Inha University in Tashkent

Application Programming in Java, Fall 2020

Group Project
Library Management System

JavaDays Team

Mamasaidov Mukhammadsaid U1910223 Section 005
Yusupov Jasur U1910236 Section 005
Allaev Bekzod U1910240 Section 005
Khasanov Asadbek U1910103 Section 003
Saidakhmedov Saidakbar U1910222 Section 005

Github URL: https://github.com/iuthub/group-project-javadays

Submission Date: 10.01.2021

User Accounts

Admin Account

Login: U1400000

Password: 1234

Librarian Account

Login: U1530000

Password: 1234

Student Accounts

Login: U1910000

Password: 1234

Login: U1910223

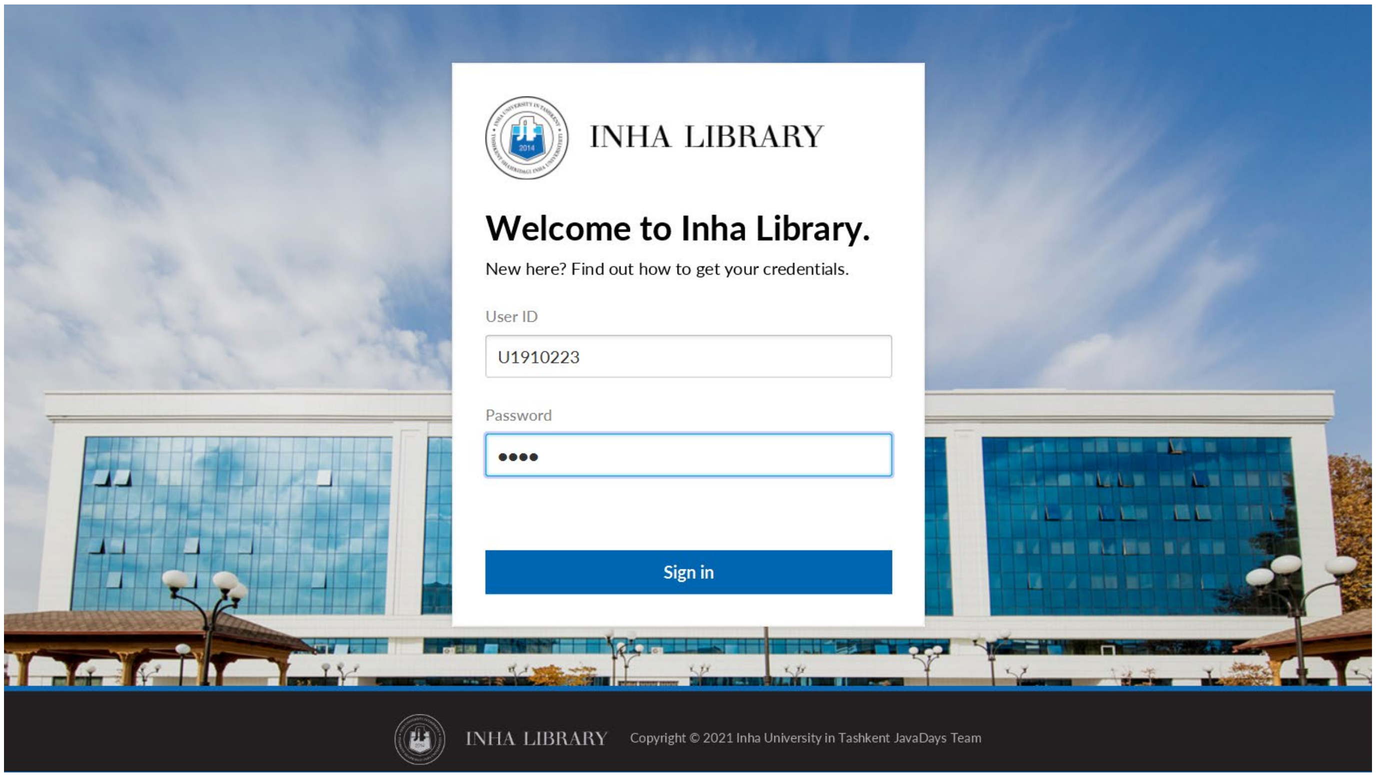
Password: 1234

Login: U1910236

Password: 1234

Description of Implemented Functionality

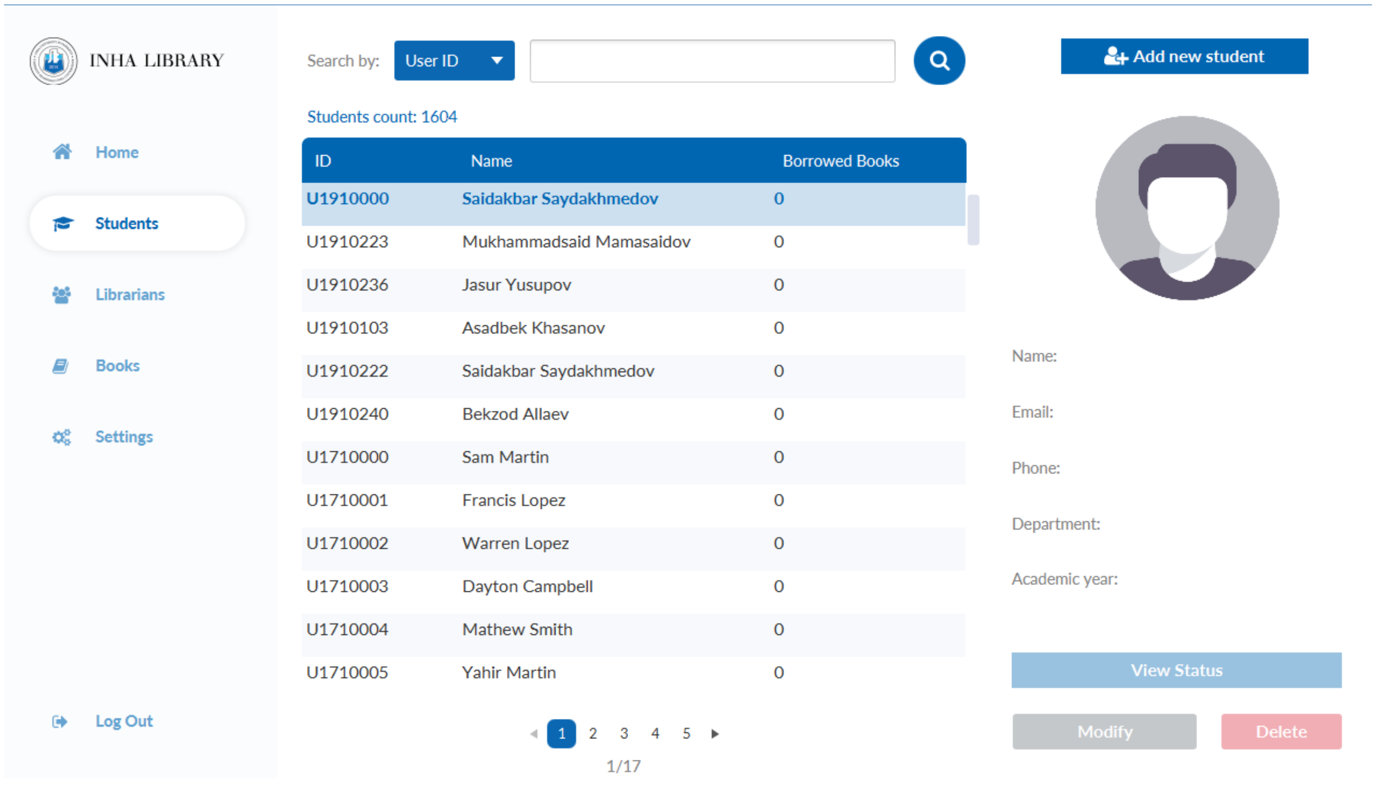
Login Interface


INHA LIBRARY
Welcome to Inha Library.
New here? Find out how to get your credentials.
User ID
U1910223
Password
••••
Sign in
INHA LIBRARY
Copyright © 2021 Inha University in Tashkent JavaDays Team

This is our Login Page Starter Window in the application. It includes validation check before sending data to database. Depending what ID was inputted it then displays corresponding Admin, Librarian and Student windows.

Overall Structure

In our application we used BorderPane as a main window Pane**.** Using its natural structure for creating navigation bars, we have a made a menu in the left child, and wrote a function that changes its center content to some fxml.



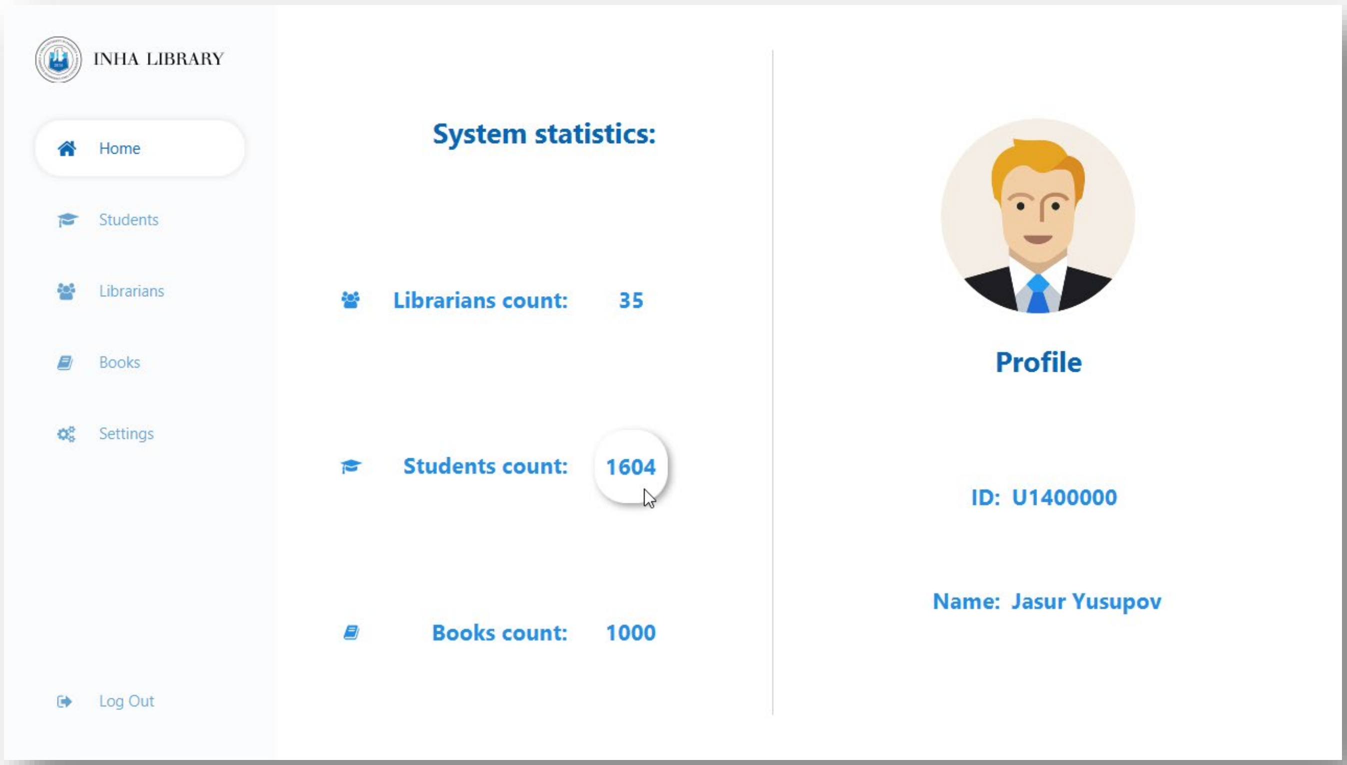
Students count: 1604

ID	Name	Borrowed Books
U1910000	Saidakbar Saydakhmedov	0
U1910223	Mukhammadsaid Mamasaidov	0
U1910236	Jasur Yusupov	0
U1910103	Asadbek Khasanov	0
U1910222	Saidakbar Saydakhmedov	0
U1910240	Bekzod Allaev	0
U1710000	Sam Martin	0
U1710001	Francis Lopez	0
U1710002	Warren Lopez	0
U1710003	Dayton Campbell	0
U1710004	Mathew Smith	0
U1710005	Yahir Martin	0

Here you can see Admin's window that is in our opinion user-friendly, since most web-applications are based on navigation bars.

Moreover, we have made a heavy use of CSS in our application. JavaFX provides specific runaround for writing CSS files with specific selectors beginning with -fx-.

Admin Interface



This is the **Home** view of administrator window, the view is simple and understandable at first glance. Left side shows statistics of the system, right side displays admin's profile.

Statistics side has a little GUI feature: If you hover the mouse over counters they will pop up and drop a shadow which is quite enjoyment for user's eyes. This is done via creating a class in css:

```
.label-result: hover {

   -fx-background-color: white;

   -fx-background-radius: 30px;

   -fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.3), 10, 0.2, 3.0, 3.0);

}
```

Where: dropshadow(<blur-type>, <color>, <radius>, <spread>, <x-offset>, <y-offset>)

Total count of Librarians, Students and Books are retrieved using SQL COUNT() function through Derby database.

General SQL query for users: SELECT COUNT(*) FROM Users WHERE Role=?

```
public int getTotalCount(Role role) throws SQLException{

   getCountStmt.setInt(1, role.getValue());

   ResultSet result = getCountStmt.executeQuery();
```

```
    result.next();

    return result.getInt(1);}
```

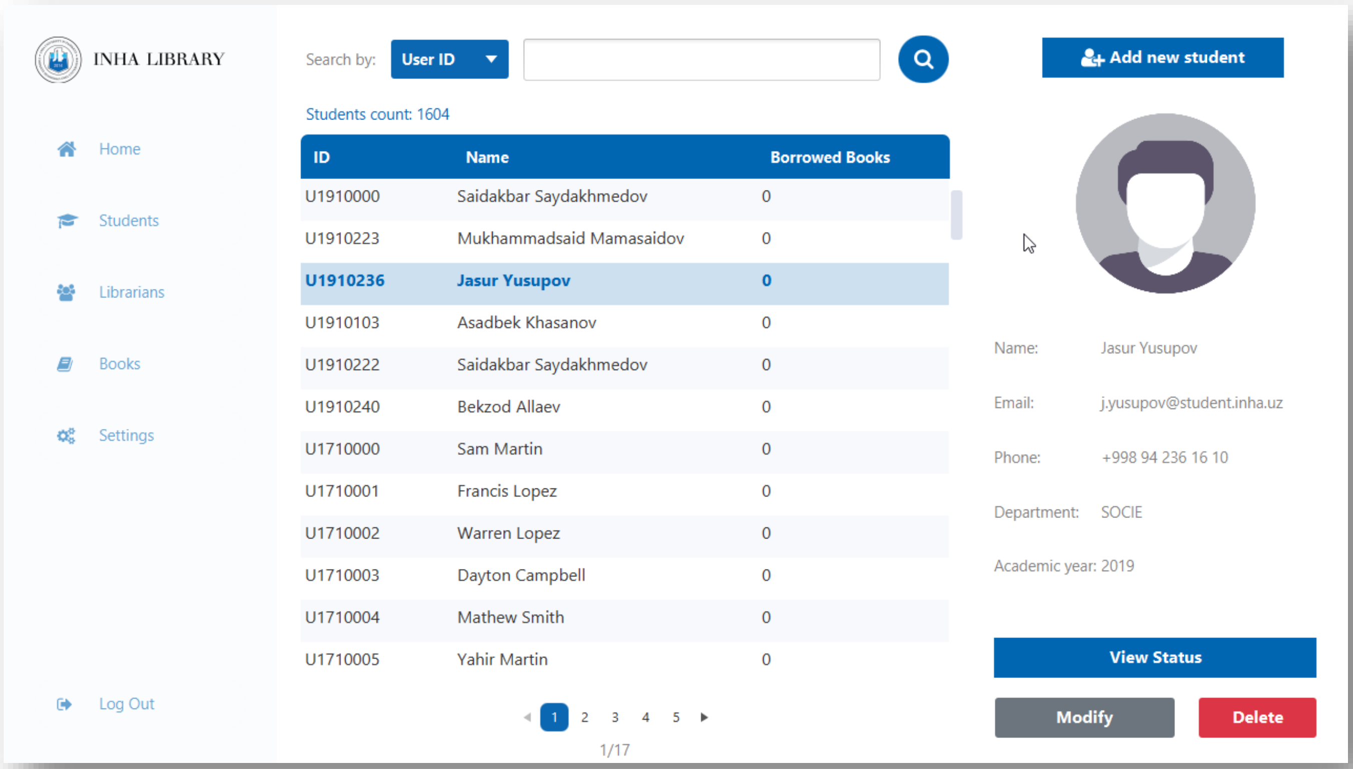
SQL qeury for books: SELECT COUNT(*) FROM Books

```
public int getTotalCount() throws SQLException{

    ResultSet r = getCountStmt.executeQuery();

    r.next();

    return r.getInt(1);}
```



Students count: 1604

ID	Name	Borrowed Books
U1910000	Saidakbar Saydakhmedov	0
U1910223	Mukhammadsaid Mamasaidov	0
U1910236	Jasur Yusupov	0
U1910103	Asadbek Khasanov	0
U1910222	Saidakbar Saydakhmedov	0
U1910240	Bekzod Allaev	0
U1710000	Sam Martin	0
U1710001	Francis Lopez	0
U1710002	Warren Lopez	0
U1710003	Dayton Campbell	0
U1710004	Mathew Smith	0
U1710005	Yahir Martin	0

1 2 3 4 5

1/17

Name:	Jasur Yusupov
Email:	j.yusupov@student.inha.uz
Phone:	+998 94 236 16 10
Department:	SOCIE
Academic year:	2019

This is the **Students** view of administrator window. On the left side there is a table which displays all users of the system. Table has a pagination functionality implemented where each page contains 100 records. Table displays students' ID, name and their borrowed books count.

