Introduction

This assignment is based on developing a LIS (Library Management System) using Java Programming Language. For that, we used GUI (Graphical User Interface) in this development so that it will become more users friendly to interact. Besides, we also added a database to store important data related to our project.

Explanation

In this documentation, we have given explanations of how to interact successfully with this a LIS. We have explained here step by step so that it will surely help users to become more user friendly with it. Below are our explanations: Required software: Before executing this program, users need to do some works so that it will run properly into their system. First, they need to make sure their system is having "JDK". If they don't have it then they can download from this below link:

https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

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 to download the IDE (Integrated Development Environment) on their system. They can download ECLIPSE or NETBEANS or IntelliJ depending on the windows (32bit/64bit).

Netbeans: https://netbeans.org/downloads/

Eclipse: http://www.eclipse.org/downloads/

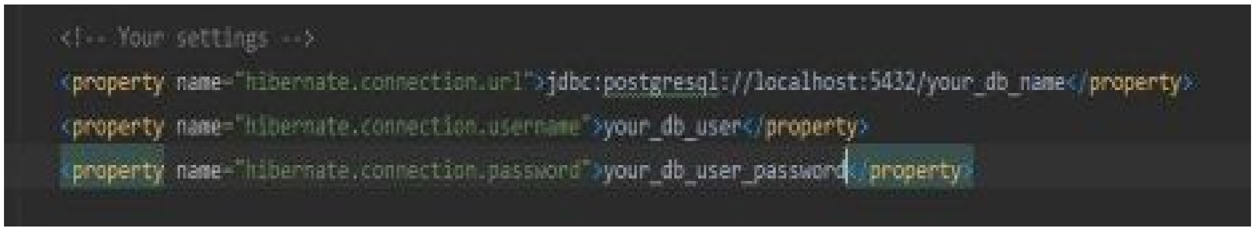
IntelliJ: https://www.jetbrains.com/idea/download/

I developed this program using "IntelliJ". Also, we need to install the PostgreSQL database which allows us to efficiently store, modify, get, and delete required data for our project. If they don't have it then they can download from this below link:

https://www.postgresql.org/download/

Project Configurations

After PostgreSQL installation we need to create one database user which will be used as a daily database user, also we need to create a database to hold our data. After the creation of database and user change the values of properties in hibernate.cfg.xml.



```
<!-- Your settings -->
<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/your_db_name</property>
<property name="hibernate.connection.username">your_db_user</property>
<property name="hibernate.connection.password">your_db_user_password</property>
```

Figure 1: Configuration file

Execution procedures

When a user executes this program, it will show the startup GUI (Graphical User Interface).

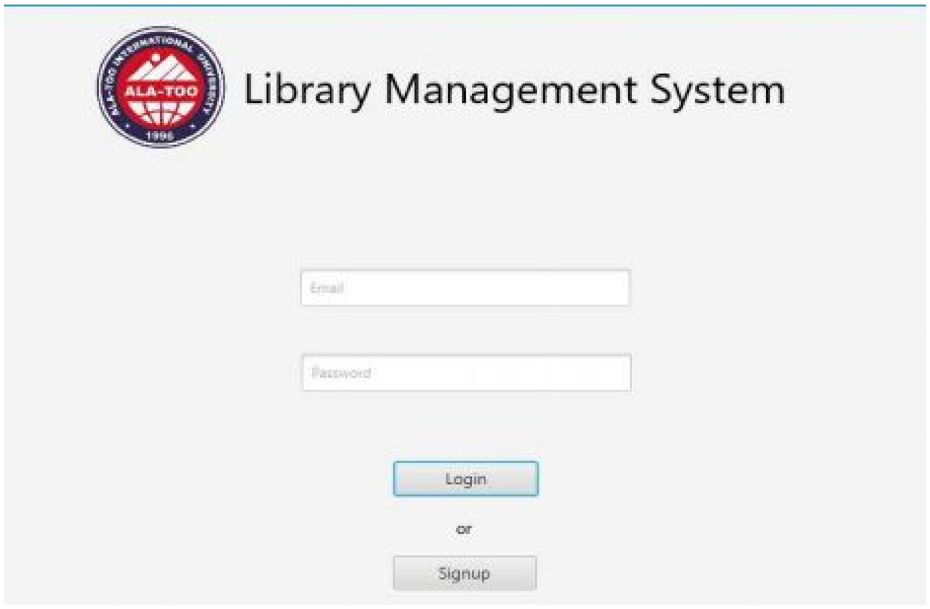


Figure 2: Login Screen

Login GUI allows to login with a **librarian** and ordinary **user** account.

