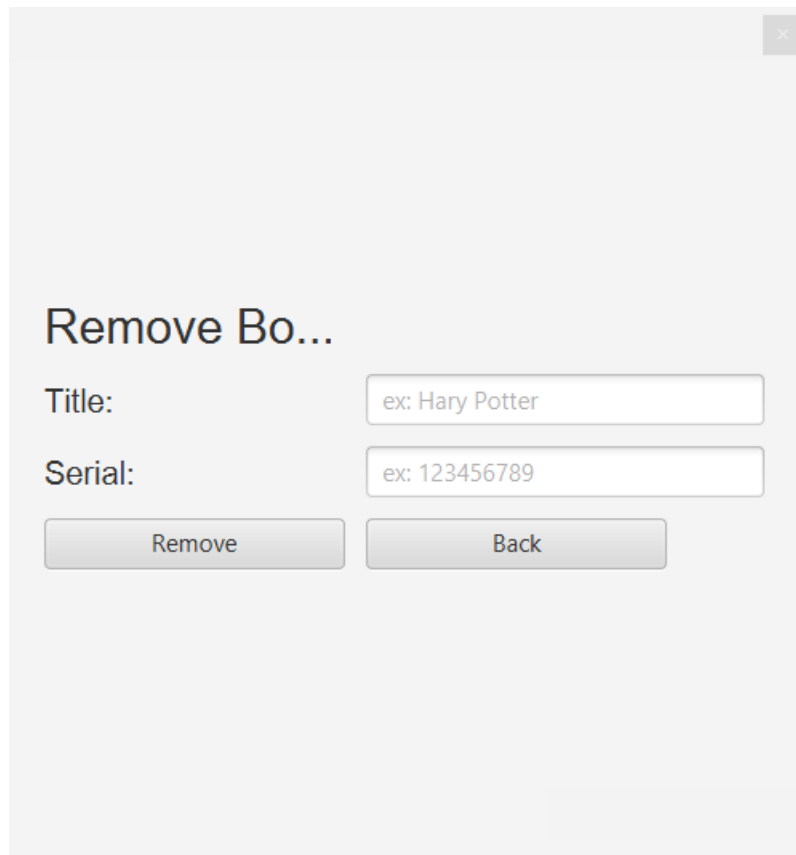
addButton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        String Title = titleField.getText();
        String Author = authorField.getText();
        String isbn = isbnField.getText();
        Book cmp = library.SearchBook(isbn, books);
        Boolean bool = true ? cmp == null : cmp != null;
        if(bool == true)
        {
            books.add(new Book(Title, Author, isbn));
            messageLabel.setText("Book Added Successfully");
        }
        else
            messageLabel.setText("Choose another Serial Number");
    }
});

backButton.setOnAction(e -> {
    // Show librarian dashboard
    showLibrarianDashboard(user, users, readers, books, librarians);
});

// Set add book scene as the root of the scene
Scene addBookScene = new Scene(addBookForm, 400, 400);
stage.setScene(addBookScene);

```

3.6.4. Remove Book



A light gray dialog box with a close button (X) in the top right corner. The title "Remove Bo..." is displayed in a large, dark font. Below the title, there are two input fields. The first is labeled "Title:" and contains the placeholder text "ex: Hary Potter". The second is labeled "Serial:" and contains the placeholder text "ex: 123456789". At the bottom of the dialog, there are two buttons: "Remove" and "Back".

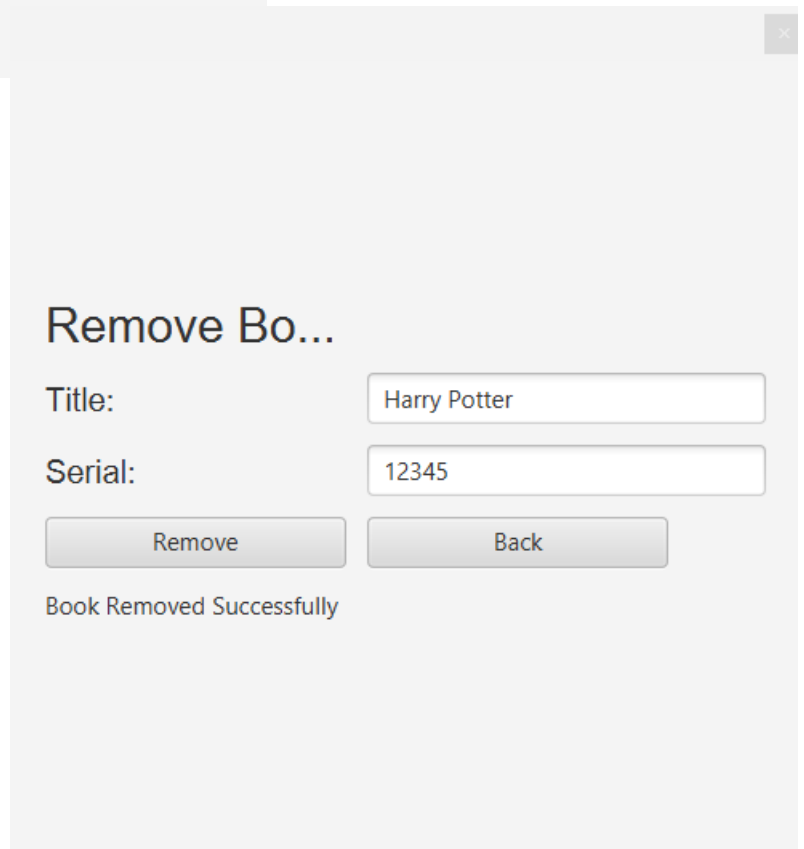
Remove Bo...

Title: ex: Hary Potter

Serial: ex: 123456789

Remove Back

Indication of successful book removal



A light gray dialog box with a close button (X) in the top right corner. The title "Remove Bo..." is displayed in a large, dark font. Below the title, there are two input fields. The first is labeled "Title:" and contains the text "Harry Potter". The second is labeled "Serial:" and contains the text "12345". At the bottom of the dialog, there are two buttons: "Remove" and "Back". Below the buttons, the text "Book Removed Successfully" is displayed.

Remove Bo...

Title: Harry Potter

Serial: 12345

Remove Back

Book Removed Successfully


```
private void showRemoveBookScene(User user , ArrayList<User> users , ArrayList<Reader> readers ,
ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    // Initialize UI components for remove book scene
    Label removeBookLabel = new Label( s: "Remove Book");
    Label titleLabel = new Label( s: "Title:");
    Label messageLabel = new Label();

    Label isbnLabel = new Label( s: "Serial:");
    TextField titleField = new TextField();
    TextField isbnField = new TextField();

    titleField.setPromptText("ex: Hary Potter");
    isbnField.setPromptText("ex: 123456789");

    Button removeButton = new Button( s: "Remove");
    Button backButton = new Button( s: "Back");

    // Set UI component properties
    removeBookLabel.setFont(new Font( s: "Arial", v: 24));
    titleLabel.setFont(new Font( s: "Arial", v: 16));

    isbnLabel.setFont(new Font( s: "Arial", v: 16));
    titleField.setPrefWidth(200);

    isbnField.setPrefWidth(200);
    removeButton.setPrefWidth(150);
    backButton.setPrefWidth(150);
}
```

```

GridPane removeBookForm = new GridPane();
removeBookForm.setHgap(10);
removeBookForm.setVgap(10);
removeBookForm.setPadding(new Insets( 20));
removeBookForm.setAlignment(Pos.CENTER);
removeBookForm.add(removeBookLabel, 0, 0);
removeBookForm.add(titleLabel, 0, 1);
removeBookForm.add(titleField, 1, 1);

removeBookForm.add(isbnLabel, 0, 2);
removeBookForm.add(isbnField, 1, 2);
removeBookForm.add(removeButton, 0, 3);
removeBookForm.add(backButton, 1, 3);
removeBookForm.add(messageLabel, 0, 4);

// Set event handlers for buttons
removeButton.setOnAction(e -> {
    // Remove book from database
    String title = titleField.getText();
    String isbn = isbnField.getText();

    Book Exist = library.BookRemoval(title , isbn , books);
    if(Exist != null)
    {
        books.remove(Exist);
        messageLabel.setText("Book Removed Successfully");
    }
    else
        messageLabel.setText("Book not found");
});

backButton.setOnAction(e -> {
    // Show librarian dashboard
    showLibrarianDashboard(user, users , readers , books , librarians);
});

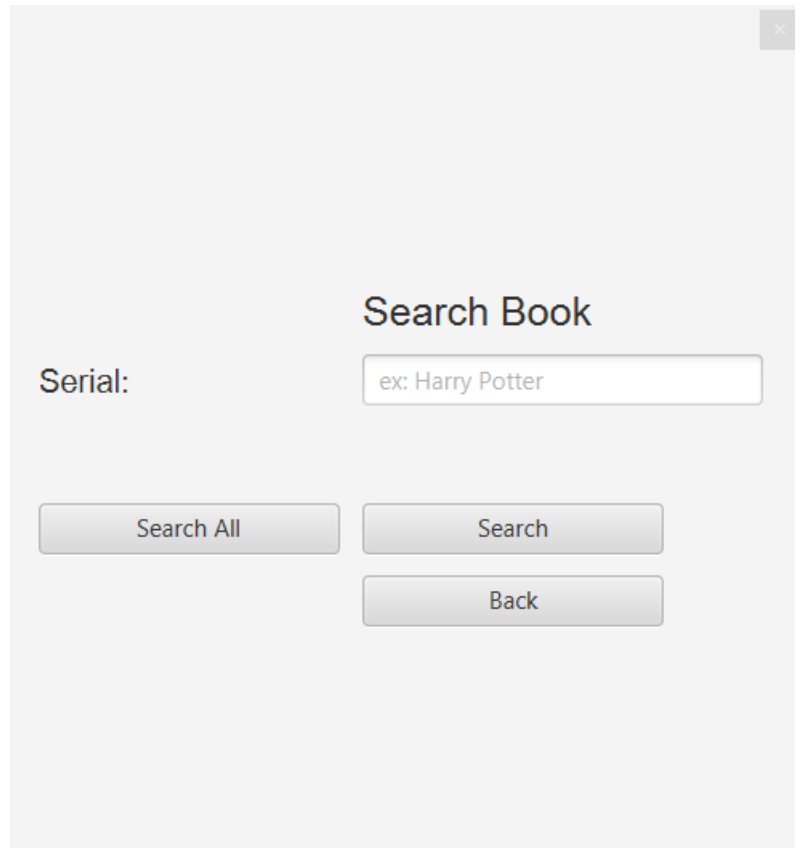
// Set remove book scene as the root of the scene
Scene removeBookScene = new Scene(removeBookForm, 400, 400);
stage.setScene(removeBookScene);

```

3.6.5. Search Book

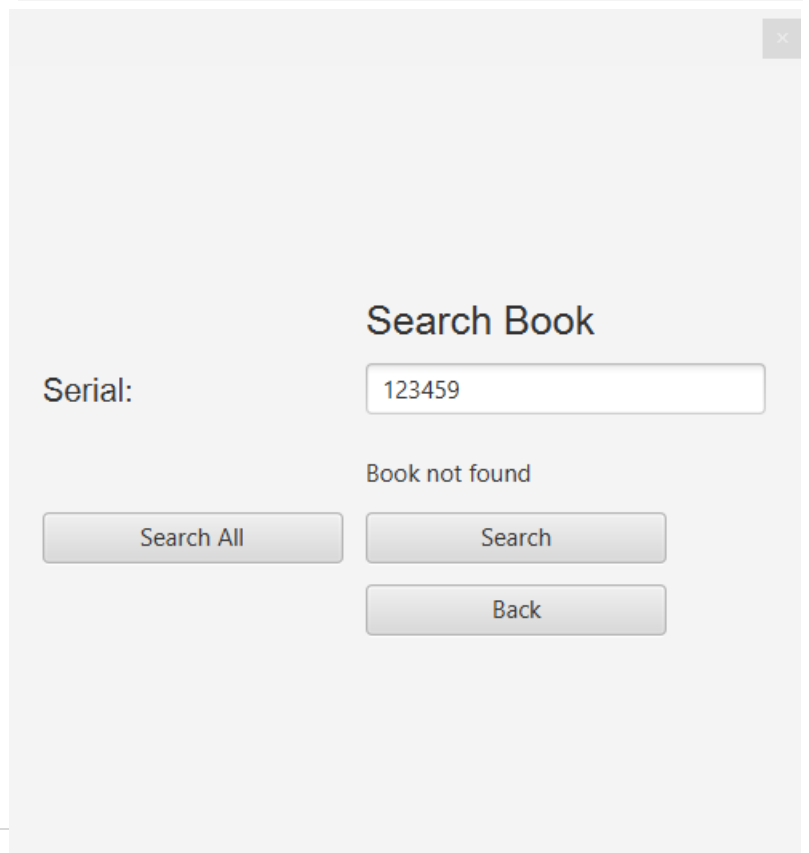
A person shall search by serial

Number or scroll through all the books



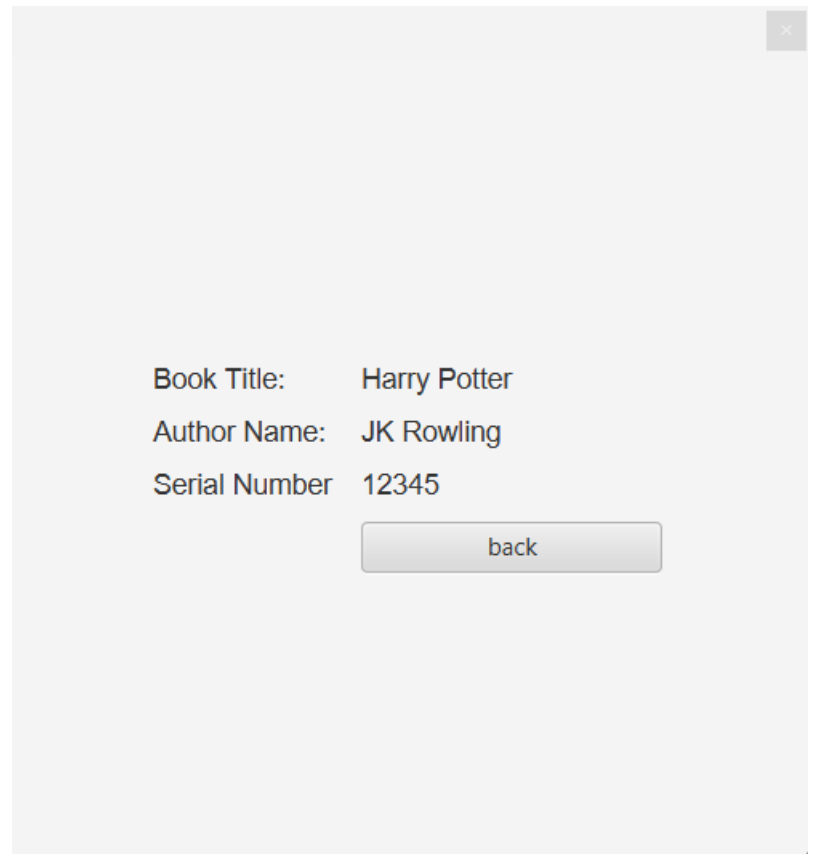
A light gray dialog box with a close button (X) in the top right corner. The title "Search Book" is centered at the top. Below the title, the label "Serial:" is positioned to the left of a text input field. The input field contains the placeholder text "ex: Harry Potter". Below the input field, there are three buttons: "Search All" on the left, and "Search" and "Back" on the right, with "Search" positioned above "Back".

If user enter a book that does not exist



A light gray dialog box with a close button (X) in the top right corner. The title "Search Book" is centered at the top. Below the title, the label "Serial:" is positioned to the left of a text input field. The input field contains the text "123459". Below the input field, the text "Book not found" is displayed. Below this text, there are three buttons: "Search All" on the left, and "Search" and "Back" on the right, with "Search" positioned above "Back".

Searching by serial



Book Title: Harry Potter
Author Name: JK Rowling
Serial Number 12345

back

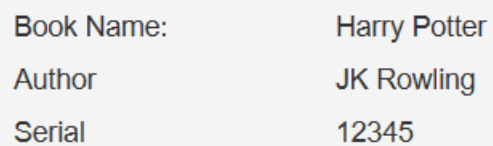
Searching Through all books

Menu: returns users to Menu

Next: goes to next book, if it exists,

If not the next button will disappear

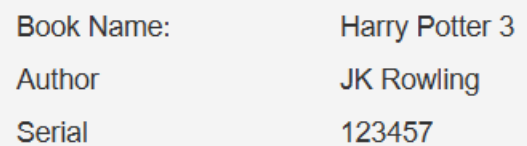
Back: returns to previous page



Book Name: Harry Potter
Author JK Rowling
Serial 12345

Next

Menu



Book Name: Harry Potter 3
Author JK Rowling
Serial 123457

back

Menu

Code:-