Pagination functionality for table view is implemented with adding OFFSET SQL Operator to SELECT Qeury:

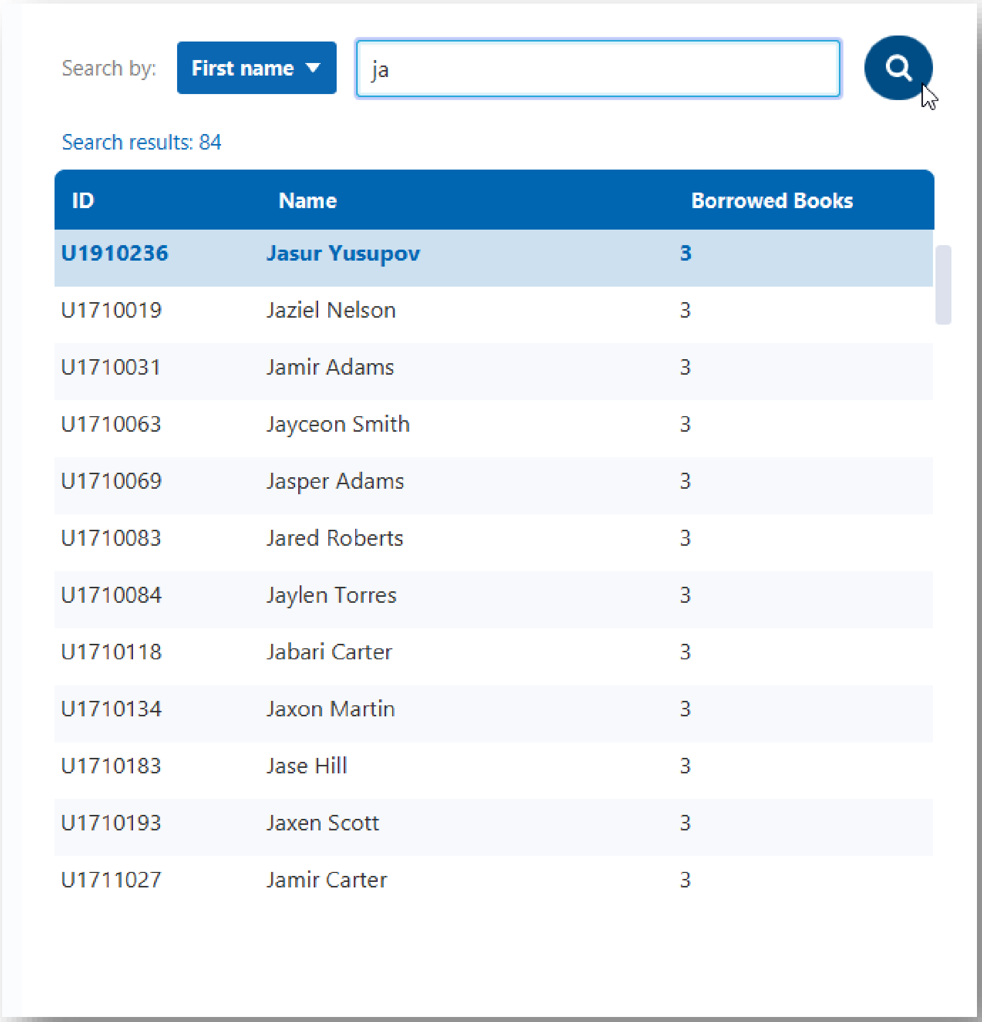
```
OFFSET ? ROWS FETCH NEXT 100 ROWS ONLY
```

For example, when page 2 is selected, `OFFSET` will be 2*100 = 200, and Derby retrieves rows from 200 to 300.

On top of the table view there is a Label that displays total count of Students in the system.

On the top, there is a Search bar with ChoiceBox where admin chooses by what paramter to search. Search can be accomplished by User ID, First name and Last name. Search button can be activated by pressing Enter on the keyboard which is quite comfortable for user.

One of the features of the Search bar is that it is programmed to ignore letter cases of the entered search query. So it doesn't matter whether the query is written in lower case or upper case.


Search by: First name ▾ ja

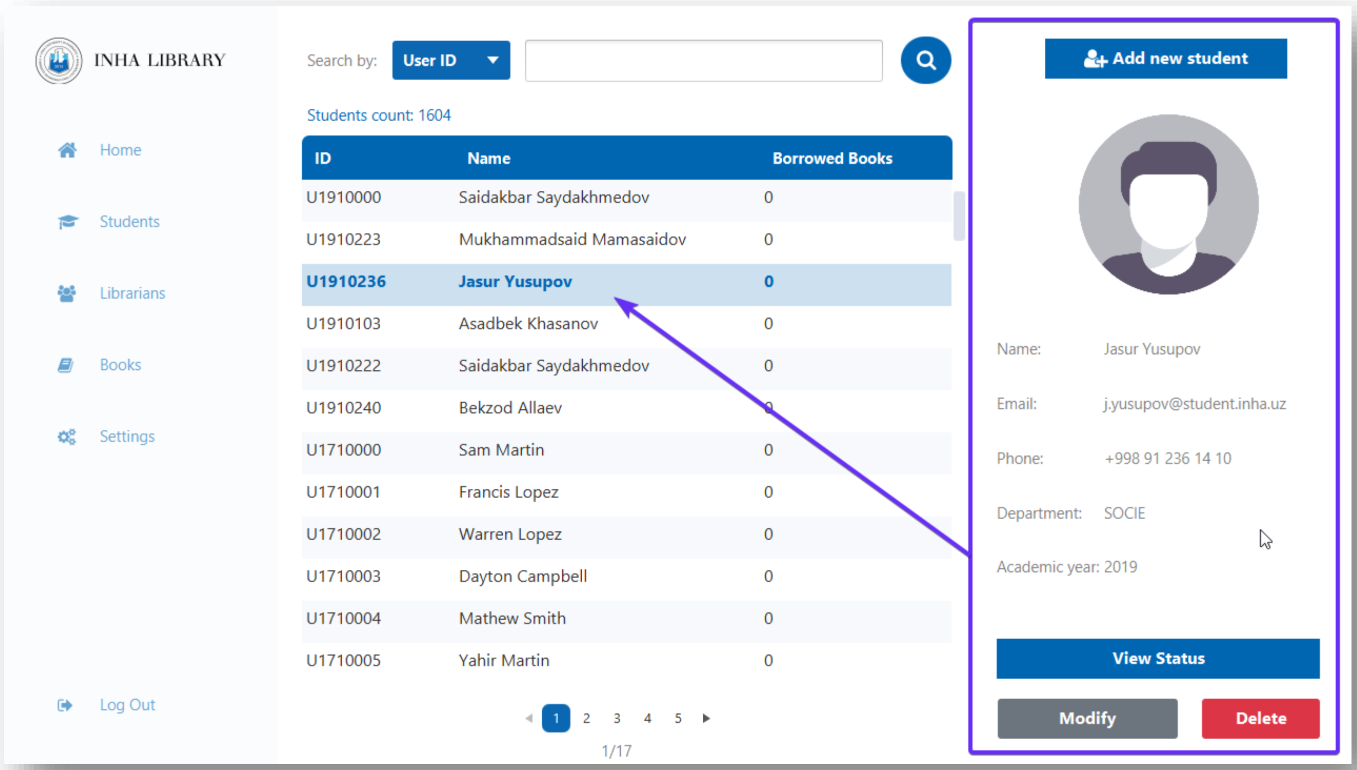
Search results: 84

ID	Name	Borrowed Books
U1910236	**Jasur Yusupov**	**3**
U1710019	Jaziel Nelson	3
U1710031	Jamir Adams	3
U1710063	Jayceon Smith	3
U1710069	Jasper Adams	3
U1710083	Jared Roberts	3
U1710084	Jaylen Torres	3
U1710118	Jabari Carter	3
U1710134	Jaxon Martin	3
U1710183	Jase Hill	3
U1710193	Jaxen Scott	3
U1711027	Jamir Carter	3

Main feature of the search functionality is that we used LIKE SQL operator to search.

For example, when user searches “ja” selecting “First name” option, table displays all students whose name starts with “Ja”.

Another key feature is that when searching by ID, user(admin) can search without typing letter “U”. For example, search queries “U1910236” and “1910236” work same and return same results.



Students count: 1604

ID	Name	Borrowed Books
U1910000	Saidakbar Saydakhmedov	0
U1910223	Mukhammadsaid Mamasaidov	0
U1910236	Jasur Yusupov	0
U1910103	Asadbek Khasanov	0
U1910222	Saidakbar Saydakhmedov	0
U1910240	Bekzod Allaev	0
U1710000	Sam Martin	0
U1710001	Francis Lopez	0
U1710002	Warren Lopez	0
U1710003	Dayton Campbell	0
U1710004	Mathew Smith	0
U1710005	Yahir Martin	0

1 2 3 4 5

1/17

Name:	Jasur Yusupov
Email:	j.yusupov@student.inha.uz
Phone:	+998 91 236 14 10
Department:	SOCIE
Academic year:	2019

On right side, there is a student control panel. When student is selected in the table, his/her information appears on right side and “View Status”, “Modify”, “Delete” buttons become available to be pressed (used). Student information is generated from student’s ID and Name. Only phone number is generated randomly.

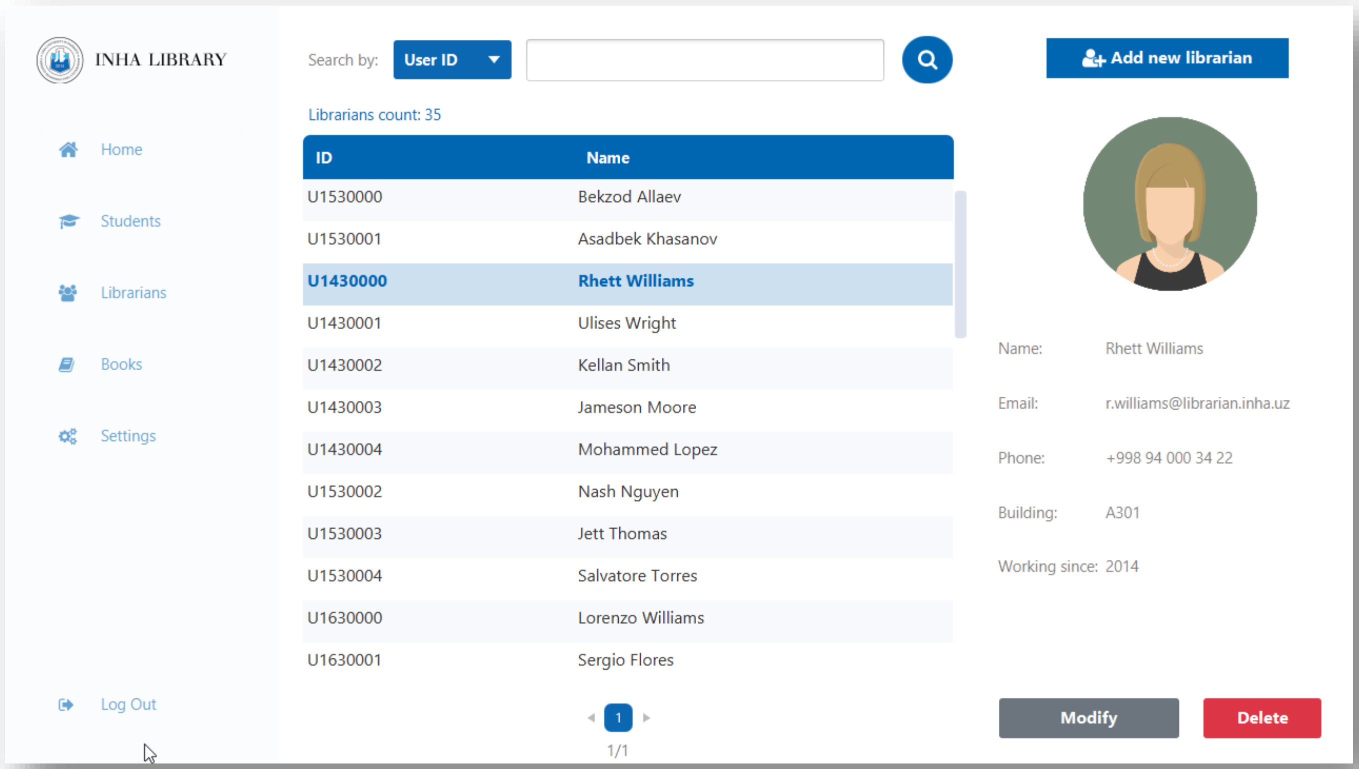
```
String generateEmail(String name){
    String[] n = name.split(" ");
    return String.format("%s.%s@student.inha.uz", n[0].substring(0, 1).toLowerCase(), n[1].toLowerCase());
}


String generateDepartment(){
    String[] d = {"SOCIE", "SOL"};
    int department = Integer.parseInt(selectedStudentId.substring(3, 5)) - 10;
    if (department > -1) return d[department];
    return d[new Random().nextInt(2)];
}


String generateAcademicYear(){
    int year = Integer.parseInt(selectedStudentId.substring(1, 3));
    if (year <= 20 && year >= 14){
        return String.format("20%d", year);
    }
    return String.format("%d", 2014 + new Random().nextInt(7));
}
```

When “View Status button is pressed, dialog showing students current status and fine pops up. Admin can modify or delete selected student by pressing “Modify” or “Delete” buttons resepectively.

On the top, there is button for adding new students to the system.

