



Kwantlen Polytechnic University

School of Business

INFO 2413: System Development Project (S50)

August 8, 2022

Library Management System

Team 7

- **Navneet Kaur Gill (100400577)**
- **Sarbjot Kaur (100394993)**
- **Kirandeep Sahota (100400613)**
- **Abhijot Saini (100405844)**
- **Jaskaran Singh (100399880)**

Table of Contents

Library Management System	1
Team 7	1
1.1 Overview	4
1.2 Goals and Objectives	5
2.1 Software Environment	7
2.2 Hardware Environment	7
2.3 Other Constraints	7
NON-FUNCTIONAL REQUIREMENTS	8
3.1 Operational Requirements	8
3.2 Performance Requirements	9
3.3 Security Requirements	9
3.4 Other Requirements	10
4.1 Library System (Use Case 1)	12
Project Plan	14
START PLAN	14
5.1 Team Organization	14
5.2 Development Tool Selection	14
WORK PLAN	15
6.1 Work Breakdown Structure - WBS	15
6.2 Gantt Chart	16
PROJECT DELIVERABLES	17
7.1 SOFTWARE	17
7.2 USER GUIDE	17
7.2.1 Getting Started for the first time:	17
7.2.2 Locating the Book's Location:	17
7.2.3 Finding the borrower list:	17
7.2.3 Registering new reader:	18
Project Design	19
STRUCTURAL DESIGNS	19
8.1 Model-View-Controller (top-level)	19

8.2 Class Diagrams (top-level)	20
BEHAVIORAL DESIGN	21
9.1 Activity Diagrams (top-level)	21
WIREFRAMES	22
Wireframes for the top-level screens (GUI).....	22
Sign In Page:	23
Sign Up Page:	24
Add book Wireframe:	25
Add Scholar Wireframe:.....	26
Project Implementation	27
Class Diagrams:	27
Test Results	28
Login page:	28
Login window: Username and password field checking Validation	29
Librarian Menu:.....	31
Issue Book: Did not issued the book, as it was not availabale, i.e., already borrowed	32
Add Book:.....	33
.....	33
Book Added into the database:	33
Borrow History:.....	37
User Guide	38
Conclusion.....	39
Reference	39
Appendix	40

Project Description

INTRODUCTION

1.1 Overview

A library management system is software built expressly to manage all the functions of a library. It is beneficial for the librarian to keep track of new books as well as books that members have borrowed and the dates on which they must be returned.

This solution will completely automate all the operations in your library. The most effective technique for handling, arranging, and systematically maintaining many books is to use library management system software.

A library database management system is used to keep library records organized. It keeps track of the overall quantity of books in the library, how many books are checked out, how many books are returned, renewed, or charged late penalties, and so on.

You will be able to identify books in seconds, issue, and reissue books quickly, and manage all data in an efficient and orderly manner with this system. The major goal of a library management system is to make it easier to provide accurate and comprehensive data about any type of book in real-time, minimizing the amount of time and effort required.

1.2 Goals and Objectives

1. Obtaining each user's credentials before accessing the application which would log in from an employee.
2. Encrypting passwords before storing them in the database to ensure that they are saved safely.
3. Creating an efficient database to save storage waste and maintain data stability.
4. Creating a validation mechanism to save only authentic data to prevent the system from crashing.
5. There would be a sign-up screen where after validation the manager could create a new account for the employee.
6. There would be a home screen with all book information displayed, as well as a search bar on top where a user could search for what they needed.
7. The search bar would have a filter option where a user could arrange things according to their needs, and everything would be displayed in a card view style.
8. There would be a menu with extra options such as borrowing, returning, adding a new book, adding a new reader, inventory status, and persons in the library.
9. To add a new reader to the system, complete information would be collected in exchange for a fee.
10. There would be a page where all users with the book details would be displayed who would not return their books on time and would add some fines associated with their reader ID.

11. The current inventory status page would display all the low goods in the library to keep track of stock.

12. People in the library option would enter the information into the system with a timestamp for each user who entered the library.

13. All information about books, including the number of books on hand, their location in the library, the names and authors of the books, and the year the book was released, will be displayed on the home screen.

14. There would be a sign-out option in the menu to keep the machine in safe hands.

GENERAL DESIGN CONSTRAINTS

2.1 Software Environment

Java, JavaFX, and MySQL are among the programming languages used to construct the application. The software will go through numerous steps to generate an application that can run on its own. As a result, the specific user requires a packet file that supports numerous coding languages for the software to function properly.

Development Environment: The Eclipse IDE will be used to create the application. The program should be written in java and have a JavaFX interface, according to the project's requirements. The programming IDE will be Eclipse because the team is already proficient with it.

Database: The database will store user and book information and will be linked to the application. The login credentials will also be saved in the database. Users must authenticate credentials before they can access the system.

2.2 Hardware Environment

It is feasible that different people (devices) will use distinct types of hardware. The software should have a user-friendly interface. To be functional, it must be compatible with a wide range of tools and equipment. However, to produce software that is compatible with other pieces of software and procedures, it may be required to use a certain piece of hardware.

2.3 Other Constraints

Ambiance: The environment also affects the performance of the software because applications' performance or dependability might be harmed by factors like heat, humidity, and water exposure. Additionally, an application's environment can include the schedule on which it runs, for example,

if the software runs 24 hours a day, it might affect the system hardware by producing more heat as a result consumption of the battery will be doubled.

Localization: The software can be deployed in different countries as a result it requires some feature in the device to work. It requires system support:

- Language
- Time zones.

NON-FUNCTIONAL REQUIREMENTS

3.1 Operational Requirements

Maintainability: This software is easy to use and does not necessitate the usage of third-party maintenance services. Users may make modifications to their software by accessing the settings.

Serviceability: Because it does not need as much maintenance and stays updated with time, it's considerably easier to use.

Recoverability: When a system crashes or fails, the capacity to recover and return to normal operations is called "recoverability. As a result, after 2-3 seconds of inactivity, it will autosave whatever files the user has opened. It will transfer all the user's data to the internet backup system if the user's machine is connected to the internet.

Usability: It can be used by different users just to log in to the system with their credentials. Moreover, two or more users can work or use the software at the same time. Just by login into different credentials.

3.2 Performance Requirements

The software system's performance criteria define how well it can accomplish specific tasks in line with predefined specifications. Students are the primary users of the library system, and the chances of them using it are likely to be high. Students make up the majority of the library's patrons. As a result, this system's reaction time, throughput, execution time, and storage capacity should all be optimized to their maximum capability. In addition, we want to include several features that will enhance the software's credibility. If there is a demand in the future, additional space can be made available.

- All functions, including registration, sending, and receiving messages, will be available in under three seconds.
- After 24 hours have passed since the previous database backup, the server will automatically perform the next database backup. As a result, it will keep track of the user's information and assist them in the future.
- The software will be capable of supporting a minimum of 50,000 concurrent users.
- A customer support team is available to assist users with the program's operation, and that team also ensures that the service is delivered successfully.

3.3 Security Requirements

There will be numerous organizations in our program that will not only keep the system secure but will also keep our users' data protected. Our software will feature the following characteristics:

Maintaining legal limitations on who can access and exchange information, as well as safeguarding personal privacy and confidential data.

Authenticity: The process of making sure that an information system can trust the identities of users that are given to it electronically. Therefore, the software can easily differentiate the users from each other, and other users cannot access other useful information.

Availability: Making sure that people can get and use information in a timely and reliable way. It means our software will also have an offline option; the user can download the PDF file. When the device is not accessing the Internet.

Accountability: A process that controls how users' processes or other systems can use system resources based on a security policy. Only authorized users, processes, or other systems can use system resources. It means all users are differentiated by their position information. For example

- The students only access the information about the book and their Performa and cannot upload any book or change information about anything whereas a library assistant can change anything like upload new books, PDF files, and many,y more.

Integrity: Keeping information from being changed or deleted without permission and making sure that information cannot be disputed and is real. It means no one can change the information of any individual until the user changes their own.

3.4 Other Requirements

Maintenance: This program is simple to use and does not require the use of outside maintenance services. The program is simple to use, and users can change the parameters to customize their software.

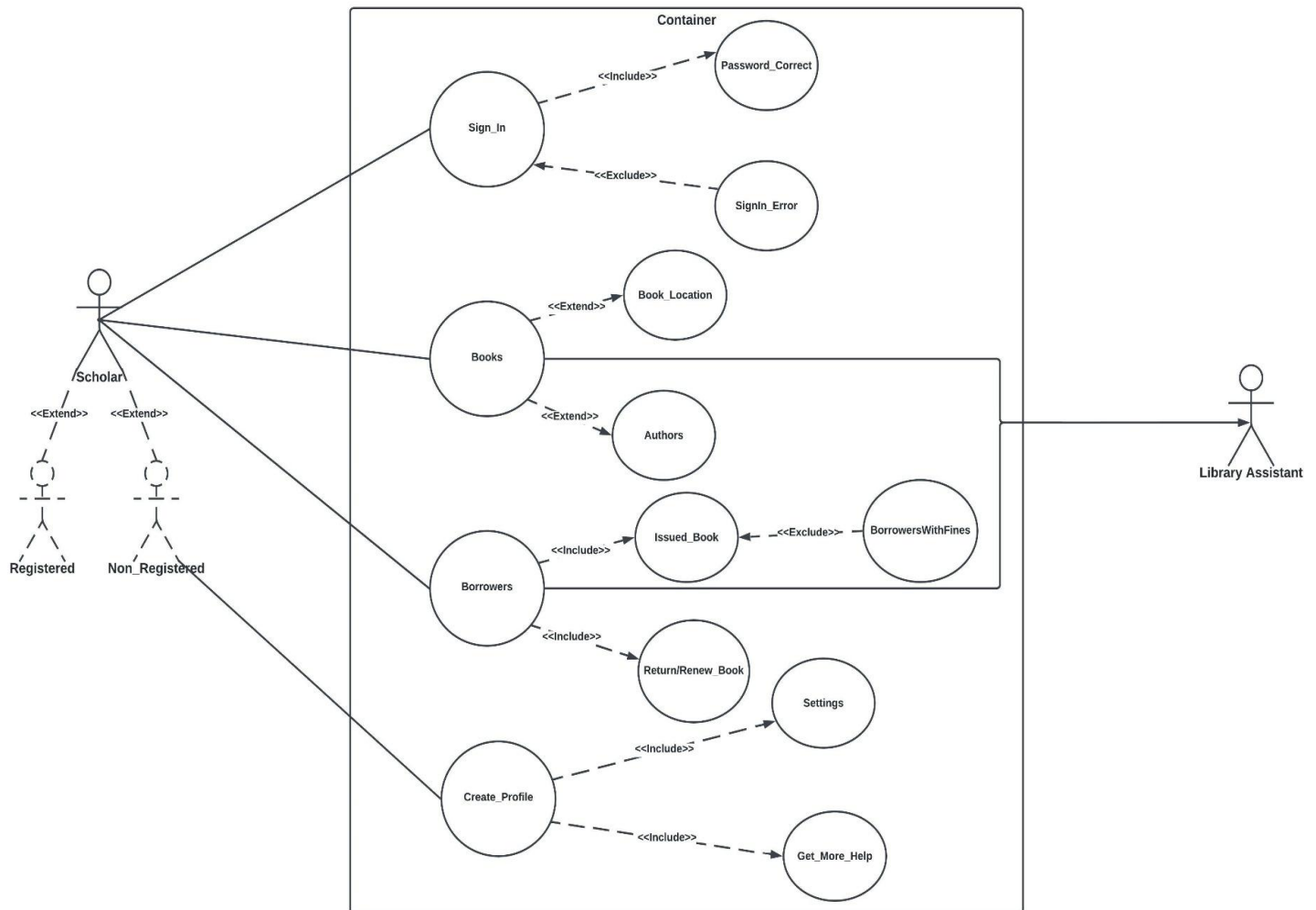
Serviceability - It is a lot easier to use because it doesn't require as much maintenance and stays updated over time.

When a system breaks or fails, the ability to recover and resume regular operations is referred to as "recoverability." As a result, it will automatically save whatever files the user has opened after 2-3 seconds of inactivity. If the user's machine is connected to the internet, it will transfer all the user's data to the online backup system.

Speed: The speed of an application's response to commands is determined by its speed. If you were to enter a word into a search engine, for instance, the speed of the engine would affect how quickly you would receive the results of your search. Evaluation of a system's speed also involves determining how well it can manage an increasing amount of work while multiple apps are being used at the same time.

FUNCTIONAL REQUIREMENTS

4.1 Library System (Use Case 1)



We anticipate that scholars will utilize our software to borrow books from the internet and read them. On-registered and registered users alike will be able to use the service. The first thing a user must do to gain access to the library's system signs up for an account. When a user enters the right

password, the system grants access. Otherwise, the user will be able to reset their password. Books, Borrowers, and Profile will be accessible as soon as they log in.

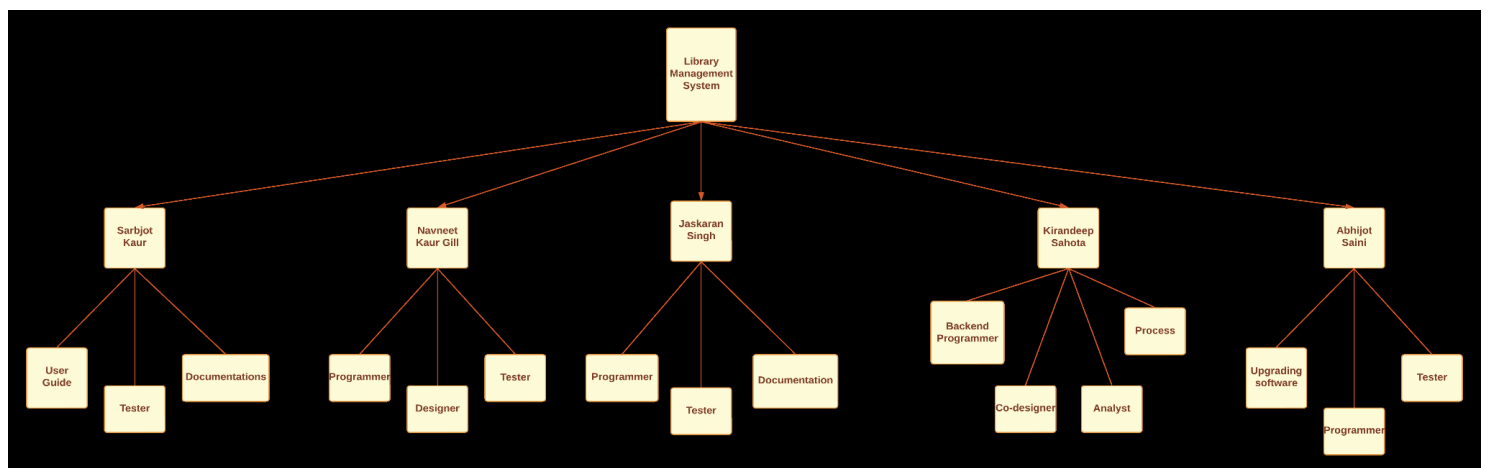
In the Books part, you will find information about books, such as their location and authors. In the Borrowed section, you will find information on which book the user has borrowed and when it will be returned. A fine will be assessed against the registered user whose return deadline has passed. Personal information, such as a user's name, university, home address, and mailing address, can be included in their profile. Along with this, it also has a setting and help option in which users can make custom settings to their software and ask for help from the IT team, they are confronted with technical problems. The library assistant will utilize the system as a super user which will control the whole library system and borrowing of each book goes through the library assistant and he also has the responsibility of receiving the fine.

Project Plan

START PLAN

5.1 Team Organization

The fact that our team splits their work based on their skills and allocates each member to specialized responsibilities that they can correctly execute is the primary factor that propels the project to its final pace. In addition, throughout each assignment, the student chooses one of the tasks from the list to do. At the end of it all, these responsibilities were split up equally. Below chart having the details of the skills that will be used to carry out the tasks by each team member for the project.