```
private void showSearchUserScene(User user , ArrayList<User> users , ArrayList<Reader> readers
    , ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    Label SearchUserLabel = new Label( s: "Search for User");
    Label TextFieldLabel = new Label( s: "ID:");
    TextField ID = new TextField();
    Button SearchButton = new Button( s: "Search");
    Button Scroll = new Button( s: "Search All");

    Button backButton = new Button( s: "Back");

    Label messageLabel = new Label();

    SearchUserLabel.setFont(new Font( s: "Arial", v: 20));
    TextFieldLabel.setFont(new Font( s: "Arial", v: 16));

    ID.setPrefWidth(200);
    backButton.setPrefWidth(150);
    SearchButton.setPrefWidth(150);
    Scroll.setPrefWidth(150);

    GridPane SearchUser = new GridPane();
    SearchUser.setHgap(10);
    SearchUser.setVgap(10);
    SearchUser.setPadding(new Insets( v: 20));
    SearchUser.setAlignment(Pos.CENTER);
    SearchUser.add(SearchUserLabel, i: 1, i1: 0);
```

```

SearchUser.add(SearchUserLabel, i: 1, i1: 0);
SearchUser.add(TextFieldLabel, i: 0, i1: 1);
SearchUser.add(ID, i: 1, i1: 1);
SearchUser.add(SearchButton, i: 1, i1: 4);
SearchUser.add(Scroll, i: 0, i1: 4);
SearchUser.add(backButton, i: 1, i1: 5);
SearchUser.add(messageLabel, i: 1, i1: 3);

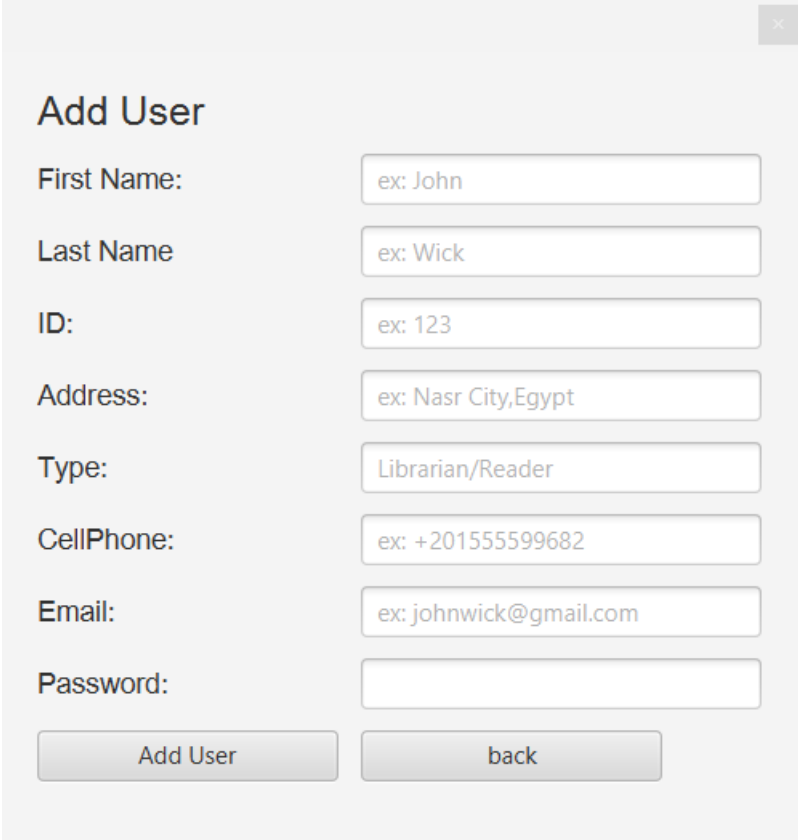
backButton.setOnAction(e -> {
    // Show librarian dashboard
    showLibrarianDashboard(user, users ,readers , books , librarians);
});
Scroll.setOnAction(e ->{
    showUserScroll(user,user, users ,readers , books , librarians);
});