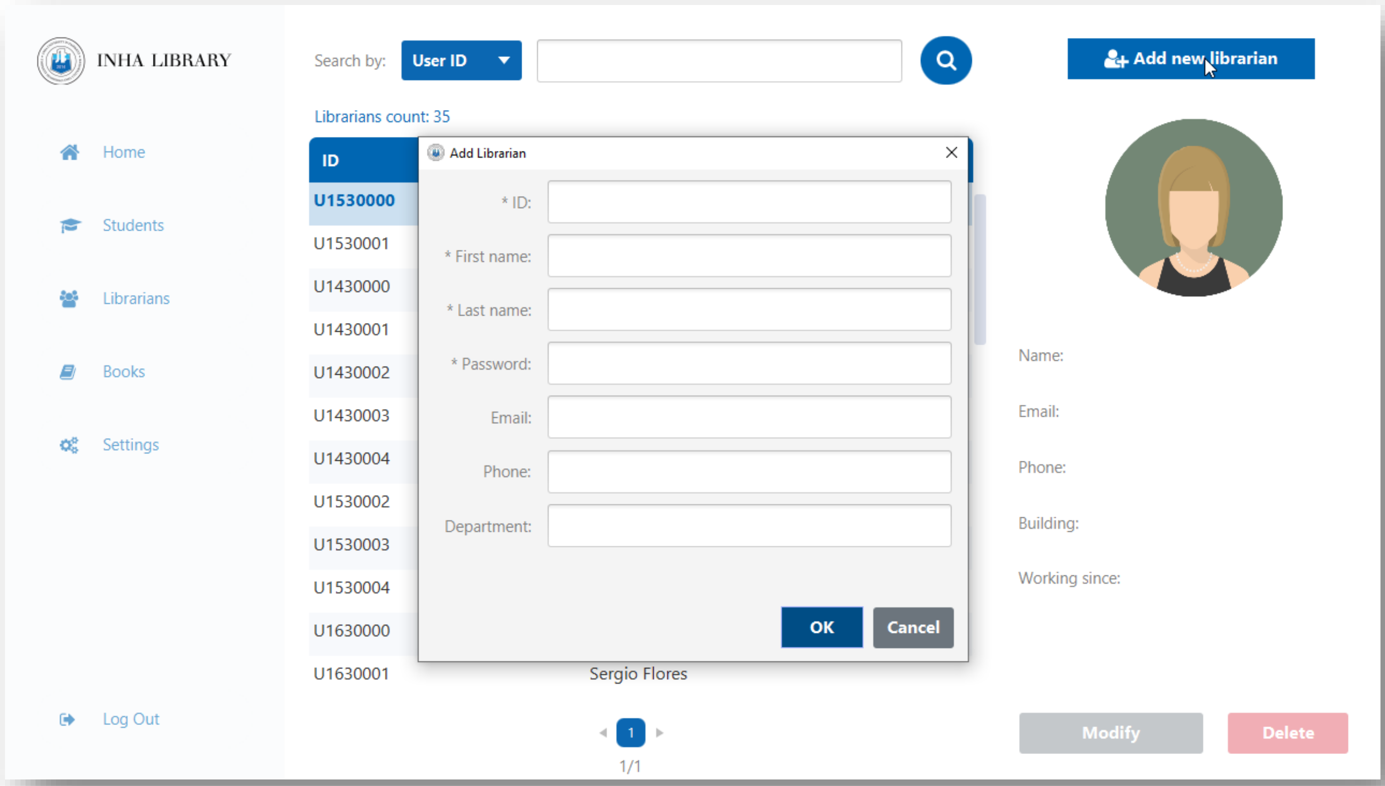


Librarians count: 35

ID	Name
U1530000	Bekzod Allaev
U1530001	Asadbek Khasanov
U1430000	Rhett Williams
U1430001	Ulises Wright
U1430002	Kellan Smith
U1430003	Jameson Moore
U1430004	Mohammed Lopez
U1530002	Nash Nguyen
U1530003	Jett Thomas
U1530004	Salvatore Torres
U1630000	Lorenzo Williams
U1630001	Sergio Flores

Name:	Rhett Williams
Email:	r.williams@librarian.inha.uz
Phone:	+998 94 000 34 22
Building:	A301
Working since:	2014

This is **Librarian** view of administrator window. It looks and behaves nearly same as student view of admin window.

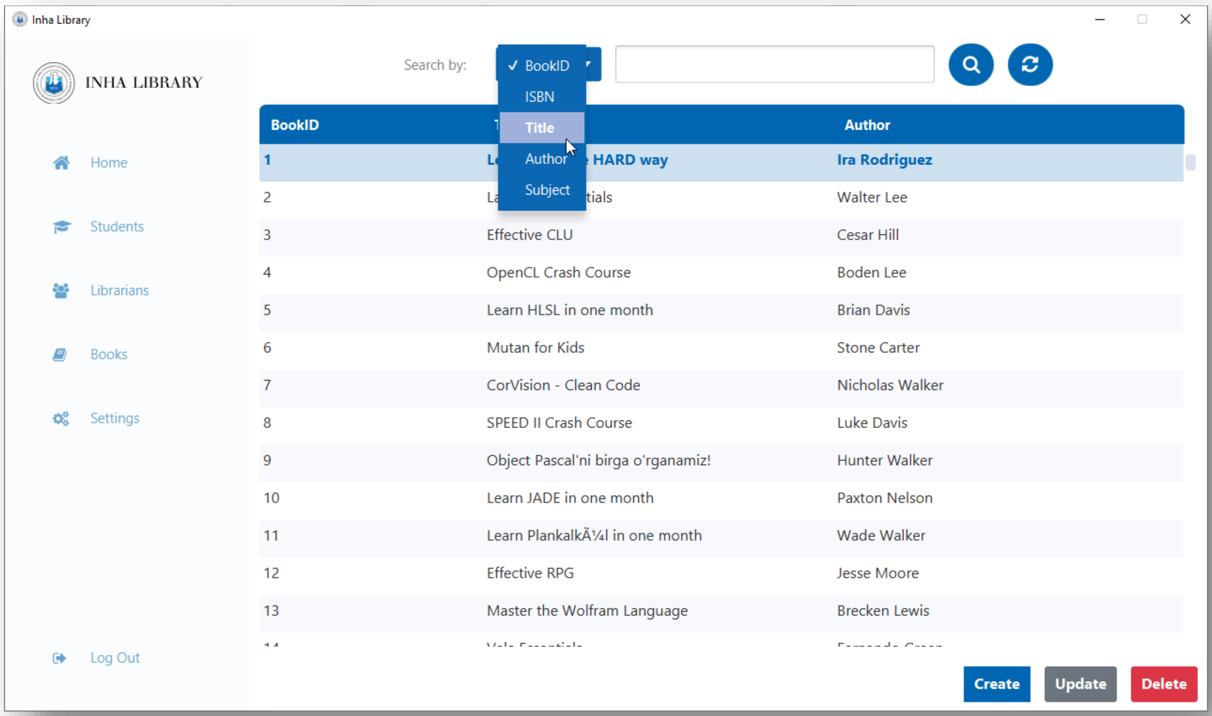


Librarians count: 35

ID	
U1530000	
U1530001	
U1430000	
U1430001	
U1430002	
U1430003	
U1430004	
U1530002	
U1530003	
U1530004	
U1630000	
U1630001	Sergio Flores

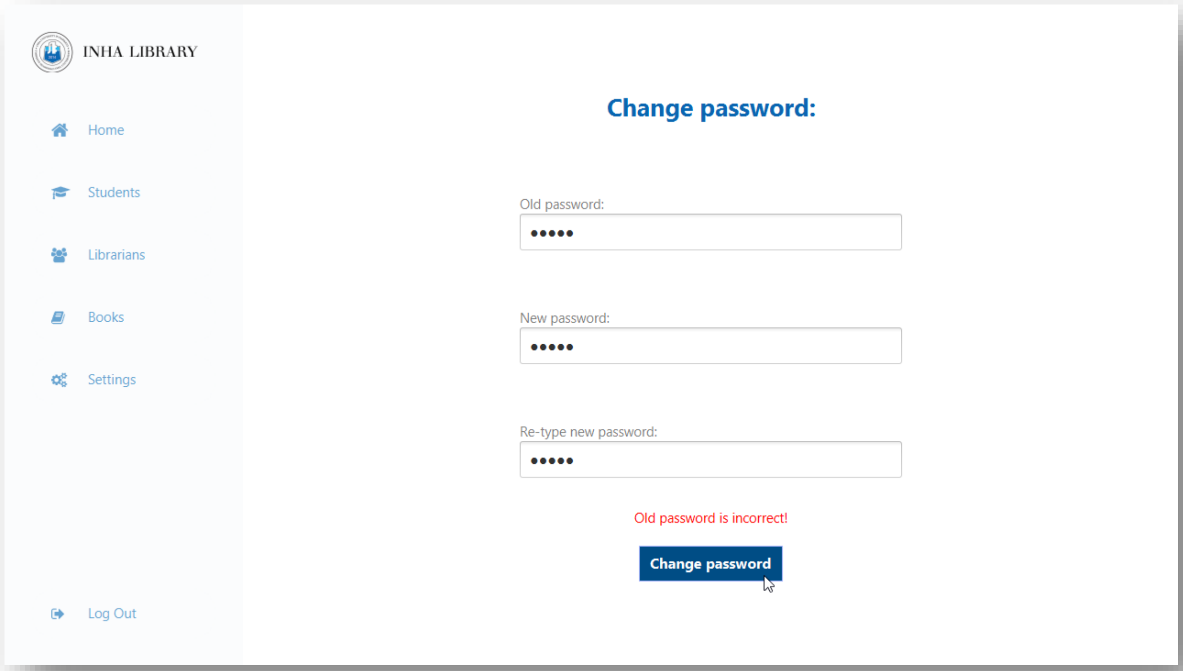
* ID:
* First name:
* Last name:
* Password:
Email:
Phone:
Department:

Name:
Email:
Phone:
Building:
Working since:



BookID	T	Author
1	L HARD way	Ira Rodriguez
2	La tials	Walter Lee
3	Effective CLU	Cesar Hill
4	OpenCL Crash Course	Boden Lee
5	Learn HLSL in one month	Brian Davis
6	Mutan for Kids	Stone Carter
7	CorVision - Clean Code	Nicholas Walker
8	SPEED II Crash Course	Luke Davis
9	Object Pascal'ni birga o'rganamiz!	Hunter Walker
10	Learn JADE in one month	Paxton Nelson
11	Learn PlankalkÃ¼l in one month	Wade Walker
12	Effective RPG	Jesse Moore
13	Master the Wolfram Language	Brecken Lewis

This is **Books** view of administrator window. In the center of the view there is a table which displays all books registered in the system. On the top, there is search bar, which behaves same as search bars in other mentioned



Change password:

Old password: •••••

New password: •••••

Re-type new password: •••••

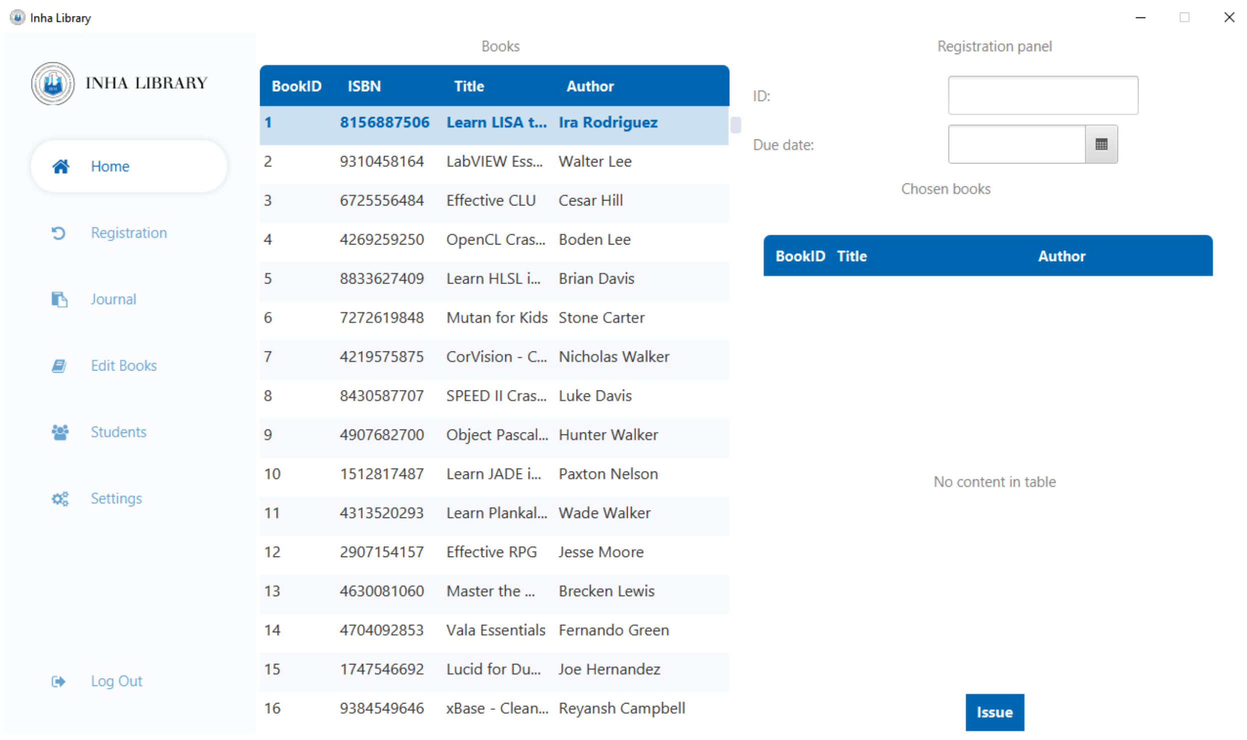
Old password is incorrect!

views (tabs). And, there are buttons on the bottom to manipulate selected books.

This is the Settings of Admin window. If admin wants to change his/her password, it can be done in this tab. There are 3 fields to be filled in order to change the password. All validation cases are implemented.

Librarian Interface

Registration view



BookID	ISBN	Title	Author
1	8156887506	Learn LISA t...	Ira Rodriguez
2	9310458164	LabVIEW Ess...	Walter Lee
3	6725556484	Effective CLU	Cesar Hill
4	4269259250	OpenCL Cras...	Boden Lee
5	8833627409	Learn HLSL i...	Brian Davis
6	7272619848	Mutan for Kids	Stone Carter
7	4219575875	CorVision - C...	Nicholas Walker
8	8430587707	SPEED II Cras...	Luke Davis
9	4907682700	Object Pascal...	Hunter Walker
10	1512817487	Learn JADE i...	Paxton Nelson
11	4313520293	Learn Plankal...	Wade Walker
12	2907154157	Effective RPG	Jesse Moore
13	4630081060	Master the ...	Brecken Lewis
14	4704092853	Vala Essentials	Fernando Green
15	1747546692	Lucid for Du...	Joe Hernandez
16	9384549646	xBase - Clean...	Reyansh Campbell

BookID	Title	Author

Semantic

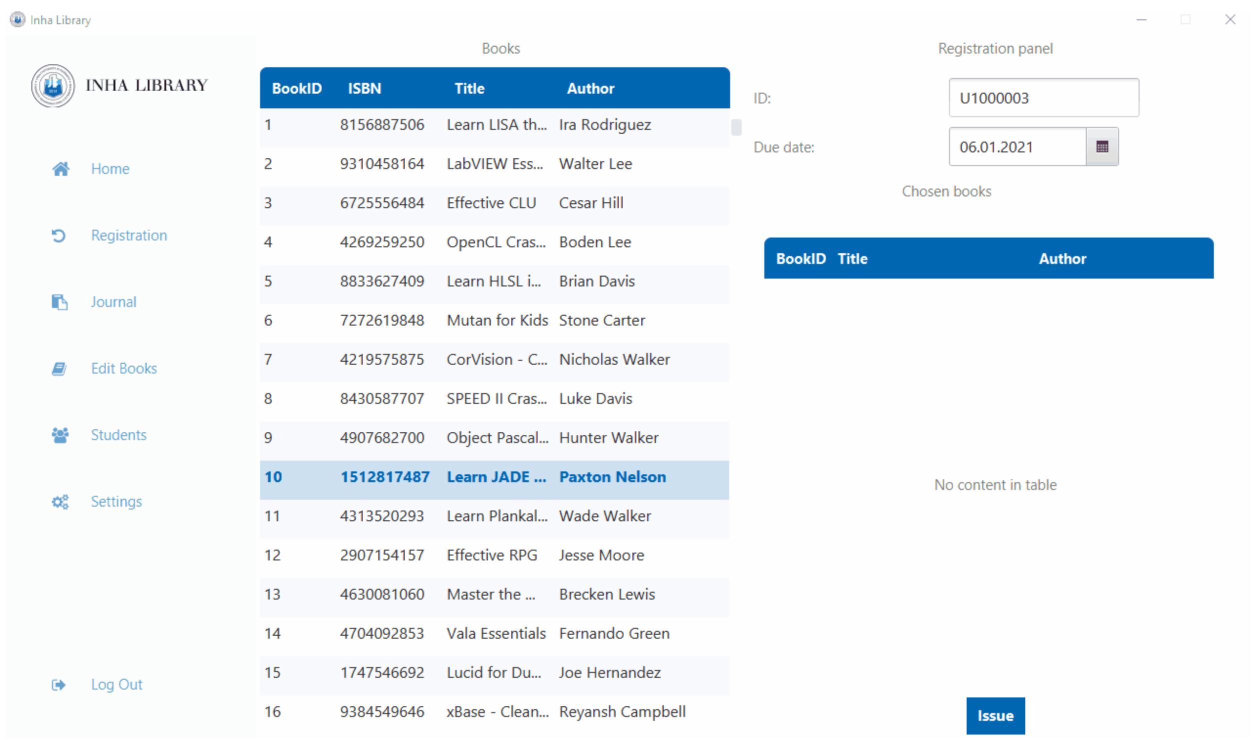
This view is used by librarian in order to give book(s) to student.

