

LIBRARY MANAGEMENT SYSTEM WITH JAVA

Technique paper – January 2020

One author:



Abdybap uulu Damirbek

Ala-Too International University Faculty of New
Technologies

See profile in [GitHub](#)

Table of Contents	
INTRODUCTION	2
EXPLANATIONS	3
<i>Dependencies</i>	4
<i>Execution procedure</i>	5
<i>Execution procedure</i>	6
<i>Function (Issued Books)</i>	6
<i>Function (View Books)</i>	7
<i>Function (Sign Out)</i>	7
ISSUE PART	8
<i>Function (Issue Book)</i>	8
<i>Function (Submission Book)</i>	9
ADMIN MENU	10
<i>Function (Add Book)</i>	11
<i>Function (View Books)</i>	12
<i>Function (Issued Books)</i>	13
<i>Function (Delete Book)</i>	14
<i>Function (View Users)</i>	15
<i>Function (Sign Out)</i>	15
DATABASE	16
UML DIAGRAMS	17
CODE EXPLANATION	19
REFERENCES	23

INTRODUCTION

This assignment is based on developing an LIS (Library Management System) using “Java programming language”. For that, I used GUI (Graphical User Interface) in this development so that it will become more users friendly to interact. All dates (Users, Books, Issued Books) stores in MySQL Database.

EXPLANATIONS

In this documentation, we have explained how to interact successfully with this LIS. We have explained here systematically so that it will surely help users to become more user friendly with it. Below are my explanations:

Dependencies:

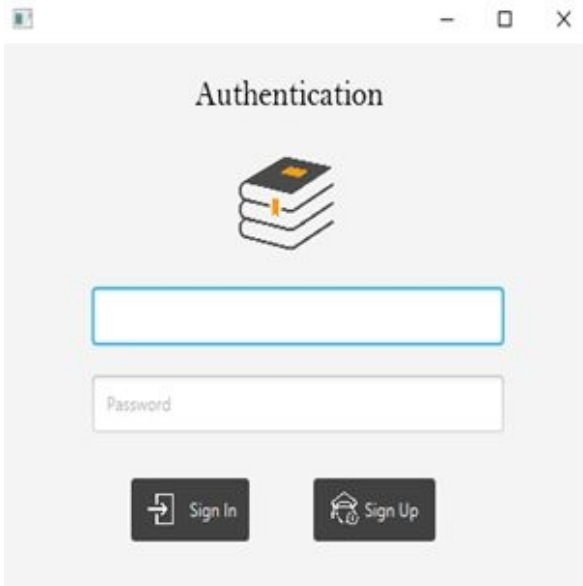
Before execute program users need to do some works so that it will run properly into their system. Firstly, they need to make sure their system is having “JDK”. If they do not have it then they can download from this below link:

[JDK link](#)

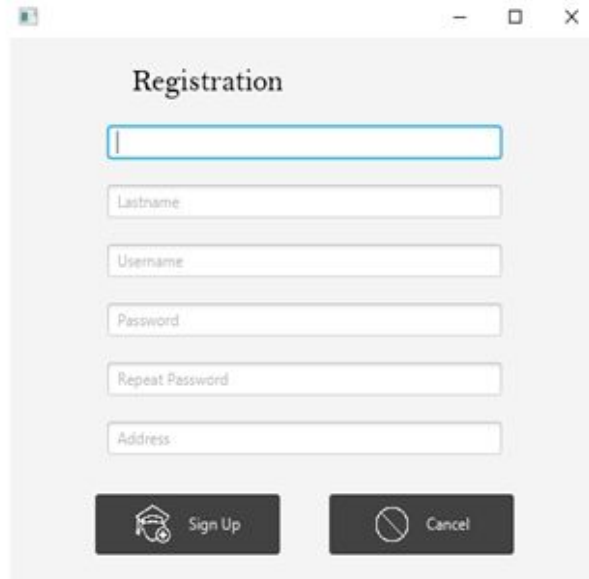
Depending on their system (Windows 64bit/32bit), they need to download and install. Then they need to add the “JAVA” files to their system “PATH” so that the system can run the program from CMD (Command Prompt). The path will show something like this “C:\Program Files (x86)\Java\jre1.8.0_25\bin;” Now just add the address besides the current path directory and save it. The other way they can execute this program in to download the IDE (Integrated Development Environment) on their system. They can download IntelliJ Idea depending on the windows (32bit/64bit). [Click to download](#).

Execution procedure:

When user executes this program, it will open login menu. There is startup GUI (Graphical User Interface) of this program (Sign In, Sign up). User has to register in system to continue with work; after all, they can just logged in through Sign In menu.

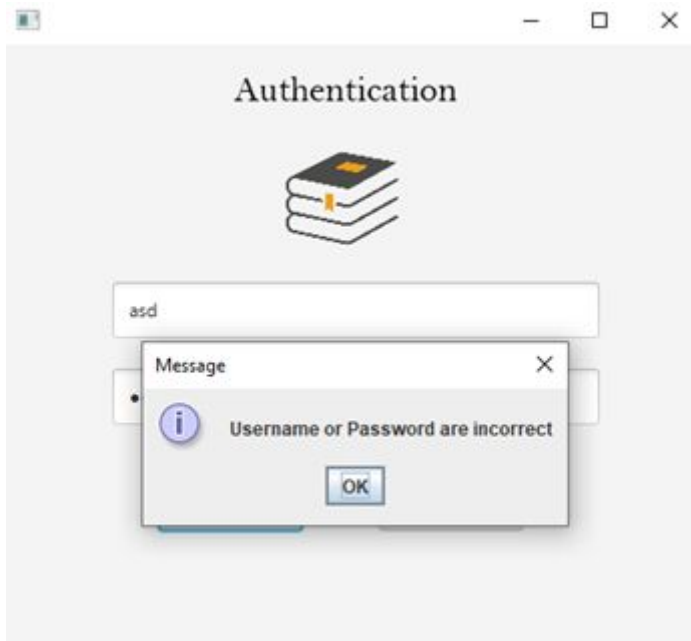
A screenshot of a window titled "Authentication". It features a stack of three books icon in the center. Below the icon are two input fields: the first is empty, and the second is labeled "Password". At the bottom, there are two buttons: "Sign In" with a right-pointing arrow icon and "Sign Up" with a house and plus icon.

Picture 1

A screenshot of a window titled "Registration". It contains five input fields: an empty field at the top, followed by "Lastname", "Username", "Password", "Repeat Password", and "Address". At the bottom, there are two buttons: "Sign Up" with a house and plus icon, and "Cancel" with a circle and slash icon.

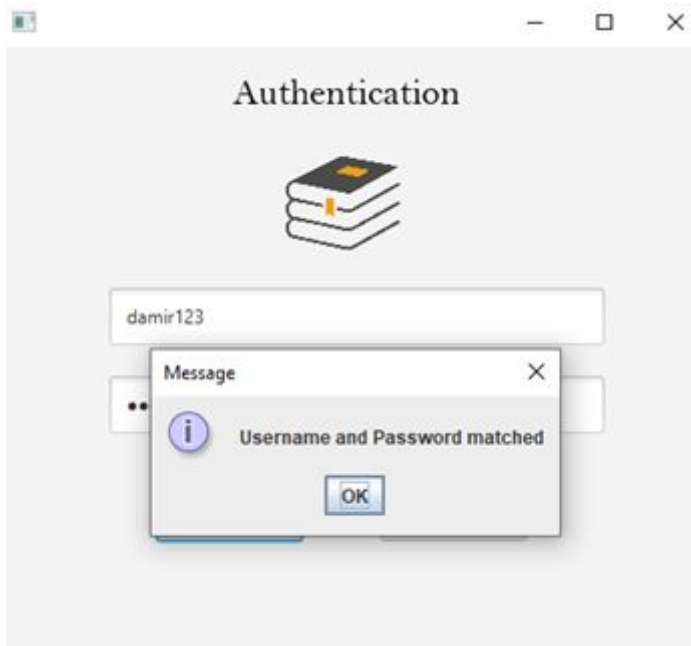
Picture 2

If *Username* or *Password* are wrong, it will “*Pop Up*” *Error* window.