Scene SearchUserScene = new Scene(SearchUser, v: 400, v1: 400);
stage.setScene(SearchUserScene);
SearchButton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        String id = ID.getText();
        User searchID = library.SearchU(id, users);
        if (searchID != null)
        {
            showUser(searchID, user, users ,readers , books , librarians);
        }
        else {
            messageLabel.setText("User not found");
        }
    }
}

```

3.6.6. Add User

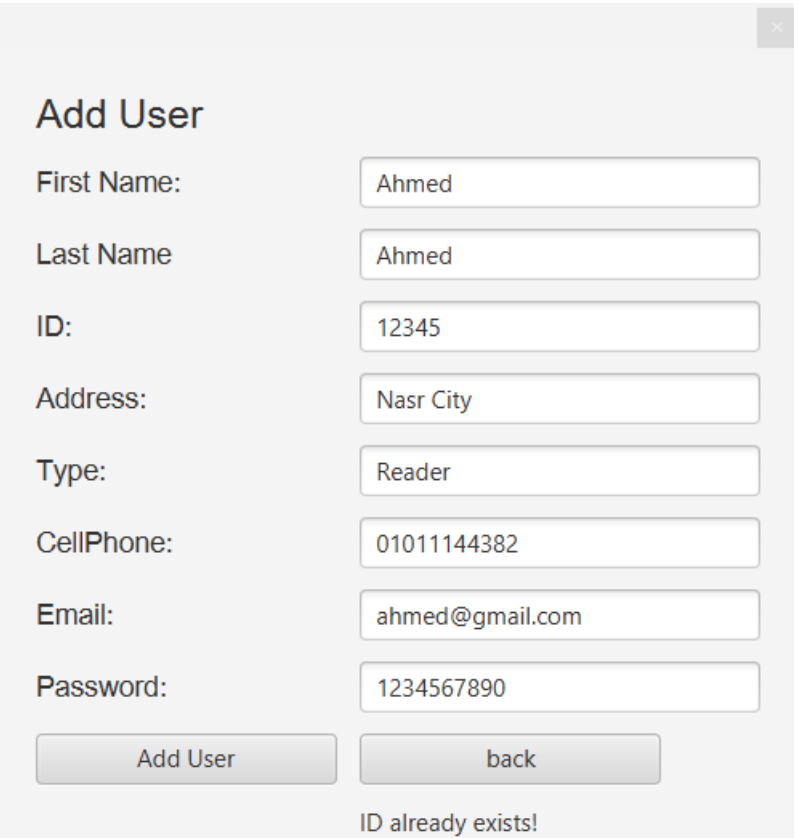
Add User screen with limitations to some data



The screenshot shows a mobile application screen titled "Add User". It features a list of input fields for user registration, each with a label and a text box containing placeholder text. The fields are: First Name (ex: John), Last Name (ex: Wick), ID (ex: 123), Address (ex: Nasr City,Egypt), Type (Librarian/Reader), CellPhone (ex: +201555599682), Email (ex: johnwick@gmail.com), and Password (empty). At the bottom, there are two buttons: "Add User" and "back". A close button (X) is in the top right corner.

First Name:	ex: John
Last Name	ex: Wick
ID:	ex: 123
Address:	ex: Nasr City,Egypt
Type:	Librarian/Reader
CellPhone:	ex: +201555599682
Email:	ex: johnwick@gmail.com
Password:	
<div>Add User</div> <div>back</div>	

Each User shall have a unique ID



The screenshot shows the same "Add User" screen, but with specific data entered and a validation error. The fields are: First Name (Ahmed), Last Name (Ahmed), ID (12345), Address (Nasr City), Type (Reader), CellPhone (01011144382), Email (ahmed@gmail.com), and Password (1234567890). The "Add User" button is disabled, and a red error message "ID already exists!" is displayed at the bottom. The "back" button is still active. A close button (X) is in the top right corner.

First Name:	Ahmed
Last Name	Ahmed
ID:	12345
Address:	Nasr City
Type:	Reader
CellPhone:	01011144382
Email:	ahmed@gmail.com
Password:	1234567890
<div>Add User</div> <div>back</div> <div>ID already exists!</div>	

User shall clearly state their type (Case insensitive)

×

Add User

First Name:

Ahmed

Last Name

Ahmed

ID:

123

Address:

Nasr City

Type:

Rea

CellPhone:

01011144382

Email:

ahmed@gmail.com

Password:

1234567890

Add User

back

Invalid Type

Correct Mail address should be used

Ex: @gmail

×

Add User

First Name:

Ahmed

Last Name

Ahmed

ID:

123

Address:

Nasr City

Type:

Reader

CellPhone:

01011144382

Email:

ahmed

Password:

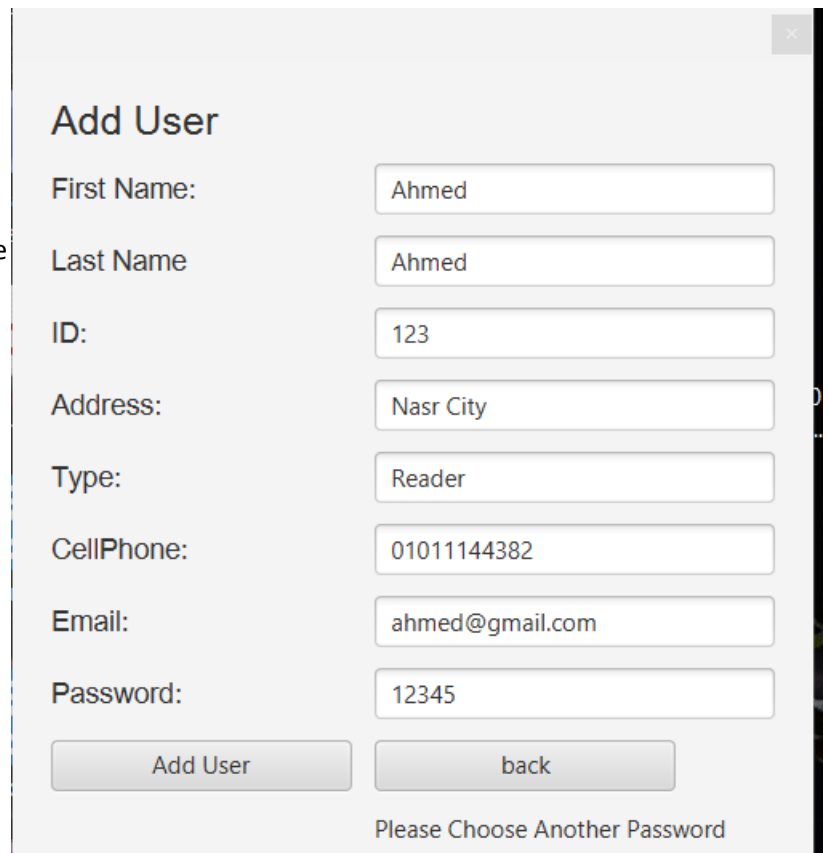
1234567890

Add User

back

Incorrect mail

System Shall detect similar passwords to ensure Uniqueness in passwords.

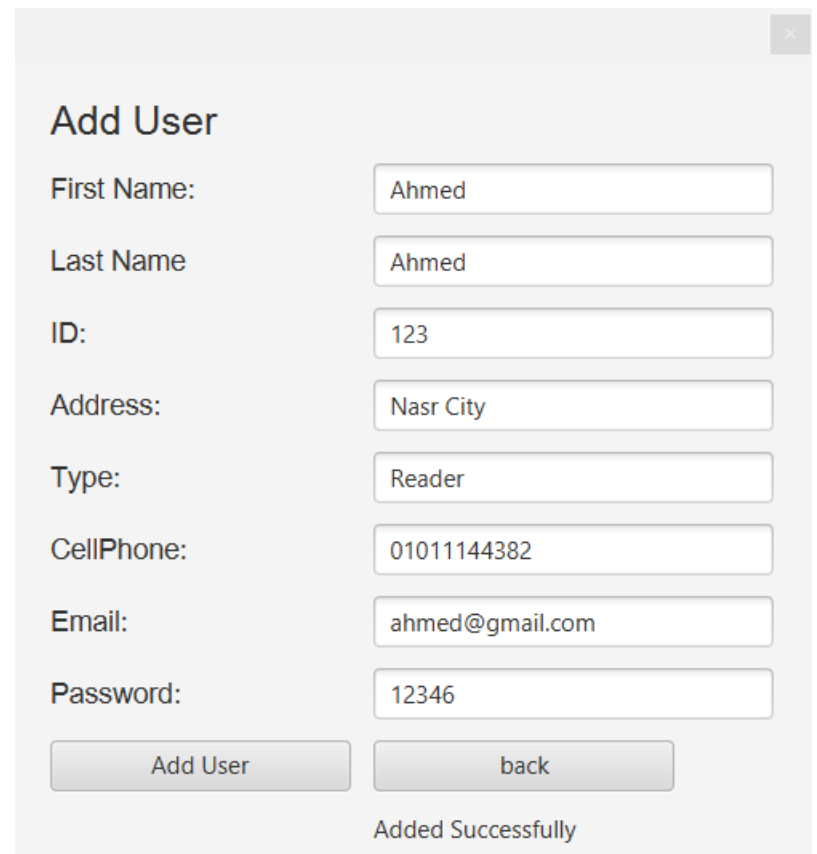


The screenshot shows a web form titled "Add User" with the following fields and values: First Name: Ahmed, Last Name: Ahmed, ID: 123, Address: Nasr City, Type: Reader, CellPhone: 01011144382, Email: ahmed@gmail.com, and Password: 12345. At the bottom, there are two buttons: "Add User" and "back". Below the buttons, a message reads "Please Choose Another Password".

First Name:	Ahmed
Last Name	Ahmed
ID:	123
Address:	Nasr City
Type:	Reader
CellPhone:	01011144382
Email:	ahmed@gmail.com
Password:	12345

Please Choose Another Password

If user oblige to all limitations, input user will be added.



The screenshot shows the same "Add User" form as above, but with the password changed to "12346". The "Add User" button is now highlighted, and a message at the bottom reads "Added Successfully".

First Name:	Ahmed
Last Name	Ahmed
ID:	123
Address:	Nasr City
Type:	Reader
CellPhone:	01011144382
Email:	ahmed@gmail.com
Password:	12346

Added Successfully

Code: -

```
private void ShowAddUser(User user , ArrayList<User> users , ArrayList<Reader> readers
    , ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    Label ShowAddUserLabel = new Label( s: "Add User");
    Label FirstName = new Label( s: "First Name:");
    TextField FirstNameAns = new TextField();
    FirstNameAns.setPromptText("ex: John");
    Label LastName = new Label( s: "Last Name");
    TextField LastNameAns = new TextField();
    LastNameAns.setPromptText("ex: Wick");
    Label ID = new Label( s: "ID: ");
    TextField IDAns = new TextField();
    IDAns.setPromptText("ex: 123");
    Label Address = new Label( s: "Address: ");
    TextField AddressAns = new TextField();
    AddressAns.setPromptText("ex: Nasr City,Egypt");
    Label Type = new Label( s: "Type: ");
    TextField TypeAns = new TextField();
    TypeAns.setPromptText("Librarian/Reader");
    Label Cell = new Label( s: "CellPhone: ");
    TextField CellAns = new TextField();
    CellAns.setPromptText("ex: +201555599682");
    Label Email = new Label( s: "Email: ");
    TextField EmailAns = new TextField();
    EmailAns.setPromptText("ex: johnwick@gmail.com");
    Label Password = new Label( s: "Password: ");
    TextField PasswordAns = new TextField();
    Button Add = new Button( s: "Add User");
    Button BackButton = new Button( s: "back");
    Label messageLabel = new Label();
```

```

ShowAddUserLabel.setFont(new Font( s: "Arial", v: 20));
FirstName.setFont(new Font( s: "Arial", v: 14));
LastName.setFont(new Font( s: "Arial", v: 14));
ID.setFont(new Font( s: "Arial", v: 14));
Address.setFont(new Font( s: "Arial", v: 14));
Type.setFont(new Font( s: "Arial", v: 14));
Cell.setFont(new Font( s: "Arial", v: 14));
Email.setFont(new Font( s: "Arial", v: 14));
Password.setFont(new Font( s: "Arial", v: 14));
FirstNameAns.setPrefWidth(200);
LastNameAns.setPrefWidth(200);
IDAns.setPrefWidth(200);
AddressAns.setPrefWidth(200);
TypeAns.setPrefWidth(200);
CellAns.setPrefWidth(200);
EmailAns.setPrefWidth(200);
PasswordAns.setPrefWidth(200);

BackButton.setPrefWidth(150);
Add.setPrefWidth(150);

GridPane Show = new GridPane();
Show.setHgap(10);
Show.setVgap(10);
Show.setPadding(new Insets( v: 20));
Show.setAlignment(Pos.CENTER);
Show.add>ShowAddUserLabel , i: 0 , i1: 1);
Show.add(FirstName, i: 0, i1: 2);
Show.add(FirstNameAns, i: 1, i1: 2);
Show.add(LastName, i: 0, i1: 3);

```

```

Show.add(LastName, i: 0, i1: 3);
Show.add(LastNameAns, i: 1, i1: 3);
Show.add(ID, i: 0, i1: 4);
Show.add(IDAns, i: 1, i1: 4);
Show.add(Address, i: 0, i1: 5);
Show.add(AddressAns, i: 1, i1: 5);
Show.add(Type, i: 0, i1: 6);
Show.add(TypeAns, i: 1, i1: 6);
Show.add(Cell, i: 0, i1: 7);
Show.add(CellAns, i: 1, i1: 7);
Show.add(Email, i: 0, i1: 8);
Show.add(EmailAns, i: 1, i1: 8);
Show.add>Password, i: 0, i1: 9);
Show.add>PasswordAns, i: 1, i1: 9);

Show.add(Add, i: 0, i1: 10);
Show.add(BackButton, i: 1, i1: 10);
Show.add(messageLabel, i: 1, i1: 11);
BackButton.setOnAction(e -> {
    // Show librarian dashboard
    showLibrarianDashboard(user, users, readers, books, librarians);
});
Add.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        String firstName = FirstNameAns.getText();
        String lastName = LastNameAns.getText();
    }
});

```

```

String Address = AddressAns.getText();
String Type = TypeAns.getText();
String id = IDAns.getText();
String Cell = CellAns.getText();
String Email = EmailAns.getText();
boolean mail = library.EmailCheck(Email);
String password = PasswordAns.getText();
boolean blocked = false;
Boolean IDChecker = library.CompareDataFields(id , users);
Boolean PasswordChecker = library.CompareDataFieldsPassword(password , users);

if (mail == false)
{
    messageLabel.setText("Incorrect mail");
}
else if(PasswordChecker == true)
{
    messageLabel.setText("Please Choose Another Password");
}
else if(IDChecker == true) {
    messageLabel.setText("ID already exists!");
}
else
{
    if (Type.equalsIgnoreCase( anotherString: "Reader"))
    {
        Type = "Reader";
        Reader R1 = new Reader(id, password, firstName, lastName, Address, Cell, Email, blocked, Type);
        users.add(R1);

```

```

        users.add(R1);
        readers.add(R1);
        messageLabel.setText("Added Successfully");

    } else if (Type.equalsIgnoreCase( anotherString: "Librarian"))
    {
        Type = "Librarian";
        Librarian L1 = new Librarian(id, password, firstName, lastName, Address, Cell, Email, blocked, Type);
        users.add(L1);
        librarians.add(L1);
        messageLabel.setText("Added Successfully");

    } else
        messageLabel.setText("Invalid Type");
    }
}

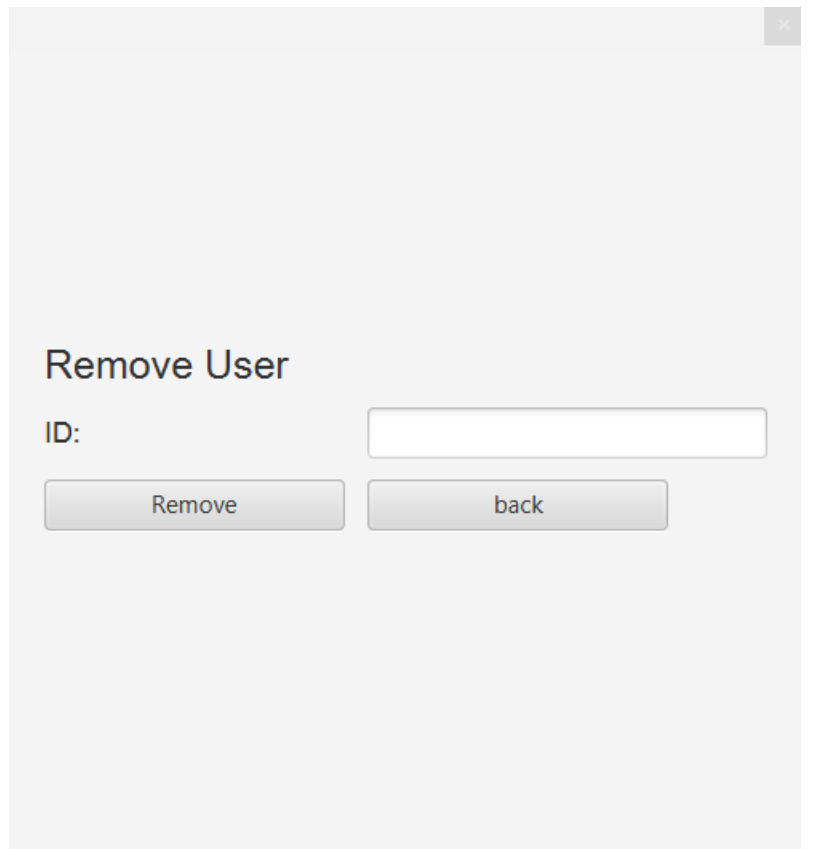
});