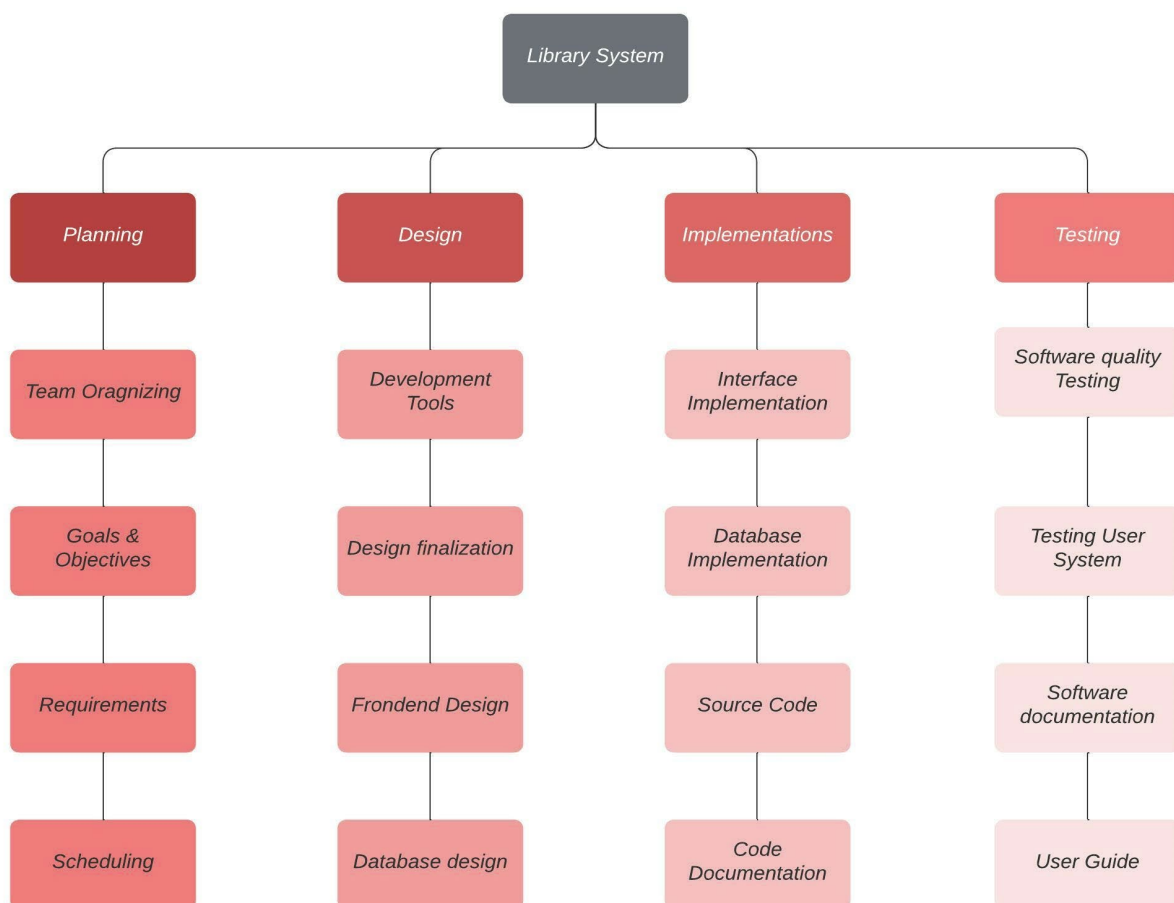
5.2 Development Tool Selection

- **Eclipse-** It is an integrated Development Environment (IDE), Eclipse is one of the most widely used Java development environments. It's free, open-source, and includes a large plugin ecosystem that allows users to modify application development functionality.
- **MySQL-** MySQL is a free, open-source database that connects databases to software to enable successful database management. It is a dependable, powerful, and stable solution with advanced features.

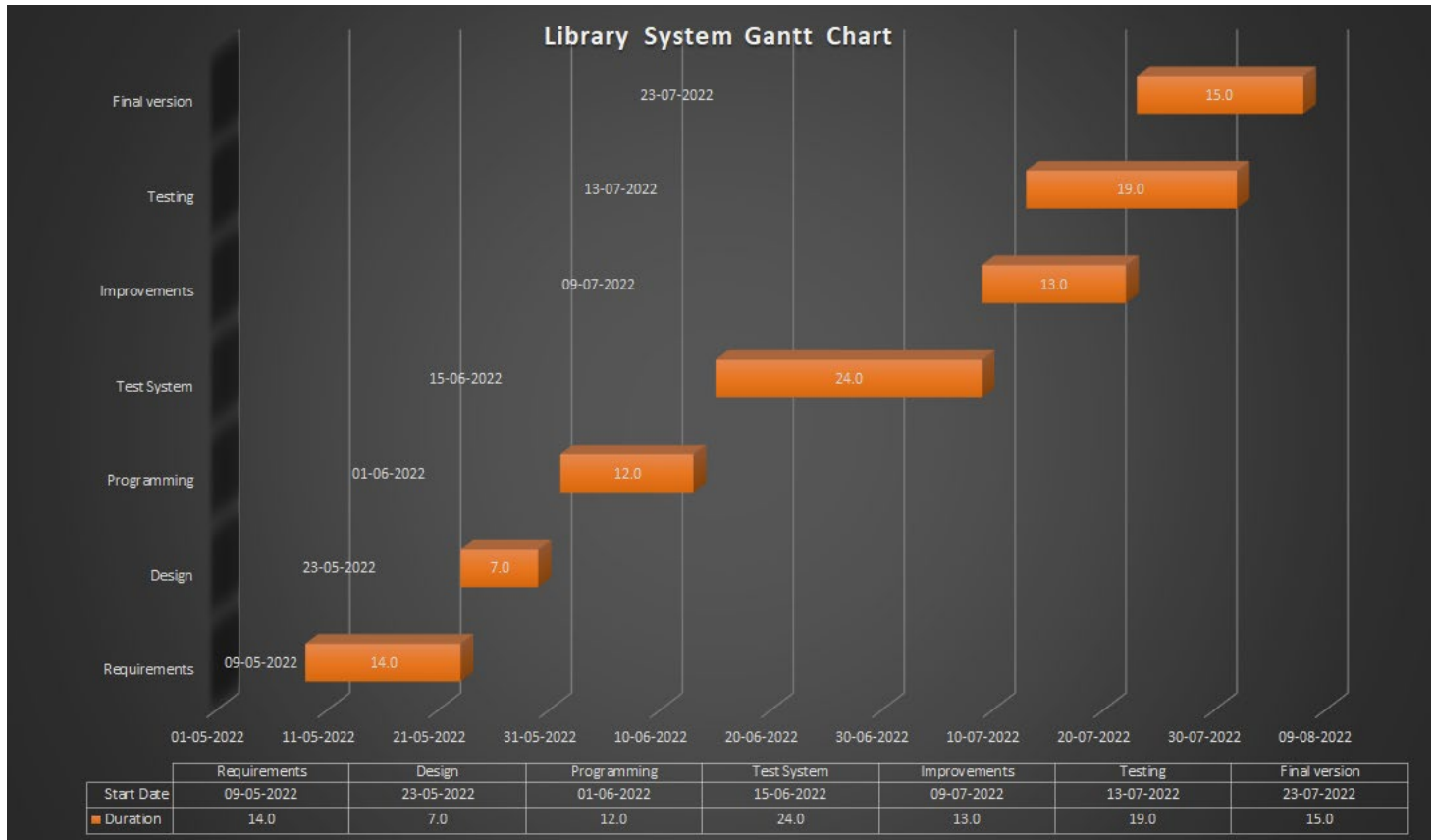
- **JDBC:** JDBC (Java Database Connectivity) is a Java API that allows you to connect to a database, issue queries and commands, and handle database result sets.

WORK PLAN

6.1 Work Breakdown Structure - WBS



6.2 Gantt Chart



PROJECT DELIVERABLES

7.1 SOFTWARE

The program is attempting to solve the problem of a library by developing a new management system capable of performing a variety of activities such as registering readers, managing book inventory by recording who borrowed the books and when they are due, and so on. It would be possible to charge the fee due to the late return of the book to the reader's account. There would be a screen with all the book's locations, including aisle numbers and sections. There would be a sign-up method that would allow a user to login into the main application and easily log out to protect the data.

7.2 USER GUIDE

7.2.1 Getting Started for the first time:

When you first start the application, a login screen will appear, where existing users can input their credentials to log in, but if you are a new user, follow the steps below:

1. To proceed to the next page, locate the Sign-up button and click it.
2. Enter your email address and create a new password, which you must confirm.
3. When the user clicks the sign-up button, the user is taken to the app's main screen.

7.2.2 Locating the Book's Location:

1. Log in to the program (If new user, please follow the 4.2.1).
2. Select Book Location from the main menu at the top of the page to access further alternatives.
3. Select the option and conduct a database search for the item.

7.2.3 Finding the borrower list:

1. Log in to the program (If new user, please follow the 4.2.1).
2. Go to the main menu and select Borrower List from the drop-down menu.
3. Select the option and look through the list of usernames.
4. Select a user to get a list of books that are overdue for that user.

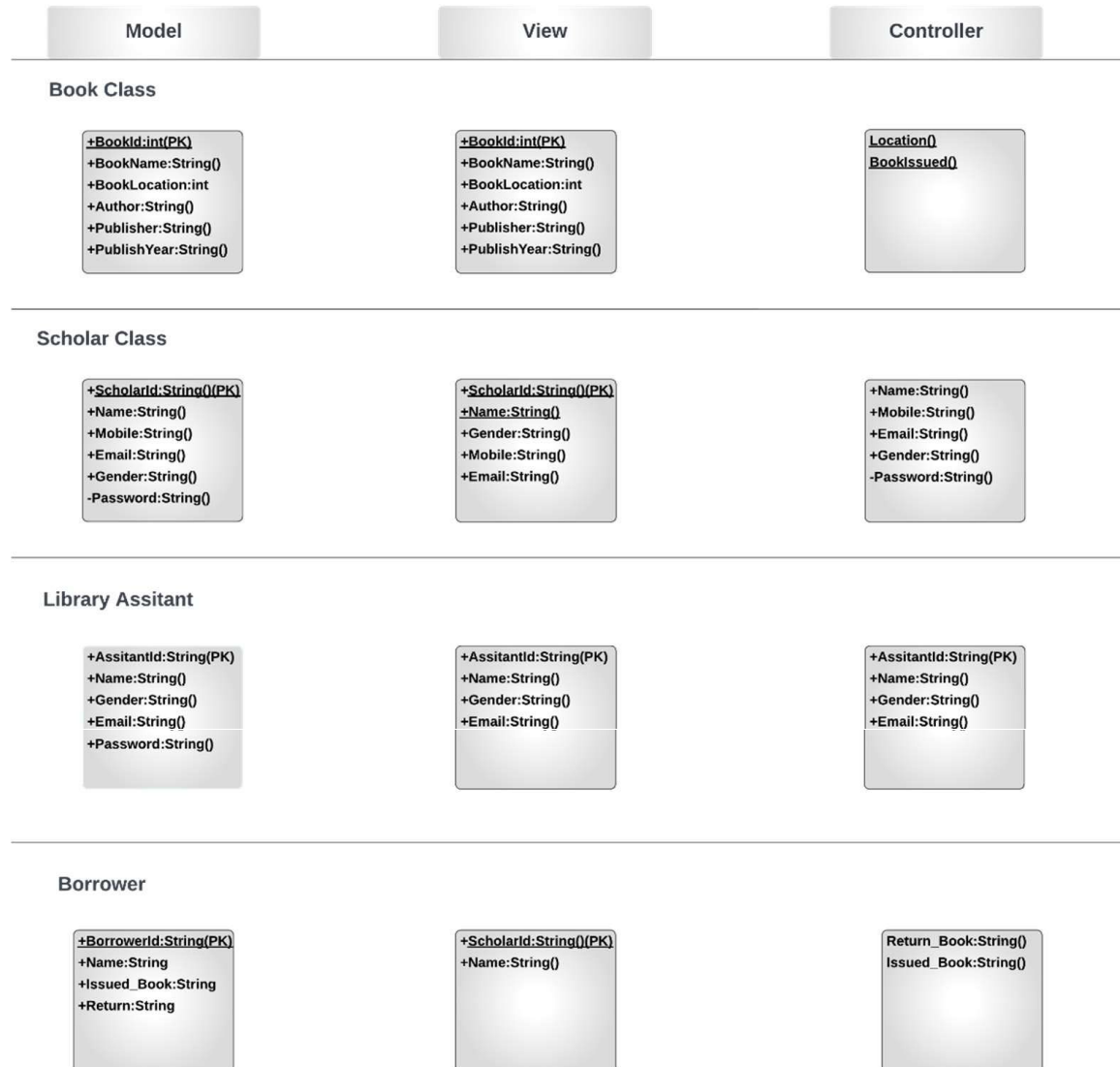
7.2.3 Registering new reader:

1. Log in to the program (If new user, please follow the 4.2.1).
2. Go to the main menu and select Borrower List from the drop-down menu.
3. Select the option and add information by clicking Add new reader button.
4. Enter your First Name, Last Name, and Address and would need to make a payment for
The registration process in the library.
5. Click on Register new user button and provide them their Library ID.

Project Design

STRUCTURAL DESIGNS

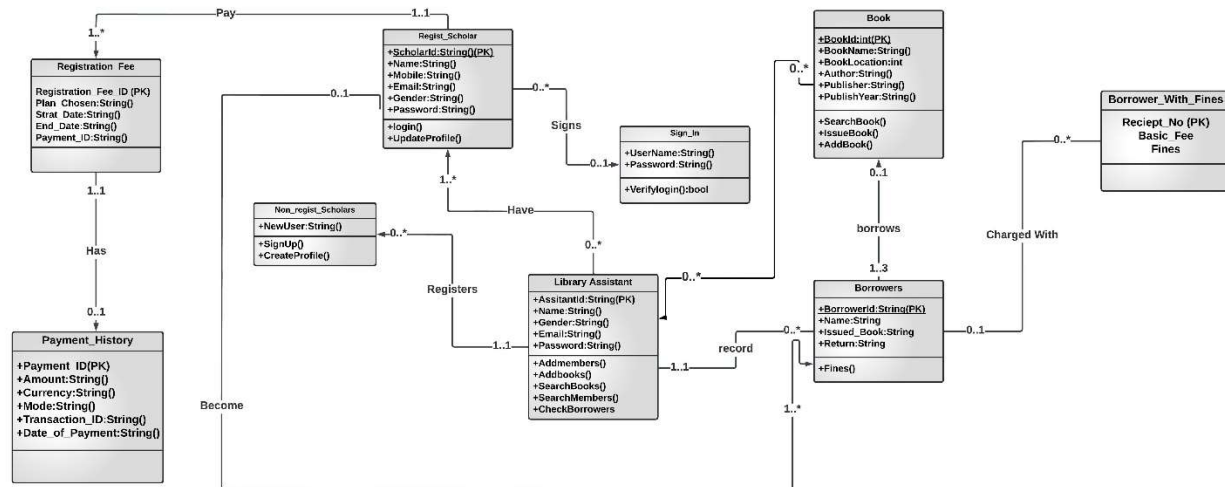
8.1 Model-View-Controller (top-level)



In the Library Management System, the main interaction will be between three classes: Books, RegistScholar and Library Assistant. The View will only cover the Books, RegistScholar and LibraryAssitant classes since these are only shown in the user interface, as shown in the sample of wireframes at the end of the document. The controller parts will update the view according to

their respective models. For example, the Controller for Books will connect the View and Model of Book Class and keep track of the location of books and their status, whether the book is issued or not. Since we there will be no other class involved in view, the updates will be done simply in the model through a direct method.

8.2 Class Diagrams (top-level)



The library Assistant keeps the basic information about the staff. It is associated with the entity 'Borrower' as it helps the **Regist_Scholar** to borrow books from the library and keep a record of them. From the relation, we know that it is optional participation for staff whereas mandatory for Borrower which implies that there can exist staff without having any borrowers to store record but on the other hand, there can't be any borrower whose information is not stored by any of the staff, also one staff member can record multiple borrowers' information.