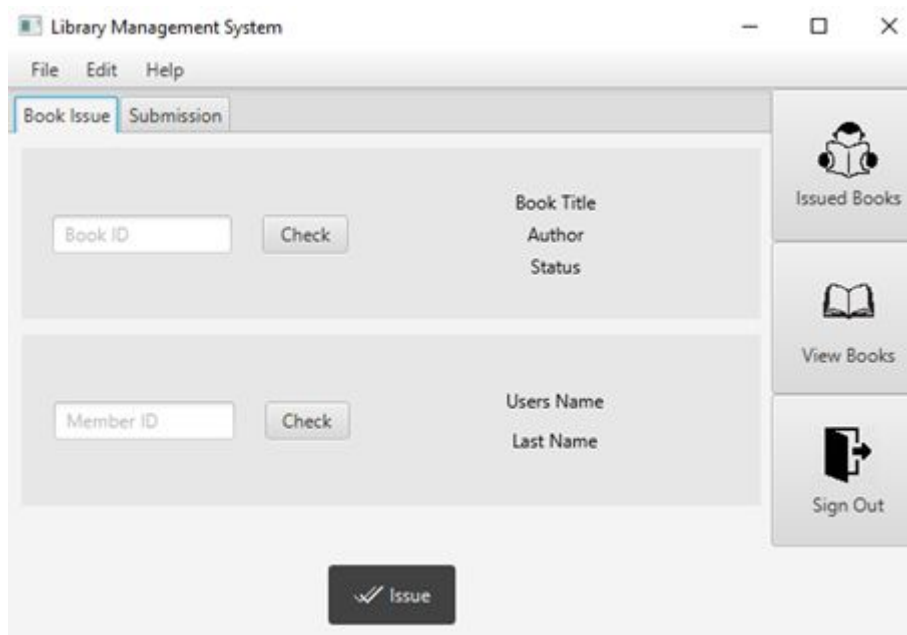


Picture 3

If all are correct, it will redirect into *Main Menu*.



Picture 4



Picture 5

Function (Issued Books):

When *Users* click *Issued Books* it will redirect into another *Window* with list of *Books* that he has taken, he can view all *Books* that he has *Issued*.

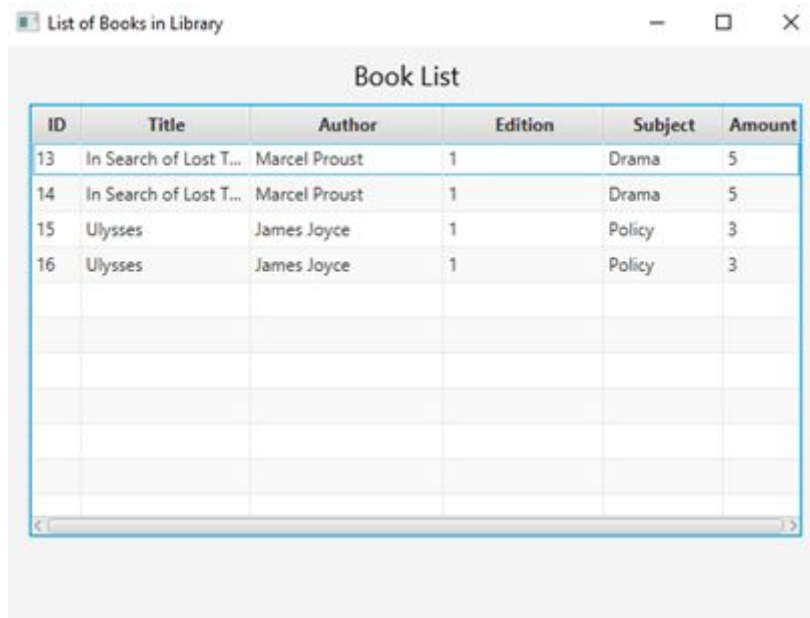
The screenshot shows a window titled "Issued Books" containing a table. The table has five columns: ID, Title, Author, Edition, and Subject. The first row contains the following data: ID 13, Title "In Search of Lost Time", Author "Marcel Proust", Edition 1, and Subject "Drama". The table has several empty rows below the first one. A scrollbar is visible at the bottom of the table.

ID	Title	Author	Edition	Subject
13	In Search of Lost Time	Marcel Proust	1	Drama

Picture 6

Function (View Books):

By clicking this [Button](#), [User](#) available to see [List](#) all of [Books](#) that [Library](#) contains.

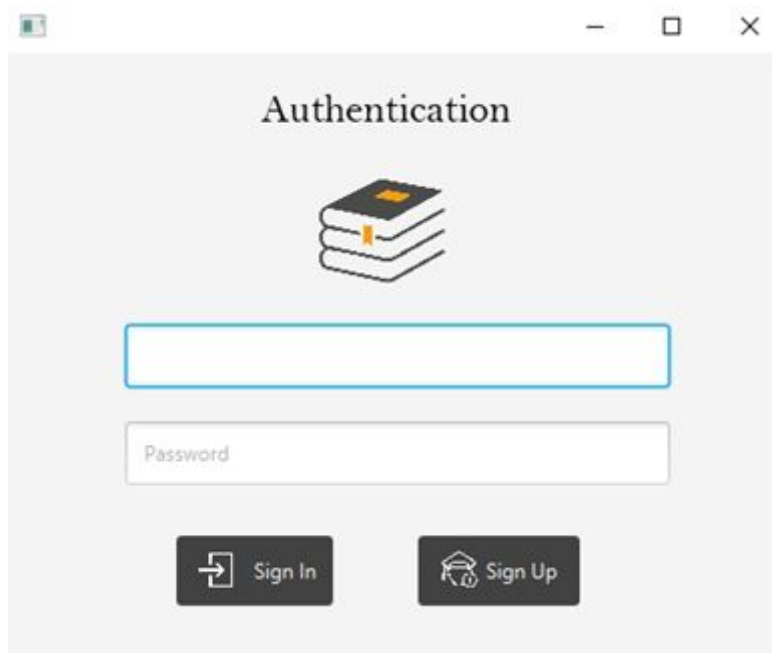


ID	Title	Author	Edition	Subject	Amount
13	In Search of Lost T...	Marcel Proust	1	Drama	5
14	In Search of Lost T...	Marcel Proust	1	Drama	5
15	Ulysses	James Joyce	1	Policy	3
16	Ulysses	James Joyce	1	Policy	3


Picture 7



Function (Sign out):

This [Button](#) uses to [Sign Out](#) to [Login Menu](#).