Scene AddUserScene = new Scene(Show, w: 400, h: 400);
stage.setScene(AddUserScene);

```

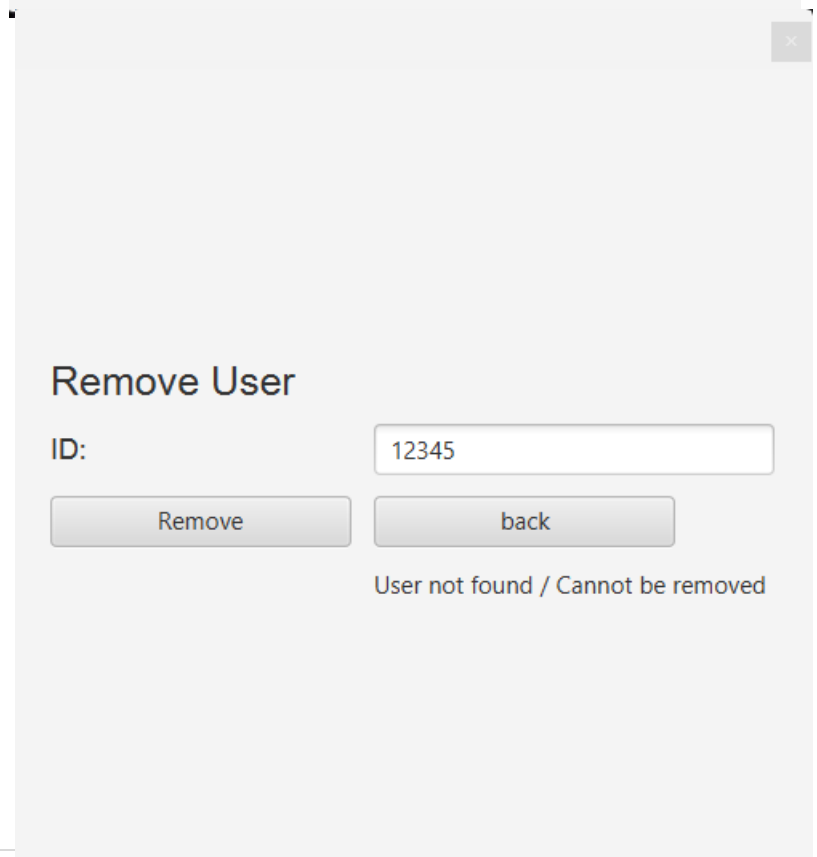
3.6.7. Remove User

Remove user scene



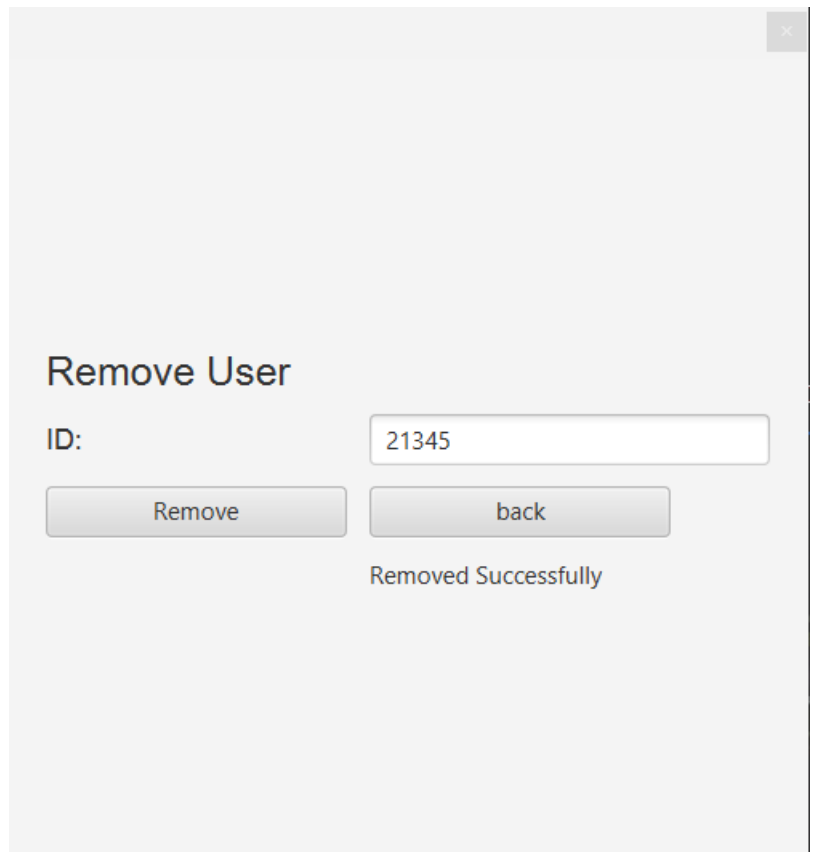
A light gray dialog box with a close button (X) in the top right corner. The title "Remove User" is centered at the top. Below the title, the label "ID:" is followed by a white text input field. At the bottom, there are two gray buttons: "Remove" on the left and "back" on the right.

Cannot remove librarian or self.
Or no user found.



A light gray dialog box with a close button (X) in the top right corner. The title "Remove User" is centered at the top. Below the title, the label "ID:" is followed by a white text input field containing the text "12345". At the bottom, there are two gray buttons: "Remove" on the left and "back" on the right. Below the buttons, the text "User not found / Cannot be removed" is displayed.

successfully removing a user and returning
all his books



Code :-

```
private void ShowRemoveUserScene(User user , ArrayList<User> users , ArrayList<Reader> readers
    , ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    Label RemoveUserLabel = new Label( s: "Remove User");

    Label ID = new Label( s: "ID:");
    TextField IDAns= new TextField();

    Button Remove = new Button( s: "Remove");
    Button BackButton = new Button( s: "back");
    Label messageLabel = new Label();

    RemoveUserLabel.setFont(new Font( s: "Arial", v: 20));
    ID.setFont(new Font( s: "Arial", v: 14));

    IDAns.setPrefWidth(200);
    BackButton.setPrefWidth(150);
    Remove.setPrefWidth(150);

    GridPane Show = new GridPane();
    Show.setHgap(10);
    Show.setVgap(10);
    Show.setPadding(new Insets( v: 20));
    Show.setAlignment(Pos.CENTER);
    Show.add(RemoveUserLabel , i: 0 , i1: 1);
    Show.add(ID, i: 0, i1: 2);
    Show.add(IDAns, i: 1, i1: 2);
    Show.add(Remove, i: 0, i1: 3);
    Show.add(BackButton, i: 1, i1: 3);
    Show.add(messageLabel, i: 1, i1: 4);
}
```

```

backButton.setOnAction(e -> {
    // Show librarian dashboard
    showLibrarianDashboard(user, users, readers, books, librarians);
});

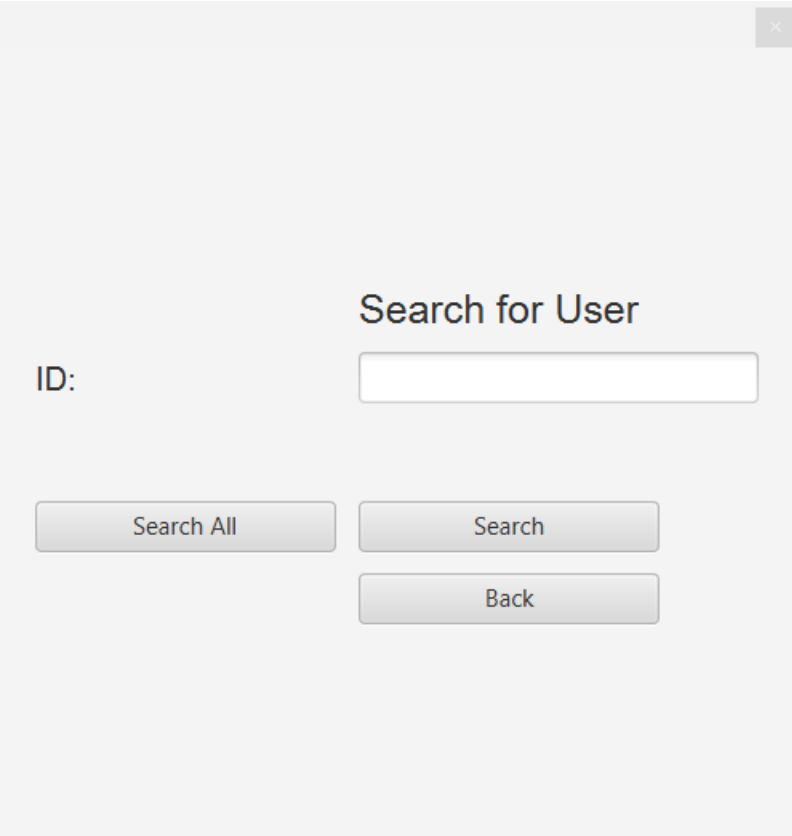
Remove.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        String id = IDAns.getText();
        Iterator<User> iterator = users.iterator();
        boolean found = false;
        while (iterator.hasNext()) {
            User user = iterator.next();
            if (user.getId().equals(id) && user.getType().equalsIgnoreCase("Reader")) {
                ArrayList<Book> books5 = ((Reader)user).getOrderList();
                Iterator<Book> iterator1 = books5.iterator();
                while (iterator1.hasNext()) {
                    Book book2 = iterator1.next();
                    book2.setIs_available(true);
                    books5.remove(book2);
                }
                users.remove(user);
                readers.remove(user);
                messageLabel.setText("Removed Successfully");
                found = true;
            }
        }
        if(!found)
            {messageLabel.setText("User not found / Cannot be removed");}
    }
});

Scene AddUserScene = new Scene(Show, 400, 400);
stage.setScene(AddUserScene);

```

3.6.8. Search User

User Shall Either search by ID or scroll through
All similar to the Search book



The screenshot shows a JavaFX application window with a light gray background and a close button in the top right corner. The window is titled "Search for User". Inside the window, there is a label "ID:" followed by a white text input field with a thin gray border. Below the input field, there are three buttons arranged horizontally. The first button is labeled "Search All", the second is labeled "Search", and the third is labeled "Back". All buttons have a light gray background and a thin gray border.

If user ID does not exist

×

Search for User

ID:

User not found

Search AllSearchBack

Search by ID

×

Full Name:	Ahmed Ahmed
ID:	123
Address:	Nasr City
Type:	Reader
CellPhone:	01011144382
Email:	ahmed@gmail.com
Is this user blocked?	not blocked

back

Search through all users

Full Name: Anthony Davies
ID: 12345
Address: 23rd Wall Street
Type: Librarian
CellPhone: +201011155234
Email: anthony@gmail.com
Is this user blocked? not blocked

Next

Menu

Full Name: L Mo
ID: 1
Address: 22nd Wall Street
Type: Reader
CellPhone: +201011155234
Email: Modric@gmail.com
Is this user blocked? not blocked

Next

back

Menu

Code :-

```
private void showSearchUserScene(User user , ArrayList<User> users , ArrayList<Reader> readers
    , ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    Label SearchUserLabel = new Label( s: "Search for User");
    Label TextFieldLabel = new Label( s: "ID:");
    TextField ID = new TextField();
    Button SearchButton = new Button( s: "Search");
    Button Scroll = new Button( s: "Search All");

    Button backButton = new Button( s: "Back");

    Label messageLabel = new Label();

    SearchUserLabel.setFont(new Font( s: "Arial", v: 20));
    TextFieldLabel.setFont(new Font( s: "Arial", v: 16));

    ID.setPrefWidth(200);
    backButton.setPrefWidth(150);
    SearchButton.setPrefWidth(150);
    Scroll.setPrefWidth(150);

    GridPane SearchUser = new GridPane();
    SearchUser.setHgap(10);
    SearchUser.setVgap(10);
    SearchUser.setPadding(new Insets( v: 20));
    SearchUser.setAlignment(Pos.CENTER);
    SearchUser.add(SearchUserLabel, i: 1, i1: 0);
```