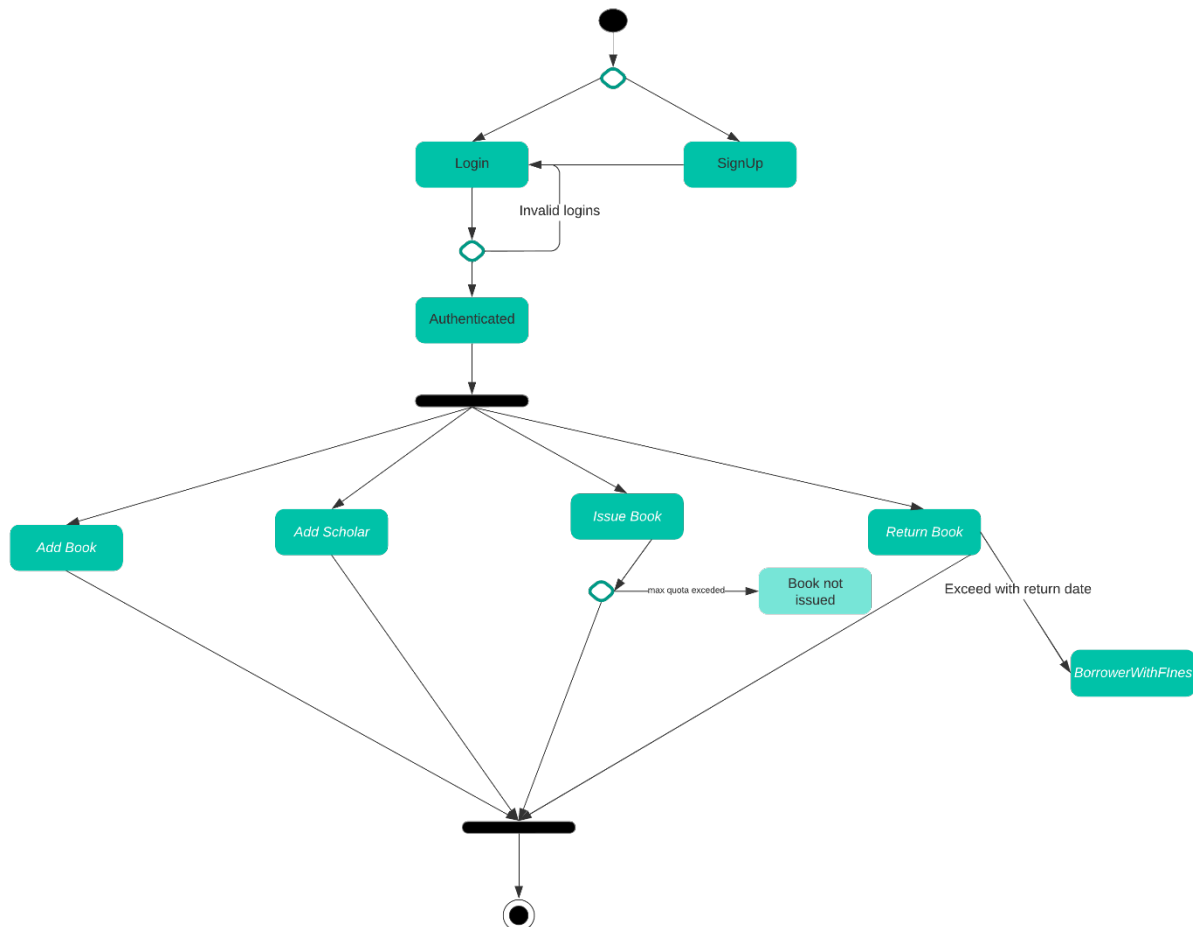
Regist_Scholar helps the system to know which scholars are registered. Only registered scholars can borrow the books. Only registered readers can borrow the book. It is quite clear that to be a Registered scholars, have to pay a registration fee monthly. The **Registration_Fee** class is responsible to collect the details like as which date the user has started his membership and when the membership would expire, which plan he has opted for. This class is associated with **Payment_History** class holds the data for all the transactions made in the past. It is related with 'Registration_Fee' to see if the transaction is done for registration fees that were allocated. From the Relation, we see that it has mandatory relation for payment_history but is optional for 'Registration_Fee' implies that there can exist registration fee records without having any payment history and if there exists a payment history, it could have multiple records. **Non Regist Scholar** class contains information about the user who entered the library but is not a registered, scholar. To borrow a book from the library, they have to first become registered Scholars. They are connected to the Library Assistant class. A library can create a profile for them.

Book class keeps the basic information about the books that are currently present in the library. Readers can search for books in this entity. As these are the book that can be borrowed so this entity is also related to the borrower to know which borrower borrowed a specific book.

Borrower class helps to find out which registered reader has borrowed a book from the library. Late returns would be charged with a fine that justifies their relationship with the class **'Borrower_With_Fines'** keeps the record of every borrower that is charged with a fine. The staff present at that moment would be responsible for keeping the record of the borrower so that defines its relationship with the Entity 'Staff'. From the relation between 'Borrower' and 'Borrower_With_Fines', we see that it has optional participation on both ends which implies that a borrower in June or June does not have a fine, and if they do, they can have multiple fines.

BEHAVIORAL DESIGN

9.1 Activity Diagrams (top-level)



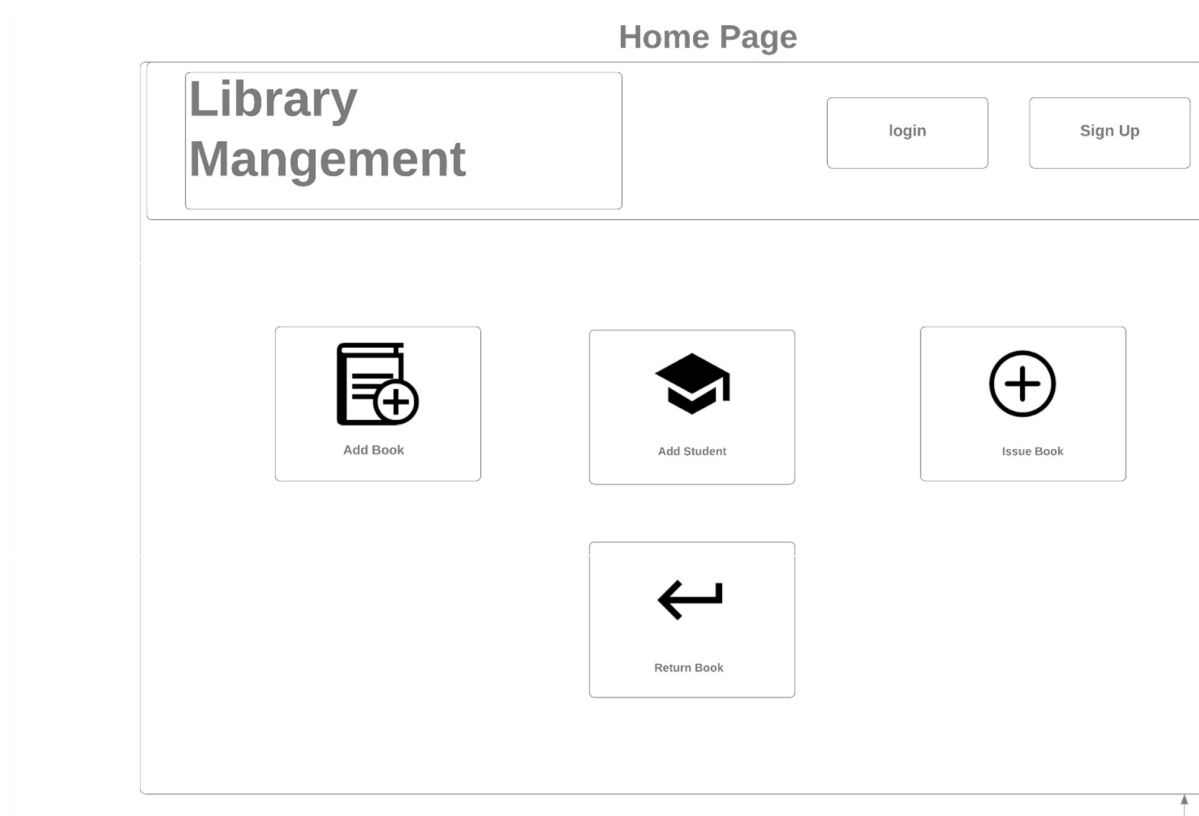
When a user signs in with their username and password, they are granted access to a higher level of functionality. Users must go back one step to log in if they enter an incorrect password. The scholar and library assistant will each have four options at the next level. Add book, issue book, add scholar, and return book are the most important steps. The borrower will be fined if the book is not returned on time.

WIREFRAMES

Wireframes for the top-level screens (GUI)

Wireframe of Homepage:

This is the main page of the wireframe by which the user can access all pages but to access this user needs to log in. Along with this, there is an option for adding a book, adding a student, Issuing a book, and returning a book.



Sign In Page:

If a user wants to access the database of the library, they need to login into the library with their credentials. They fill in their Username and Password to login into the system.

**Library
Mangement**

loginSign Up

Welcome
To
Library

Username

Text

Password

Text

Login

New User ?


Sign Up

Sign Up Page:

On this page, scholars can sign up for the library with their Student ID by which we differentiate Whether the individual belongs to a university or not. Moreover, there is more information on the page that requires filling in for sign-up.

Library Mangement

[login](#)[Sign Up](#)

Sign Up 

First Name

Last Name

Id

Username

Password

Sign Up

Add book Wireframe:

If there is any new book in the library or it is not in the library database. To add them to the library, there should be a Book Id, name, location, author of the book, and publisher. So, the librarian can use this option to add the book and students can issue that book.

Library Mangement

[login](#)[Sign Up](#)

Add Book

Book Id

Text

Book Name

Text

Book Location

Text

Author

Text

Publisher

Text

Add Book

Add Scholar Wireframe:

This page is going to be used by the library assistant to add the scholars to the library database directly or at the scholar's request. Therefore, whenever scholars come to issue or return the book as result their information data can be easily accessed.

Library Mangement

login

Sign Up

Add Scholar

Scholar Id

Text

First Name

Text

Last Name

Text

Gender

Text

Phone

Text

E-mail

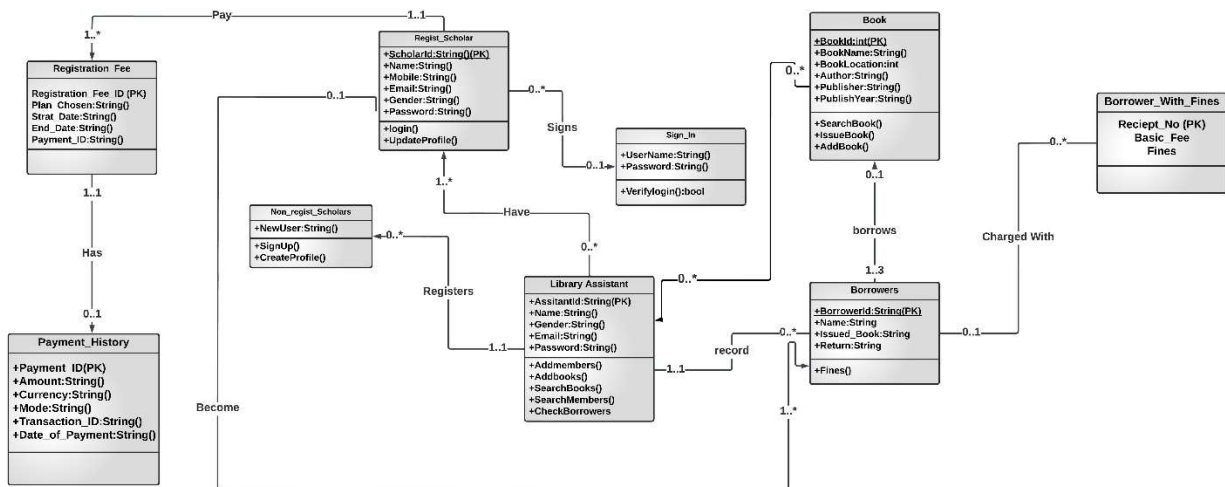
Text

Add Scholar

Project Implementation

- Hardware Platform
 - Windows PC
- Software Platform
 - Eclipse
- Programming Language
 - Java
- Development Tools
 - Eclipse
 - Scene Builder
 - Java UI libraries

Class Diagrams:



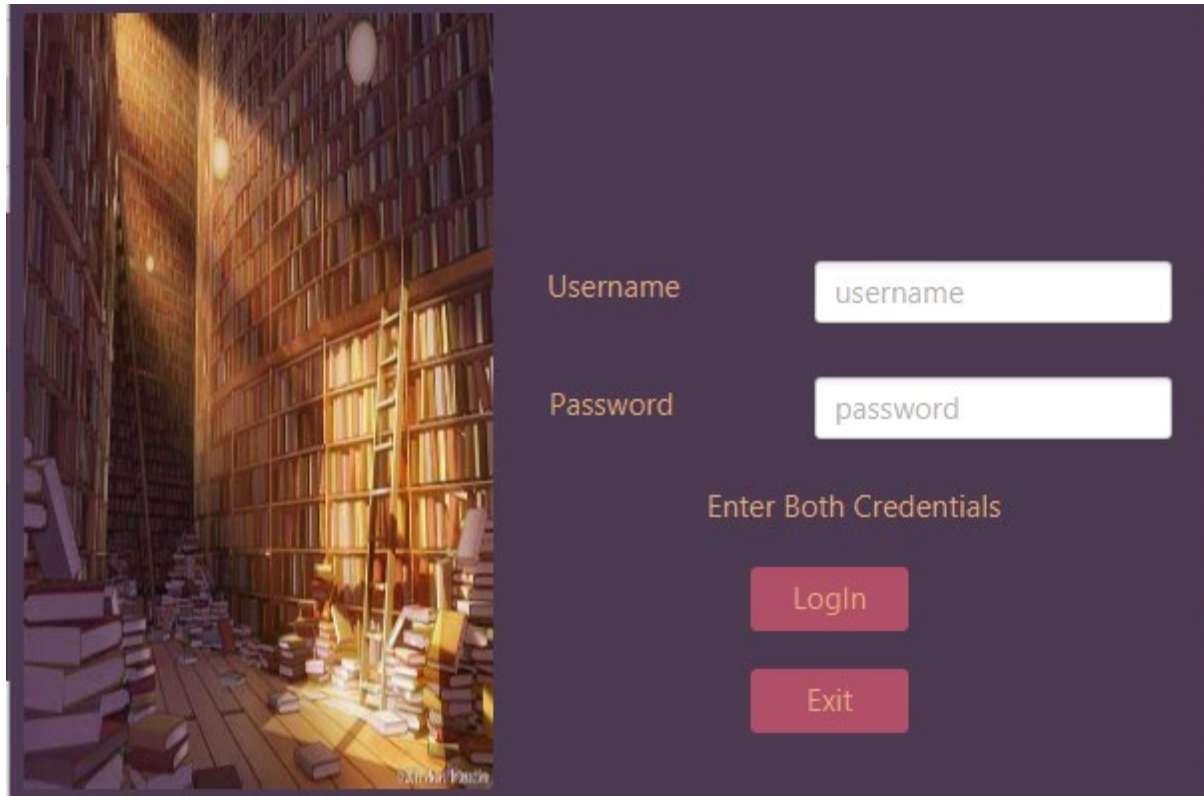
Test Results

Login page:



Figure 1

Login window: Username and password field checking Validation



The image shows a login window with a dark purple background. On the left side, there is a vertical image of a library with tall bookshelves and a wooden ladder. The right side of the window contains the login form. It has two labels, 'Username' and 'Password', in a light orange font. Each label is followed by a white rectangular input field. The first input field contains the text 'username' and the second contains 'password'. Below these fields, the text 'Enter Both Credentials' is displayed in the same light orange font. At the bottom of the form, there are two red rectangular buttons with white text: 'Login' and 'Exit'.