Authentication



 Sign In  Sign Up

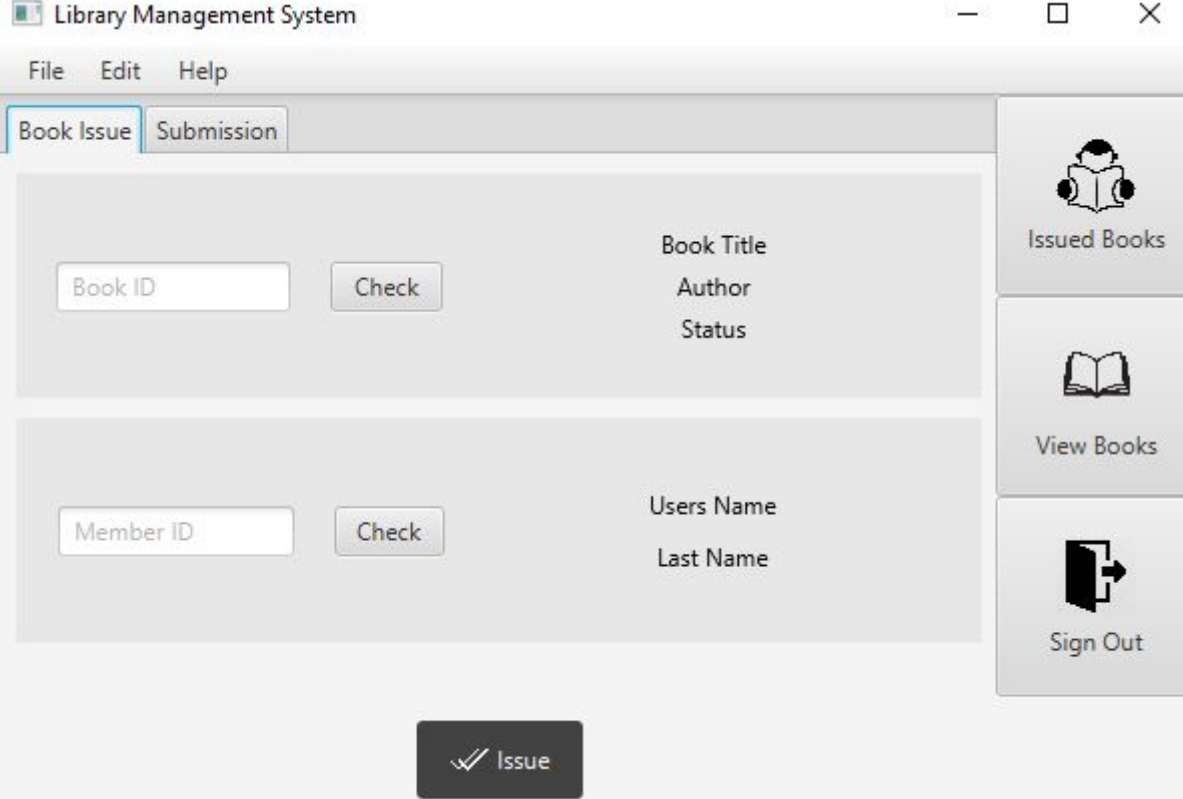
Picture 8

ISSUE PART

Function (Issue Book):

Firstly, *User* have to check which *Books* is available in *View Books* and choose which one he want to *Issue*. After he has chosen it, he enters its *ID* into *Book ID Field*. He can also check if that *Book*, he had chosen by clicking *Button Check*. After clicking in place *Book Title*, *Author*, *Status* will appear dates of *Book*.

He also need to enter own *User ID* and (is not necessary) can check the correctness. As soon as *User* finishes checking the dates, he must press *Button Issue*.



The screenshot shows a window titled "Library Management System" with a menu bar (File, Edit, Help) and two tabs: "Book Issue" (active) and "Submission". The "Book Issue" tab contains two main sections. The top section has a "Book ID" input field and a "Check" button. To the right of these are labels for "Book Title", "Author", and "Status". The bottom section has a "Member ID" input field and a "Check" button. To the right of these are labels for "Users Name" and "Last Name". At the bottom center of the window is a large dark button with a checkmark icon and the text "Issue". On the right side of the window is a vertical sidebar with three buttons: "Issued Books" (with a book icon), "View Books" (with an open book icon), and "Sign Out" (with a door icon).

Picture 9

System will open “*Pop Up*” *Success Window* to notify that you have successfully *Issued Book* if it is *OK* and by clicking *Issue Button* it will ask about your choice: “Are you sure want to issue this book?”



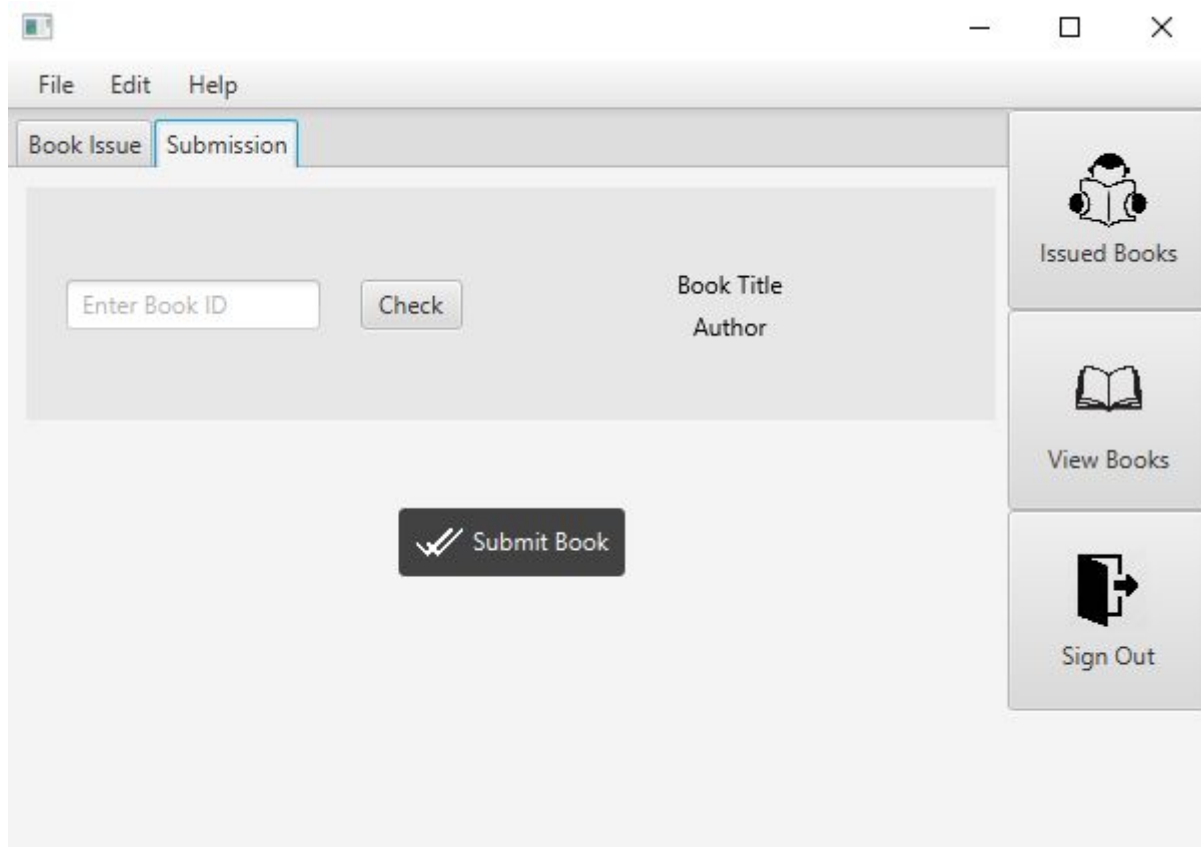
Picture 10



Picture 11

Function (Submission Book):