```

SearchUser.add(TextFieldLabel, i: 0, i1: 1);
SearchUser.add(ID, i: 1, i1: 1);
SearchUser.add(SearchButton, i: 1, i1: 4);
SearchUser.add(Scroll, i: 0, i1: 4);
SearchUser.add(backButton, i: 1, i1: 5);
SearchUser.add(messageLabel, i: 1, i1: 3);

backButton.setOnAction(e -> {
    // Show librarian dashboard
    showLibrarianDashboard(user, users, readers, books, librarians);
});
Scroll.setOnAction(e ->{
    showUserScroll(user,user, users, readers, books, librarians);
});

Scene SearchUserScene = new Scene(SearchUser, v: 400, v1: 400);
stage.setScene(SearchUserScene);
SearchButton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        String id = ID.getText();
        User searchID = library.SearchU(id, users);
        if (searchID != null)
        {
            showUser(searchID, user, users, readers, books, librarians);
        }
        else {
            messageLabel.setText("User not found");
        }
    }
}

```

```

private void showUserScroll(User user, User Change, ArrayList<User> users , ArrayList<Reader> readers
    , ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    User user1 = users.get(iteration);
    int x = users.size();
    Label FullName = new Label( s: "Full Name:");
    Label FullNameAns = new Label( s: user1.getFirstName() + " " + user1.getLastName());
    Label ID = new Label( s: "ID: ");
    Label IDAns = new Label(user1.getId());
    Label Address = new Label( s: "Address: ");
    Label AddressAns = new Label(user1.getAddress());
    Label Type = new Label( s: "Type: ");
    Label TypeAns = new Label(user1.getType());
    Label Cell = new Label( s: "CellPhone: ");
    Label CellAns = new Label(user.getCellPhone());
    Label Email = new Label( s: "Email: ");
    Label EmailAns = new Label(user1.getEmail());
    Label isBlocked = new Label( s: "Is this user blocked? ");
    Label isBlockedAns = new Label(user1.isBlockedStr());
    Button BackButton = new Button( s: "back");
    Button MainMenu = new Button( s: "Menu");
    Button Next = new Button( s: "Next");

    FullName.setFont(new Font( s: "Arial", v: 14));
    FullNameAns.setFont(new Font( s: "Arial", v: 14));
    ID.setFont(new Font( s: "Arial", v: 14));
    IDAns.setFont(new Font( s: "Arial", v: 14));
    Address.setFont(new Font( s: "Arial", v: 14));
    AddressAns.setFont(new Font( s: "Arial", v: 14));
    Type.setFont(new Font( s: "Arial", v: 14));

```

```

TypeAns.setFont(new Font( s: "Arial", v: 14));
Cell.setFont(new Font( s: "Arial", v: 14));
CellAns.setFont(new Font( s: "Arial", v: 14));
Email.setFont(new Font( s: "Arial", v: 14));
EmailAns.setFont(new Font( s: "Arial", v: 14));
isBlocked.setFont(new Font( s: "Arial", v: 14));
isBlockedAns.setFont(new Font( s: "Arial", v: 14));

BackButton.setPrefWidth(150);
Next.setPrefWidth(150);
MainMenu.setPrefWidth(150);

GridPane Show = new GridPane();
Show.setHgap(10);
Show.setVgap(10);
Show.setPadding(new Insets( v: 20));
Show.setAlignment(Pos.CENTER);
Show.add(FullName, i: 0, i1: 1);
Show.add(FullNameAns, i: 1, i1: 1);
Show.add(ID, i: 0, i1: 2);
Show.add(IDAns, i: 1, i1: 2);
Show.add(Address, i: 0, i1: 3);
Show.add(AddressAns, i: 1, i1: 3);
Show.add(Type, i: 0, i1: 4);
Show.add(TypeAns, i: 1, i1: 4);
Show.add(Cell, i: 0, i1: 5);
Show.add(CellAns, i: 1, i1: 5);
Show.add(Email, i: 0, i1: 6);
Show.add(EmailAns, i: 1, i1: 6);
Show.add(isBlocked, i: 0, i1: 7);
Show.add(isBlockedAns, i: 1, i1: 7);

```

```

if(iteration>0)
    Show.add(BackButton, i: 1, i1: 8);
if(iteration<x-1)
    Show.add(Next, i: 0, i1: 8);

Show.add(MainMenu, i: 0, i1: 9);

BackButton.setOnAction(e -> {
    // Show librarian dashboard
    iteration--;
    showUserScroll(user,user1, users ,readers , books , librarians);
});
Next.setOnAction(e -> {
    iteration++;
    showUserScroll(user , user1, users ,readers , books , librarians);
});
MainMenu.setOnAction(e -> {
    // Show librarian dashboard
    iteration = 0;
    showSearchUserScene(user, users, readers, books, librarians);
});

Scene ShowScene = new Scene(Show, v: 400, v1: 400);
stage.setScene(ShowScene);
}

```

3.6.9. Block User

Librarians could block users if they failed to
Return the books in 5 days.

Cannot Block Librarians

Block Users

ID:

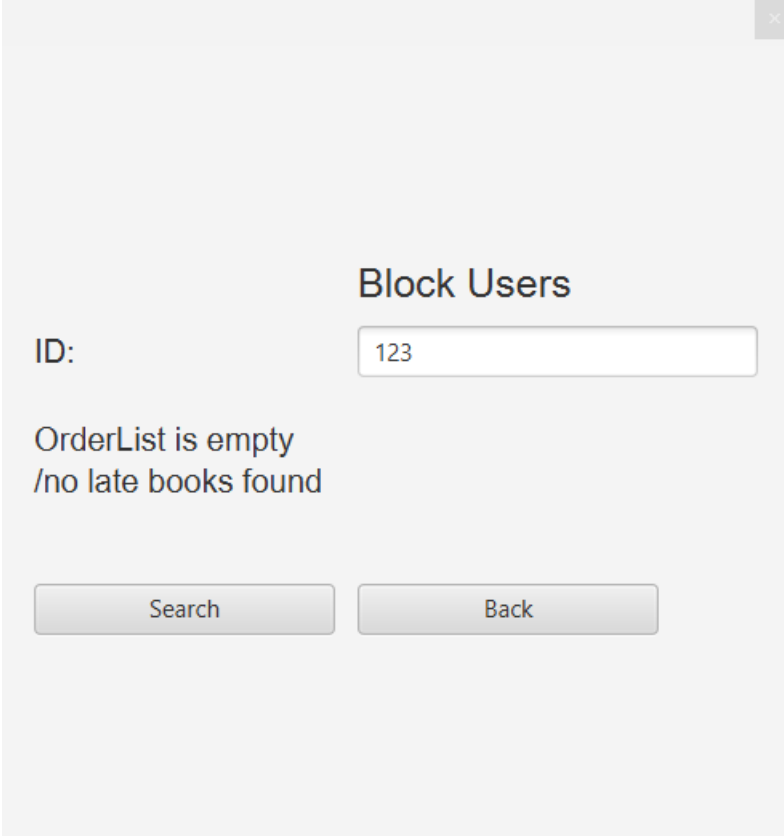
12345

User is a Librarian

Search

Back

Cannot block who returned books on time
Or rented no books



This screenshot shows a 'Block Users' dialog box. At the top right is a close button (X). The title 'Block Users' is centered. Below it, the label 'ID:' is followed by a text input field containing the number '123'. A message 'OrderList is empty /no late books found' is displayed in the center. At the bottom, there are two buttons: 'Search' and 'Back'.

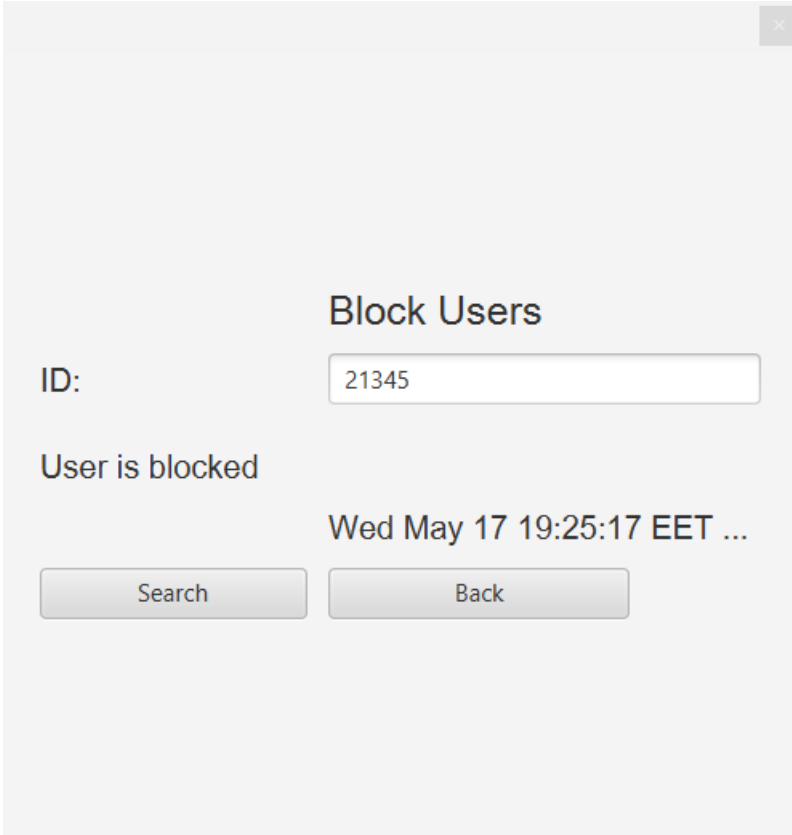
Block Users

ID: 123

OrderList is empty
/no late books found

Search Back

Blocking users when the book is 5 days late
And time of rental is displayed



This screenshot shows the 'Block Users' dialog box after a successful block. The title 'Block Users' is centered. Below it, the label 'ID:' is followed by a text input field containing the number '21345'. A message 'User is blocked' is displayed in the center. Below that, the timestamp 'Wed May 17 19:25:17 EET ...' is shown. At the bottom, there are two buttons: 'Search' and 'Back'.

Block Users

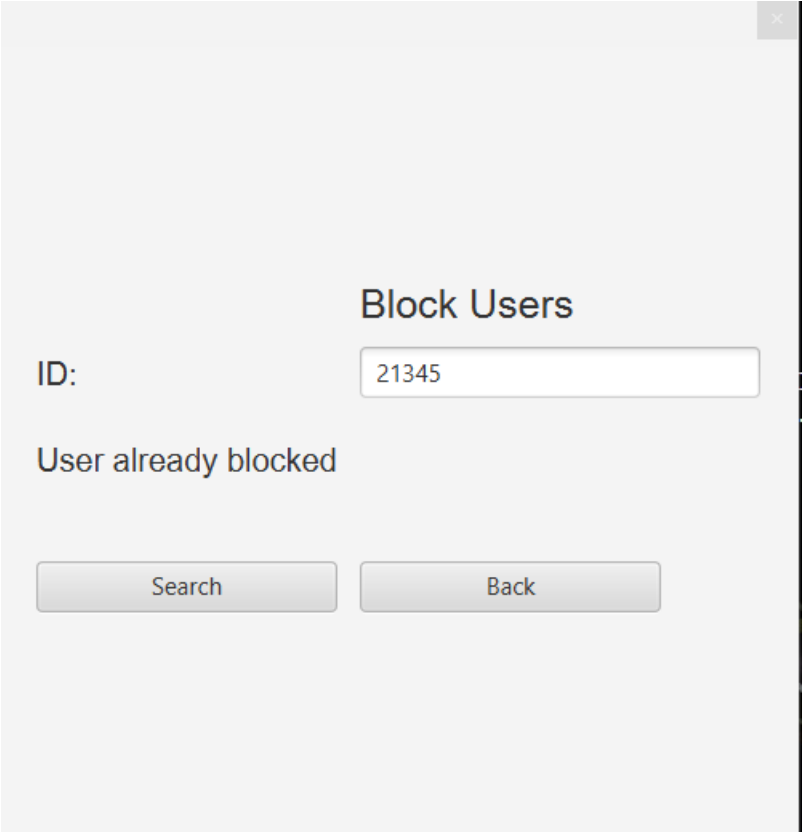
ID: 21345

User is blocked

Wed May 17 19:25:17 EET ...

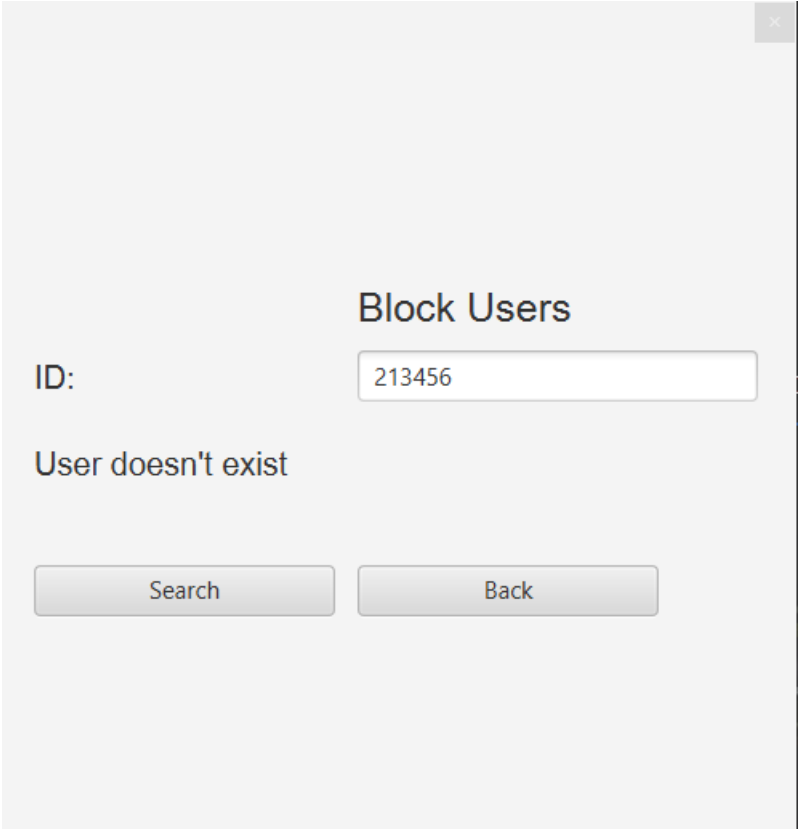
Search Back

Cannot block already blocked users.



A screenshot of a web application window titled "Block Users". The window has a light gray background and a close button in the top right corner. Below the title, there is a label "ID:" followed by a text input field containing the value "21345". Below the input field, the text "User already blocked" is displayed in a dark blue font. At the bottom of the window, there are two buttons: "Search" and "Back".

Cannot block a user that doesn't exist



A screenshot of a web application window titled "Block Users". The window has a light gray background and a close button in the top right corner. Below the title, there is a label "ID:" followed by a text input field containing the value "213456". Below the input field, the text "User doesn't exist" is displayed in a dark blue font. At the bottom of the window, there are two buttons: "Search" and "Back".

Code:-