Now a user has to option where to login with his **email** and **password** or user can create a new account for himself by making registration to the system. Let's consider the second option and look at the signup process.

Sign Up Procedure

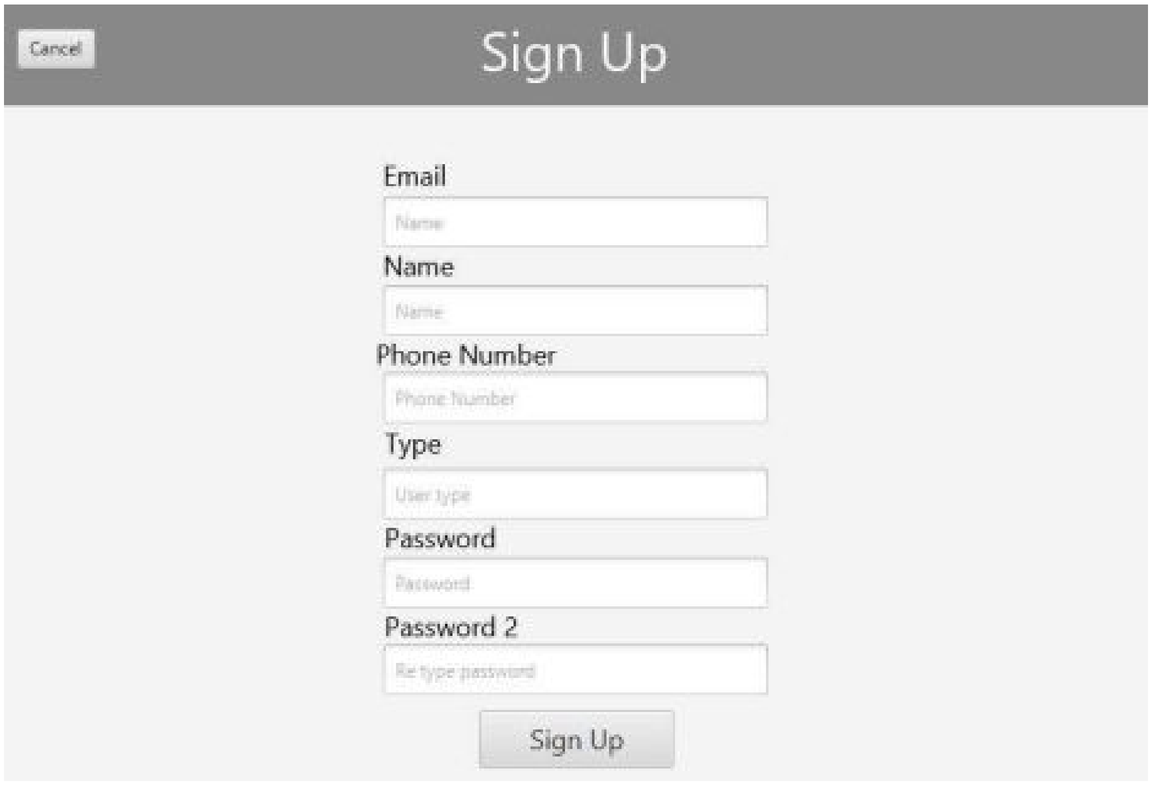


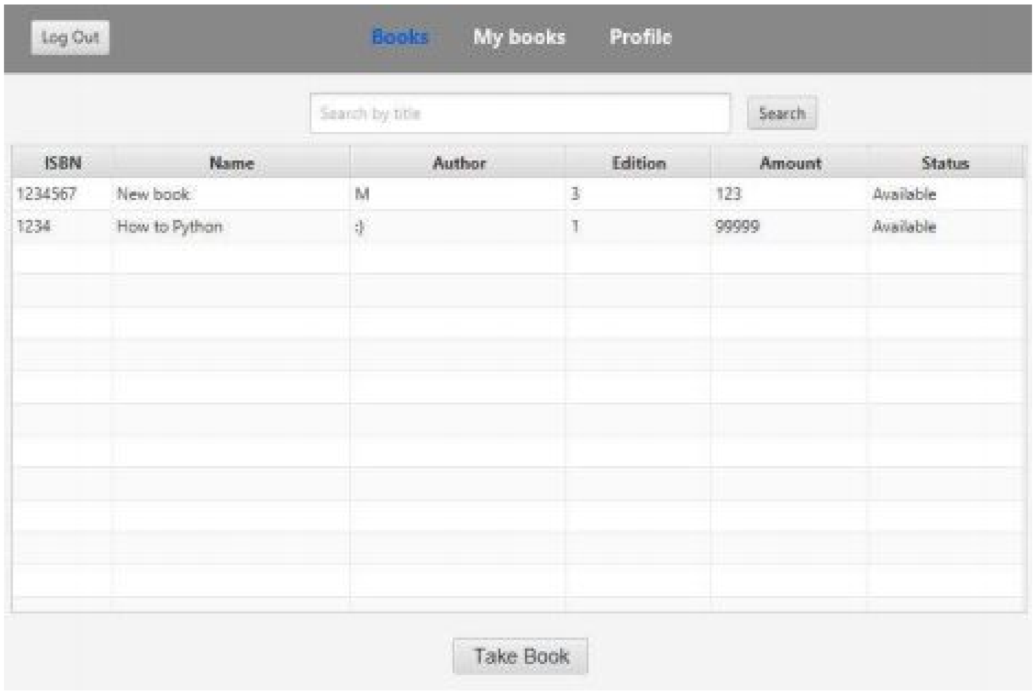
Figure 3: Signup Screen

Here user needs to fill all required data about himself like name, email address, phone number, type of users like Student, Staff Member, and etc. Finally, the user needs to provide a password, both **<u>password and password2 need to be identical</u>**. Also, none of the form fields can be empty or user will see an error telling that some fields are empty. If the user made **<u>successful registration</u>** he will be returned to **<u>log in screen</u>**. If user miss clicked the signup button he can be immediately returned to the **<u>login screen</u>** by pressing the **<u>Cancel</u>** button.