UI

You can see two tables. Left table shows list of all books in library and right one shows books that student want to take. Up to the chosen books you can see date picker this controls is used to set due date (date when student must return book(s)) and ID field where librarian enters student's ID.

UX

First of all, you set user's ID then when student must return books. In order to add books, you just press on book from the left table and then this book moves to right one (thus book is chosen). If student want to remove a book from chosen books librarian just press on this book and this book is returned to the library. At the end of the process, you librarian press Issue button and all fields become empty.



Books

BookID	ISBN	Title	Author
1	8156887506	Learn LISA th...	Ira Rodriguez
2	9310458164	LabVIEW Ess...	Walter Lee
3	6725556484	Effective CLU	Cesar Hill
4	4269259250	OpenCL Cras...	Boden Lee
5	8833627409	Learn HLSL i...	Brian Davis
6	7272619848	Mutan for Kids	Stone Carter
7	4219575875	CorVision - C...	Nicholas Walker
8	8430587707	SPEED II Cras...	Luke Davis
9	4907682700	Object Pascal...	Hunter Walker
10	**1512817487**	**Learn JADE ...**	**Paxton Nelson**
11	4313520293	Learn Plankal...	Wade Walker
12	2907154157	Effective RPG	Jesse Moore
13	4630081060	Master the ...	Brecken Lewis
14	4704092853	Vala Essentials	Fernando Green
15	1747546692	Lucid for Du...	Joe Hernandez
16	9384549646	xBase - Clean...	Reyansh Campbell

Registration panel

ID: U1000003

Due date: 06.01.2021

Chosen books

BookID	Title	Author

No content in table

Issue

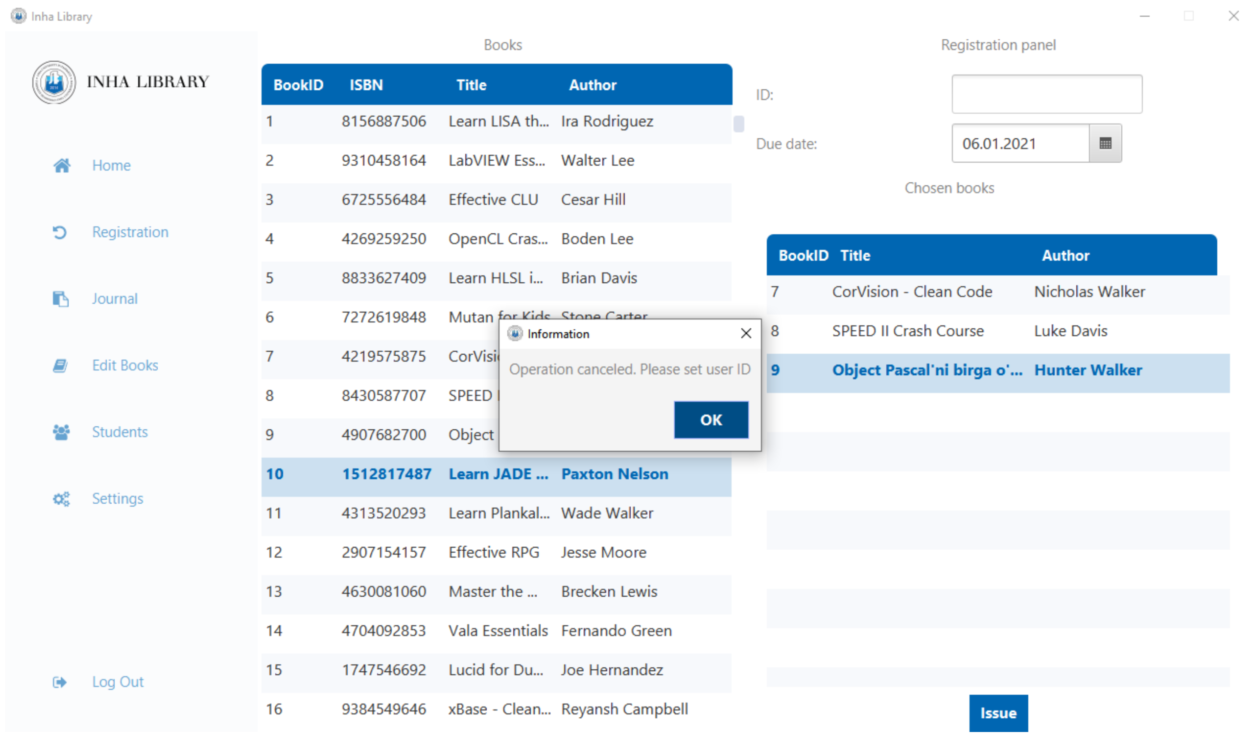
Validation

Validation on this view is:

- ID and Due date fields must be filled, there must be at least one chosen book;

- ID must be valid student's ID;

- Student cannot choose book that he/she chose and didn't return;



Books

BookID	ISBN	Title	Author
1	8156887506	Learn LISA th...	Ira Rodriguez
2	9310458164	LabVIEW Ess...	Walter Lee
3	6725556484	Effective CLU	Cesar Hill
4	4269259250	OpenCL Cras...	Boden Lee
5	8833627409	Learn HLSL i...	Brian Davis
6	7272619848	Mutan for Kids	Stone Carter
7	4219575875	CorVisi	
8	8430587707	SPEED	
9	4907682700	Object	
10	**1512817487**	**Learn JADE ...**	**Paxton Nelson**
11	4313520293	Learn Plankal...	Wade Walker
12	2907154157	Effective RPG	Jesse Moore
13	4630081060	Master the ...	Brecken Lewis
14	4704092853	Vala Essentials	Fernando Green
15	1747546692	Lucid for Du...	Joe Hernandez
16	9384549646	xBase - Clean...	Reyansh Campbell

Information

Operation canceled. Please set user ID

OK

Registration panel

ID:

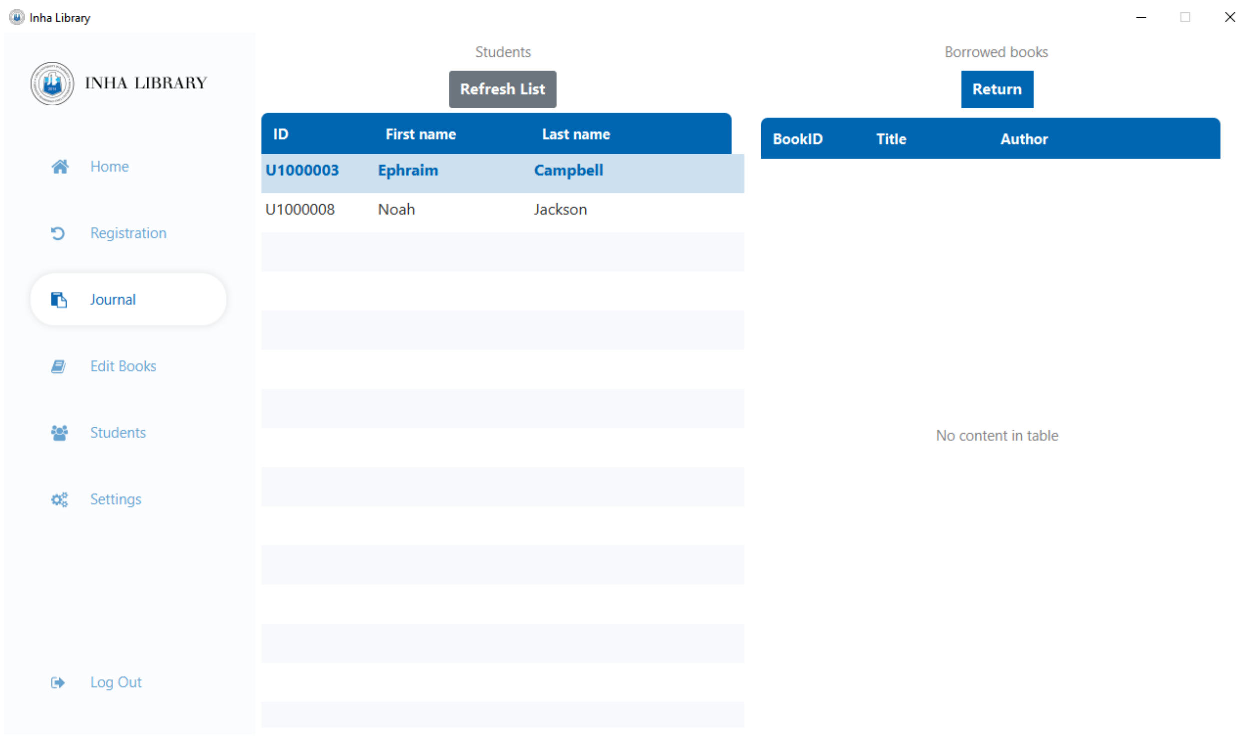
Due date: 06.01.2021

Chosen books

BookID	Title	Author
7	CorVision - Clean Code	Nicholas Walker
8	SPEED II Crash Course	Luke Davis
9	**Object Pascal'ni birga o'...**	**Hunter Walker**

Issue

Journal view



ID	First name	Last name
U1000003	Ephraim	Campbell
U1000008	Noah	Jackson

BookID	Title	Author

Semantic

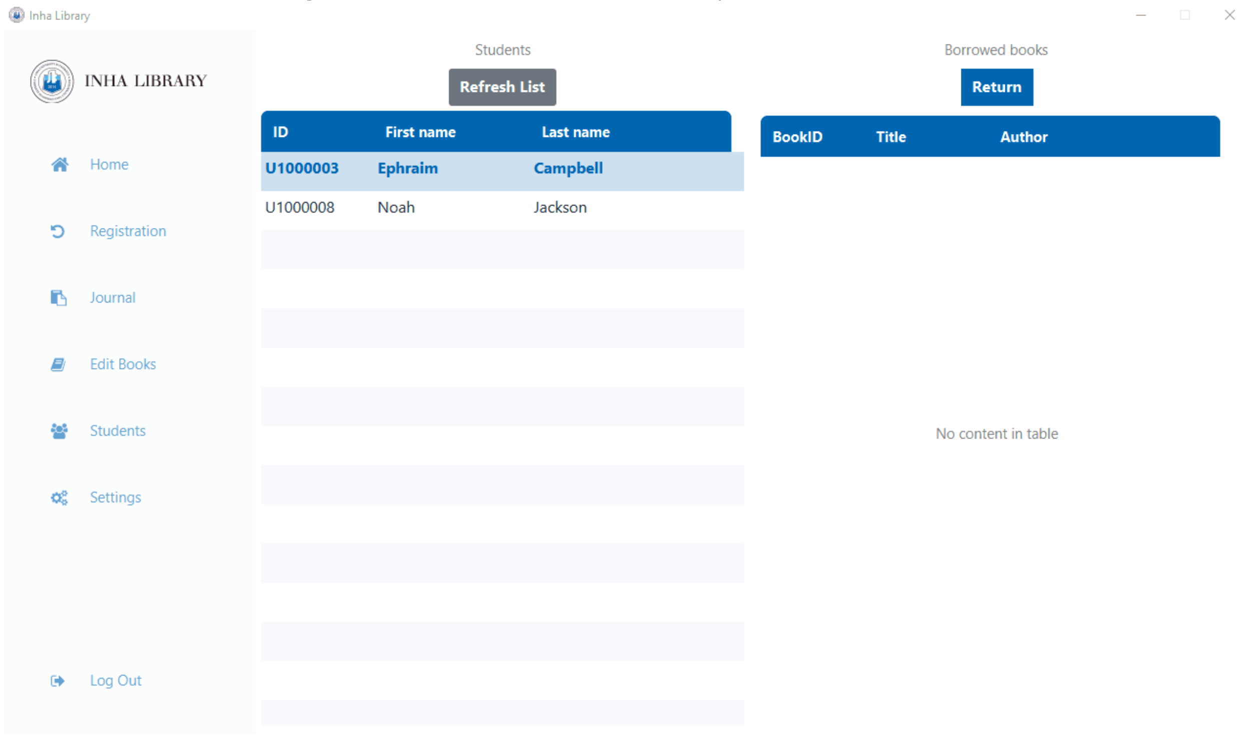
Librarian use this view in order to see students who took book from library and which book(s) they have taken. Also, they can return issued books back to the library through this view.

UI

In the left table there is list of students who took book from library and on the right, there is table of books that they have taken. Refresh button is used to refresh the list of students who took the book. Return button is used to return book to the library back. In case if student returns book later then return date, he/she receives fine.

UX

If librarian wants to see books taken by particular student, he/she just press on this student in the left table and then in the right table list of chosen books is presented.

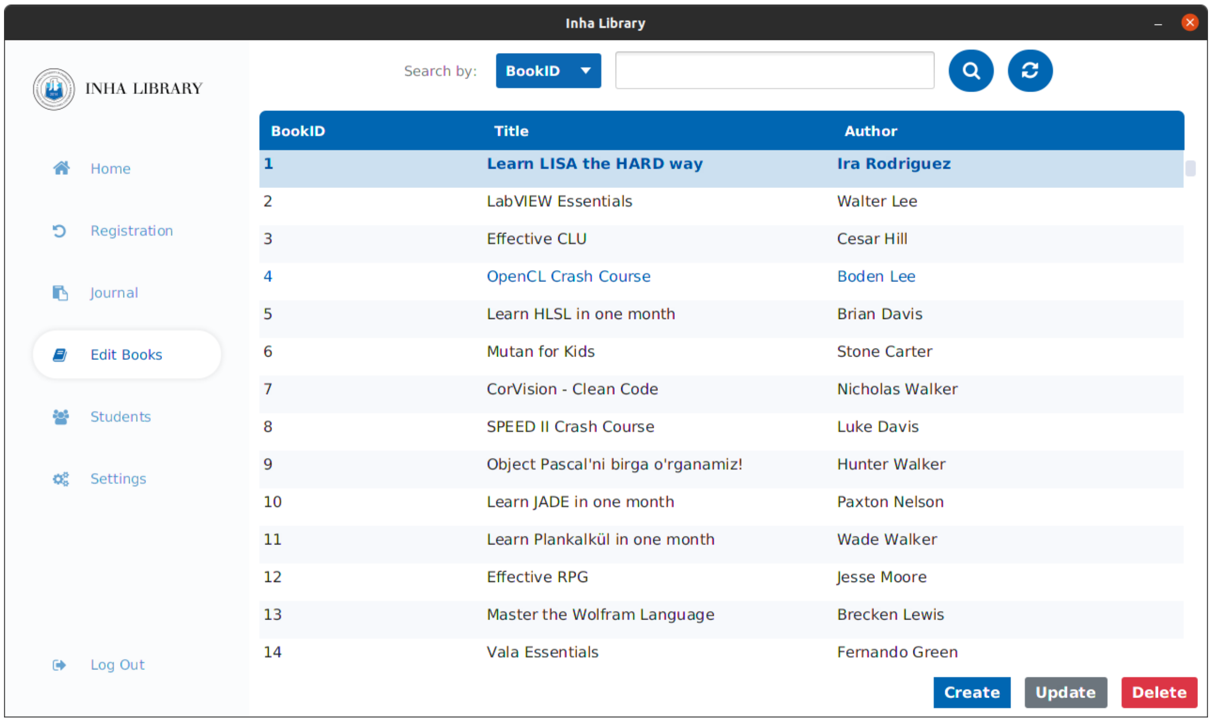


ID	First name	Last name
U1000003	Ephraim	Campbell
U1000008	Noah	Jackson

BookID	Title	Author

No content in table

When pressed Edit Books button on the navigation bar, the window changes to the following look:

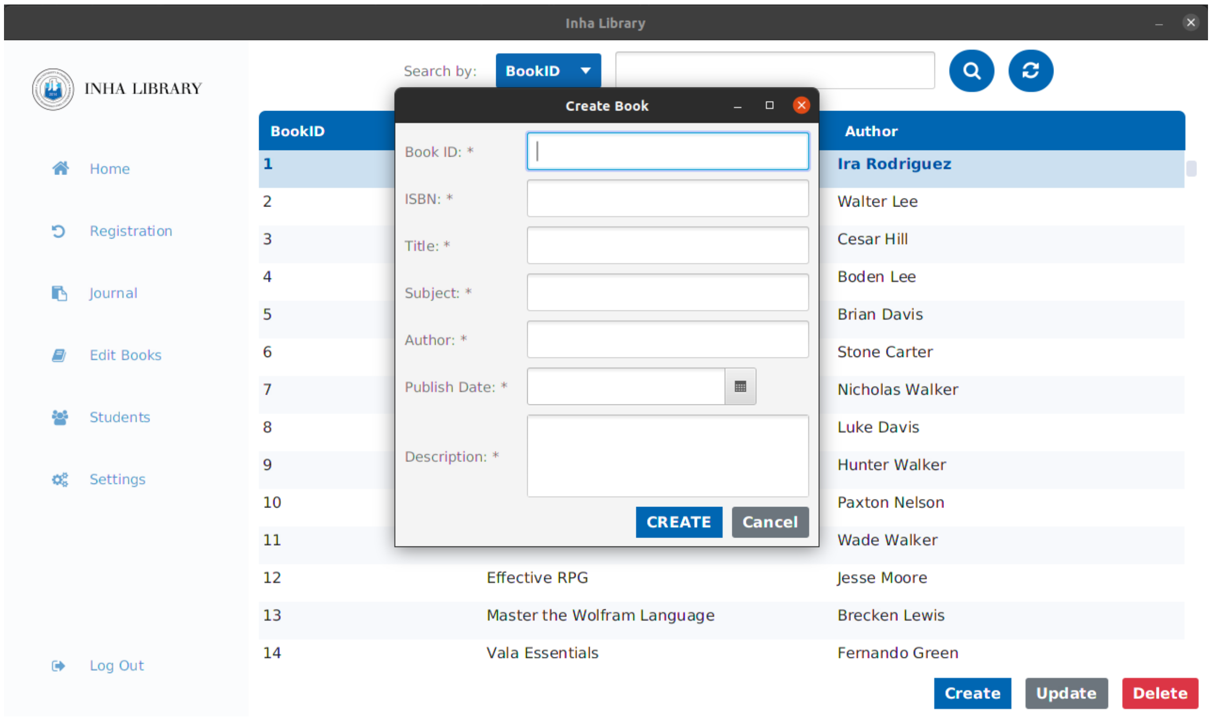


BookID	Title	Author
1	Learn LISA the HARD way	Ira Rodriguez
2	LabVIEW Essentials	Walter Lee
3	Effective CLU	Cesar Hill
4	OpenCL Crash Course	Boden Lee
5	Learn HLSL in one month	Brian Davis
6	Mutan for Kids	Stone Carter
7	CorVision - Clean Code	Nicholas Walker
8	SPEED II Crash Course	Luke Davis
9	Object Pascal'ni birga o'rganamiz!	Hunter Walker
10	Learn JADE in one month	Paxton Nelson
11	Learn Plankalkül in one month	Wade Walker
12	Effective RPG	Jesse Moore
13	Master the Wolfram Language	Brecken Lewis
14	Vala Essentials	Fernando Green

There is this table that shows the list of all books in the library. Users can use the elements on the top of the table to search for a specific book. There is also Refresh button that can be used to refresh the table or to see the list of all books after some searches were made.

On the bottom, there are three buttons: Create, Update and Delete that can be used to create, update and delete books, respectively.

When pressed on Create button, this dialog window pops up:

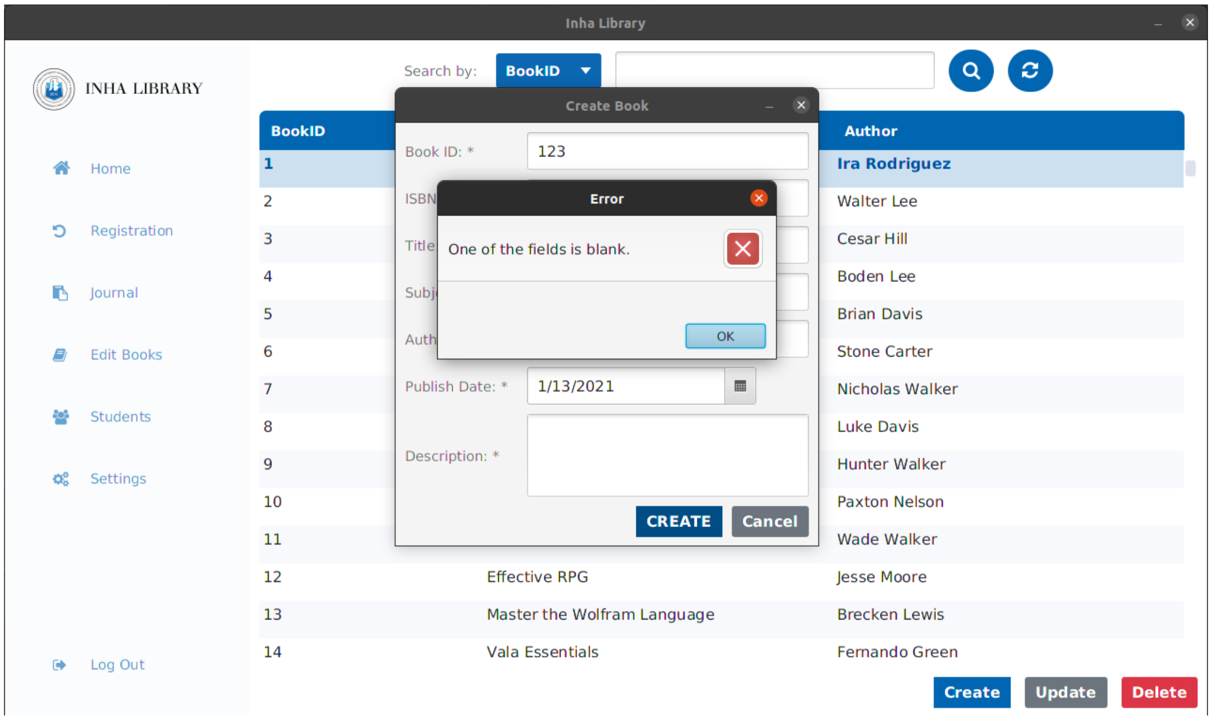


Book ID: *	
ISBN: *	
Title: *	
Subject: *	
Author: *	
Publish Date: *	
Description: *	

BookID	Title	Author
1		Ira Rodriguez
2		Walter Lee
3		Cesar Hill
4		Boden Lee
5		Brian Davis
6		Stone Carter
7		Nicholas Walker
8		Luke Davis
9		Hunter Walker
10		Paxton Nelson
11		Wade Walker
12	Effective RPG	Jesse Moore
13	Master the Wolfram Language	Brecken Lewis
14	Vala Essentials	Fernando Green

Little stars in front of labels mean that field must be filled in order to successfully create book.

If above condition is not satisfied, the application notifies the user that he/she left at least one of the must-fill fields blank:



BookID		Author
1		Ira Rodriguez
2		Walter Lee
3		Cesar Hill
4		Boden Lee
5		Brian Davis
6		Stone Carter
7		Nicholas Walker
8		Luke Davis
9		Hunter Walker
10		Paxton Nelson
11		Wade Walker
12	Effective RPG	Jesse Moore
13	Master the Wolfram Language	Brecken Lewis
14	Vala Essentials	Fernando Green

Book ID: * 123

Publish Date: * 1/13/2021

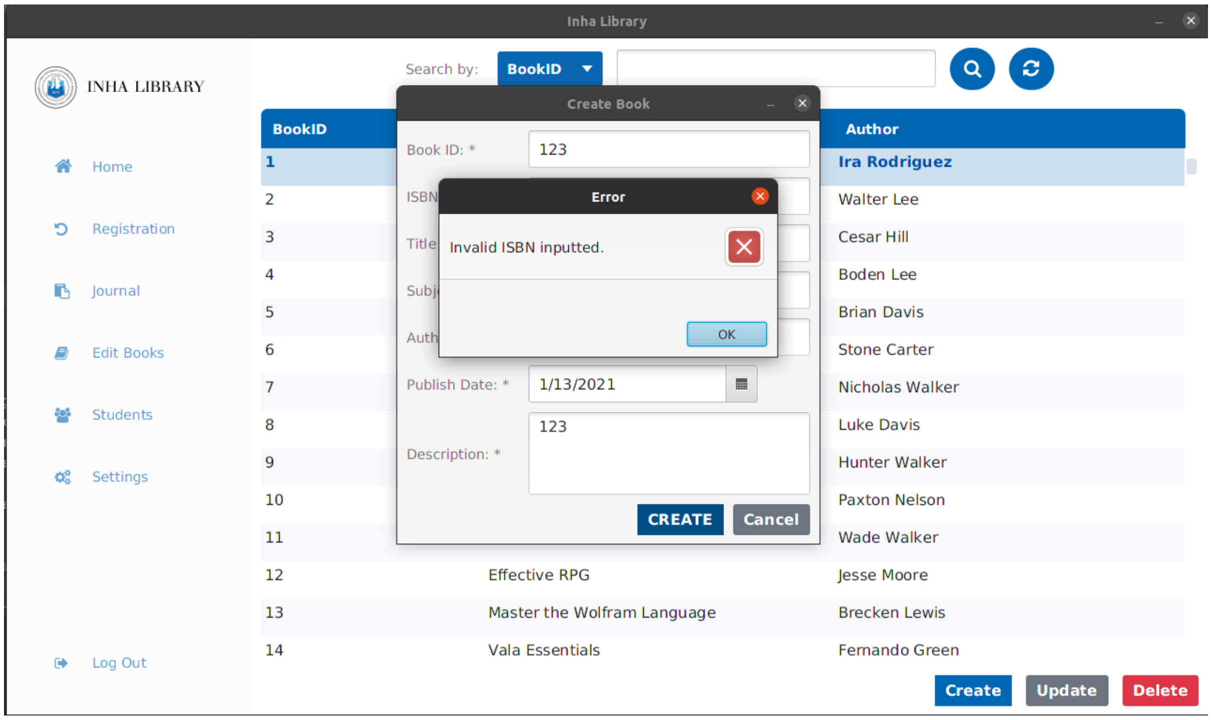
Description: *

Error

One of the fields is blank.

OK

There is also ISBN validation that checks if the inputted ISBN is valid. If not, the following alert window pops up, informing the user:



BookID		Author
1		Ira Rodriguez
2		Walter Lee
3		Cesar Hill
4		Boden Lee
5		Brian Davis
6		Stone Carter
7		Nicholas Walker
8		Luke Davis
9		Hunter Walker
10		Paxton Nelson
11		Wade Walker
12	Effective RPG	Jesse Moore
13	Master the Wolfram Language	Brecken Lewis
14	Vala Essentials	Fernando Green

Book ID: * 123

Publish Date: * 1/13/2021

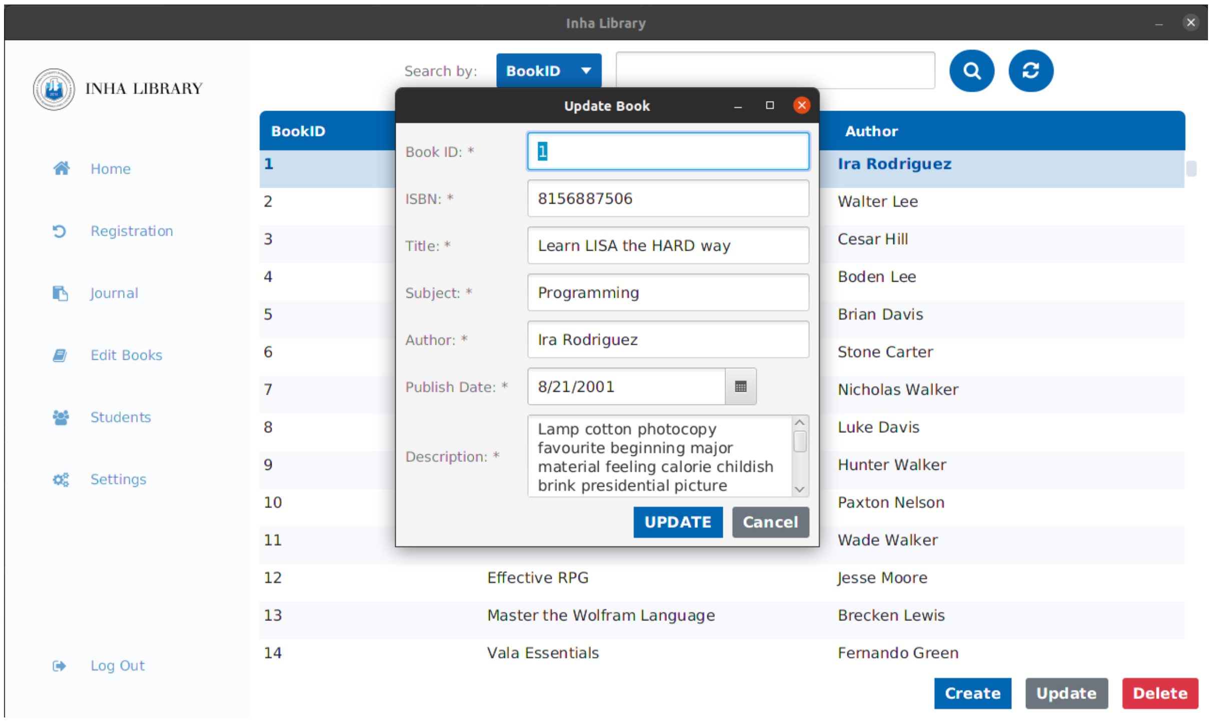
Description: * 123

Error

Invalid ISBN inputted.

OK

When users want to update some book, that specific book should be selected on the table first. By default, the first one is chosen. When everything is all right, this dialog window should pop up, already filled with book specifications.


Inha Library

INHA LIBRARY

Home
Registration
Journal
Edit Books
Students
Settings
Log Out

Search by: BookID

Update Book

Book ID: *	1
ISBN: *	8156887506
Title: *	Learn LISA the HARD way
Subject: *	Programming
Author: *	Ira Rodriguez
Publish Date: *	8/21/2001
Description: *	Lamp cotton photocopy favourite beginning major material feeling calorie childish brink presidential picture

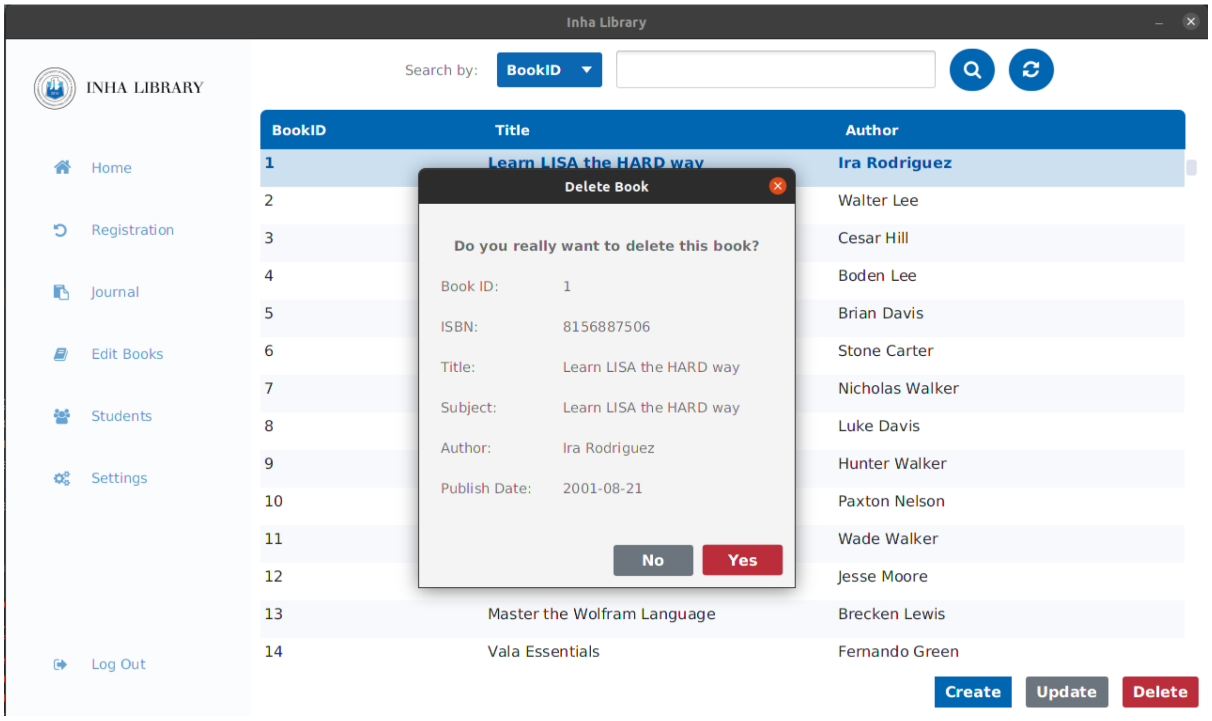
UPDATE Cancel

BookID		Author
1		Ira Rodriguez
2		Walter Lee
3		Cesar Hill
4		Boden Lee
5		Brian Davis
6		Stone Carter
7		Nicholas Walker
8		Luke Davis
9		Hunter Walker
10		Paxton Nelson
11		Wade Walker
12	Effective RPG	Jesse Moore
13	Master the Wolfram Language	Brecken Lewis
14	Vala Essentials	Fernando Green

Create Update Delete

Just like the Update feature, on Deletion, the book should be selected first. Again, by default, the first one on the table is chosen.