```
private void showBlockUserScene(User user , ArrayList<User> users , ArrayList<Reader> readers ,
                                ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    Label SearchUserLabel = new Label( s: "Block Users");
    Label TextFieldLabel = new Label( s: "ID:");
    TextField ID = new TextField();
    Button SearchButton = new Button( s: "Search");
    Button backButton = new Button( s: "Back");
    Label messageLabel = new Label();
    Label messageLabe = new Label();

    SearchUserLabel.setFont(new Font( s: "Arial", v: 20));
    TextFieldLabel.setFont(new Font( s: "Arial", v: 16));
    messageLabel.setFont(new Font( s: "Arial", v: 16));
    messageLabe.setFont(new Font( s: "Arial", v: 16));

    ID.setPrefWidth(200);
    backButton.setPrefWidth(150);
    SearchButton.setPrefWidth(150);

    GridPane SearchUser = new GridPane();
    SearchUser.setHgap(10);
    SearchUser.setVgap(10);
    SearchUser.setPadding(new Insets( v: 20));
    SearchUser.setAlignment(Pos.CENTER);
    SearchUser.add(SearchUserLabel, i: 1, i1: 0);
    SearchUser.add(TextFieldLabel, i: 0, i1: 1);
    SearchUser.add(ID, i: 1, i1: 1);
    SearchUser.add(SearchButton, i: 0, i1: 5);
    SearchUser.add(backButton, i: 1, i1: 5);
```

```

SearchUser.add(messageLabel, i: 0, i1: 3);
SearchUser.add(messageLabe, i: 1, i1: 4);

backButton.setOnAction(e -> {
    // Show Librarian dashboard
    showLibrarianDashboard(user, users, readers, books, librarians);
});

Scene SearchUserScene = new Scene(SearchUser, w: 400, h: 400);
stage.setScene(SearchUserScene);

SearchButton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {

        String id = ID.getText();
        User user = library.SearchU(id, users);
        if(user != null)
        {
            if(user.getType().equalsIgnoreCase(anotherString: "librarian"))
            {
                messageLabel.setText("User is a Librarian");
                messageLabe.setText("");
            }
            else
            {
                {

```

```

{
    Reader r1 = (Reader)user;
    Book Block = library.Block(r1);
    if (r1.isBlocked() == true) {
        messageLabel.setText("User already blocked");
        messageLabe.setText("");
    } else {

        if (Block != null) {
            user.setBlocked(true);
            messageLabel.setText("User is blocked");
            messageLabe.setText("");
            messageLabe.setText(Block.getRentdate().toString());
            ArrayList<Book> books5 = r1.getOrderList();
            Iterator<Book> iterator = books5.iterator();
            while (iterator.hasNext()) {
                Book book2 = iterator.next();
                book2.setIs_available(true);
                books5.remove(book2);
            }
        } else {
            messageLabel.setText("OrderList is empty\n" +
                "/no late books found");
            messageLabe.setText("");
        }
    }
}

```

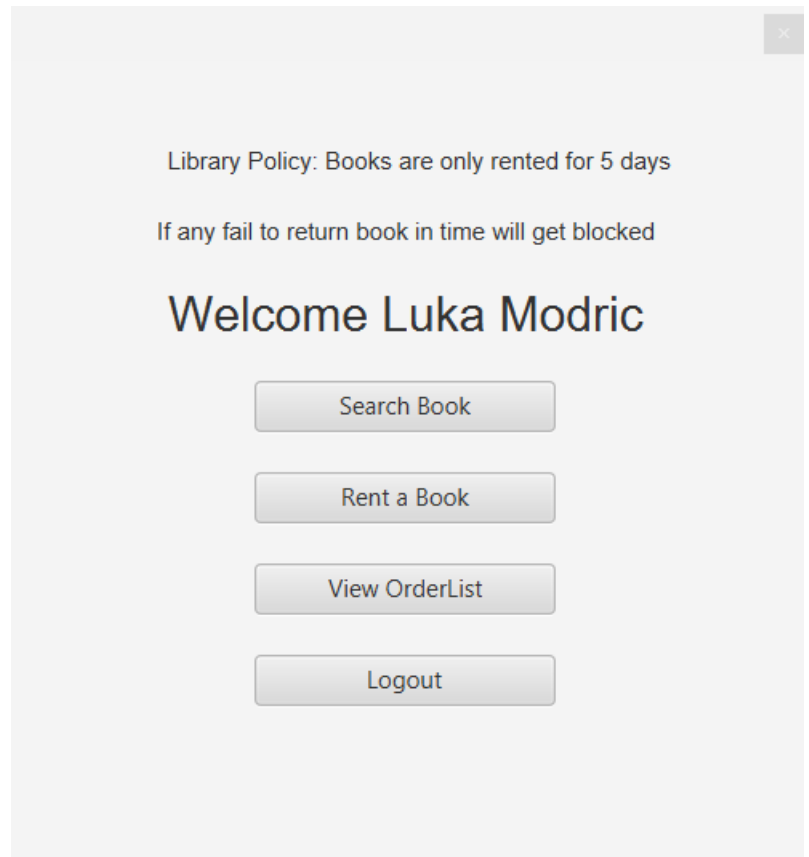
```

else
{
    messageLabel.setText("User doesn't exist");
    messageLabe.setText("");
}

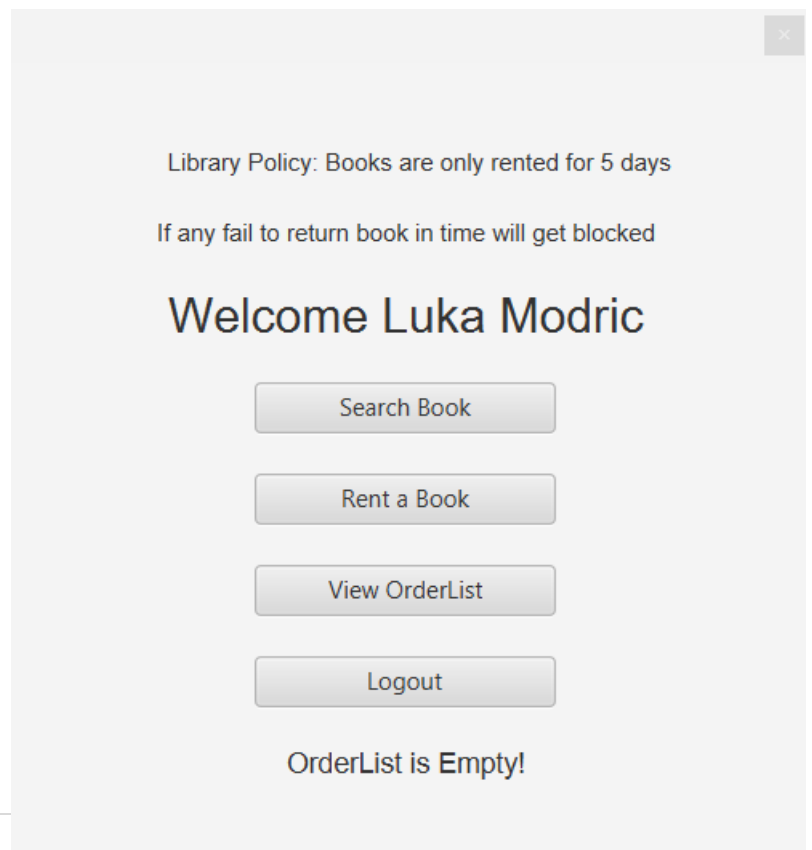
```

3.6.10. Reader Dashboard

Reader Dashboard which states the policy of library



Cannot access an empty order list



Code: -

```
private void showReaderDashboard(User user, ArrayList<User> users, ArrayList<Reader> readers,
                                ArrayList<Book> books, ArrayList<Librarian> librarians)
{
    // Initialize UI components for reader dashboard
    Label NewLabel = new Label( s: "    Library Policy: Books are only rented for 5 days");
    Label NewLabel1 = new Label( s: "If any fail to return book in time will get blocked");
    Label readerLabel = new Label( s: "Welcome " + user.getFirstName() + " " + user.getLastName());
    Button searchButton = new Button( s: "Search Book");
    Button RentBook = new Button( s: "Rent a Book");
    Button OrderList = new Button( s: "View OrderList");
    Button logoutButton = new Button( s: "Logout");
    Label lbl = new Label();

    // Set UI component properties
    NewLabel.setFont(new Font( s: "Arial", v: 12));
    NewLabel1.setFont(new Font( s: "Arial", v: 12));
    readerLabel.setFont(new Font( s: "Arial", v: 24));
    lbl.setFont(new Font( s: "Arial" , v: 14));
    searchButton.setPrefWidth(150);

    OrderList.setPrefWidth(150);
    RentBook.setPrefWidth(150);
    logoutButton.setPrefWidth(150);

    // Create layout for reader dashboard
    VBox readerBox = new VBox( v: 20, NewLabel , NewLabel1, readerLabel, searchButton,
                               RentBook, OrderList, logoutButton ,lbl);
    readerBox.setAlignment(Pos.CENTER);
}
```

```

searchButton.setOnAction(e -> {

    if(books.size() == 0)
    {
        lbl.setText("No Books exists");
    }
    else
    ShowSearchBook(user, users, readers, books, librarians);
});

RentBook.setOnAction(e ->
{
    if(books.size() == 0)
    {
        lbl.setText("No Books exists");
    }
    else
    ShowRentBookScene(user, users, readers, books, librarians);
});

OrderList.setOnAction(e -> {
    Reader r1 = (Reader)user;
    if(r1.getOrderList().size() == 0)
    {

        lbl.setText("OrderList is Empty!");
    }
    else
    ShowOrderList(user, users, readers, books, librarians);
});

```