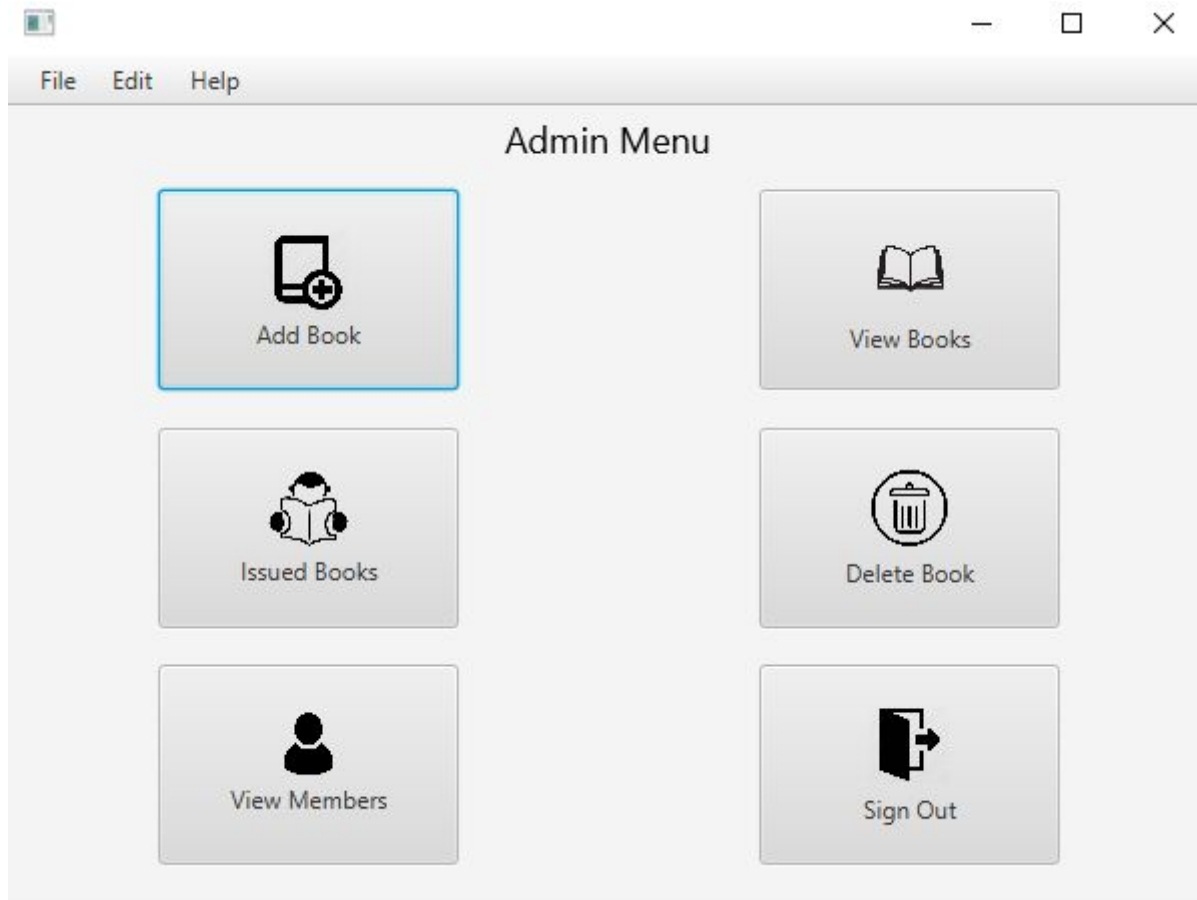
There is same procedure, *User* enters *Book ID* (*does not necessary*) and checks the dates, if it is correct he presses the *Button Submit Book*. There would “*Pop Up*” *Window* with *Success* message.



Picture 12

ADMIN MENU

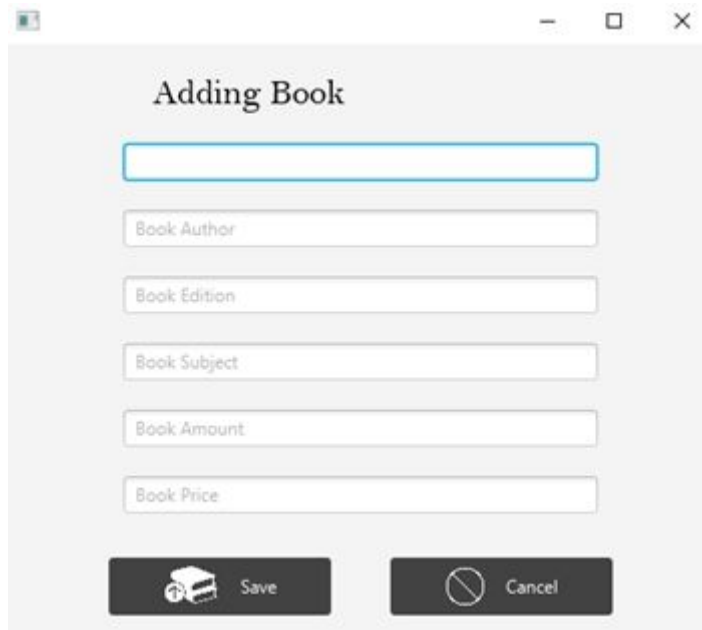
Only *Admin* can view *Admin Menu* and make changes here. There are six *Buttons*: *Add Book*, *View Books*, *Issued Books (all of Users)*, *Delete Book*, *View Members* and *Sign Out*.



Picture 13

Function (Add Book):

By clicking *Button Add Book*, it will redirect into another *Window* where there will be six *Text Fields* (*Book Title*, *Book Author*, *Book Edition*, *Book Subject*, *Book Amount* and *Book Price*). *Admin* has to fill all of these *Fields* and press *Button Save*.

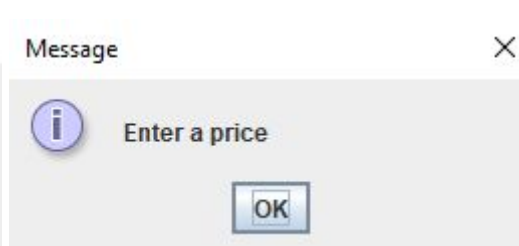


Picture 14

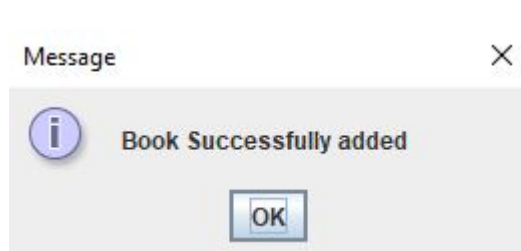
If there is any *Field*, which is not filled, it will “*Pop Up*” *Error Window*, otherwise *Success Window*. It also checks correctness of input.