When everything is all right, users must see the fallowing dialog window asking for confirmation:


Inha Library

INHA LIBRARY

Home
Registration
Journal
Edit Books
Students
Settings
Log Out

Search by: BookID

Delete Book

Do you really want to delete this book?

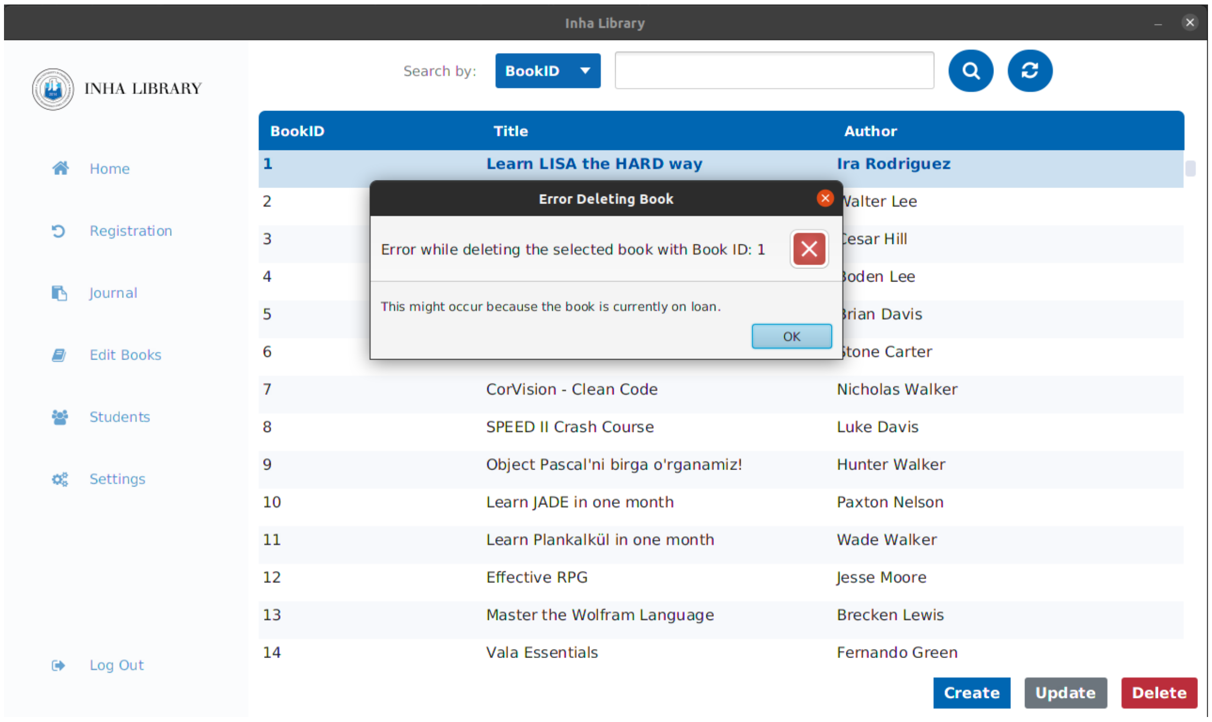
Book ID:	1
ISBN:	8156887506
Title:	Learn LISA the HARD way
Subject:	Learn LISA the HARD way
Author:	Ira Rodriguez
Publish Date:	2001-08-21

No Yes

BookID	Title	Author
1	Learn LISA the HARD way	Ira Rodriguez
2		Walter Lee
3		Cesar Hill
4		Boden Lee
5		Brian Davis
6		Stone Carter
7		Nicholas Walker
8		Luke Davis
9		Hunter Walker
10		Paxton Nelson
11		Wade Walker
12		Jesse Moore
13	Master the Wolfram Language	Brecken Lewis
14	Vala Essentials	Fernando Green

Create Update Delete

One more thing to mention: it is not possible to delete books from the database that are currently on loan. This is because there is no way of deleting book from one table when it is referenced in another. When tried to do so, alert window shows up and informs the user of the condition:



BookID	Title	Author
1	Learn LISA the HARD way	Ira Rodriguez
2		Walter Lee
3		Cesar Hill
4		Boden Lee
5		Brian Davis
6		Stone Carter
7	CorVision - Clean Code	Nicholas Walker
8	SPEED II Crash Course	Luke Davis
9	Object Pascal'ni birga o'rganamiz!	Hunter Walker
10	Learn JADE in one month	Paxton Nelson
11	Learn Plankalkül in one month	Wade Walker
12	Effective RPG	Jesse Moore
13	Master the Wolfram Language	Brecken Lewis
14	Vala Essentials	Fernando Green

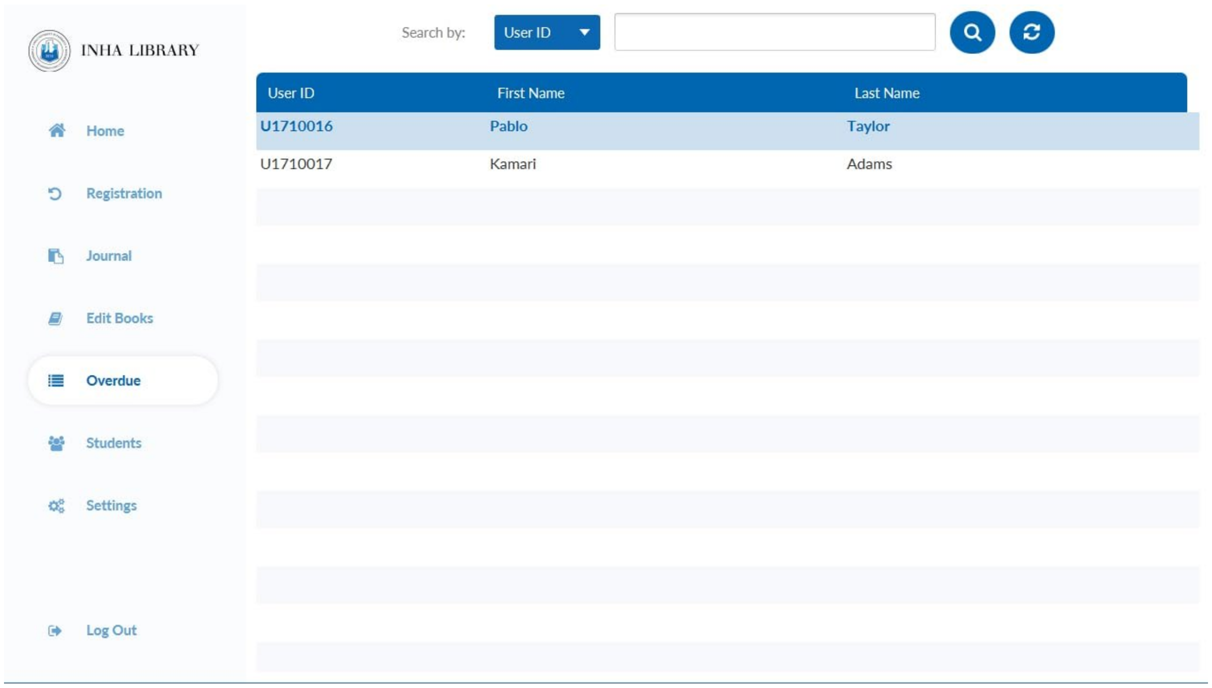
Error Deleting Book

Error while deleting the selected book with Book ID: 1

This might occur because the book is currently on loan.

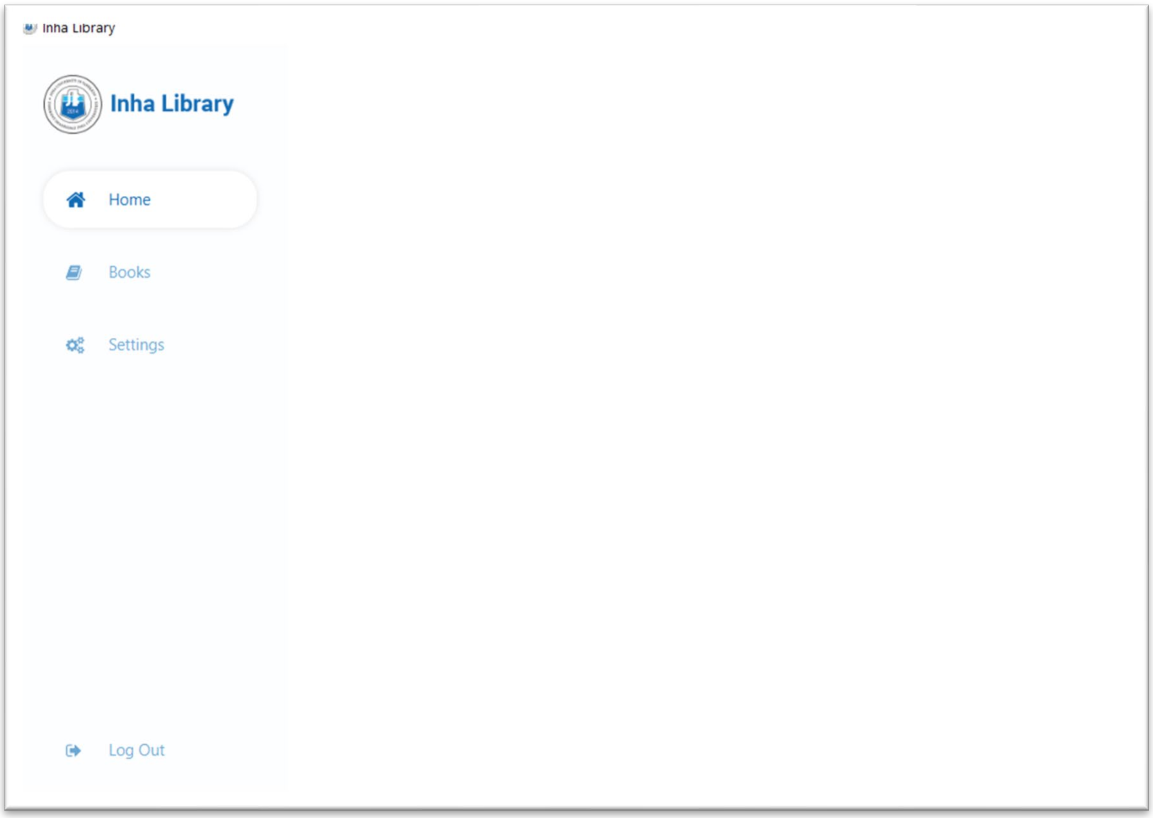
OK

In this view, one can see the list of students with overdue. It also supports searching & refreshing the table. One challenging task was to pull data from the database for this table. Because, it would require to filter through two DB tables: IssuedBooks and Users… Still, the way the query is written is inefficient.



User ID	First Name	Last Name
U1710016	Pablo	Taylor
U1710017	Kamari	Adams

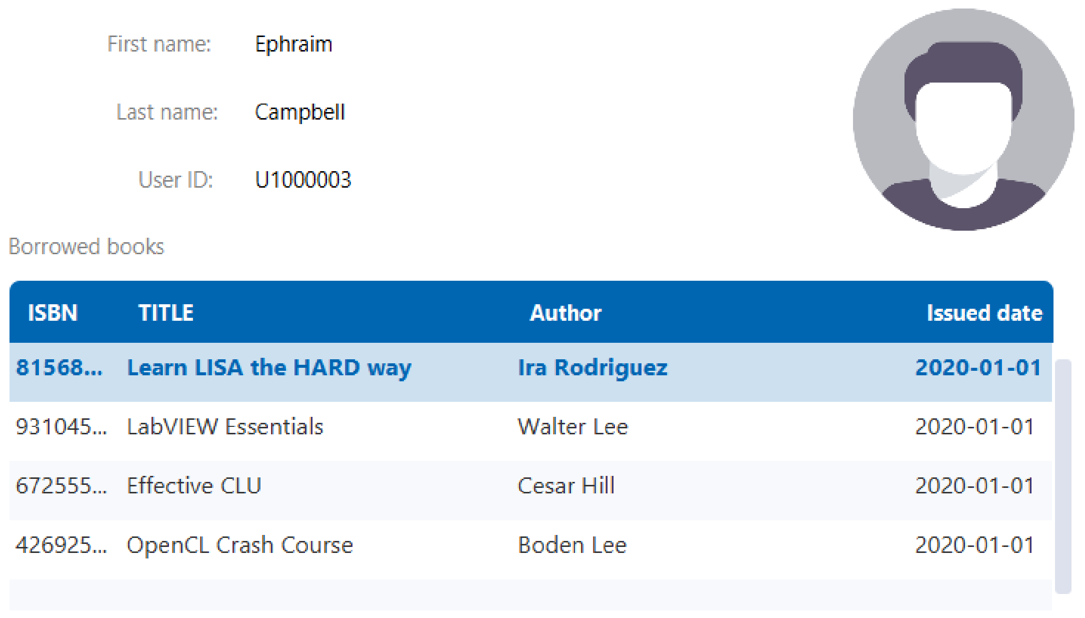
Student Interface



After Logging In student will have options to view Personal information, view Books list, change Settings of his window and Log Out

Functionality

Requirement 1. First requirement is to show student his/her personal information as well as books that the student has borrowed.



First name:	Ephraim
Last name:	Campbell
User ID:	U1000003

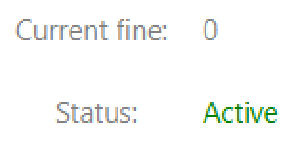
Borrowed books

ISBN	TITLE	Author	Issued date
81568...	Learn LISA the HARD way	Ira Rodriguez	2020-01-01
931045...	LabVIEW Essentials	Walter Lee	2020-01-01
672555...	Effective CLU	Cesar Hill	2020-01-01
426925...	OpenCL Crash Course	Boden Lee	2020-01-01

By searching student's ID in Data Base, we get and set his/her name and surname, as well as ID number to Window. To fill "Borrowed books" table we collect books' ID that was borrowed by this specific student ("Select * FROM IssuedBooks Where BookID =?"). Following that we take ISBN, Title and Author name by book ID from books_tbl.sql Data Base and include them to table issuedBooksRows.add(**new** IssuedBookRow(result.getString("ISBN"), result.getString("Title"), result.getString("Author"), resultForID.getDate("IssuedDate")));

Repository usage: For having list of borrowed books with four mentioned properties, our team have created method in IssuedBooksRepository which returns ObservableList<IssuedBook>.