Username

Password

Enter Both Credentials

Figure 2

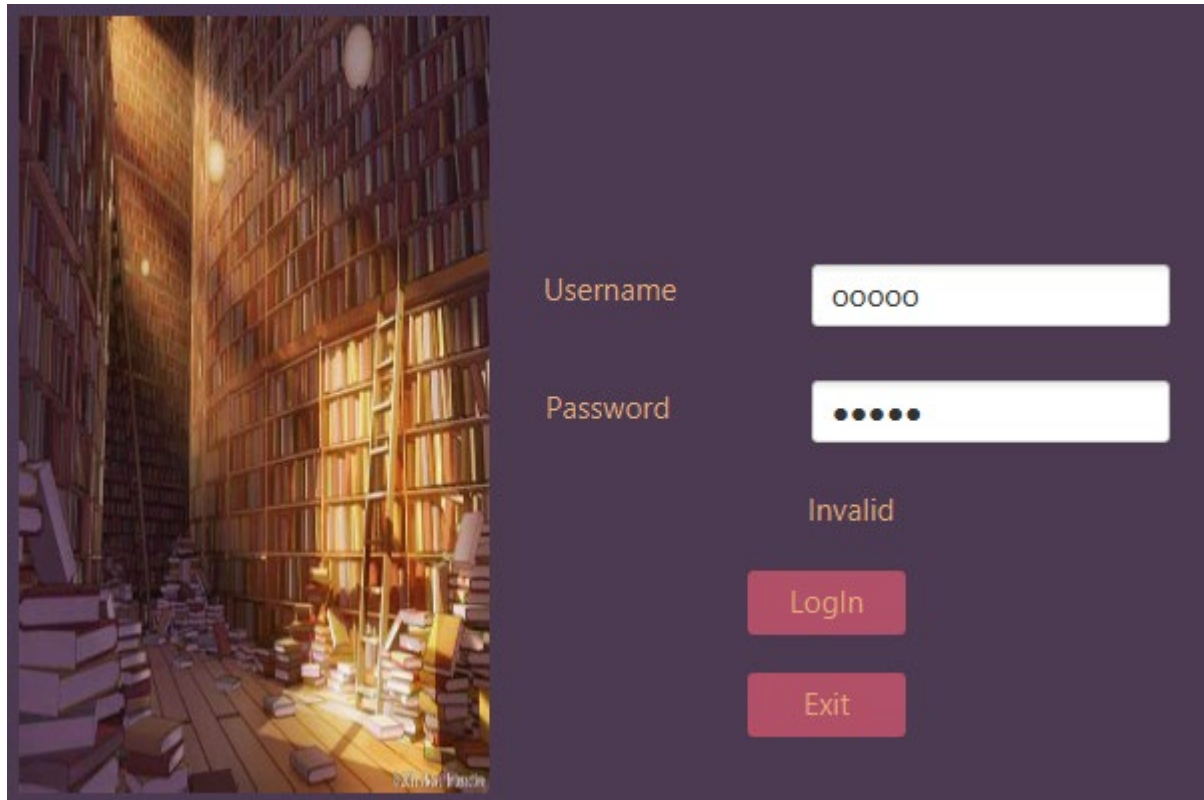


Figure 3

Librarian Menu:

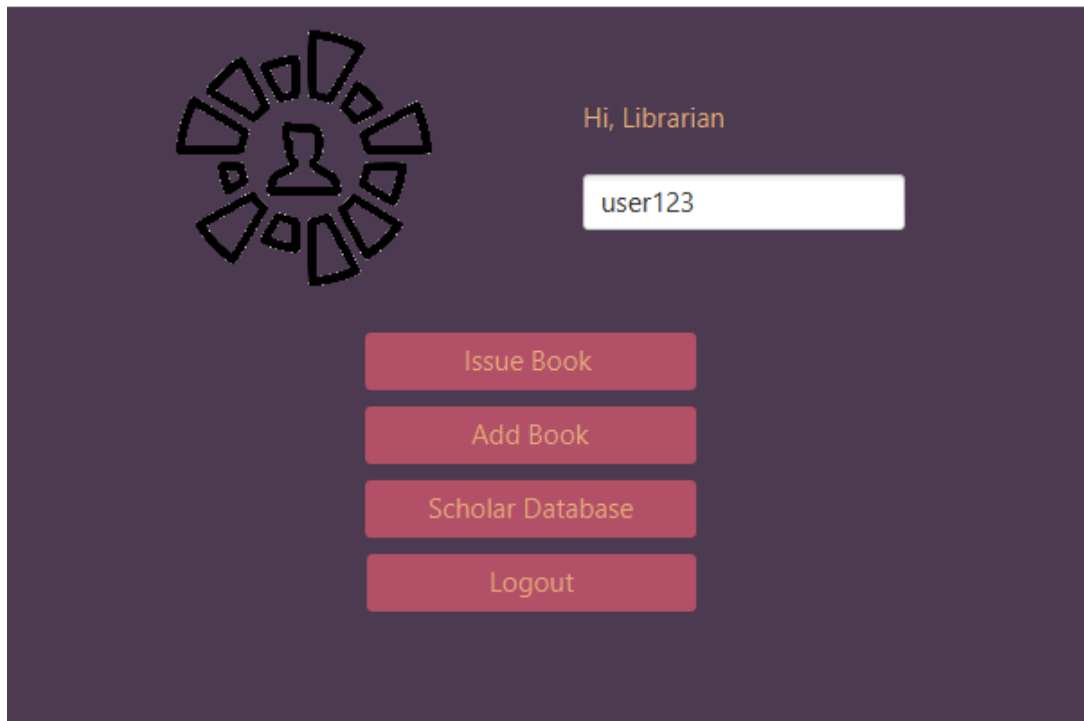


Figure 4

Issue Book: Did not issued the book, as it was not availabale, i.e., already borrowed

Book ID	Availability	Request
804168609-5	no	no
595564621-3	no	no
791052080-8	yes	yes
159855586-3	no	no
479777127-5	yes	no
697095535-4	yes	no
468319143-1	yes	yes
576082806-1	no	no
593630911-8	no	no

Book Issuing failed: Book Unavailable

595564621-3

Issue Book

Quit

Figure 5

Add Book:

Book Added to Database

Book Id

Book Name

Author

Publisher

Publish Year

Figure 6

Book Added into the database:

	bookTitle	bookId	bookAuthor	bookPub	bookAvail	bookRequest	borrowId
	Operation Ganymed	479777127-5	Minot	Daugherty Inc	yes	no	-----
	The Garden of Sinners - Chapter ...	697095535-4	Van Halle	Littel, Hidle and Kessler	yes	no	-----
	Hey, Happy!	468319143-1	Maw	Hackett-Harber	yes	yes	-----
	Deep Water	576082806-1	Ruusa	Ruecker, Morissette ...	no	no	12347789
	Cheyenne Autumn	593630911-8	Kleinerman	Schaefer, Friesen an...	no	no	2112112
					yes	no	NULL
	Inserted	2413	Team7	Team7	yes	no	NULL

Figure 7

Main Menu:

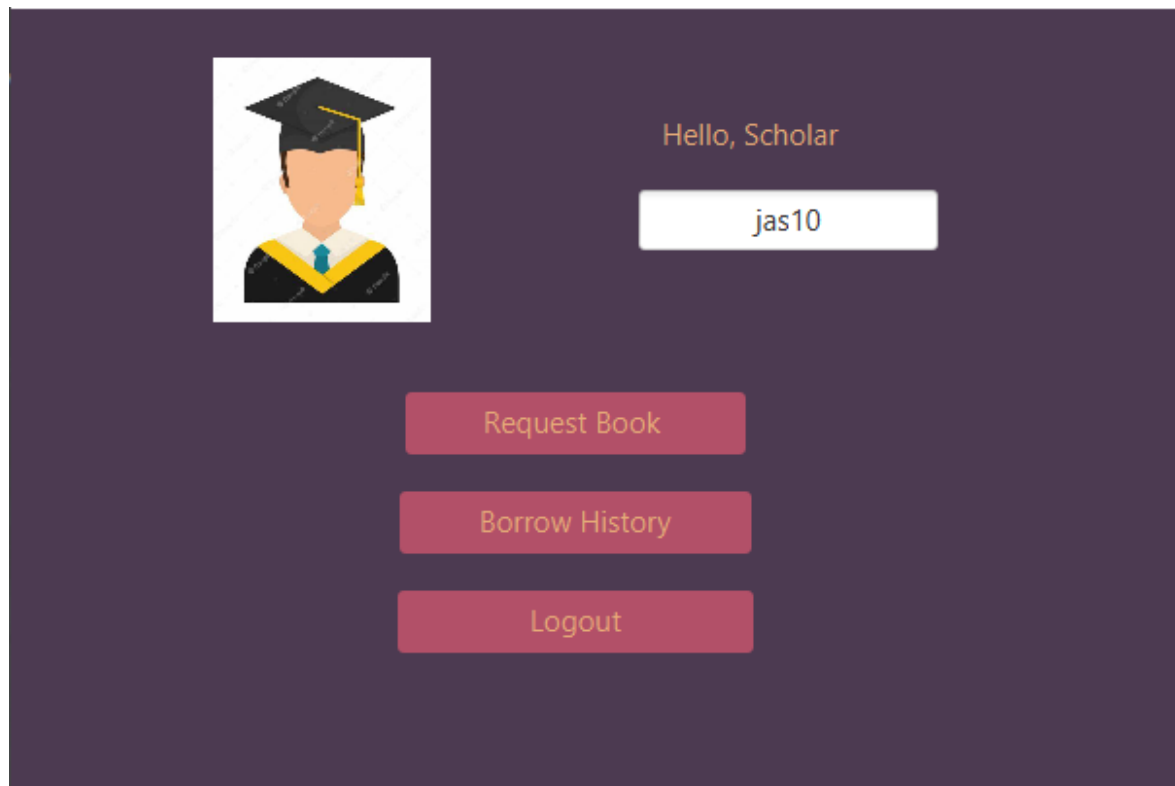


Figure 8

Scholar Database:

ID	Name	Contact	E-mail	username	password
52527891	Harlie	144-607-7724	hrumford2@how...	hmattingley2	30rv1x6b8P5L
76957897	Elwyn	197-884-3193	edudin5@msu.edu	ecorry5	7tczul
67527899	Kordula	393-181-1172	kleindecker1@us...	klien66	ag234daP
75178967	Sibbie	650-581-5970	scornock4@topli...	sziemen4	AlXs8D36
2957893	Wilow	853-842-6089	wheimann3@wsj...	wdalesio3	Bezzu5iaP
99578091	Montague	102-661-7817	mmccorkell9@ec...	mhogbourn...	dgzhvRmS
29957890	Jaskaran	194-388-9324	srraundl0@theti...	Jas10	letin
44657892	Abhijot	987-284-7429	codocherty8@sin...	Abhi07	pass
77578896	Paula	522-302-9481	pnurden6@drup...	pfrontczak6	sBnqtEWpQzg)
34757894	Tanny	221-933-0828	tmaine7@omnitu...	tallsopp7	VZK7DJXp

<

>

Quit

Figure 9

RequestBook:

ID	Title	Author	Publisher	Availabili
067102702-6	Hangmen Also Die	Coules	Hangmen Also Die	no
159855586-3	Mannequin 2: On the...	Blucher	Mannequin 2: On t...	no
468319143-1	Hey, Happy!	Maw	Hey, Happy!	yes
479777127-5	Operation Ganymed	Minot	Operation Ganymed	yes
576082806-1	Deep Water	Ruusa	Deep Water	no
593630911-8	Cheyenne Autumn	Kleinerman	Cheyenne Autumn	no
595564621-3	Border The	Amesbury	Border The	no
697095535-4	The Garden of Sinner...	Van Halle	The Garden of Sinn...	yes
791052080-8	Romance on the Hig...	Labbati	Romance on the Hi...	yes

<

>

Not Available

593630911-8

Request Book

Quit

Figure 10

Borrow History:

Borrow ID	Book Id	borrowDate	returnDate	Fine
12347789	067102702-6	2022-08-01	2022-09-01	\$0
2112112	593630911-8	2022-07-01	2022-07-27	\$0
12340089	576082806-1	2022-08-01	--/--/----	\$30
<div>< <input type="text"/> ></div> <div>Quit</div>				

Figure 11

User Guide

As we created the library management system, the software can be downloaded from the institution's website and is directly linked to the library's database. A software is user-friendly, the user requires only minimal assistance to operate it. A user does not need to install any additional software to utilise this application. The user only needs to install software on their computer and connect directly to the cloud. To log into the system, users must know their credentials, and users of software include both scholars and librarians.

- Figure 1 demonstrates that the interface provides two options: librarian or student login option.
- As we progress, the user selects the librarian option, which opens a new interface prompting them to enter their credentials in figure2.
- In figure3, the user entered an invalid credential, demonstrating that these credentials are invalid.
- When librarian is entered, the system displays four options. Add book where the librarian can add a new book to the database, Issue Book indicates that the librarian can issue the book to the student, and the Scholar database displays the student's borrowing history if he borrowed any books from the library.
- If the user clicks the Issue book option, the database will open, and the book id will need to be entered. As the user enters the book's identifier, the system indicates whether the book is available or has been requested by another user. If the book is available, the user can easily issue it by clicking the option to issue the book. **Figure3** demonstrates that a book is not issued because it is unavailable.
- The next step is to determine how to add a new book to the library. Choose the Add book option from the main menu to add a new book to the library database. For a librarian to add a book to a page, there are attributes for adding the book's title and other attributes. You can also see that the book has been added to the library's database in Figure7.
- In the scholar database **Figure9**, only the librarian can view the username and password. Therefore, if a user forgets his or her password, he or she can change it with the help of the librarian.
- When a scholar logs into their account, the requested page is displayed. Therefore, if a student can request a book that will be returned by another student, confirmation will be sent via email or text message, which will be implemented in the future.
- When it becomes connected to the cloud. If a user has borrowed any a book from the library, he can view his borrowing history at **Figure10**.

Conclusion

Throughout the process of writing our code, we meticulously planned everything out. As soon as we had each task clearly defined in our daily stand-ups, we launched our integrated development environment, Eclipse, and got to work. Before opening the Eclipse, we always sketched out the coding solution on paper, selecting the appropriate methods, variables, controllers, and classes. After that, we did a round of peer review before finally sitting down to write the code. We always maintained open lines of communication, paid attention to the details, stood up for each other when problems occurred, and used group meeting to obtain solutions for issues. We used team meetings to solve problems. It's important to remember that the context of a software application is different from that of the end user. We needed to know the needs of the users as well as the environment in which the specific programmes will run in order to successfully complete the project.

Reference

We did not take any help from any source. We just used book to learn the JavaFX and Java.