Picture 15



Picture 16



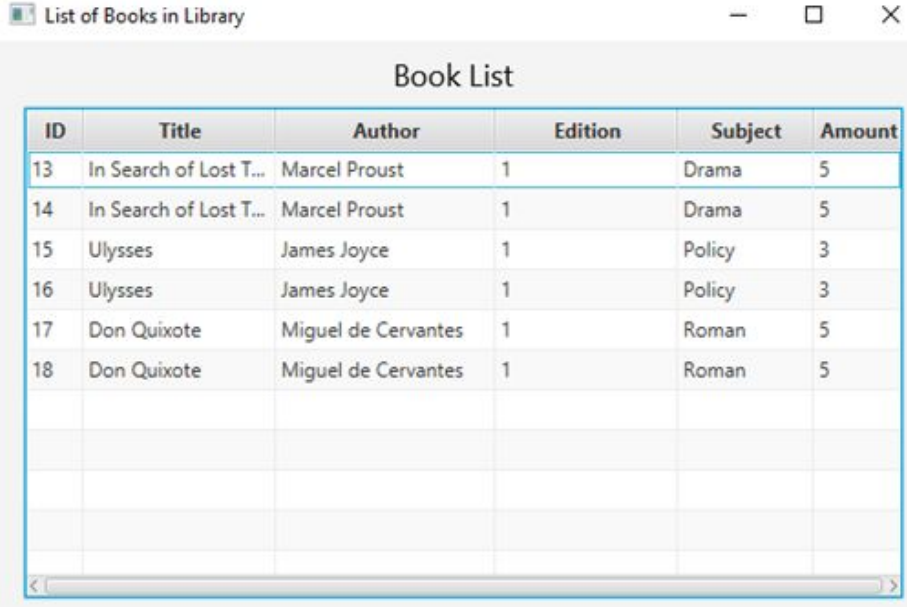
Picture 17



Picture 18

Function (View Books):

By clicking [Button View Books](#), it will open a [Window](#) with [List of Books](#). There [Admin \(also User\)](#) can see all of six field that he has filled before.

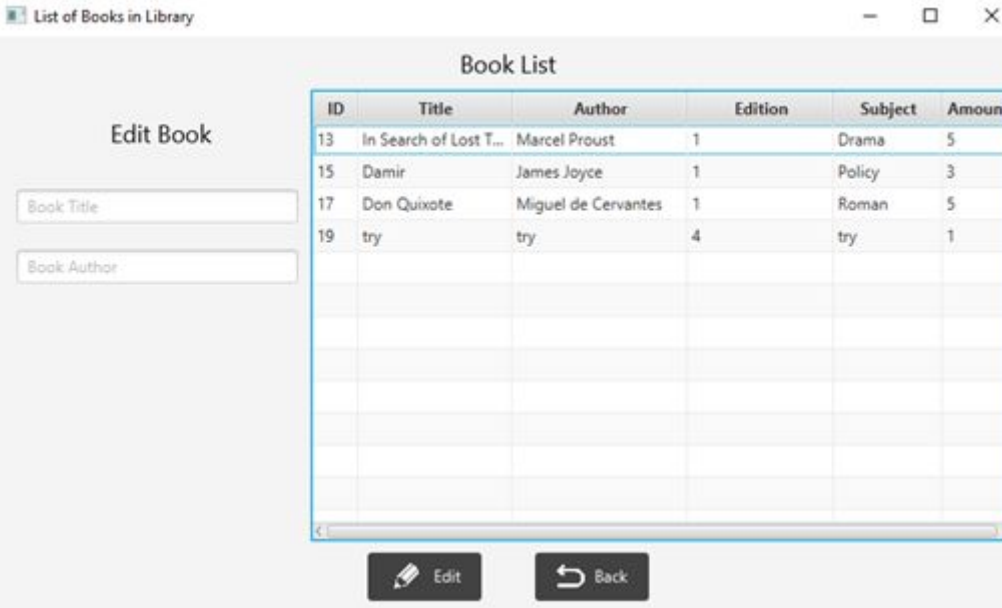


Book List

ID	Title	Author	Edition	Subject	Amount
13	In Search of Lost T...	Marcel Proust	1	Drama	5
14	In Search of Lost T...	Marcel Proust	1	Drama	5
15	Ulysses	James Joyce	1	Policy	3
16	Ulysses	James Joyce	1	Policy	3
17	Don Quixote	Miguel de Cervantes	1	Roman	5
18	Don Quixote	Miguel de Cervantes	1	Roman	5

Picture 18

Additional function for [Admin](#) is [Edit](#) option, there two fields that any of these fields could be fill. If he wants change data one of [Books](#), he must choose row corresponding to that [Book](#) and enter the data for which he wants to change.



Book List

Edit Book

Book Title

Book Author

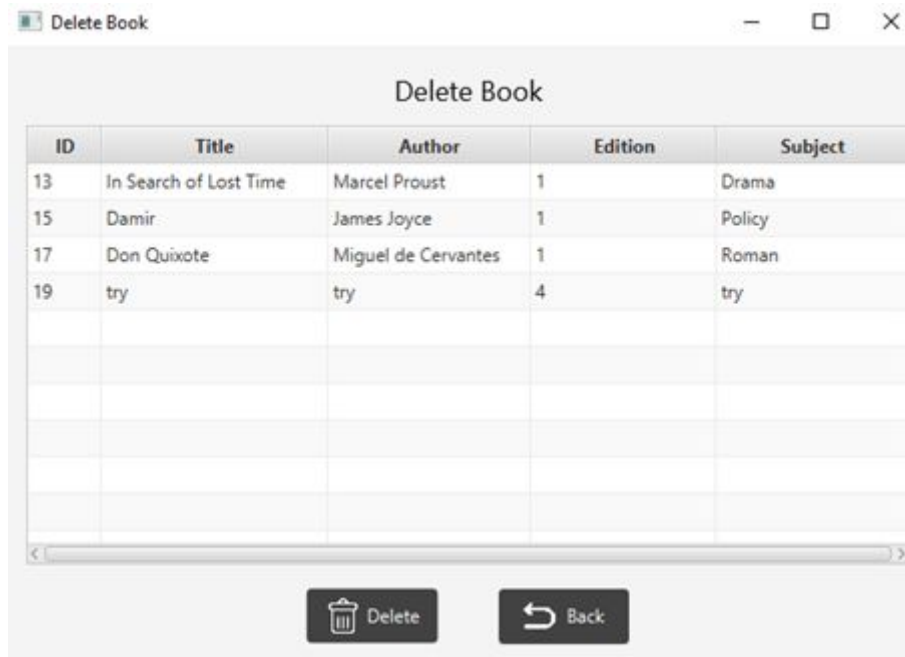
ID	Title	Author	Edition	Subject	Amount
13	In Search of Lost T...	Marcel Proust	1	Drama	5
15	Damir	James Joyce	1	Policy	3
17	Don Quixote	Miguel de Cervantes	1	Roman	5
19	try	try	4	try	1

Edit Back

Picture 19

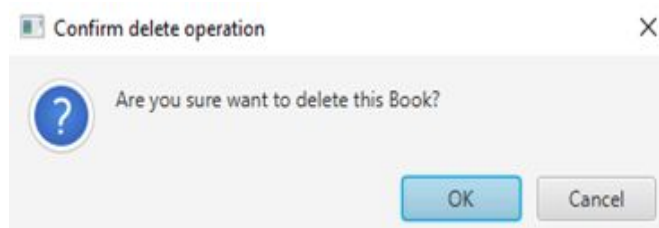
Function (Delete Book):

By clicking, *System* opens a window like *Book List*. There will be *Table* with all *Books*. *Admin* have to select row which he wants to *Delete* after click *Button Delete* and it will *Delete*. Also if *Admin* change mind and do not want to *Delete Books*, he press the *Button Back* and goes back to *Admin Menu*.