Requirement 2. Second requirement is to show student his/her current fine and status (whether it is Banned or Active).



Current fine:	0
Status:	Active

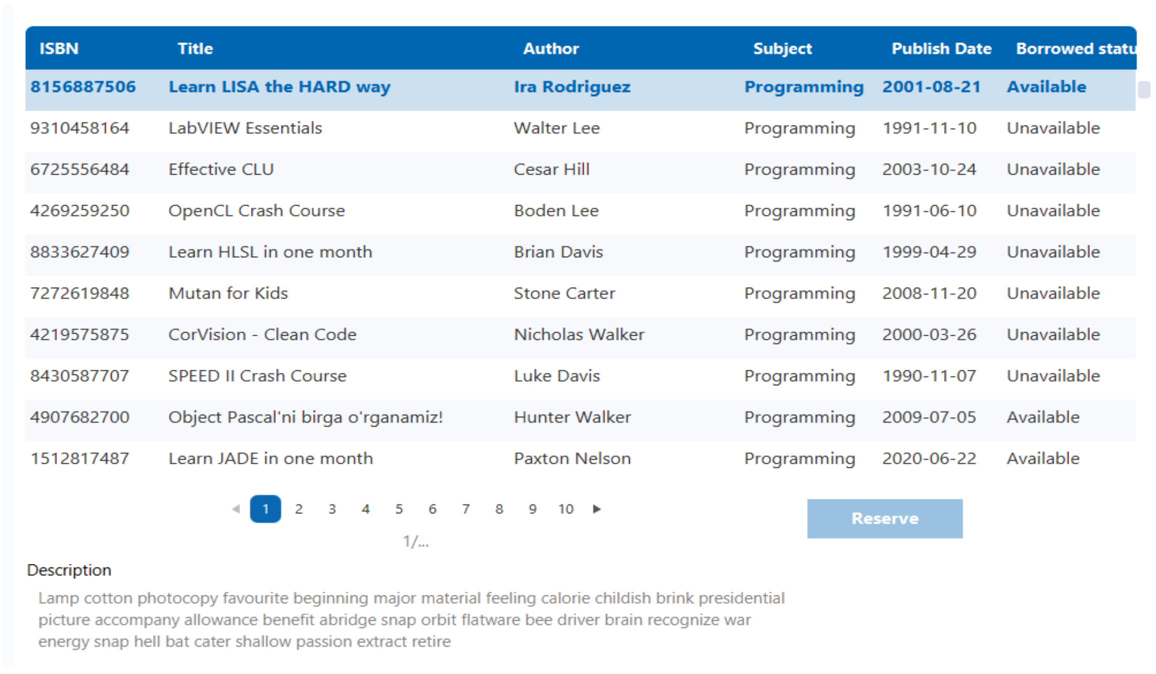
To implement this function, we had to create new Data Base called students_tbl.sql, which contains three columns called UserID, Fine, Status; consequently, each student will have own Fine and Status.

Fine will be charged by librarian, if student will not return borrowed book until return date. On top of this, status of student will change to "Banned" when having a fine and will remain still until fine will be paid.

Table with fine and status:

```
CREATE TABLE Students (
   UserID VARCHAR(8) NOT NULL REFERENCES Users(UserID),
   Fine INT NOT NULL DEFAULT 0,  Status INT NOT NULL DEFAULT 0,
   PRIMARY KEY (UserID)
);
```

Requirement 3. Third requirement is to show all books and filter them by borrowed status, title, subject, author, ISBN and publish date.



ISBN	Title	Author	Subject	Publish Date	Borrowed statu
8156887506	Learn LISA the HARD way	Ira Rodriguez	Programming	2001-08-21	Available
9310458164	LabVIEW Essentials	Walter Lee	Programming	1991-11-10	Unavailable
6725556484	Effective CLU	Cesar Hill	Programming	2003-10-24	Unavailable
4269259250	OpenCL Crash Course	Boden Lee	Programming	1991-06-10	Unavailable
8833627409	Learn HLSL in one month	Brian Davis	Programming	1999-04-29	Unavailable
7272619848	Mutan for Kids	Stone Carter	Programming	2008-11-20	Unavailable
4219575875	CorVision - Clean Code	Nicholas Walker	Programming	2000-03-26	Unavailable
8430587707	SPEED II Crash Course	Luke Davis	Programming	1990-11-07	Unavailable
4907682700	Object Pascal'ni birga o'rganamiz!	Hunter Walker	Programming	2009-07-05	Available
1512817487	Learn JADE in one month	Paxton Nelson	Programming	2020-06-22	Available

Description

Lamp cotton photocopy favourite beginning major material feeling calorie childish brink presidential picture accompany allowance benefit abridge snap orbit flatware bee driver brain recognize war energy snap hell bat cater shallow passion extract retire

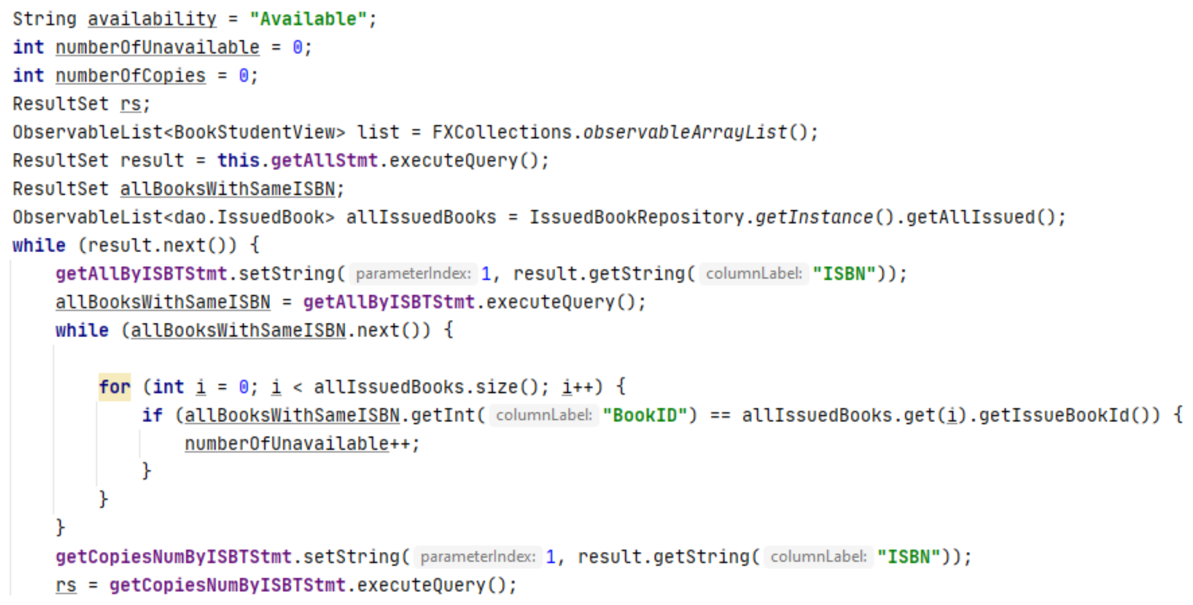
Repository usage: To fill table with six above shown properties, we created new books repository and add method called getBookWithBorrowedSt(). The mentioned method returns ObservableList<BookStudentView>, which then is set to table. BookStudentView is a model for book with properties that student should see.

Filter: By clicking on different column names student can filter them by ascending and descending order. Also, we included pagination control under the table for switching pages.

Table controllers: When user clicks on a book, description of the book will appear in Description box below the table. There is a method in Books repository, which returns description by ISBN of book:

```
labelDescription.setText(BooksRepository.getInstance().getDescriptionByISBN(ISBN));
```

Availability: Availability of book is identified by method below, which counts books copies in Books_tbl.sql and saves books ID in list. Since IssuedBooks_tbl.sql does not contain ISBN column, we had to check whether each copy of book is issued to check unavailability.

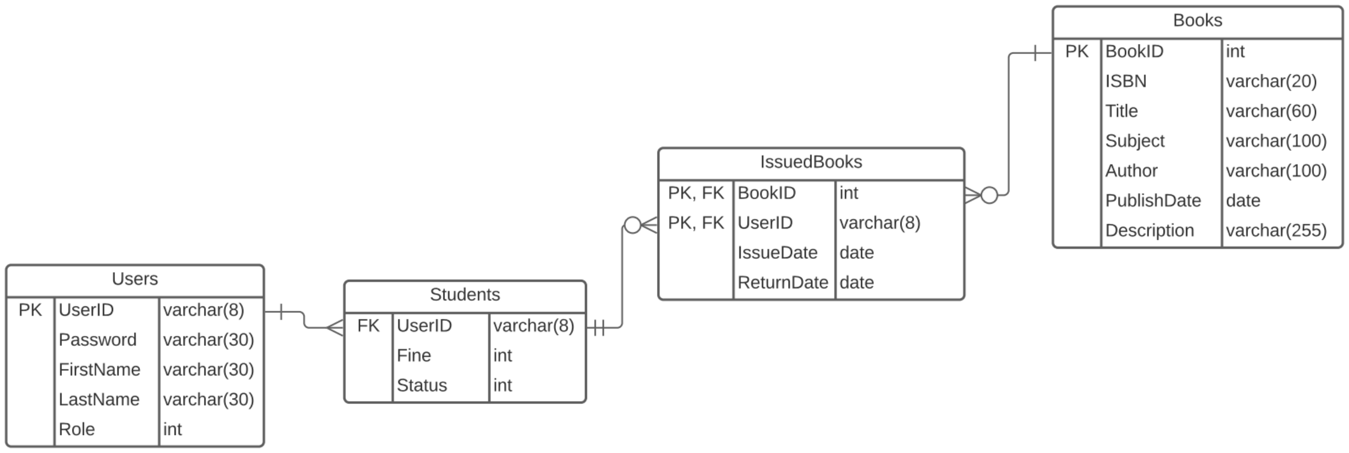


```
String availability = "Available";
int numberOfUnavailable = 0;
int numberOfCopies = 0;
ResultSet rs;
ObservableList<BookStudentView> list = FXCollections.observableArrayList();
ResultSet result = this.getAllStmt.executeQuery();
ResultSet allBooksWithSameISBN;
ObservableList<dao.IssuedBook> allIssuedBooks = IssuedBookRepository.getInstance().getAllIssued();
while (result.next()) {
    getAllByISBTStmt.setString( parameterIndex: 1, result.getString( columnLabel: "ISBN"));
    allBooksWithSameISBN = getAllByISBTStmt.executeQuery();
    while (allBooksWithSameISBN.next()) {

        for (int i = 0; i < allIssuedBooks.size(); i++) {
            if (allBooksWithSameISBN.getInt( columnLabel: "BookID") == allIssuedBooks.get(i).getIssueBookId()) {
                numberOfUnavailable++;
            }
        }
    }
    getCopiesNumByISBTStmt.setString( parameterIndex: 1, result.getString( columnLabel: "ISBN"));
    rs = getCopiesNumByISBTStmt.executeQuery();
```

Requirement 4. When user clicks on an unavailable book Reserve button will be available to use.
btnReserve.setDisable(false);
This button is used to reserve book ahead. (the functionality of Reserve button is still in progress)

Entity Relationship Diagram

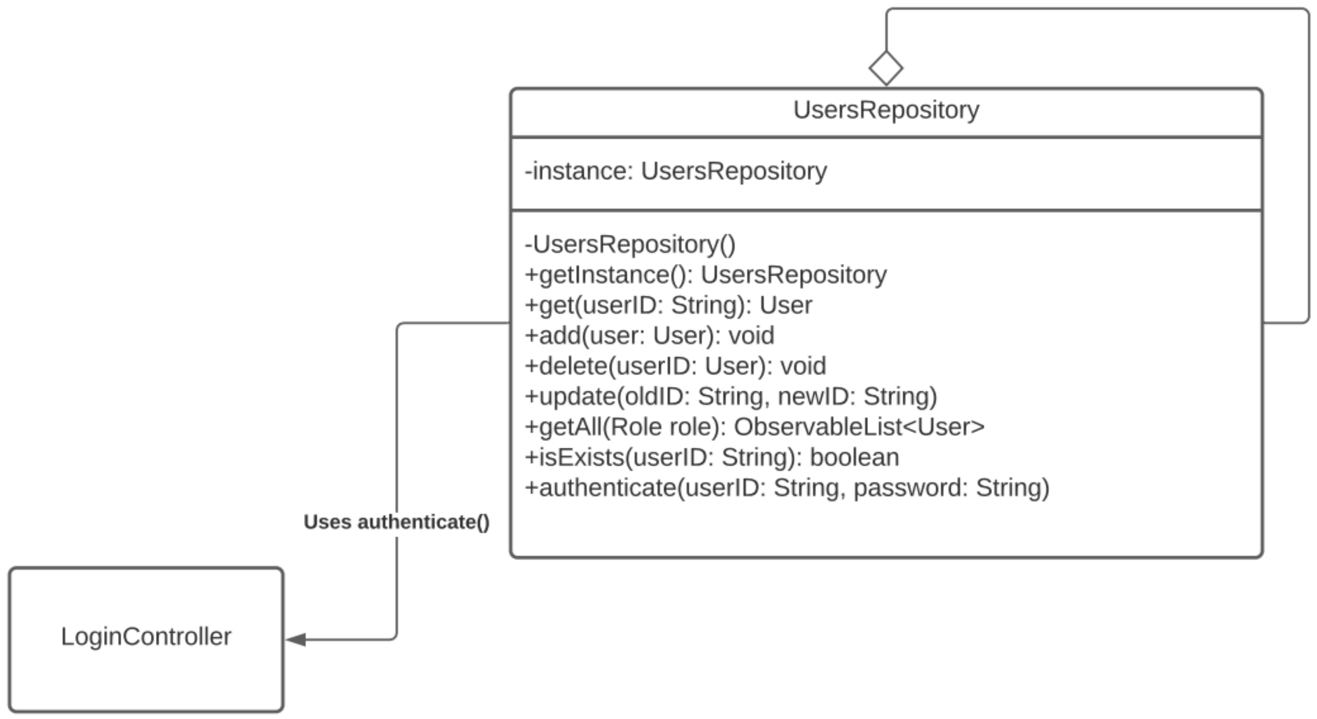


To avoid unnecessary tables, we included all users in Users table. To distinguish between their roles, we included Role column, where 0 corresponds to Admin, 1 to Librarian, and 2 to Student.

We looked up in our local Inha Library how books are managed. Each books is given a specific library-number which is independent of ISBN, so we decided to make BookID which is defined by library itself as a primary key in our Books table. After that, we made IssuedBooks table, which has composite primary key made of BookID and UserID. To avoid unnecessary traversal in Users, we made intermediate Students table, which has UserID as a foreign key from Users.