Oracle JavaFx(<https://docs.oracle.com/javase/8/javase-clienttechnologies.htm>)

Appendix

MySQL Database script:

```
CREATE DATABASE world;
```

```
create table scholar (
```

```
  scholarId varchar(8),
```

```
  Name VARCHAR(30),
```

```
  contact VARCHAR(12),
```

```
  email VARCHAR(40),
```

```
  gender VARCHAR(10),
```

```
  username VARCHAR(30),
```

```
  password VARCHAR(30)
```

```
);
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('29957890',  
'Jaskaran', '194-388-9324', 'srroundl0@thetimes.co.uk', 'Male', 'Jas10', 'letin');
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('67527899',  
'Kordula', '393-181-1172', 'kleindecker1@usa.gov', 'Female', 'klien66', 'ag234daP');
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('52527891',  
'Harlie', '144-607-7724', 'hrumford2@howstuffworks.com', 'Female', 'hmattingley2', '30rv1x6b8P5L');
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('2957893',  
'Wilow', '853-842-6089', 'wheimann3@wsj.com', 'Female', 'wdalesio3', 'Bezzu5iaP');
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('75178967',  
'Sibbie', '650-581-5970', 'scornock4@toplist.cz', 'Female', 'sziemen4', 'AIXs8D36');
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('76957897',  
'Elwyn', '197-884-3193', 'edudin5@msu.edu', 'Male', 'ecorry5', '7tczul');
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('77578896',  
'Paula', '522-302-9481', 'pnurden6@drupal.org', 'Female', 'pfrontczak6', 'sBnqtEWpQzgX');
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('34757894',  
'Tanny', '221-933-0828', 'tmaine7@omniture.com', 'Male', 'tallsopp7', 'VZK7DJXp');
```



```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('44657892', 'Abhijot', '987-284-7429', 'codocherty8@sina.com.cn', 'Male', 'Abhi07', 'pass');
```

```
insert into scholar (scholarId, Name, contact, email, gender, username, password) values ('99578091', 'Montague', '102-661-7817', 'mmccorkell9@economist.com', 'Male', 'mhogbourne9', 'dgzhvRmS');
```

```
create table book (
```

```
bookTitle VARCHAR(70),
```

```
bookId VARCHAR(50),
```

```
bookAuthor VARCHAR(50),
```

```
bookPub VARCHAR(50),
```

```
bookAvail VARCHAR(20),
```

```
bookRequest VARCHAR(5),
```

```
borrowId VARCHAR(8)
```

```
);
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('Hangmen Also Die', '067102702-6', 'Coules', 'Feeney Inc', 'no', 'no', '12347789');
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('Floundering', '804168609-5', 'Reckus', 'Kunze, Dach and Bins', 'no', 'no', '10047789');
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('Border The', '595564621-3', 'Amesbury', 'Baumbach, Bauch and Aufderhar', 'no', 'no', '10847789');
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('Romance on the High Seas', '791052080-8', 'Labbati', 'Von-Ankunding', 'yes', 'yes', '');
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('Mannequin 2: On the Move', '159855586-3', 'Blucher', 'Mann-Strosin', 'no', 'no', 5);
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('Operation Ganymed', '479777127-5', 'Minot', 'Daugherty Inc', 'yes', 'no', '');
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('The Garden of Sinners - Chapter 5: Paradox Paradigm', '697095535-4', 'Van Halle', 'Littel, Hickie and Kessler', 'yes', 'no', '');
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('Hey, Happy!', '468319143-1', 'Maw', 'Hackett-Harber', 'yes', 'yes', '');
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values ('Deep Water', '576082806-1', 'Ruusa', 'Ruecker, Morissette and Runte', 'no', 'no', '12340089');
```

```
insert into book (bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest, borrowId) values
('Cheyenne Autumn', '593630911-8', 'Kleinerman', 'Schaefer, Friesen and Russel', 'no', 'no', '2112112');
```

```
create table borrower (
```

```
  borrowId varchar(8),
```

```
  scholarId varchar(9),
```

```
  bookId varchar(15),
```

```
  borrowDate varchar(10),
```

```
  returnDate varchar(10),
```

```
  fine varchar(5)
```

```
);
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('12347789',
'29957890', '067102702-6', '2022-08-01', '2022-09-01', '$0');
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('2112112',
'29957890', '593630911-8', '2022-07-01', '2022-07-27', '$0');
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('12340089',
'29957890', '576082806-1', '2022-08-01', '', '$30');
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('14558890', '',
'804168609-5', '', '', '');
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('14598882', '',
'479777127-5', '', '', '');
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('77912112', '',
'576082806-1', '', '', '');
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('87654220', '',
'804168609-5', '', '', '');
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('56115215', '',
'595564621-3', '', '', '');
```

```
insert into borrower (borrowId, scholarId, bookId, borrowDate, returnDate, fine) values ('89761241', '',
'576082806-1', '', '', '');
```

```
create table librarian (
```

```

librarianId VARCHAR(8),
username VARCHAR(50),
password VARCHAR(50),
name VARCHAR(50),
contact VARCHAR(50),
gender VARCHAR(50)
);

insert into librarian (librarianId, name, contact, gender, username, password) values ('13398218',
'Hadleigh', '436-142-0251', 'Female', 'user123', 'pass123');

insert into librarian (librarianId, name, contact, gender, username, password) values ('1234821', 'Tyrus',
'708-977-2433', 'Male', 'tseedull1', 'c8m2yiQayHr');

insert into librarian (librarianId, name, contact, gender, username, password) values ('1311821', 'Zolly',
'453-248-4436', 'Female', 'zhugland2', 'HIADYo');

insert into librarian (librarianId, name, contact, gender, username, password) values ('7421695', 'Lucian',
'766-433-0347', 'Female', 'Isisson3', 'jo2MNC47JE');

insert into librarian (librarianId, name, contact, gender, username, password) values ('2324512',
'Adriano', '583-524-1195', 'Male', 'alamba4', '2XF3IZN5mGe');

insert into librarian (librarianId, name, contact, gender, username, password) values ('5167760',
'Francisco', '534-287-8409', 'Male', 'fvarnals5', 'te2aoy');

insert into librarian (librarianId, name, contact, gender, username, password) values ('5216511', 'Paige',
'702-980-8400', 'Female', 'ptenpenny6', '2ympNeav');

insert into librarian (librarianId, name, contact, gender, username, password) values ('2156765', 'Erny',
'742-760-2207', 'Female', 'elemmens7', 'TEReDHh');

insert into librarian (librarianId, name, contact, gender, username, password) values ('4514224', 'Chaim',
'342-393-4092', 'Male', 'cwalcar8', 'hUht83soR');

insert into librarian (librarianId, name, contact, gender, username, password) values ('9245163',
'Allister', '873-717-5660', 'Male', 'aspikins9', 'pexvys4Qid');

```

JAVA Code:

1. Application package:

- Main.java

```
public class Main extends Application {
```

```

@Override

public void start(Stage primaryStage) {

    try {

        Parent root = FXMLLoader.load(getClass().getResource("MainLogin.fxml"));

        Scene scene = new Scene(root, 600 ,400);

        scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());

        primaryStage.setTitle("Login Page");

        primaryStage.setScene(scene);

        primaryStage.show();

    } catch(Exception e) {

        e.printStackTrace();

    }

}

public static void main(String[] args) {

    launch(args);

}

}

```

- MainLoginController.java

```

public class MainLoginController implements Initializable {

```

```

@FXML

```

```

private Button btnlgn1;

```

```

@FXML

```

```

private Button btnlgn2;

```

```

@Override

```

```

public void initialize(URL url, ResourceBundle res) {

```

```

// TODO Auto-generated method stub

```

```

}

```

```

public void btnlgn1SetOnAction(ActionEvent event) throws IOException{
    FXMLLoader loader = new FXMLLoader(getClass().getResource("/Scholar/ScholarLoginWindow.fxml"));
    Parent root = loader.load();

    Scene scene = new Scene(root, 600, 400);
    scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
    Stage stage = (Stage) btnlgn1.getScene().getWindow();
    //stage.close();
    stage.setTitle("Scholar Login");
    stage.setScene(scene);
    stage.show();
}

```

- DbConnection.java

```

public class dbConnection {

    public Connection dbLink;

    public Connection getConnection() {

        String dbuser = "root";           //mysql user name
        String dbpass = "root@unforget7";  //mysql password
        String url = "jdbc:mysql://localhost/world?autoReconnect=true&useSSL=false";

        try {
            dbLink = DriverManager.getConnection(url, dbuser, dbpass);
        }
        catch(Exception e){
            e.printStackTrace();
            e.getCause();
        }
    }
}

```

```

return dbLink;

}

}

public void btnlgn2SetOnAction(ActionEvent event) throws IOException{

FXMLLoader loader = new
FXMLLoader(getClass().getResource("/librarian/LibrarianloginWindow.fxml"));

Parent root = loader.load();

Scene scene = new Scene(root, 600, 400);

scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());

Stage stage = (Stage) btnlgn2.getScene().getWindow();

//stage.close();

stage.setTitle("Librarian Login");

stage.setScene(scene);

stage.show();

}

}

```

- application.css

```

.root{

-fx-background-color:#4C3A51;

-fx-font-size: 15;

-fx-font-family: "Palatino";

-fx-alignment: center;

-fx-text-fill: E7AB79;

}

```

```

.label{

-fx-font-family: "Palatino";

-fx-text-fill: #E7AB79;

```

```
}
```

```
.textfield{
```

```
-fx-background-color:#B25068;
```

```
-fx-text-fill: #E7AB79;
```

```
}
```

```
.button{
```

```
-fx-background-color:#B25068;
```

```
-fx-text-fill: #E7AB79;
```

```
-fx-font-family: "Palatino";
```

```
}
```

```
.hbox{
```

```
-fx-background-color:#4C3A51;
```

```
-fx-font-family: "Palatino";
```

```
}
```

```
.vbox{
```

```
-fx-background-color:#4C3A51;
```

```
-fx-font-family: "Palatino";
```

```
}
```

```
.table-view{
```

```
-fx-background-color:#4C3A51;
```

```
-fx-font-family: "Palatino";
```

```
-fx-text-fill: #E7AB79;
```

```
}
```

```
.table-row-cell{  
    -fx-background-color: #774360;  
    -fx-font-family: "Palatino";  
    -fx-text-fill: #E7AB79;  
}
```

```
.table-view .column-header{  
    -fx-background-color: #4C3A51;  
    -fx-font-family: "Palatino";  
    -fx-text-fill: #E7AB79;  
}
```

```
.table-view .virtual-flow .scroll-bar:vertical,  
.table-view .virtual-flow .scroll-bar:vertical .track,  
.table-view .virtual-flow .scroll-bar:vertical .track-background,  
.table-view .virtual-flow .scroll-bar:horizontal,  
.table-view .virtual-flow .scroll-bar:horizontal .track,  
.table-view .virtual-flow .scroll-bar:horizontal .track-background {  
    -fx-background-color: #B25068;  
  
}
```

```
.table-view .scroll-bar{  
    -fx-background-color: #E7AB79;  
}
```

2. Book:

- Book.java

```
public class Book {
```



```
private String bookTitle;  
private String bookId;  
private String bookAuthor;  
private String bookPub;  
private String bookAvail;  
private String bookRequest;  
private String borrowId;
```

```
public Book(String bookTitle, String bookId, String bookAuthor, String bookPub, String bookAvail,  
String bookRequest, String borrowId) {  
    super();  
    this.bookTitle = bookTitle;  
    this.bookId = bookId;  
    this.bookAuthor = bookAuthor;  
    this.bookPub = bookPub;  
    this.bookAvail = bookAvail;  
    this.bookRequest = bookRequest;  
    this.borrowId = borrowId;  
}
```

```
Book(String bookTitle, String bookId, String bookAuthor,  
String bookPub, String bookAvail){  
    this.bookTitle = bookTitle;  
    this.bookId = bookId;  
    this.bookAuthor = bookAuthor;  
    this.bookPub = bookPub;  
    this.bookAvail = bookAvail;  
}
```

```
public Book(String bookId, String bookAvail, String bookRequest, String borrowId) {  
    this.bookId = bookId;  
    this.bookAvail = bookAvail;  
    this.bookRequest = bookRequest;  
    this.borrowId = borrowId;  
}
```

```
public String getBookRequest() {  
    return bookRequest;  
}
```

```
public String getBorrowerId() {  
    return borrowId;  
}
```

```
public String getBookTitle() {  
    return bookTitle;  
}
```

```
public String getBookId() {  
    return bookId;  
}
```

```
public String getBookAuthor() {  
    return bookAuthor;  
}
```

```
public String getBookPub() {  
    return bookPub;  
}
```

```
public String getBookAvail() {  
    return bookAvail;  
}
```

```
}
```

```
}
```

- BookController.java

```
public class BookController implements Initializable{
```

```
@FXML
```

```
private TextField bookIdTxt;
```

```
@FXML
```

```
private TextField bookNameTxt;
```

```
@FXML
```

```
private TextField authorTxt;
```

```
@FXML
```

```
private TextField publisherTxt;
```

```
@FXML
```

```
private TextField yearTxt;
```

```
@FXML
```

```
private Label addBooktxt;
```

```
@FXML
```

```
private Button quit;
```

```
@FXML
```

```
private Button addBook;
```

```
@Override
```

```
public void initialize(URL arg0, ResourceBundle arg1) {
```

```
}
```

```
public void quitOnAction(ActionEvent event) {
```

```
Stage stage = (Stage) quit.getScene().getWindow();
```

```
stage.close();
}

public void addBookOnAction(ActionEvent event) {

    dbConnection connectNow = new dbConnection();
    Connection connectdb = connectNow.getConnection();

    String bookId = bookIdTxt.getText();
    String bookTitle = bookNameTxt.getText();
    String bookAuthor = authorTxt.getText();
    String bookPub = publisherTxt.getText();
    String bookAvail = "yes";
    String bookRequest = "no";

    String insertBook = "insert into book(bookTitle, bookId, bookAuthor, bookPub, bookAvail, bookRequest)
    values ('";

    String insertValues = bookTitle+ "',' + bookId + "',' + bookAuthor + "',' + bookPub + "',' + bookAvail +
    "',' + bookRequest + '");";

    String insertToAdd = insertBook + insertValues;

    try {
        Statement st = connectdb.createStatement();
        st.executeUpdate(insertToAdd);
        addBooktxt.setText("Book Added to Database");

    }catch(Exception e) {
        e.printStackTrace();
        e.getCause();
    }
```

```
}  
}
```

- BookDbController.java

```
public class BookDbController implements Initializable {
```

```
@FXML
```

```
private TableView<Book> bookTable;
```

```
@FXML
```

```
private TableColumn<Book, String> bookTitle;
```

```
@FXML
```

```
private TableColumn<Book, String> bookId;
```

```
@FXML
```

```
private TableColumn<Book, String> bookAuthor;
```

```
@FXML
```

```
private TableColumn<Book, String> bookPub;
```

```
@FXML
```

```
private TableColumn<Book, String> bookAvail;
```

```
@FXML
```

```
private Button requestBook;
```

```
@FXML
```

```
private Button quit;
```

```
@FXML
```

```
private TextField requestStatus;
```

```
@FXML
```

```
private TextField bookIdtxt;
```

```
ObservableList<Book> oList = FXCollections.observableArrayList();
```

```
@Override
```

```
public void initialize(URL url, ResourceBundle res) {

    bookTitle.setCellValueFactory(new PropertyValueFactory<>("bookTitle"));
    bookId.setCellValueFactory(new PropertyValueFactory<>("bookId"));
    bookAuthor.setCellValueFactory(new PropertyValueFactory<>("bookAuthor"));
    bookPub.setCellValueFactory(new PropertyValueFactory<>("bookTitle"));
    bookAvail.setCellValueFactory(new PropertyValueFactory<>("bookAvail"));

    showDb();

    bookTable.setItems(oList);
}
```

```
public void showDb() {
    dbConnection connectNow = new dbConnection();
    Connection connectdb = connectNow.getConnection();
```

```
String data = "SELECT * FROM world.book";
try {
    Statement st = connectdb.createStatement();
    ResultSet rs = st.executeQuery(data);
```

```
while(rs.next()) {
    oList.add(new Book(rs.getString("bookTitle"),
        rs.getString("bookId"),
        rs.getString("bookAuthor"),
        rs.getString("bookPub"),
        rs.getString("bookAvail")));
}
}catch(Exception e){
}
```

```
}
```

```
public void requestBookbtnOnAction(ActionEvent event) {  
    validateRequest();  
}
```

```
public void validateRequest() {  
    dbConnection connectNow = new dbConnection();  
    Connection connectdb = connectNow.getConnection();
```

```
    String verifyAvail = "SELECT count(1) FROM book WHERE bookId = '" + bookIdtxt.getText() +  
    "' And bookAvail = 'yes' AND bookRequest = 'no'; ";  
    try {  
        Statement st = connectdb.createStatement();  
        ResultSet queryResult = st.executeQuery(verifyAvail);  
        while(queryResult.next()) {  
            if(queryResult.getInt(1) == 1) {  
                String reqBook = "UPDATE book set bookRequest= 'yes' where bookId = '" + bookIdtxt.getText() + """;  
  
                try {  
                    Statement is = connectdb.createStatement();  
                    is.executeUpdate(reqBook);  
                }catch(Exception e) {  
                    e.printStackTrace();  
                    e.getCause();  
                }  
                //issueStatuslbl.setText("Book Issued");  
                requestStatus.setText("Book Requested");  
            }  
        }  
    }
```

```

else {
requestStatus.setText("Not Available");
}
}
}catch(Exception e) {
}
}

```

```

public void quitOnAction(ActionEvent event) {
Stage stage = (Stage) quit.getScene().getWindow();
stage.close();
}

```

- IssueBookController.java

```

public class IssueBookController implements Initializable {

```

```

@FXML

```

```

private TableView<Book> requestTable;

```

```

@FXML

```

```

private TableColumn<Book, String> bookId;

```

```

@FXML

```

```

private TableColumn<Book, String> bookAvail;

```

```

@FXML

```

```

private TableColumn<Book, String> bookRequest;

```

```

@FXML

```

```

private TableColumn<Book, String> borrowId;