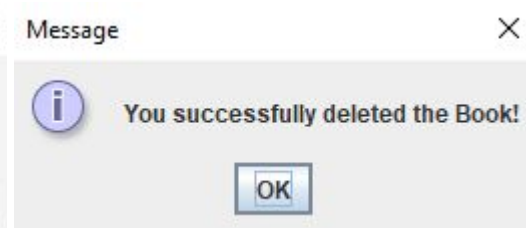


Picture 20

System also ask if you are sure want to delete this *Book*. If it is *OK*, it will show *Success Message*.



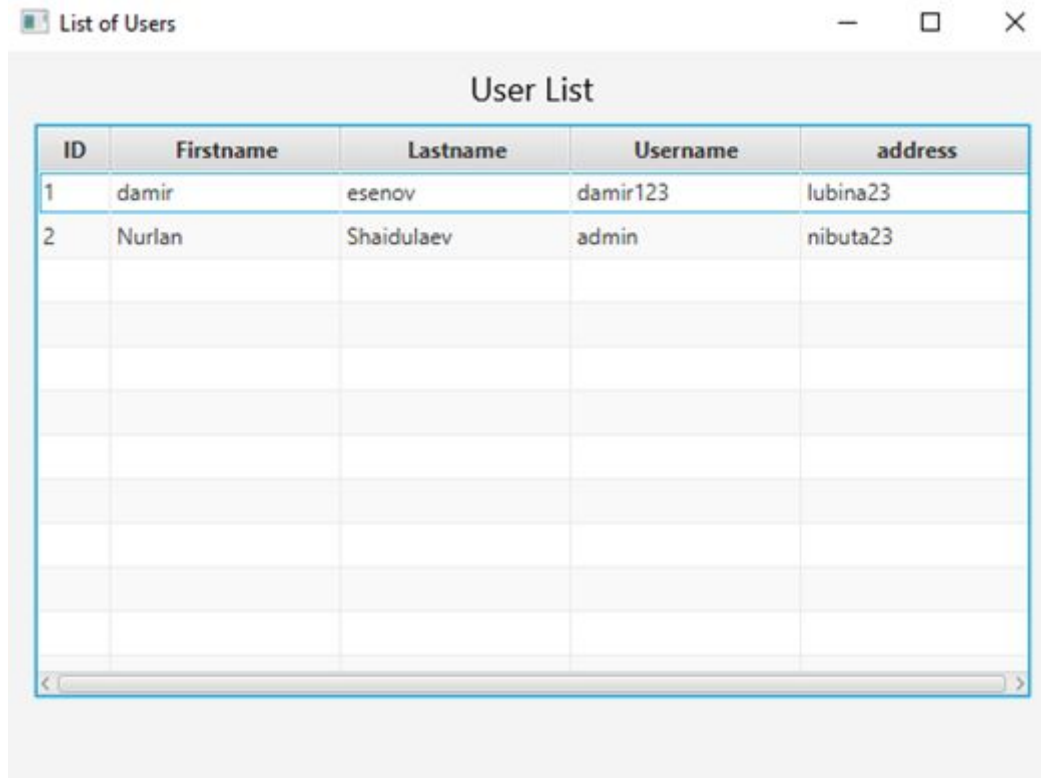
Picture 21



Picture 22

Function (View Users):

This window uses only to monitor which [Users](#) has registered.



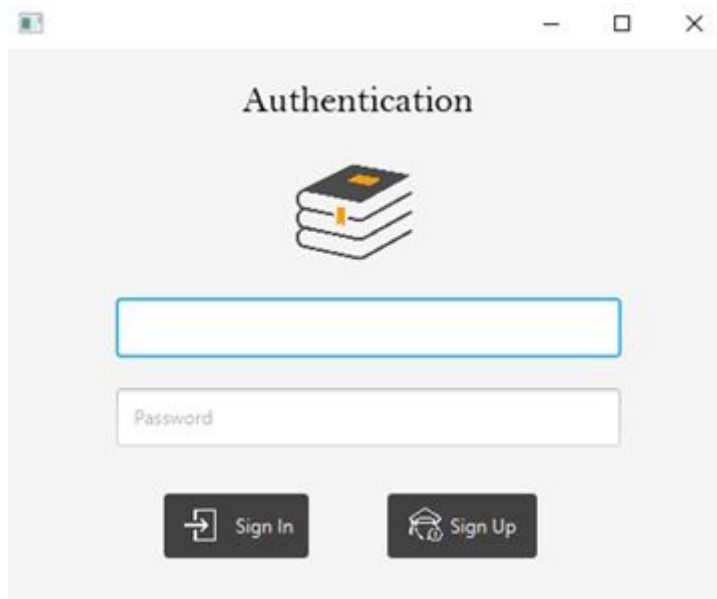
The screenshot shows a window titled "List of Users" with a standard Windows-style title bar (minimize, maximize, close buttons). The main content area is titled "User List" and contains a table with the following data:

ID	Firstname	Lastname	Username	address
1	damir	esenov	damir123	lubina23
2	Nurlan	Shaidulaev	admin	nibuta23

Picture 23

Function (Sign Out):

By clicking this [Button](#), it will redirect to [Login Menu](#).



The screenshot shows a window titled "Authentication" with a standard Windows-style title bar. The main content area features a stack of books icon, a text input field, a password input field labeled "Password", and two buttons: "Sign In" (with a right-pointing arrow icon) and "Sign Up" (with a house and plus icon).

Picture 24

MYSQL WORKBENCH

In this program, I have used MySQL Database ([link for download](#))

There are three *Tables* in *Library Database*: *book_table*, *issued_books_table*, *users_table*.

Users_table:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
⚡ idusers_table	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
◇ first_name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ last_name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ user_name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ password	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ address	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Picture 25

```
// Dates for User Table
public static final String USER_ID = "idusers_table";
public static final String USER_FN = "first_name";
public static final String USER_LN = "last_name";
public static final String USERNAME = "user_name";
public static final String USER_ADDRESS = "address";
```

Picture 26

Book_table:





Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
⚡ idbook_table	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
◇ book_title	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ book_author	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ book_edition	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ book_subject	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ book_amount	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ book_price	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Picture 27


```
// Dates for Book Table
public static final String BOOK_ID = "idbook_table";
public static final String BOOK_TITLE = "book_title";
public static final String BOOK_AUTHOR = "book_author";
public static final String BOOK_EDITION = "book_edition";
public static final String BOOK_SUBJECT = "book_subject";
public static final String BOOK_AMOUNT = "book_amount";
public static final String BOOK_PRICE = "book_price";
```

Picture 28

Issued_books_table:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 book_id	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 user_id	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 issue_time	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