UML Diagrams

Data Access

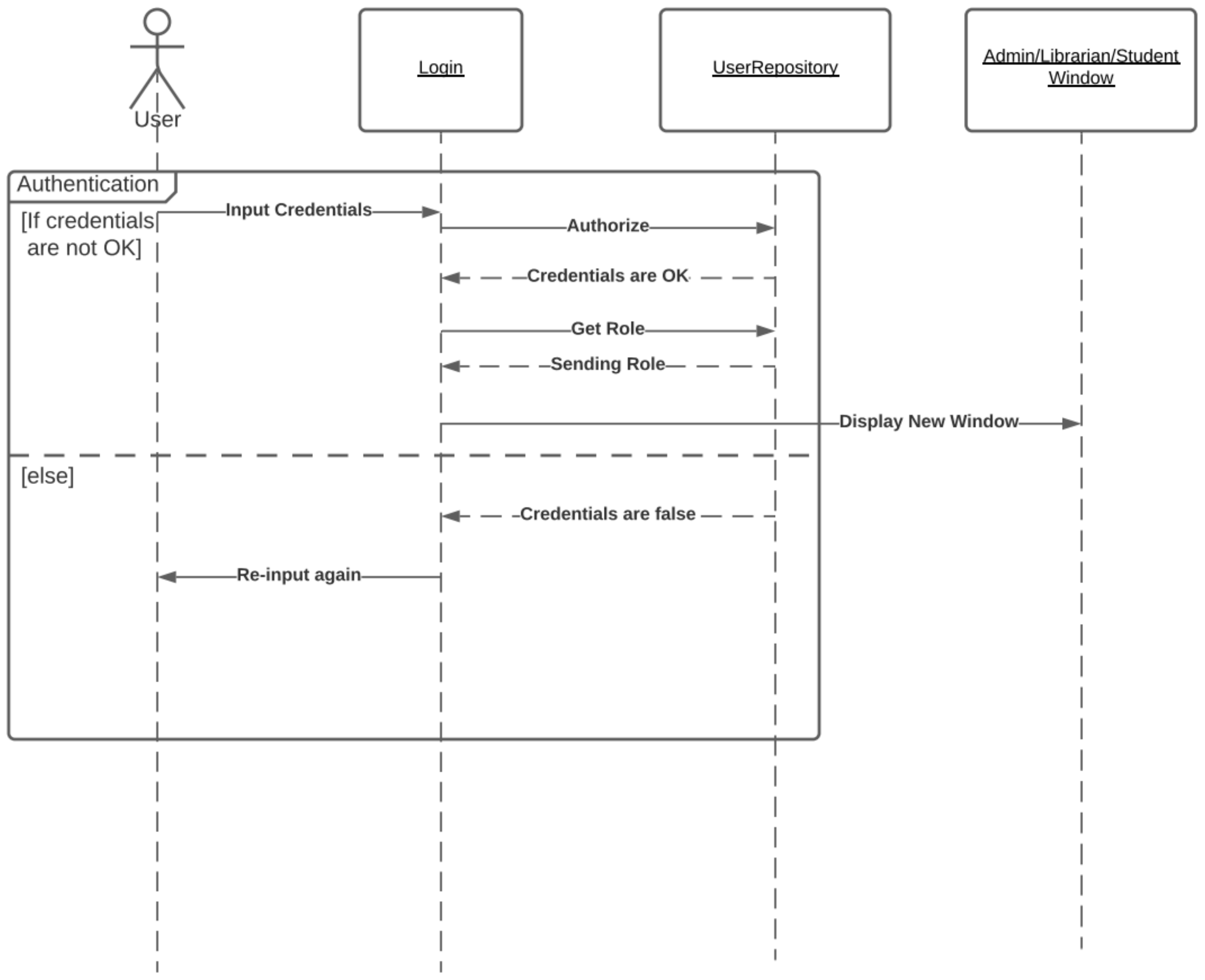


UsersRepository
-instance: UsersRepository
-UsersRepository()
+getInstance(): UsersRepository
+get(userID: String): User
+add(user: User): void
+delete(userID: User): void
+update(oldID: String, newID: String)
+getAll(Role role): ObservableList<User>
+isExists(userID: String): boolean
+authenticate(userID: String, password: String)
Uses authenticate()
LoginController

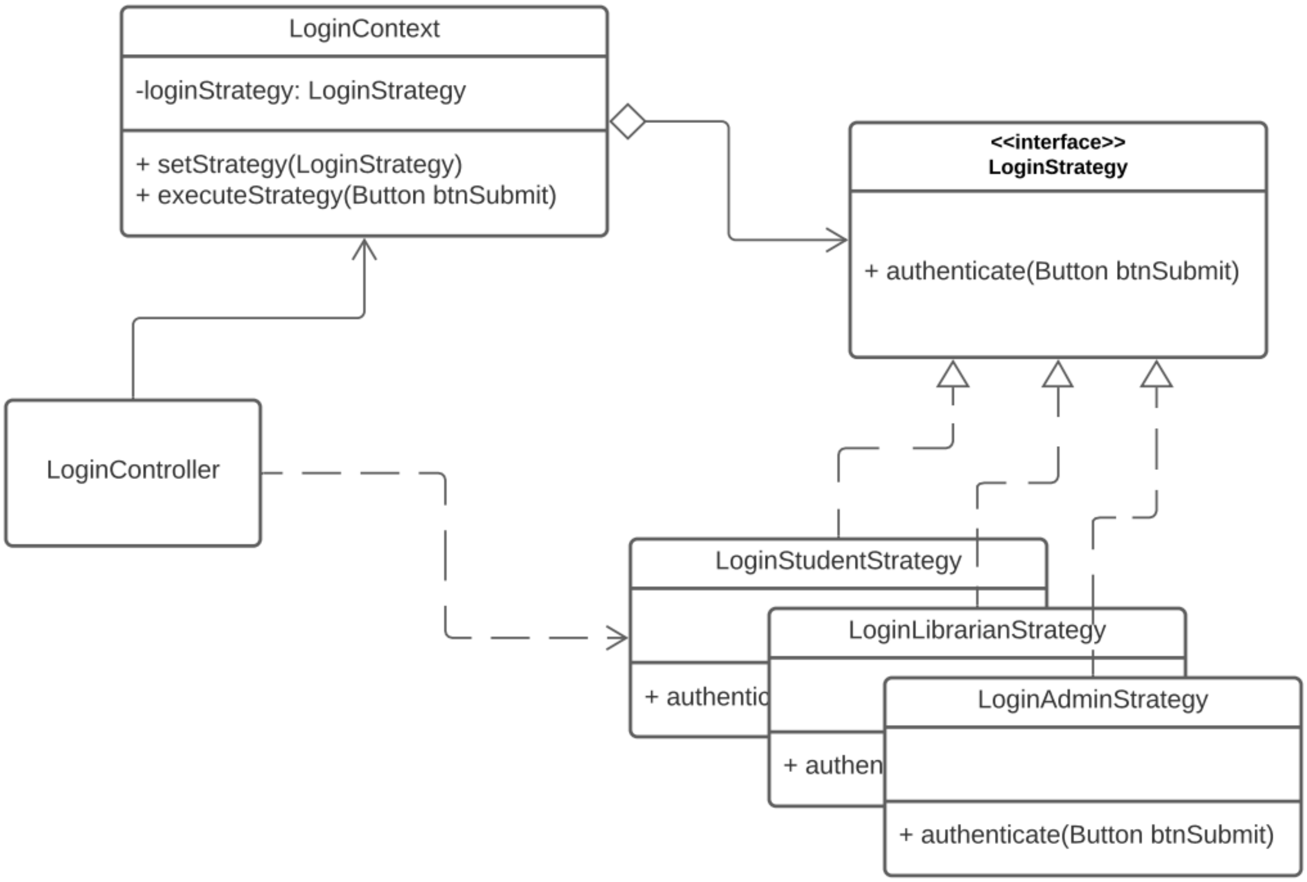
Here we use Singleton pattern to make specific requests to database. Data Access Objects are used extensively by many client classes through the whole program.

Login

We have made several interaction diagrams to design specific parts of our application. The first thing a user must do is to login. So, we applied Strategy pattern to dynamically decide which window to open if user is admin, librarian, or student.



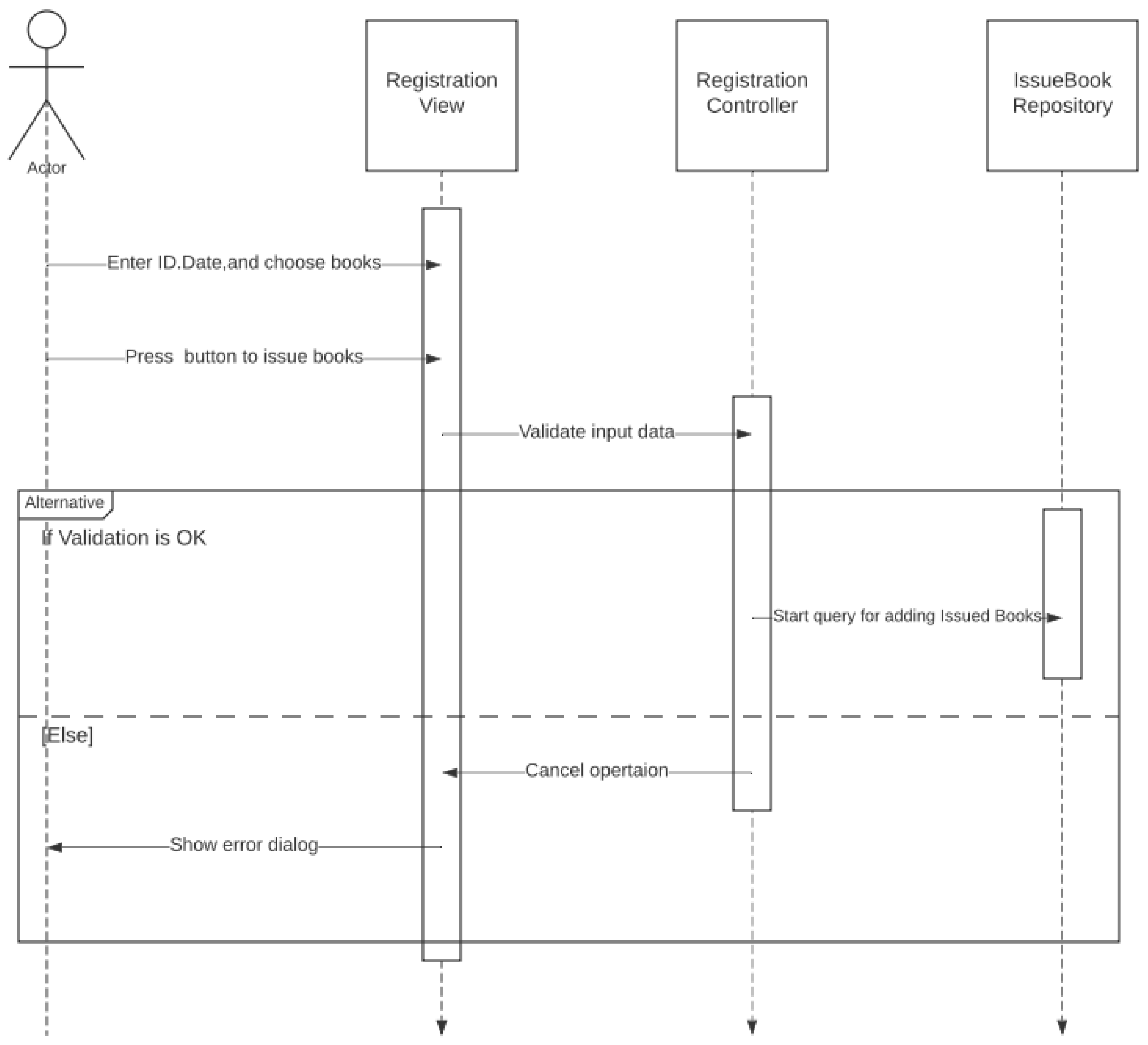
As you can see, we have a sequence diagram between user and our authentication logic. We have made use of UserRepository dao to verify if user inputted correct credentials or not. After that, LoginController class will decide which algorithm of opening windows to run using Strategy pattern.


LoginContext
-loginStrategy: LoginStrategy
+ setStrategy(LoginStrategy)
+ executeStrategy(Button btnSubmit)
<<interface>>
LoginStrategy
+ authenticate(Button btnSubmit)
LoginController
LoginStudentStrategy
+ authentic
LoginLibrarianStrategy
+ authen
LoginAdminStrategy
+ authenticate(Button btnSubmit)

Strategy pattern is a behavioral design pattern that chooses which algorithm to run at runtime. In our case, we make choice between three controllers: AdminWindowController, LibrarianWindowController, and StudentWindowController.

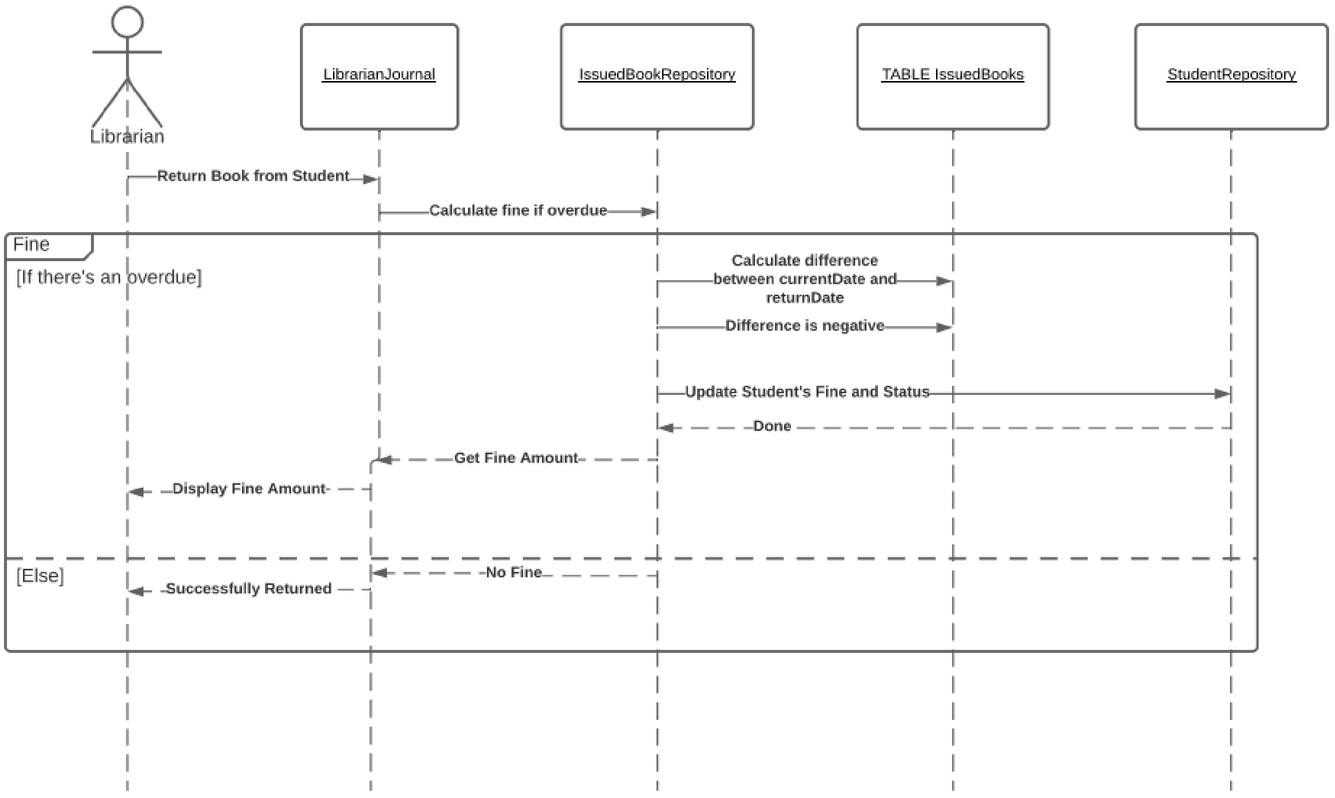
Librarian Functions

Throughout our program, we have many struggles designing librarian's functions. LMS wouldn't be LMS without Librarians, as it turns out. So, we have made interaction diagrams for them.



In the figure above, we have illustrated Librarian's register functions, i.e., issuing books to a particular student. Librarian enters UserID, ReturnDate and chooses the list of books to issue. After that, it gets created in issuedbooks table in our database. We tried to use only one data access object in one controller where possible, reducing project dependencies to minimum.

Another interaction diagram concerns fine issuing process.



Librarian selects the student, then the book from his borrowed books to return and pressed return. After that we calculate the difference between the current day and the return day of that borrowed book. If the latter was overdue already, we multiply the difference of day by 5 and assign a fine to a student. In Student.class we have included FINE_LIMIT static integer to set the limit after which the student is blocked.

Conclusion

As it turns out, we have never developed end-to-user software with GUI. It was difficult for us to design a UML with class and sequence diagrams, so we had to dive into working process directly, which was not the case where you'd expect clean and SOLID code. We could have used some more patterns, such as Mediator pattern for our heavy Controllers, because they start to grow into God objects.

Our database is not perfect; we acknowledge it, during the time we have been learning about SQL we discovered that many tasks that we made by mere selection and then retrieving from Java could be done using simple SQL queries.

Advantages of working in teams:

- Students learn to communicate with other programmers
- Students learn to read and understand teammates code
- Students learn how to work with Git and Github (Commit, Push, Pull, Merge)
- Students learn to write comments about what they added to an existing code

In theory, team members should be coordinated by one experienced person, who can envision every part of the program and who can leave to others only the implementation part. In our case, there was no such a person, so we wrote a code through trial and error.