```

```

@FXML

```

```

private Button issueBookbtn;

```

```

@FXML

```

```

private Button quit;

```



```
@FXML
```

```
private TextField bookIdTxt;
```

```
@FXML
```

```
private TextField issueStatusLbl;
```

```
ObservableList<Book> reqList = FXCollections.observableArrayList();
```

```
@Override
```

```
public void initialize(URL url, ResourceBundle res) {
```

```
    bookId.setCellValueFactory(new PropertyValueFactory<>("bookId"));
```

```
    bookAvail.setCellValueFactory(new PropertyValueFactory<>("bookAvail"));
```

```
    bookRequest.setCellValueFactory(new PropertyValueFactory<>("bookRequest"));
```

```
    borrowId.setCellValueFactory(new PropertyValueFactory<>("borrowId"));
```

```
    showDb();
```

```
    requestTable.setItems(reqList);
```

```
}
```

```
public void showDb() {
```

```
    dbConnection connectNow = new dbConnection();
```

```
    Connection connectdb = connectNow.getConnection();
```

```
    String data = "SELECT * FROM world.book";
```

```
    try {
```

```
        Statement st = connectdb.createStatement();
```

```
        ResultSet rs = st.executeQuery(data);
```

```
        while(rs.next()) {
```

```
            reqList.add(new Book(rs.getString("bookId"),
```

```

rs.getString("bookAvail"),
rs.getString("bookRequest"),
rs.getString("borrowId"));
}
}catch(Exception e){
}
}

```

```

public void issueBookbtnOnAction(ActionEvent event) {
checkavailability();
}

```

```

public void checkavailability() {
dbConnection connectNow = new dbConnection();
Connection connectdb = connectNow.getConnection();

```

```

String verifyAvail = "SELECT count(1) FROM book WHERE bookId ='" + bookIdTxt.getText() +
"" And bookAvail = 'yes' AND bookRequest = 'yes'; ";
try {
Statement st = connectdb.createStatement();
ResultSet queryResult = st.executeQuery(verifyAvail);
while(queryResult.next()) {
if(queryResult.getInt(1) == 1) {
String issueBookDb = "UPDATE book set bookAvail = 'no', bookRequest= 'no' where bookId = '"+
bookIdTxt.getText() +"";";
try {
Statement is = connectdb.createStatement();
is.executeUpdate(issueBookDb);
}catch(Exception e) {

```

```

e.printStackTrace();
e.getCause();
}
issueStatusLabel.setText("Book Issued");
}
else {
issueStatusLabel.setText("Book Issuing failed: Book Unavailable");
}
}
}catch(Exception e) {
}
}

```

```

public void quitOnAction(ActionEvent event) {
    Stage stage = (Stage) quit.getScene().getWindow();
    stage.close();
}
}

```

- AddBook.fxml:

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.VBox?>

```

```

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="book.BookController">

    <center>

        <AnchorPane prefHeight="200.0" prefWidth="200.0" BorderPane.alignment="CENTER">

            <children>

                <Button fx:id="addBook" contentDisplay="CENTER" layoutX="156.0" layoutY="350.0"
mnemonicParsing="false" onAction="#addBookOnAction" prefHeight="25.0" prefWidth="141.0"
text="Add Book" />

                <Label fx:id="addBooktxt" alignment="CENTER" layoutX="156.0" layoutY="21.0"
prefHeight="31.0" prefWidth="287.0" />

                <VBox layoutX="330.0" layoutY="76.0" prefHeight="273.0" prefWidth="215.0"
spacing="20.0">

                    <children>

                        <TextField fx:id="bookIdTxt" />

                        <TextField fx:id="bookNameTxt" prefHeight="34.0" prefWidth="215.0" />

                        <TextField fx:id="authorTxt" prefHeight="34.0" prefWidth="215.0" />

                        <TextField fx:id="publisherTxt" prefHeight="33.0" prefWidth="215.0" />

                        <TextField fx:id="yearTxt" prefHeight="32.0" prefWidth="215.0" />

                    </children>

                </VBox>

                <VBox layoutX="105.0" layoutY="72.0" prefHeight="275.0" prefWidth="188.0"
spacing="26.0">

                    <children>

                        <Label alignment="CENTER" prefHeight="28.0" prefWidth="188.0" text="Book Id" />

                        <Label alignment="CENTER" prefHeight="26.0" prefWidth="188.0" text="Book Name"
/>

                        <Label alignment="CENTER" prefHeight="25.0" prefWidth="188.0" text="Author" />

                        <Label alignment="CENTER" prefHeight="24.0" prefWidth="188.0" text="Publisher" />

                        <Label alignment="CENTER" prefHeight="28.0" prefWidth="188.0" text="Publish
Year" />

                    </children>

```

```

        </VBox>

        <Button fx:id="quit" layoutX="333.0" layoutY="349.0" mnemonicParsing="false"
onAction="#quitOnAction" prefHeight="25.0" prefWidth="145.0" text="Quit" />

    </children>

</AnchorPane>

</center>

</BorderPane>

    • BookDb.fxml:

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.HBox?>


<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="book.BookDbController">

    <children>

        <TableView fx:id="bookTable" prefHeight="314.0" prefWidth="600.0"
AnchorPane.bottomAnchor="86.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">

            <columns>

                <TableColumn fx:id="bookId" editable="false" prefWidth="79.0" text="ID" />

                <TableColumn fx:id="bookTitle" editable="false" prefWidth="139.0" text="Title" />

                <TableColumn fx:id="bookAuthor" editable="false" prefWidth="131.0" text="Author" />

                <TableColumn fx:id="bookPub" editable="false" prefWidth="118.0" text="Publisher" />

                <TableColumn fx:id="bookAvail" editable="false" prefWidth="132.0" text="Availability"
/>
    
```

```

        </columns>

    </TableView>

    <HBox alignment="CENTER" layoutX="141.0" layoutY="348.0" prefHeight="56.0"
    prefWidth="600.0" spacing="10.0" AnchorPane.leftAnchor="0.0"
    AnchorPane.rightAnchor="0.0">

        <children>

            <TextField fx:id="bookIdtxt" prefHeight="28.0" prefWidth="250.0" promptText="Enter
            Book Id To request" />

            <Button fx:id="requestBook" mnemonicParsing="false"
            onAction="#requestBookbtnOnAction" prefHeight="31.0" prefWidth="152.0" text="Request
            Book" />

            <Button fx:id="quit" mnemonicParsing="false" onAction="#quitOnAction"
            prefHeight="32.0" prefWidth="147.0" text="Quit" />

        </children>

    </HBox>

    <TextField fx:id="requestStatus" alignment="CENTER" editable="false" layoutX="97.0"
    layoutY="323.0" prefHeight="25.0" prefWidth="395.0" promptText="Book Request Status" />

</children>
</AnchorPane>

```

- IssueBook.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

```

```

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.HBox?>

```

```

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="book.IssueBookController">

```

```

<children>

    <TableView fx:id="requestTable" prefHeight="312.0" prefWidth="600.0"
AnchorPane.bottomAnchor="88.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">

        <columns>

            <TableColumn fx:id="bookId" editable="false" prefWidth="123.0" text="Book ID" />

            <TableColumn fx:id="borrowId" editable="false" prefWidth="143.0" text="Borrower Id" />

            <TableColumn fx:id="bookAvail" editable="false" prefWidth="177.0" text="Availability"
/>

            <TableColumn fx:id="bookRequest" editable="false" prefWidth="156.0" text="Request"
/>

        </columns>

    </TableView>

    <HBox alignment="CENTER" layoutX="141.0" layoutY="348.0" prefHeight="56.0"
prefWidth="600.0" spacing="10.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0">

        <children>

            <TextField fx:id="bookIdTxt" prefHeight="28.0" prefWidth="195.0" promptText="Enter
Borrower Id To request" />

            <Button fx:id="IssueBookbtn" mnemonicParsing="false"
onAction="#issueBookbtnOnAction" prefHeight="31.0" prefWidth="152.0" text="Issue Book" />

            <Button fx:id="quit" mnemonicParsing="false" onAction="#quitOnAction"
prefHeight="32.0" prefWidth="147.0" text="Quit" />

        </children>

    </HBox>

    <TextField fx:id="issueStatuslbl" alignment="CENTER" editable="false" layoutX="109.0"
layoutY="323.0" prefHeight="28.0" prefWidth="419.0" promptText="Book Issue Status" />

</children>

</AnchorPane>

```

3. Scholar:

- Scholar.java

```
public class Scholar{
```

```
private String scholarId;
```

```
private String name;
```

```
private String contact;
```

```
private String email;
```

```
private String username;
```

```
private String password;
```

```
Scholar(){
```

```
}
```

```
Scholar(String scholarId, String name, String contact, String email, String username, String password) {
```

```
this.scholarId = scholarId;
```

```
this.name = name;
```

```
this.contact = contact;
```

```
this.email = email;
```

```
this.username = username;
```

```
this.password = password;
```

```
}
```

```
public void setScholarId(String scholarId) {
```

```
this.scholarId = scholarId;
```

```
}
```

```
public void setName(String name) {
```

```
this.name = name;
```

```
}
```

```
public void setContact(String contact) {
```

```
this.contact = contact;
```



```
}
```

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

```
public void setUsername(String username) {  
    this.username = username;  
}
```

```
public void setPassword(String password) {  
    this.password = password;  
}
```

```
public String getScholarId() {  
    return scholarId;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public String getContact() {  
    return contact;  
}
```

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

```
public String getUsername() {  
    return username;  
}
```

```
public String getPassword() {  
    return password;  
}  
}
```

- AddScholarController.java

```
public class AddScholarController implements Initializable {
```

```
@FXML
```

```
private TextField nameTxt;
```

```
@FXML
```

```
private TextField contactTxt;
```

```
@FXML
```

```
private TextField emailTxt;
```

```
@FXML
```

```
private TextField usernameTxt;
```

```
@FXML
```

```
private PasswordField passwordTxt;
```

```
@FXML
```

```
private PasswordField passConfirmTxt;
```

```
@FXML
```

```
private Button AddScholarbtn;
```

```
@FXML
```

```
private Label prompt;
```

```
@Override

public void initialize(URL arg0, ResourceBundle arg1) {

// TODO Auto-generated method stub

}


public void AddScholarBtnOnAction(ActionEvent event) {

if(passwordTxt.getText().equals(passConfirmTxt.getText())) {

registerUser();

prompt.setText("User is registered");

}

else {

prompt.setText("Passwords does not match!");

}

}


public void registerUser() {


dbConnection connectNow = new dbConnection();

Connection connectdb = connectNow.getConnection();


String name = nameTxt.getText();

String contact = contactTxt.getText();

String username = usernameTxt.getText();

String password = passwordTxt.getText();

String email = emailTxt.getText();


double genrateId = 1+ (Math.random()*999999);

String id = Double.toString(genrateId);
```

```
String insertData = "INSERT INTO scholar(id, name, contact, email, username, password) VALUES (";  
String insertValues = id+ "',' + name + "',' + contact + "',' + email + "',' + username + "',' + password +  
"');";
```

```
String insertToRegister = insertData + insertValues;
```

```
try {  
    Statement st = connectdb.createStatement();  
    st.executeUpdate(insertToRegister);  
} catch (Exception e) {  
    e.printStackTrace();  
    e.getCause();  
}  
}  
}
```

- MenuController.java

```
public class MenuController implements Initializable{
```

```
@FXML
```

```
private Button requestBook;
```

```
@FXML
```

```
private Button accountInfoBtn;
```

```
@FXML
```

```
private Button borrowHistory;
```

```
@FXML
```

```
private Button logoutbtn;
```

```
@FXML
```

```
TextField userTxt;
```

```
private String username;
```

```
public String getUsername() {  
    return username;  
}
```

```
public void setUsername(String username) {  
    this.username = username;  
}
```

```
@Override
```

```
public void initialize(URL arg0, ResourceBundle arg1) {  
}
```

```
public void bookDbbtnOnAction(ActionEvent event) throws IOException{
```

```
FXMLLoader loader = new FXMLLoader(getClass().getResource("/Book/BookDb.fxml"));
```

```
Parent root = loader.load();
```

```
Scene scene = new Scene(root, 600, 400);
```

```
scene.getStylesheets().add(getClass().getResource("/application/application.css").toExternalForm());
```

```
Stage stage = (Stage) requestBook.getScene().getWindow();
```

```
stage.setTitle("Book Database");
```

```
stage.setScene(scene);
```

```
stage.show();
```

```
}
```

```
public void accountInfobtnOnAction(ActionEvent action) throws IOException{
```

```
dbConnection connectNow = new dbConnection();
```

```
Connection connectdb = connectNow.getConnection();
```

```
FXMLLoader loader = new FXMLLoader(getClass().getResource("/scholar/ScholarInfo.fxml"));
```

```
Parent root = loader.load();
```

```
String username = userTxt.getText();
```

```
ScholarInfoController infoCon = loader.getController();
```

```
infoCon.setUsername(username);
```

```
infoCon.usernameTxt.setText(username);
```

```
String contact = "SELECT contact FROM world.scholar where username='"+ getUsername() +"'";
```

```
try{
```

```
Statement st = connectdb.createStatement();
```

```
ResultSet rs = st.executeQuery(contact);
```

```
infoCon.contactTxt.setText(rs.getString("contact"));
```

```
}catch(Exception e) {
```

```
}
```

```
Scene scene = new Scene(root, 600, 400);
```

```
scene.getStylesheets().add(getClass().getResource("/application/application.css").toExternalForm());
```

```
Stage stage = (Stage) accountInfoBtn.getScene().getWindow();
```

```
stage.setTitle("Account Details");
```

```
stage.setScene(scene);
```

```
stage.show();
```

```
}
```

```
public void borrowHistorybtnOnAction(ActionEvent action) throws IOException {
```

```
FXMLLoader loader = new FXMLLoader(getClass().getResource("/Borrower/BorrowHistory.fxml"));
```

```
Parent root = loader.load();
```

```
BorrowerController bCon = new BorrowerController();
```

```
dbConnection connectNow = new dbConnection();
```

```
Connection connectdb = connectNow.getConnection();
```

```
String getScholarId = "SELECT scholarId FROM scholar WHERE username ='" + username + "'";
```

```
try {
```

```
Statement st = connectdb.createStatement();
```

```
ResultSet rs = st.executeQuery(getScholarId);
```

```
while(rs.next()) {
```

```
bCon.setScholarId(rs.getString("scholarId"));
```

```
bCon.setScholarId(rs.getString(0));
```

```
}
```

```
}catch(Exception e) {
```

```
}
```

```
Scene scene = new Scene(root, 600, 400);
```

```
scene.getStylesheets().add(getClass().getResource("/application/application.css").toExternalForm());
```

```
Stage stage = (Stage) borrowHistory.getScene().getWindow();
```

```
stage.setTitle("Borrow History");
```

```
stage.setScene(scene);
```

```
stage.show();
```

```
}
```

```
public void logoutbtnOnAction(ActionEvent action) {
```

```
Stage stage = (Stage) logoutbtn.getScene().getWindow();
```

```
stage.close();
```

```
}
```

```
}
```

- ScholarDbController.java

```
public class ScholarDbController implements Initializable {
```

```
@FXML
```

```
private TableView<Scholar> scholarTable;
```

```
@FXML
```

```
private TableColumn<Scholar, String> scholarId;
```

```
@FXML
```

```
private TableColumn<Scholar, String> name;
```

```
@FXML
```

```
private TableColumn<Scholar, String> contact;
```

```
@FXML
```

```
private TableColumn<Scholar, String> email;
```

```
@FXML
```

```
private TableColumn<Scholar, String> username;
```

```
@FXML
```

```
private TableColumn<Scholar, String> password;
```

```
@FXML
```

```
private Button quit;
```

```
ObservableList<Scholar> oList = FXCollections.observableArrayList();
```

```
@Override
```

```
public void initialize(URL arg0, ResourceBundle arg1) {
```

```
scholarId.setCellValueFactory(new PropertyValueFactory<>("scholarId"));
```

```
name.setCellValueFactory(new PropertyValueFactory<>("name"));
```

```
contact.setCellValueFactory(new PropertyValueFactory<>("contact"));
```

```
email.setCellValueFactory(new PropertyValueFactory<>("email"));
```

```
username.setCellValueFactory(new PropertyValueFactory<>("username"));
```

```
password.setCellValueFactory(new PropertyValueFactory<>("password"));
```

```
showDb();
```



```

scholarTable.setItems(oList);
}

public void showDb() {
    dbConnection connectNow = new dbConnection();
    Connection connectdb = connectNow.getConnection();

    String data = "SELECT * FROM world.scholar";
    try {
        Statement st = connectdb.createStatement();
        ResultSet rs = st.executeQuery(data);

        while(rs.next()) {
            oList.add(new Scholar(rs.getString("scholarId"),
            rs.getString("Name"),
            rs.getString("contact"),
            rs.getString("email"),
            rs.getString("username"),
            rs.getString("password")));
        }
    }catch(Exception e){
    }
}

public void quitOnAction(ActionEvent event) {
    Stage stage = (Stage) quit.getScene().getWindow();
    stage.close();
}
}