After registration user will be allowed to log in to the library management system.

User GUI

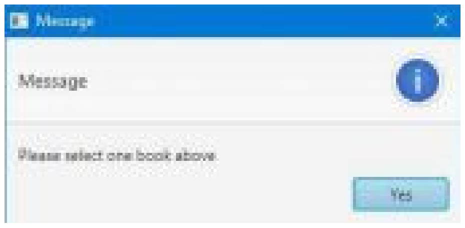
After successful authentication user will see Books screen with all books that are currently registered at library management system by the librarian.



ISBN	Name	Author	Edition	Amount	Status
1234567	New book	M	3	123	Available
1234	How to Python	:)	1	99999	Available

Figure 4: Books available at LMS

Here user can **<u>choose a book</u>** and take it by clicking **<u>Take Book</u>** button if no books are chosen he will see the alert '**<u>Please select one book above'</u>.**

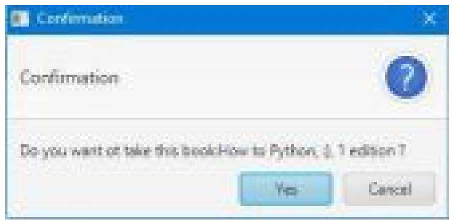

Message

Message

Please select one book above

Figure 5: No books selected alert

If the user selected book, he will see a confirmation box asking whether he wants to add this book to his list.


Confirmation

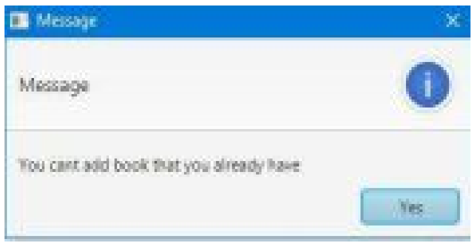
Confirmation

Do you want ot take this bookHow to Python, ,|, 1 edition ?