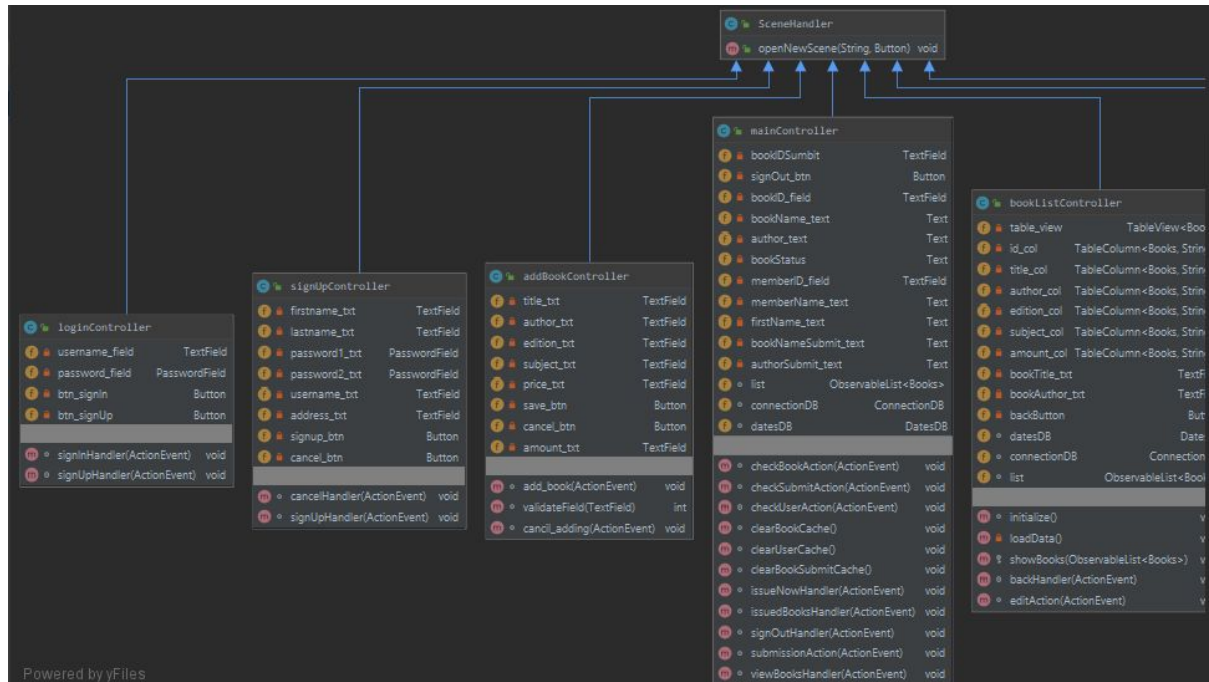
Picture 29

```
// Dates for Issued Table
public static final String ISSUE_ID = "id";
public static final String BOOK_ID_ISSUE = "book_id";
public static final String USER_ID_ISSUE = "user_id";
public static final String ISSUE_TIME = "issue_time";
```

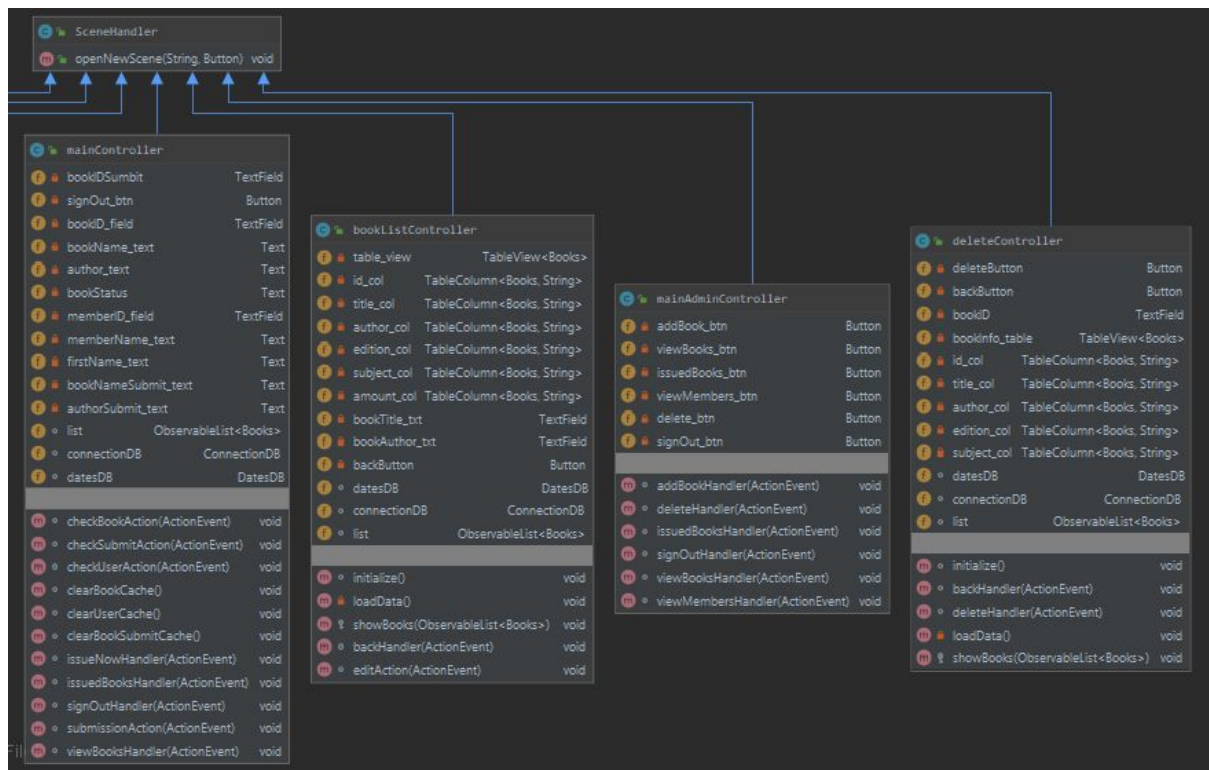
Picture 30

UML DIAGRAMS

Controller Diagrams:

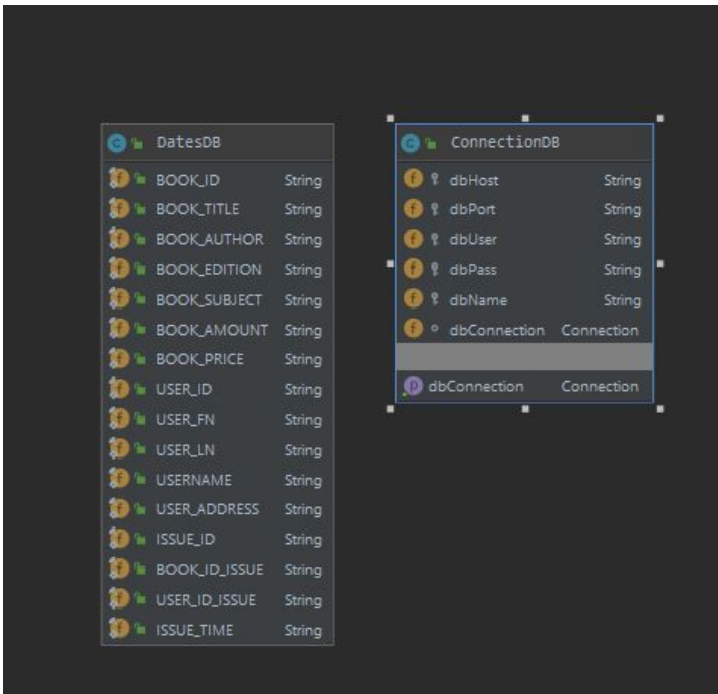


Picture 31



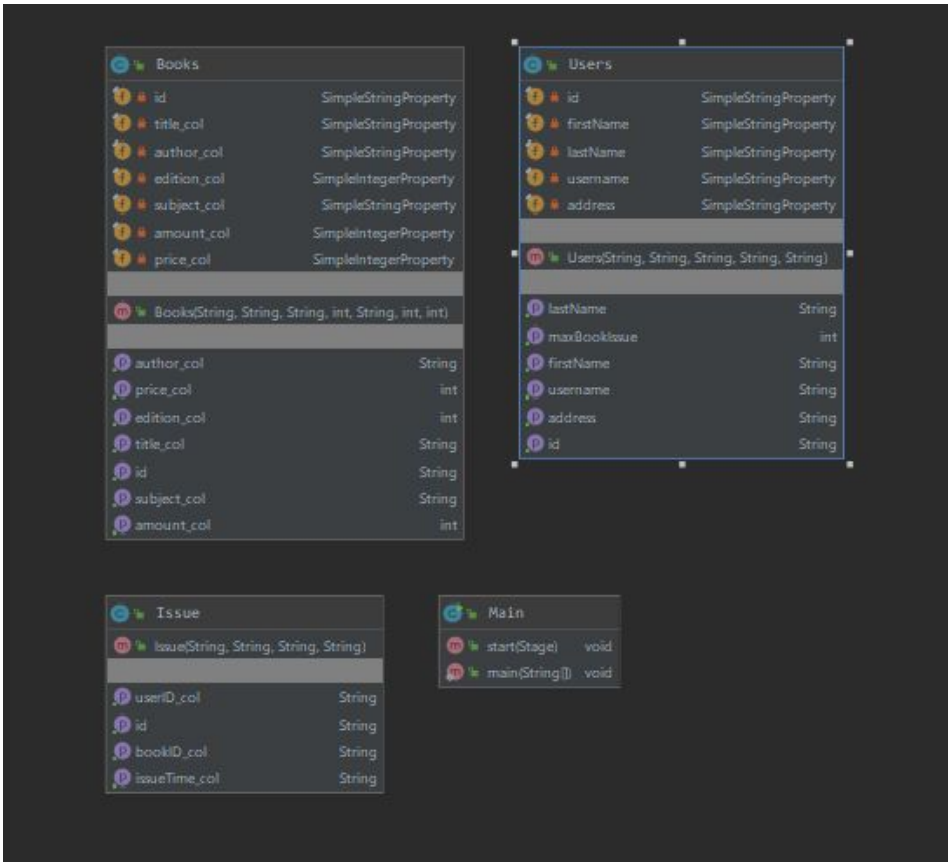
Picture 32

Database Diagram:



Picture 33

Model Diagram:



Picture 30

CODE EXPLANATION

There is explanation of code and what it does. I have explained here systematically so that it will surely help users to become more user friendly with it.

Controller Folder:

There are all controllers for all fxml files. They need to able the system work right.

addBookController.java:

- **add_book()** – this method responds for taking dates from *Text Fields* and insert it into DB.
- **cancel_adding()** – this method stands for cancel *Adding Book*, if *User & Admin* change they mind.

bookListController.java:

- **initialize()** – this method responds for executing code as soon as fxml file opens. It contains two functions: **loadData()** and **showBooks()**.
- **loadData()** – this method stands for loading *Book* dates from *DB* and storing it into *ArrayList*.
- **showBooks(Observable<Books> list)** – this method has entering *ArrayList* with type of dates *Book*. It stands for inserting dates from list into *Columns in Table*.
- **editAction()** - this method is available only for *Admin*, it gets two dates from *Text Field* and change *Books Dates in DB*.

deleteController.java:

- **backHandler()** – this method stands for going back into *Main Menu*.
 - **deleteHandler()** – this is main method, it deletes *Book from DB*.
 - **initialize()** – this method responds for executing code as soon as fxml file opens. It contains two functions: **loadData()** and **showBooks()**.
 - **loadData()** – this method stands for loading *Book* dates from *DB* and storing it into *ArrayList*.
 - **showBooks(Observable<Books> list)** – this method has entering *ArrayList* with type of dates *Books*. It stands for inserting dates from list into *Columns in Table*.
-

issuedController.java:

- **initialize()** – this method responds for executing code as soon as fxm1 file opens. It contains two functions: **loadData()** and **showBooks()**.
- **loadData()** – this method stands for loading Issue dates from DB and storing it into ArrayList.
- **showBooks(Observable<Issue> list)** – this method has entering *ArrayList* with type of dates *Books*. It stands for inserting dates from list into *Columns in Table*.

issuedUserController.java:

- **initialize()** – this method responds for executing code as soon as fxm1 file opens. It contains two functions: **loadData()** and **showBooks()**.
- **loadData()** – this method stands for loading *Issued* dates from *DB* and storing it into *ArrayList*.
- **showBooks(Observable<Issue> list)** – this method has entering *ArrayList* with type of dates *Issue*. It stands for inserting dates from list into *Columns in Table*.

loginController.java:

- **signInHandler()** – this method response for checking if *Username* and *Password* matches in *DB*, if it does it will redirect into *Main Menu*.
- **signUpHandler()** – this method stands for redirecting into *Sign Up Window* for registering new *User*.

mainAdminController.java:

- **addBookHandler()** – this method needs to redirect *Admin* into *Add Book Window*.
 - **deleteHandler()** – this method stands for redirecting *Admin* into *Delete Book Window*.
 - **issuedBooksHandler()** – this method response for redirecting *Admin* into *List of Issued Books* by *Users*.
 - **signOutHandler()** – this method stands for *Signing Out* and redirecting into *Login Menu*.
 - **viewBooksHandler()** – this method response for redirecting *Admin* into *View Books Window*.
-

- **viewMembersHandler()** – this method needs for redirecting *Admin* into *List of Users*.

mainController.java:

- **checkBookAction()** – this method needs for checking if *User* has chosen right *Book*, it takes *ID* from *Text Field* and make search in *DB*, takes *Book Title, Author and Status* and shows it to *User*.
- **checkSubmitAction()** – this method response for checking if *User* has chosen right *Book for Submitting*.
- **checkUserAction()** – this method needs for checking if *User* filled correct *ID*.
- **clearBookCache(), clearUserCache() and clearBookSubmitCache()** – needs for clearing dates that was filling.
- **issueNowHandler()** – responds for issuing *Book*.
- **issuedBooksHandler()** – this method responds for redirecting *User* into *List of Issued Books*.
- **signOutHandler()** – this method stands for *Signing Out* and redirecting into *Login Menu*.
- **submissionAction()** – this method responds for submitting *Book* to *Library*. It takes *Book ID* and make search in *DB* after matching it deletes from that *Table*.
- **viewBooksHandler()** – this method response for redirecting *Admin* into *View Books Window*.

SceneHandler.java – needs for opening new *Window* and closing old one.

signUpController.java:

- **cancelHandler()** – stands for cancel *Sign Up* operation, it will redirect to *Login Menu*.
 - **signUpHandler()** – responds for adding *User* into *DB*.
-

userListController.java:

- **initialize()** – this method responds for executing code as soon as fxml file opens. It contains two functions: **loadData()** and **showUsers()**.
- **loadData()** – this method stands for loading *Users* dates from *DB* and storing it into *ArrayList*.
- **showBooks(Observable<Issue> list)** – this method has entering *ArrayList* with type of dates *Users*. It stands for inserting dates from list into *Columns in Table*.

DatabaseHandler Folder:

ConnectionDB.java – this class needs to establish connection between *IntelliJ* and *DB*.

DatesDB.java – this class responds for storing dates for three tables: *book_table*, *users_table*, *issued_books_table*.

References:

<https://bytescout.com/blog/20-important-sql-queries.html>

<https://cdn.mysql.com//Downloads/MySQLInstaller/mysql-installer-community-8.0.18.0.msi>

<https://dev.mysql.com/downloads/workbench/>

<https://material.io/resources/icons/?style=baseline>

https://www.youtube.com/watch?v=9d3X8eBov1M&list=PLhs1urmduZ29jTcE1ca8Z6bZNvH_39ayL