```

- ScholarInfoController.java

```
public class ScholarInfoController implements Initializable {

    @FXML
    private TextField IdTxt;

    @FXML
    private TextField nameTxt;

    @FXML
    public TextField contactTxt;

    @FXML
    private TextField emailTxt;

    @FXML
    TextField usernameTxt;

    @FXML
    private PasswordField passwordTxt;

    @FXML
    private TextField genderTxt;

    @FXML
    private Button quit;

    private String username;

    ObservableList<Scholar> userList = FXCollections.observableArrayList();

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

@Override

```
public void initialize(URL arg0, ResourceBundle arg1) {
```

```
    dbConnection connectNow = new dbConnection();
```

```
    Connection connectdb = connectNow.getConnection();
```

```
    String contact = "SELECT contact FROM world.scholar where username='" + getUsername() + "';";
```

```
    try {
```

```
        Statement st = connectdb.createStatement();
```

```
        ResultSet rs = st.executeQuery(contact);
```

```
        emailTxt.setText(rs.getString(0));
```

```
    }catch(Exception e){
```

```
    }
```

```
    nameTxt.setText("iside");
```

```
}
```

```
public void quitOnAction(ActionEvent event) {
```

```
    Stage stage = (Stage) quit.getScene().getWindow();
```

```
    stage.close();
```

```
}
```

```
public void setIdTxt(TextField idTxt) {
```

```
    IdTxt = idTxt;
```

```
}
```

```
public void setNameTxt(TextField nameTxt) {
```

```
    this.nameTxt = nameTxt;
```

```
}
```

```
public void setContactTxt(TextField contactTxt) {
```

```
    this.contactTxt = contactTxt;
```

```
}
```

```
public void setEmailTxt(TextField emailTxt) {
```

```

this.emailTxt = emailTxt;
}

public void setPasswordTxt(PasswordField passwordTxt) {
this.passwordTxt = passwordTxt;
}

public void setGenderTxt(TextField genderTxt) {
this.genderTxt = genderTxt;
}

public void setQuit(Button quit) {
this.quit = quit;
}
}

```

- ScholarLoginController.java

```

public class ScholarLoginController implements Initializable {

```

```

@FXML

```

```

private Button exitbtn;

```

```

@FXML

```

```

private Label logintxt;

```

```

@FXML

```

```

public TextField usernameTxt;

```

```

@FXML

```

```

public PasswordField passwordTxt;

```

```

@FXML

```

```

private Button loginbtn;

```

```

@Override

```

```

public void initialize(URL url, ResourceBundle res) {

```

```
}
```

```
public void loginbtnOnAction(ActionEvent event) throws IOException {
```

```
if(usernameTxt.getText().isBlank() == false && passwordTxt.getText().isBlank() == false) {
```

```
    validateLogin();
```

```
}
```

```
else {
```

```
    logintxt.setText("Enter Both Credentials");
```

```
}
```

```
}
```

```
public void exitbtnOnAction(ActionEvent event) {
```

```
    Stage stage = (Stage) exitbtn.getScene().getWindow();
```

```
    stage.close();
```

```
}
```

```
public void validateLogin() {
```

```
    dbConnection connectNow = new dbConnection();
```

```
    Connection connectdb = connectNow.getConnection();
```

```
    String verifylogin = "SELECT count(1) FROM scholar WHERE username ='" + usernameTxt.getText() + "'  
AND password = '" + passwordTxt.getText() + "' ";
```

```
    try {
```

```
        Statement st = connectdb.createStatement();
```

```
        ResultSet queryResult = st.executeQuery(verifylogin);
```

```
        while(queryResult.next()) {
```

```
            if(queryResult.getInt(1) == 1) {
```

```

logintxt.setText("Valid");

FXMLLoader loader = new FXMLLoader(getClass().getResource("/Scholar/MenuWindow.fxml"));

Parent root = loader.load();

String username = usernameTxt.getText();

MenuController menuController = loader.getController();

menuController.userTxt.setText(username);

menuController.setUsername(username);


Scene scene = new Scene(root, 600 ,400);

scene.getStylesheets().add(getClass().getResource("/application/application.css").toExternalForm());

Stage stage = (Stage) loginbtn.getScene().getWindow();

stage.setTitle("Main Menu");

stage.setScene(scene);

stage.show();

}

else {

logintxt.setText("Invalid");

}

}

}catch(Exception e) {

}

}

}

```

- AddScholar.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.PasswordField?>
```

```

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="scholar.AddScholarController">

    <bottom>

        <AnchorPane prefHeight="374.0" prefWidth="600.0" BorderPane.alignment="CENTER">

            <children>

                <TextField fx:id="nameTxt" layoutX="277.0" layoutY="93.0" />

                <TextField fx:id="contactTxt" layoutX="277.0" layoutY="161.0" />

                <TextField fx:id="usernameTxt" layoutX="278.0" layoutY="230.0" />

                <Button fx:id="AddScholarbtn" layoutX="248.0" layoutY="335.0" mnemonicParsing="false"
onAction="#signUpbtnOnAction" text="Add Scholar" />

                <Label layoutX="108.0" layoutY="97.0" text="Name" />

                <Label fx:id="contact" layoutX="110.0" layoutY="165.0" text="Contact" />

                <Label layoutX="108.0" layoutY="225.0" text="Username" />

                <Label layoutX="110.0" layoutY="259.0" text="Password" />

                <Label layoutX="110.0" layoutY="298.0" text="Confirm Password" />

                <Label fx:id="prompt" layoutX="248.0" layoutY="67.0" />

                <PasswordField fx:id="passConfirmTxt" layoutX="277.0" layoutY="294.0" />

                <PasswordField fx:id="passwordTxt" layoutX="277.0" layoutY="264.0" />

                <Button fx:id="cancelbtn" layoutX="352.0" layoutY="335.0" mnemonicParsing="false"
onAction="#signUpbtnOnAction" prefHeight="25.0" prefWidth="61.0" text="Cancel" />

                <Label fx:id="contact1" layoutX="114.0" layoutY="196.0" text="E-mail" />

                <TextField fx:id="emailTxt" layoutX="277.0" layoutY="192.0" />

            </children>

        </AnchorPane>

    </bottom>

```

</BorderPane>

- MenuWindow.fxml

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.VBox?>

<?import javafx.scene.text.Font?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="scholar.MenuController">

<center>

<AnchorPane prefHeight="200.0" prefWidth="200.0" BorderPane.alignment="CENTER">

<children>

<TextField fx:id="userTxt" alignment="CENTER" editable="false" layoutX="323.0" layoutY="93.0"
prefHeight="28.0" prefWidth="154.0" promptText="Show username" />

<VBox alignment="CENTER" layoutX="112.0" layoutY="151.0" prefHeight="225.0"
prefWidth="357.0" spacing="20.0">

<children>

<Button id="MenuBookBtn" fx:id="requestBook" mnemonicParsing="false"
onAction="#bookDbbtnOnAction" prefHeight="25.0" prefWidth="175.0" text="Request Book " />

<Button fx:id="accountInfoBtn" mnemonicParsing="false"
onAction="#accountInfoBtnOnAction" prefHeight="25.0" prefWidth="179.0" text="Account Info" />

<Button id="MenuUpdatebtn" fx:id="borrowHistory" mnemonicParsing="false"
onAction="#borrowHistorybtnOnAction" prefHeight="25.0" prefWidth="181.0" text="Borrow History"
/>


```

        <Button id="MenuLogoutbtn" fx:id="logoutbtn" mnemonicParsing="false"
onAction="#logoutbtnOnAction" prefHeight="25.0" prefWidth="183.0" text="Logout" />

    </children>

</VBox>

<Label layoutX="335.0" layoutY="48.0" prefHeight="32.0" prefWidth="142.0" text="Hello,
Scholar">

    <font>

        <Font size="19.0" />

    </font>

</Label>

<ImageView fitHeight="136.0" fitWidth="181.0" layoutX="119.0" layoutY="26.0"
pickOnBounds="true" preserveRatio="true" />

<ImageView fitHeight="136.0" fitWidth="145.0" layoutX="104.0" layoutY="25.0"
pickOnBounds="true" preserveRatio="true">

    <image>

        <Image url="@../images/scholar.jpg" />

    </image>

</ImageView>

</children></AnchorPane>

</center>

</BorderPane>

```

- ScholarDb.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.HBox?>

```

```
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="scholar.ScholarDbController">
```

```
<children>
```

```
<TableView fx:id="scholarTable" prefHeight="200.0" prefWidth="200.0"
AnchorPane.bottomAnchor="50.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
```

```
<columns>
```

```
<TableColumn fx:id="scholarId" prefWidth="79.0" text="ID" />
```

```
<TableColumn fx:id="name" prefWidth="121.0" text="Name" />
```

```
<TableColumn fx:id="contact" prefWidth="90.0" text="Contact" />
```

```
<TableColumn fx:id="email" prefWidth="103.0" text="E-mail" />
```

```
<TableColumn fx:id="username" prefWidth="95.0" text="username" />
```

```
<TableColumn fx:id="password" prefWidth="141.0" text="password" />
```

```
</columns>
```

```
</TableView>
```

```
<HBox alignment="CENTER" layoutX="141.0" layoutY="348.0" prefHeight="56.0" prefWidth="600.0"
spacing="10.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0">
```

```
<children>
```

```
<Button fx:id="quit" mnemonicParsing="false" onAction="#quitOnAction" prefHeight="32.0"
prefWidth="147.0" text="Quit" />
```

```
</children>
```

```
</HBox>
```

```
</children>
```

```
</AnchorPane>
```

- ScholarInfo.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.PasswordField?>
```

```

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.VBox?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="scholar.ScholarInfoController">

    <center>

        <AnchorPane prefHeight="374.0" prefWidth="600.0" BorderPane.alignment="CENTER">

            <children>

                <Label fx:id="prompt" layoutX="248.0" layoutY="67.0" />

                <Button fx:id="quit" layoutX="253.0" layoutY="345.0" mnemonicParsing="false"
onAction="#quitOnAction" prefHeight="25.0" prefWidth="94.0" text="Quit" />

                <VBox alignment="CENTER" layoutX="347.0" layoutY="67.0" prefHeight="290.0"
prefWidth="140.0" spacing="20.0">

                    <children>

                        <TextField fx:id="nameTxt" prefHeight="24.0" prefWidth="140.0" />

                        <TextField fx:id="contactTxt" />

                        <TextField fx:id="emailTxt" />

                        <TextField fx:id="genderTxt" />

                        <PasswordField fx:id="usernameTxt" />

                        <PasswordField fx:id="passwordTxt" />

                    </children>

                </VBox>

                <VBox alignment="CENTER" layoutX="113.0" layoutY="67.0" prefHeight="282.0"
prefWidth="152.0" spacing="25.0">

                    <children>

                        <Label alignment="CENTER" prefHeight="17.0" prefWidth="167.0" text="Name" />

                        <Label fx:id="contact" alignment="CENTER" prefHeight="17.0" prefWidth="156.0"
text="Contact" />

```

```
        <Label fx:id="contact1" alignment="CENTER" prefHeight="17.0" prefWidth="156.0" text="E-mail" />
```

```
        <Label alignment="CENTER" prefHeight="17.0" prefWidth="226.0" text="Gender" />
```

```
        <Label alignment="CENTER" prefHeight="17.0" prefWidth="175.0" text="Username" />
```

```
        <Label alignment="CENTER" prefHeight="15.0" prefWidth="157.0" text="Password" />
```

```
    </children>
```

```
    </VBox>
```

```
    </children>
```

```
    </AnchorPane>
```

```
    </center>
```

```
    </BorderPane>
```

- ScholarLoginWindow.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.PasswordField?>
```

```
<?import javafx.scene.control.TextField?>
```

```
<?import javafx.scene.image.Image?>
```

```
<?import javafx.scene.image.ImageView?>
```

```
<?import javafx.scene.layout.AnchorPane?>
```

```
<?import javafx.scene.layout.BorderPane?>
```

```
<?import javafx.scene.text.Font?>
```

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="scholar.ScholarLoginController">
```

```
    <left>
```

```
        <AnchorPane prefHeight="400.0" prefWidth="249.0" BorderPane.alignment="CENTER">
```

```
            <children>
```

```
<ImageView fitHeight="389.0" fitWidth="234.0" layoutX="7.0" layoutY="6.0">
    <image>
        <Image url="@../images/loginImage.jpg" />
    </image>
</ImageView>
</children>
</AnchorPane>
</left>
<center>
    <AnchorPane prefHeight="400.0" prefWidth="292.0" BorderPane.alignment="CENTER">
        <children>
            <TextField fx:id="usernameTxt" layoutX="154.0" layoutY="130.0" prefHeight="25.0"
prefWidth="179.0" promptText="username" />
            <PasswordField fx:id="passwordTxt" layoutX="154.0" layoutY="188.0" prefHeight="25.0"
prefWidth="179.0" promptText="password" />
            <Label layoutX="21.0" layoutY="131.0" prefHeight="23.0" prefWidth="112.0" text="Username"
textFill="#ddda0c">
                <font>
                    <Font name="System Bold" size="13.0" />
                </font>
            </Label>
            <Label layoutX="21.0" layoutY="190.0" prefHeight="22.0" prefWidth="112.0" text="Password"
textFill="#8e972c">
                <font>
                    <Font name="System Bold" size="14.0" />
                </font>
            </Label>
            <Button fx:id="loginbtn" alignment="CENTER" layoutX="122.0" layoutY="283.0"
mnemonicParsing="false" onAction="#loginbtnOnAction" prefHeight="27.0" prefWidth="79.0"
text="LogIn" />
        </children>
    </AnchorPane>
</center>
</right>
</children>
</BorderPane>
</Scene>
</Stage>
</FXML>
```

```
<Button fx:id="exitbtn" alignment="CENTER" layoutX="122.0" layoutY="334.0"
mnemonicParsing="false" onAction="#exitbtnOnAction" prefHeight="27.0" prefWidth="79.0"
text="Exit" />
```

```
<Label fx:id="logintxt" alignment="CENTER" layoutX="69.0" layoutY="238.0" prefHeight="29.0"
prefWidth="210.0" textFill="#ebe8e8">
```

```
<font>
```

```
<Font name="System Bold" size="12.0" />
```

```
</font>
```

```
</Label>
```

```
</children>
```

```
</AnchorPane>
```

```
</center>
```

```
</BorderPane>
```

4. Borrower:

- Borrower.java

```
public class Borrower {
```

```
    private String borrowId;
```

```
    private String scholarId;
```

```
    private String bookId;
```

```
    private String returnDate;
```

```
    private String borrowDate;
```

```
    private String fine;
```

```
    Borrower(String borrowId, String bookId,
```

```
             String borrowDate, String returnDate, String fine){
```

```
        this.borrowId = borrowId;
```

```
        this.bookId = bookId;
```

```
        this.returnDate = returnDate;
```

```
        this.borrowDate = borrowDate;
```

```
this.fine = fine;
```

```
}
```

```
public String getReturnDate() {
```

```
    return returnDate;
```

```
}
```

```
public String getFine() {
```

```
    return fine;
```

```
}
```

```
public String getBorrowId() {
```

```
    return borrowId;
```

```
}
```

```
public String getBookId() {
```

```
    return bookId;
```

```
}
```

```
public String getBorrowDate() {
```

```
    return borrowDate;
```

```
}
```

```
}
```