```

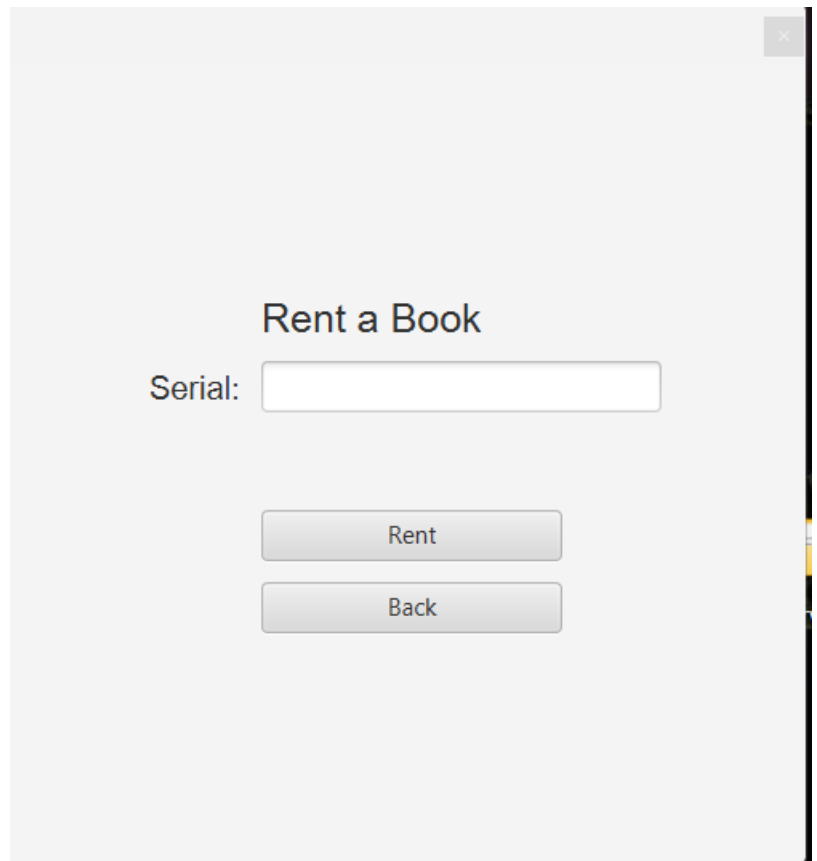
logoutButton.setOnAction(e -> {
    // Show login scene
    stage.setScene(loginScene);
    stage.show();
});

// Set reader dashboard as the root of the scene
Scene readerScene = new Scene(readerBox, 400, 400);
stage.setScene(readerScene);
}

```

3.6.11. Rent Book

Rent book through serial, if user not blocked



A screenshot of a web application window titled "Rent a Book". The window has a light gray background and a close button in the top right corner. The title "Rent a Book" is centered at the top. Below the title, the label "Serial:" is followed by a text input field. Below the input field, there are two buttons: "Rent" and "Back", stacked vertically.

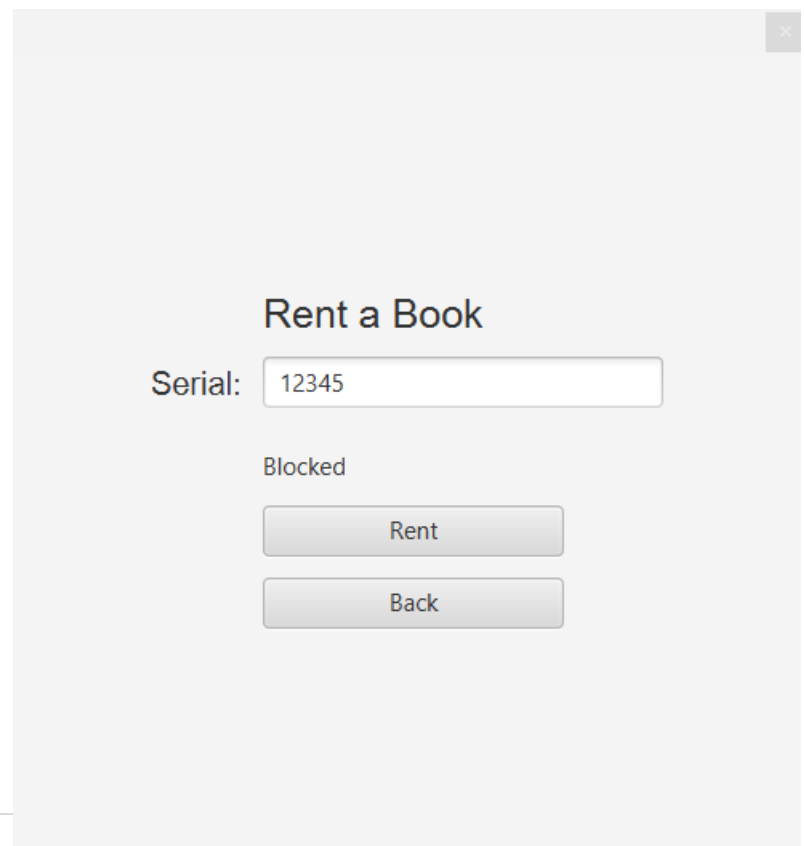
Rent a Book

Serial:

Rent

Back

System will not allow blocked user to rent books



A screenshot of a web application window titled "Rent a Book". The window has a light gray background and a close button in the top right corner. The title "Rent a Book" is centered at the top. Below the title, the label "Serial:" is followed by a text input field containing the value "12345". Below the input field, the word "Blocked" is displayed. Below "Blocked", there are two buttons: "Rent" and "Back", stacked vertically.

Rent a Book

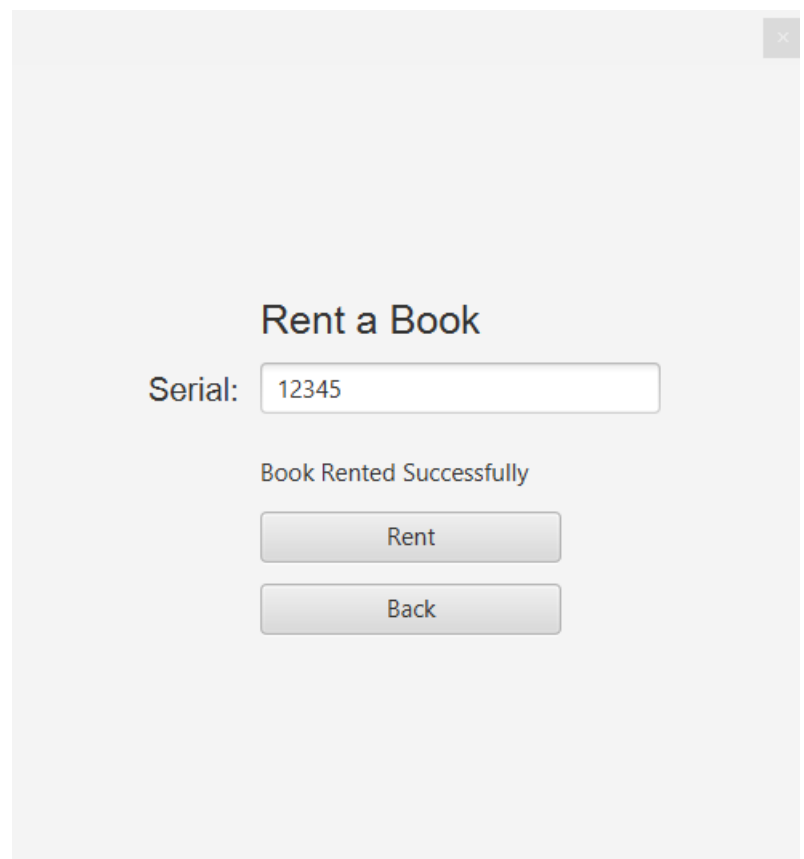
Serial:

Blocked

Rent

Back

Successfully renting a book



A screenshot of a web application window titled "Rent a Book". The window has a light gray background and a close button in the top right corner. The main content area is white. At the top, the title "Rent a Book" is displayed in a bold, black font. Below the title, the text "Serial:" is followed by a text input field containing the value "12345". Underneath the input field, the message "Book Rented Successfully" is shown in a smaller, gray font. At the bottom of the dialog, there are two buttons: "Rent" and "Back", both with a light gray background and rounded corners.

Rent a Book

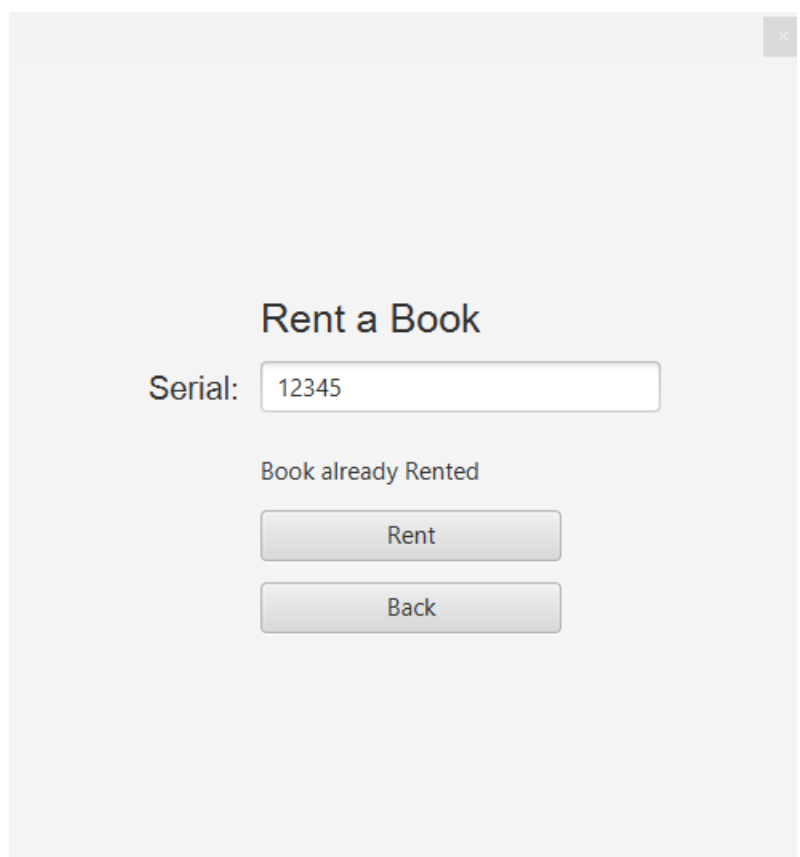
Serial: 12345

Book Rented Successfully

Rent

Back

Cannot rent an already rented book



A screenshot of a web application window titled "Rent a Book". The window has a light gray background and a close button in the top right corner. The main content area is white. At the top, the title "Rent a Book" is displayed in a bold, black font. Below the title, the text "Serial:" is followed by a text input field containing the value "12345". Underneath the input field, the message "Book already Rented" is shown in a smaller, gray font. At the bottom of the dialog, there are two buttons: "Rent" and "Back", both with a light gray background and rounded corners.

Rent a Book

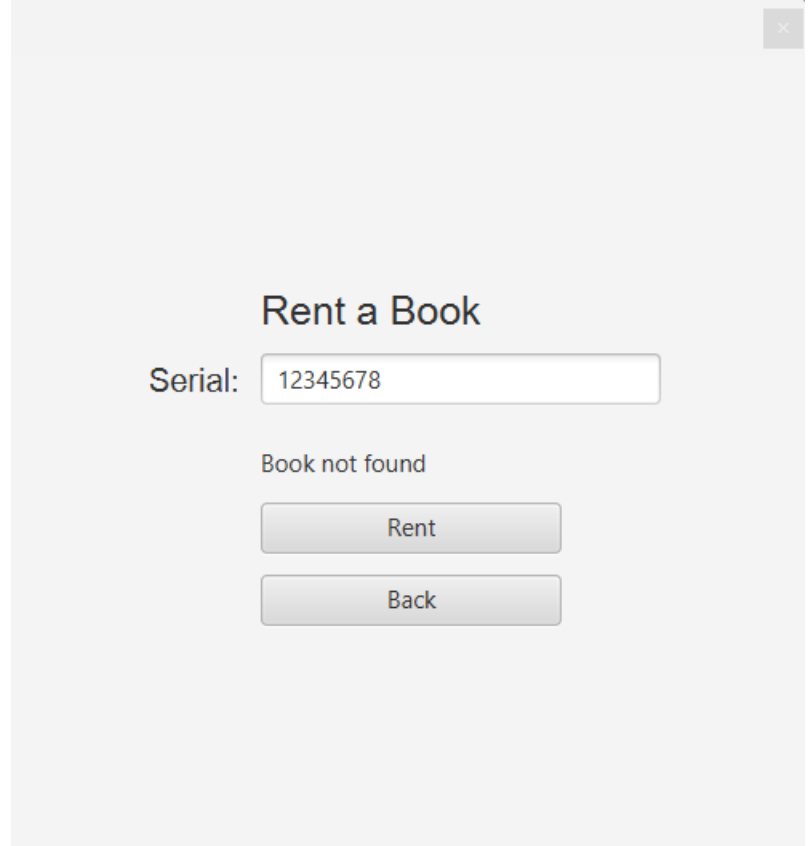
Serial: 12345

Book already Rented

Rent

Back

If book doesn't exist



Rent a Book

Serial: 12345678

Book not found

Rent

Back

Code :-

```
private void ShowRentBookScene(User user , ArrayList<User> users , ArrayList<Reader> readers
    , ArrayList<Book> books , ArrayList<Librarian> librarians)
{
    Label SearchBookLabel = new Label( s: "Rent a Book");
    Label TextFieldLabel = new Label( s: "Serial:");
    TextField Serial = new TextField();
    Button SearchButton = new Button( s: "Rent");
    Button backButton = new Button( s: "Back");
    Label messageLabel = new Label();