- BorrowerController.java

```
public class BorrowerController implements Initializable{
```

```
@FXML
```

```
private TableView<Borrower> historyTable;
```

```
@FXML
```

```
private TableColumn<Borrower, String> borrowId;
```

```
@FXML
```

```
private TableColumn<Borrower, String> bookId;
```

```
@FXML
```

```
private TableColumn<Borrower, String> borrowDate;
@FXML
private TableColumn<Borrower, String> returnDate;
@FXML
private TableColumn<Borrower, String> fine;
@FXML
private Button requestBookbtn;
@FXML
private TextField bookIdTxt;
@FXML
private Label requestStatusTxt;
String scholarId;
@FXML
private Button quit;

public String getScholarId() {
    return scholarId;
}

public void setScholarId(String scholarId) {
    this.scholarId = scholarId;
}

ObservableList<Borrower> bList = FXCollections.observableArrayList();

@Override
public void initialize(URL url, ResourceBundle res) {
    borrowId.setCellValueFactory(new PropertyValueFactory<>("borrowId"));
    bookId.setCellValueFactory(new PropertyValueFactory<>("bookId"));
```



```
borrowDate.setCellValueFactory(new PropertyValueFactory<>("borrowDate"));
returnDate.setCellValueFactory(new PropertyValueFactory<>("returnDate"));
fine.setCellValueFactory(new PropertyValueFactory<>("fine"));
```

```
connectDb();
historyTable.setItems(bList);
}
```

```
public void connectDb() {
    dbConnection connectNow = new dbConnection();
    Connection connectdb = connectNow.getConnection();
```

```
String Id = "29957890";
String data = "SELECT * FROM world.borrower where scholarId = '"+ Id +"'";
try {
    Statement st = connectdb.createStatement();
    ResultSet rs = st.executeQuery(data);
```

```
while(rs.next()) {
    bList.add(new Borrower(rs.getString("borrowID"),
    rs.getString("bookId"),
    rs.getString("borrowDate"),
    rs.getString("returnDate"),
    rs.getString("fine")));
}
}catch(Exception e){
}
}
```

```
public void quitbtnOnAction(ActionEvent event) {
```

```

Stage stage = (Stage) quit.getScene().getWindow();
stage.close();
}
}

```

- BorrowerHistory.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.TableColumn?>
```

```
<?import javafx.scene.control.TableView?>
```

```
<?import javafx.scene.layout.AnchorPane?>
```

```
<?import javafx.scene.layout.HBox?>
```

```

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="borrower.BorrowerController">

```

```
<children>
```

```

    <TableView fx:id="historyTable" prefHeight="200.0" prefWidth="200.0"
AnchorPane.bottomAnchor="50.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">

```

```
<columns>
```

```
<TableColumn fx:id="borrowId" prefWidth="79.0" text="Borrow ID" />
```

```
<TableColumn fx:id="bookId" prefWidth="139.0" text="Book Id" />
```

```
<TableColumn fx:id="borrowDate" prefWidth="131.0" text="borrowDate" />
```

```
<TableColumn fx:id="returnDate" prefWidth="118.0" text="returnDate" />
```

```
<TableColumn fx:id="fine" prefWidth="132.0" text="Fine" />
```

```
</columns>
```

```
</TableView>
```

```

    <HBox alignment="CENTER" layoutX="141.0" layoutY="348.0" prefHeight="56.0" prefWidth="600.0"
spacing="10.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0">

```

```

        <children>

            <Button fx:id="quit" mnemonicParsing="false" onAction="#quitbtnOnAction" prefHeight="32.0"
prefWidth="147.0" text="Quit" />

        </children>

    </HBox>

</children>

</AnchorPane>

```

5. Librarian:

- LibrarianLoginController.java

```
public class LibrarianLoginController implements Initializable {
```

```
@FXML
```

```
private Button exitbtn;
```

```
@FXML
```

```
private Label logintxt;
```

```
@FXML
```

```
public TextField usernameTxt;
```

```
@FXML
```

```
public PasswordField passwordTxt;
```

```
@FXML
```

```
private Button loginbtn;
```

```
@Override
```

```
public void initialize(URL url, ResourceBundle res) {
```

```
}
```

```
public void loginbtnOnAction(ActionEvent event) throws IOException {
```

```
if(usernameTxt.getText().isBlank()== false && passwordTxt.getText().isBlank() == false) {  
    validateLogin();  
}  
else {  
    logintxt.setText("Enter Both Credentials");  
}  
}
```

```
public void exitbtnOnAction(ActionEvent event) {  
    Stage stage = (Stage) exitbtn.getScene().getWindow();  
    stage.close();  
  
}
```

```
public void validateLogin() {  
    dbConnection connectNow = new dbConnection();  
    Connection connectdb = connectNow.getConnection();  
  
    String verifylogin = "SELECT count(1) FROM librarian WHERE username =" + usernameTxt.getText() + "  
    AND password = " + passwordTxt.getText() + " ";  
  
    try {  
        Statement st = connectdb.createStatement();  
        ResultSet queryResult = st.executeQuery(verifylogin);  
  
        while(queryResult.next()) {  
            if(queryResult.getInt(1) == 1) {  
                logintxt.setText("Valid");  
                FXMLLoader loader = new FXMLLoader(getClass().getResource("/librarian/MenuWindow.fxml"));  
                Parent root = loader.load();
```

```

String username = usernameTxt.getText();
MenuController menuController = loader.getController();
menuController.txtuse.setText(username);
menuController.setUsername(username);

Scene scene = new Scene(root, 600 ,400);
scene.getStylesheets().add(getClass().getResource("/application/application.css").toExternalForm());
Stage stage = (Stage) loginbtn.getScene().getWindow();
stage.setTitle("Main Menu");
stage.setScene(scene);
stage.show();
}
else {
logintxt.setText("Invalid");
}
}
}catch(Exception e) {
}
}
}

```

- MenuController.java

```
public class MenuController implements Initializable{
```

```
@FXML
```

```
private Button issueBook;
```

```
@FXML
```

```
Button addBookbtn;
```

```
@FXML
```

```
private Button ScholarDbbtn;
```

@FXML

private Button MenuLogoutbtn;

@FXML

TextField txtuse;

@Override

public void initialize(URL url, ResourceBundle res) {

}

private String username;

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public void issueBookbtnOnAction(ActionEvent event) throws IOException{

FXMLLoader loader = new FXMLLoader(getClass().getResource("/Book/IssueBook.fxml"));

Parent root = loader.load();

Scene scene2 = new Scene(root, 600, 400);

scene2.getStylesheets().add(getClass().getResource("/application/application.css").toExternalForm());

Stage stage2 = (Stage) issueBook.getScene().getWindow();

stage2.setTitle("Issue Book");

stage2.setScene(scene2);

stage2.show();

```
}
```

```
public void scholarDbbtnOnAction(ActionEvent event) throws IOException{
```

```
FXMLLoader loader = new FXMLLoader(getClass().getResource("/Scholar/ScholarDb.fxml"));
```

```
Parent root = loader.load();
```

```
Scene scene = new Scene(root, 600, 400);
```

```
scene.getStylesheets().add(getClass().getResource("/application/application.css").toExternalForm());
```

```
Stage stage = (Stage) ScholarDbbtn.getScene().getWindow();
```

```
//stage.close();
```

```
stage.setTitle("Scholar List");
```

```
stage.setScene(scene);
```

```
stage.show();
```

```
}
```

```
public void MenuAddBookOnAction(ActionEvent event) throws IOException{
```

```
FXMLLoader loader = new FXMLLoader(getClass().getResource("/book/AddBook.fxml"));
```

```
Parent root = loader.load();
```

```
Scene scene = new Scene(root, 600, 400);
```

```
scene.getStylesheets().add(getClass().getResource("/application/application.css").toExternalForm());
```

```
Stage stage = (Stage) addBookbtn.getScene().getWindow();
```

```
stage.setTitle("Add Book Information");
```

```
stage.setScene(scene);
```

```
stage.show();
```

```
}
```

```

public void logoutbtnOnAction(ActionEvent event) throws IOException {
    Stage stage = (Stage) MenuLogoutbtn.getScene().getWindow();
    stage.close();
}
}

```

- RegisterController.java

```

public class registerController implements Initializable {

```

```

@FXML

```

```

private TextField fNameetxt;

```

```

@FXML

```

```

private TextField lNameetxt;

```

```

@FXML

```

```

private TextField idtxt;

```

```

@FXML

```

```

private TextField usernametxt;

```

```

@FXML

```

```

private PasswordField passwordtxtr;

```

```

@FXML

```

```

private PasswordField passConfirmtxtr;

```

```

@FXML

```

```

private Button AddScholarbtn;

```

```

@FXML

```

```

private Label prompt;

```

```

@Override

```

```

public void initialize(URL arg0, ResourceBundle arg1) {

```

```

}

```



```
public void signUpbtnOnAction(ActionEvent event) {  
    if(passwordtxttr.getText().equals(passConfirmtxttr.getText())) {  
        registerUser();  
        prompt.setText("User is registered");  
    }  
    else {  
        prompt.setText("Passwords does not match!");  
    }  
}
```

```
public void registerUser() {  
    dbConnection connectNow = new dbConnection();  
    Connection connectdb = connectNow.getConnection();  
    String fName = fNameetxt.getText();  
    String lName = lNameetxt.getText();  
    String username = usernametxttr.getText();  
    String password = passwordtxttr.getText();  
    String id = idtxt.getText();  
  
    String insertData = "INSERT INTO world.testData(id, first_name, last_name, username, password)  
VALUES ('";  
    String insertValues = id + "','" + fName + "','" + lName + "','" + username + "','" + password + "');";  
    String insertToRegister = insertData + insertValues;  
  
    try {  
        Statement st = connectdb.createStatement();  
        st.executeUpdate(insertToRegister);  
    }catch(Exception e) {
```

```
e.printStackTrace();
e.getCause();
}
}
}
```

- LibrarianloginWindow.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.PasswordField?>
```

```
<?import javafx.scene.control.TextField?>
```

```
<?import javafx.scene.image.Image?>
```

```
<?import javafx.scene.image.ImageView?>
```

```
<?import javafx.scene.layout.AnchorPane?>
```

```
<?import javafx.scene.layout.BorderPane?>
```

```
<?import javafx.scene.text.Font?>
```

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="librarian.LibrarianLoginController">
```

```
<left>
```

```
<AnchorPane prefHeight="400.0" prefWidth="249.0" BorderPane.alignment="CENTER">
```

```
<children>
```

```
<ImageView fitHeight="388.0" fitWidth="235.0" layoutX="7.0" layoutY="6.0">
```

```
<image>
```

```
<Image url="@../images/loginImage.jpg" />
```

```
</image>
```

```

        </ImageView>
    </children>
</AnchorPane>
</left>
<center>
    <AnchorPane prefHeight="400.0" prefWidth="292.0" BorderPane.alignment="CENTER">
        <children>
            <TextField fx:id="usernameTxt" layoutX="154.0" layoutY="130.0" prefHeight="25.0"
prefWidth="179.0" promptText="username" />

            <PasswordField fx:id="passwordTxt" layoutX="154.0" layoutY="188.0" prefHeight="25.0"
prefWidth="179.0" promptText="password" />

            <Label layoutX="20.0" layoutY="130.0" prefHeight="25.0" prefWidth="109.0" text="Username"
textFill="#ddda0c">
                <font>
                    <Font name="System Bold" size="13.0" />
                </font>
            </Label>

            <Label layoutX="21.0" layoutY="187.0" prefHeight="28.0" prefWidth="107.0" text="Password"
textFill="#d7eb00">
                <font>
                    <Font name="System Bold" size="14.0" />
                </font>
            </Label>

            <Button fx:id="loginbtn" alignment="CENTER" layoutX="122.0" layoutY="283.0"
mnemonicParsing="false" onAction="#loginbtnOnAction" prefHeight="27.0" prefWidth="79.0"
text="LogIn" />

            <Button fx:id="exitbtn" alignment="CENTER" layoutX="122.0" layoutY="334.0"
mnemonicParsing="false" onAction="#exitbtnOnAction" prefHeight="27.0" prefWidth="79.0"
text="Exit" />

            <Label fx:id="logintxt" alignment="CENTER" layoutX="69.0" layoutY="238.0" prefHeight="29.0"
prefWidth="210.0" textFill="#ebe8e8">
                <font>

```

```

        <Font name="System Bold" size="12.0" />
    </font>
</Label>
</children>
</AnchorPane>
</center>
</BorderPane>
    • MenuWindow.fxml
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="librarian.MenuController">

    <center>

        <AnchorPane prefHeight="200.0" prefWidth="200.0" BorderPane.alignment="CENTER">

            <children>

                <TextField fx:id="txtuse" editable="false" layoutX="320.0" layoutY="93.0" prefHeight="30.0"
prefWidth="179.0" promptText="Show username" />

                <VBox alignment="CENTER" layoutX="153.0" layoutY="155.0" prefHeight="206.0"
prefWidth="276.0" spacing="10.0">

```

```

    <children>

        <Button id="MenuBookBtn" fx:id="issueBook" mnemonicParsing="false"
onAction="#issueBookbtnOnAction" prefHeight="25.0" prefWidth="184.0" text="Issue Book " />

        <Button id="MenuInfobtn" fx:id="addBookbtn" mnemonicParsing="false"
onAction="#MenuAddBookOnAction" prefHeight="25.0" prefWidth="184.0" text="Add Book" />

        <Button id="MenuUpdatebtn" fx:id="ScholarDbbtn" mnemonicParsing="false"
onAction="#scholarDbbtnOnAction" prefHeight="25.0" prefWidth="184.0" text="Scholar Database" />

        <Button id="MenuLogoutbtn" fx:id="MenuLogoutbtn" mnemonicParsing="false"
onAction="#logoutbtnOnAction" prefHeight="25.0" prefWidth="183.0" text="Logout" />

    </children>

</VBox>

<ImageView fitHeight="142.0" fitWidth="175.0" layoutX="94.0" layoutY="13.0"
pickOnBounds="true" preserveRatio="true">

    <image>

        <Image url="@../images/librarian.png" />

    </image>

</ImageView>

<Label layoutX="320.0" layoutY="42.0" prefHeight="38.0" prefWidth="178.0" text="Hi,
Librarian">

    <font>

        <Font size="19.0" />

    </font></Label>

</children></AnchorPane>

</center>

</BorderPane>

```

- RegisterWindow.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.PasswordField?>
```

```

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="librarian.registerController">

    <bottom>

        <AnchorPane prefHeight="374.0" prefWidth="600.0" BorderPane.alignment="CENTER">

            <children>

                <TextField fx:id="fNametxt" layoutX="277.0" layoutY="93.0" />

                <TextField fx:id="lNametxt" layoutX="277.0" layoutY="124.0" />

                <TextField fx:id="idtxt" layoutX="277.0" layoutY="161.0" />

                <TextField fx:id="usernamestr" layoutX="277.0" layoutY="213.0" />

                <Button fx:id="AddScholarbtn" layoutX="248.0" layoutY="335.0" mnemonicParsing="false"
onAction="#signUpbtnOnAction" text="Add Scholar" />

                <Label layoutX="108.0" layoutY="97.0" text="First Name" />

                <Label layoutX="107.0" layoutY="128.0" text="Last Name" />

                <Label layoutX="110.0" layoutY="165.0" text="ID" />

                <Label layoutX="108.0" layoutY="225.0" text="Username" />

                <Label layoutX="110.0" layoutY="259.0" text="Password" />

                <Label layoutX="110.0" layoutY="298.0" text="Confirm Password" />

                <Label fx:id="prompt" layoutX="248.0" layoutY="67.0" />

                <PasswordField fx:id="passConfirmtxtr" layoutX="277.0" layoutY="294.0" />

                <PasswordField fx:id="passwordtxtr" layoutX="277.0" layoutY="264.0" />

                <Button fx:id="signUpbtn1" layoutX="352.0" layoutY="335.0" mnemonicParsing="false"
onAction="#signUpbtnOnAction" prefHeight="25.0" prefWidth="61.0" text="Cancel" />

            </children>

        </AnchorPane>

    </bottom>

```

</BorderPane>