    SearchBookLabel.setFont(new Font( s: "Arial", v: 20));
    TextFieldLabel.setFont(new Font( s: "Arial", v: 16));
    Serial.setPrefWidth(200);
    backButton.setPrefWidth(150);
    SearchButton.setPrefWidth(150);

    GridPane SearchBook = new GridPane();
    SearchBook.setHgap(10);
    SearchBook.setVgap(10);
    SearchBook.setPadding(new Insets( v: 20));
    SearchBook.setAlignment(Pos.CENTER);
    SearchBook.add(SearchBookLabel, i: 1, i1: 0);
    SearchBook.add(TextFieldLabel, i: 0, i1: 1);
    SearchBook.add(Serial, i: 1, i1: 1);
    SearchBook.add(SearchButton, i: 1, i1: 4);
    SearchBook.add(backButton, i: 1, i1: 5);
    SearchBook.add(messageLabel, i: 1, i1: 3);
}
```

```

backButton.setOnAction(e -> {
    // Show librarian dashboard
    showReaderDashboard(user, users ,readers , books , librarians);
});

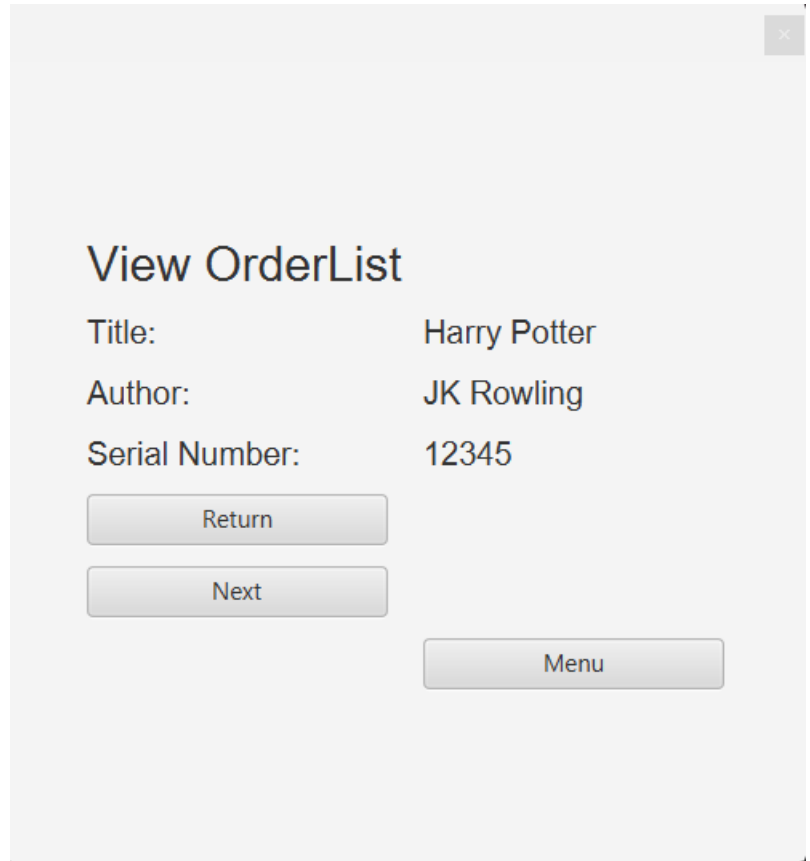
Scene SearchBookScene = new Scene(SearchBook, w: 400, h: 400);
stage.setScene(SearchBookScene);

SearchButton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        String SerialAns = Serial.getText();
        Book book = library.SearchBook(SerialAns , books);
        if (book != null)
        {
            if(book.Is_available() == true && user.isBlocked() == false)
            {
                book.setIs_available(false);
                book.setRentdate(new Date());
                Reader r1 = (Reader)user;
                r1.addToOrderList(book);
                messageLabel.setText("Book Rented Successfully");
            }
            else if(book.Is_available() == false){
                messageLabel.setText("Book already Rented");
            }
            else if(user.isBlocked() == true)
            {
                messageLabel.setText("Blocked");
            }
        }
    }
});
else {
    messageLabel.setText("Book not found");
}

```

3.6.12. View Order list

User can view all his rented books and scroll
Through them , as well as return them.

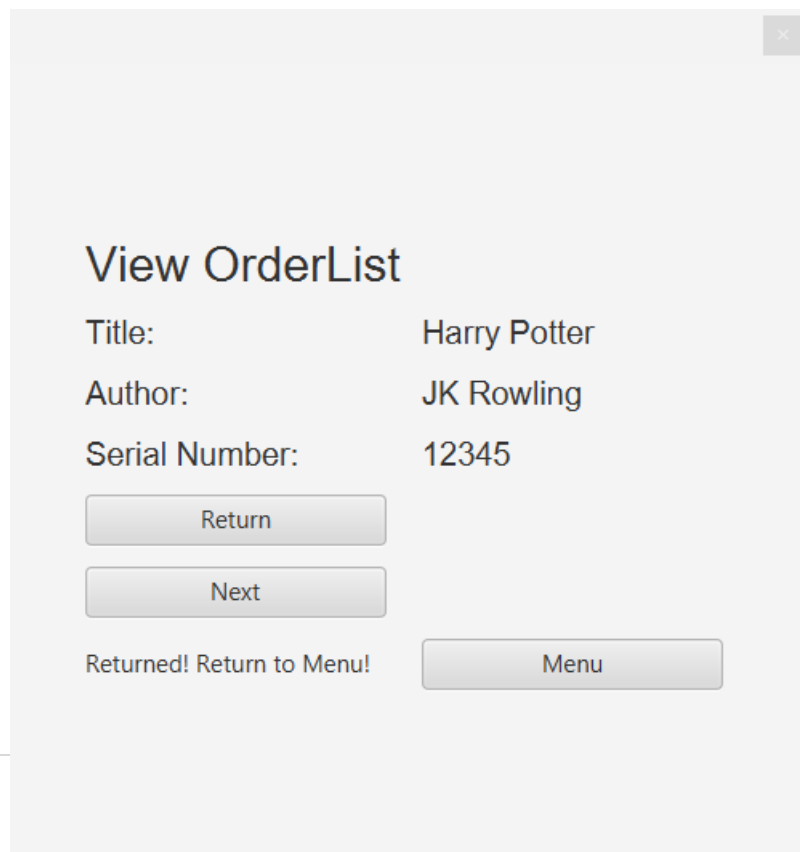


A screenshot of a software window titled "View OrderList". The window has a light gray background and a close button in the top right corner. It displays the following information:

Title:	Harry Potter
Author:	JK Rowling
Serial Number:	12345

Below the table, there are three buttons: "Return", "Next", and "Menu". The "Return" and "Next" buttons are stacked vertically on the left, and the "Menu" button is on the right.

When user return a book , he must return to
Menu.

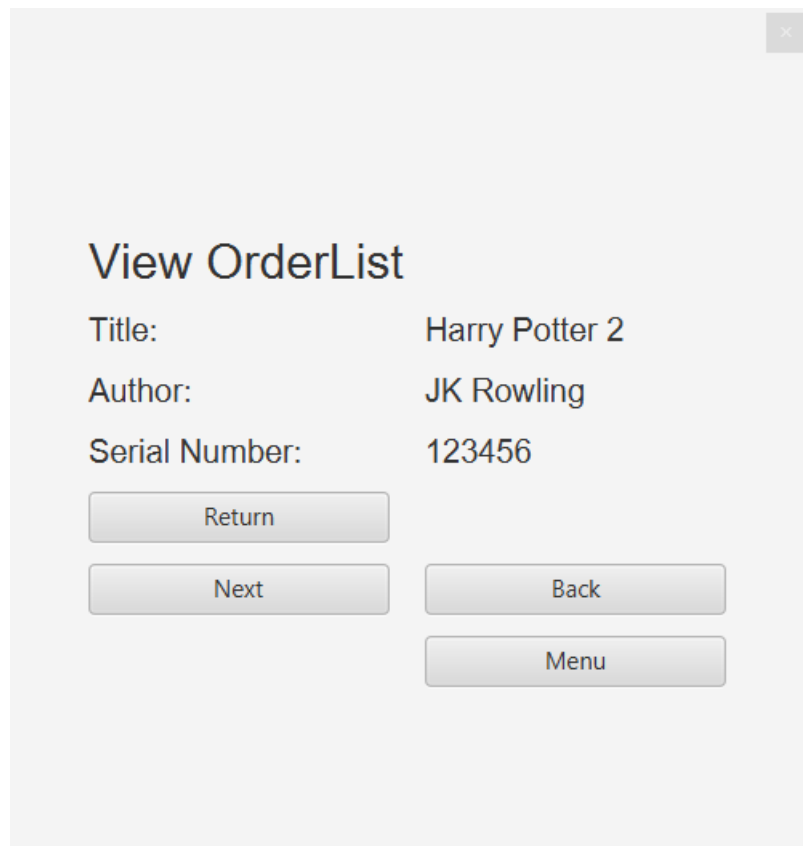


A screenshot of the same "View OrderList" window after a book has been returned. The window displays the same book details as the previous screenshot:

Title:	Harry Potter
Author:	JK Rowling
Serial Number:	12345

Below the table, there are three buttons: "Return", "Next", and "Menu". The "Return" and "Next" buttons are stacked vertically on the left. A new message, "Returned! Return to Menu!", is displayed below the "Return" and "Next" buttons. The "Menu" button is on the right.

User can scroll through his order list



Code :-

```
private void ShowOrderList( User user, ArrayList<User> users, ArrayList<Reader> readers,
                           ArrayList<Book> books, ArrayList<Librarian> librarians)
{
    Reader reader = (Reader)user;
    Book books2 = reader.getOrderList().get(iteration);
    int x = reader.getOrderList().size();

    Label addBookLabel = new Label( s: "View OrderList");
    addBookLabel.setFont(new Font( s: "Arial", v: 24));

    Label titleLabel = new Label( s: "Title:");
    Label authorLabel = new Label( s: "Author:");
    Label isbnLabel = new Label( s: "Serial Number:");
    Label titleField = new Label(books2.getTitle());
    Label authorField = new Label(books2.getAuthor());
    Label isbnField = new Label(books2.getSerial());
    Label txt = new Label();
    // Set UI component properties

    titleLabel.setFont(new Font( s: "Arial", v: 16));
    authorLabel.setFont(new Font( s: "Arial", v: 16));
    isbnLabel.setFont(new Font( s: "Arial", v: 16));
    titleField.setFont(new Font( s: "Arial", v: 16));
    authorField.setFont(new Font( s: "Arial", v: 16));
    isbnField.setFont(new Font( s: "Arial", v: 16));

    Button backButton = new Button( s: "Back");
    Button Next = new Button( s: "Next");
```

```

Button MainMenu = new Button( s: "Menu");
Button Return = new Button( s: "Return");
backButton.setPrefWidth(150);
MainMenu.setPrefWidth(150);
Next.setPrefWidth(150);
Return.setPrefWidth(150);
GridPane addBookForm = new GridPane();
addBookForm.setHgap(10);
addBookForm.setVgap(10);
addBookForm.add(titleLabel, i: 0, i1: 1);
addBookForm.add(titleField, i: 1, i1: 1);
addBookForm.add(authorLabel, i: 0, i1: 2);
addBookForm.add(authorField, i: 1, i1: 2);
addBookForm.add(isbnLabel, i: 0, i1: 3);
addBookForm.add(isbnField, i: 1, i1: 3);
addBookForm.add(Return, i: 0, i1: 4);
addBookForm.setPadding(new Insets( v: 20));
addBookForm.setAlignment(Pos.CENTER);
addBookForm.add(addBookLabel, i: 0, i1: 0);
addBookForm.add(txt, i: 0, i1: 6);
    if(iteration < x-1)
        addBookForm.add(Next , i: 0 , i1: 5);

addBookForm.add(MainMenu , i: 1 , i1: 6);

if(iteration > 0)
    addBookForm.add(backButton, i: 1, i1: 5);

```

```

Next.setOnAction(e -> {
    iteration++;
    ShowOrderList(user, users, readers, books, librarians);
});
MainMenu.setOnAction(e -> {
    // Show librarian dashboard
    iteration = 0;
    showReaderDashboard(user, users, readers, books, librarians);
});
backButton.setOnAction(e -> {

    iteration--;
    ShowOrderList(user, users, readers, books, librarians);
});
Return.setOnAction(e -> {
    reader.getOrderList().remove(books2);
    books2.setIs_available(true);
    txt.setText("Returned! Return to Menu!");
});

```

```

// Set add book scene as the root of the scene
Scene addBookScene = new Scene(addBookForm, v: 400, v1: 400);
stage.setScene(addBookScene);

```