Figure 6: Confirmation

Taking a book will decrease the number of available books. If a number of specific books reach zero the book will be unavailable to all users and its status will become **Unavailable**.

If the user **already took this book** he will see the alert box telling that user **can't add this book to his list**.


Message

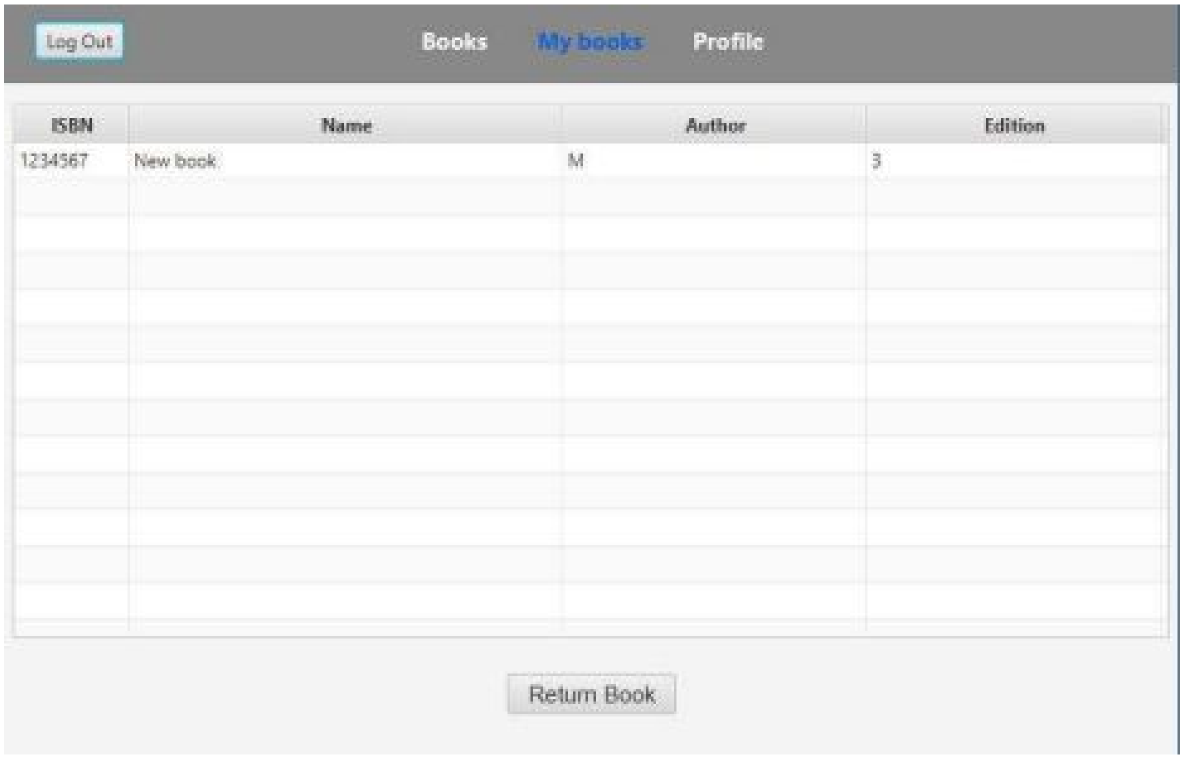
Message

You cant add book that you already have

Figure 7: User already having a specific book

All the books that were taken can be seen at **My Books** tab.

Inside of **My Books** tab, we can see all the books that were taken by the user.



ISBN	Name	Author	Edition
1234567	New book	M	3

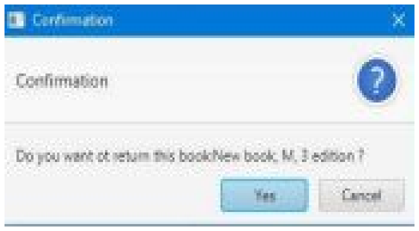
Figure 9: Books that were taken by the current user

By clicking **Return Book** user can return a book. If no book was chosen user will see an alert that no was book was taken, similar to **Books** Screen.

If the book was select user will see a confirmation box was asking whether he wants to return a specific book.

Returning Book will remove the selected book from the user's book list and add to the general book pool and Increase the number of available books.

field empty.

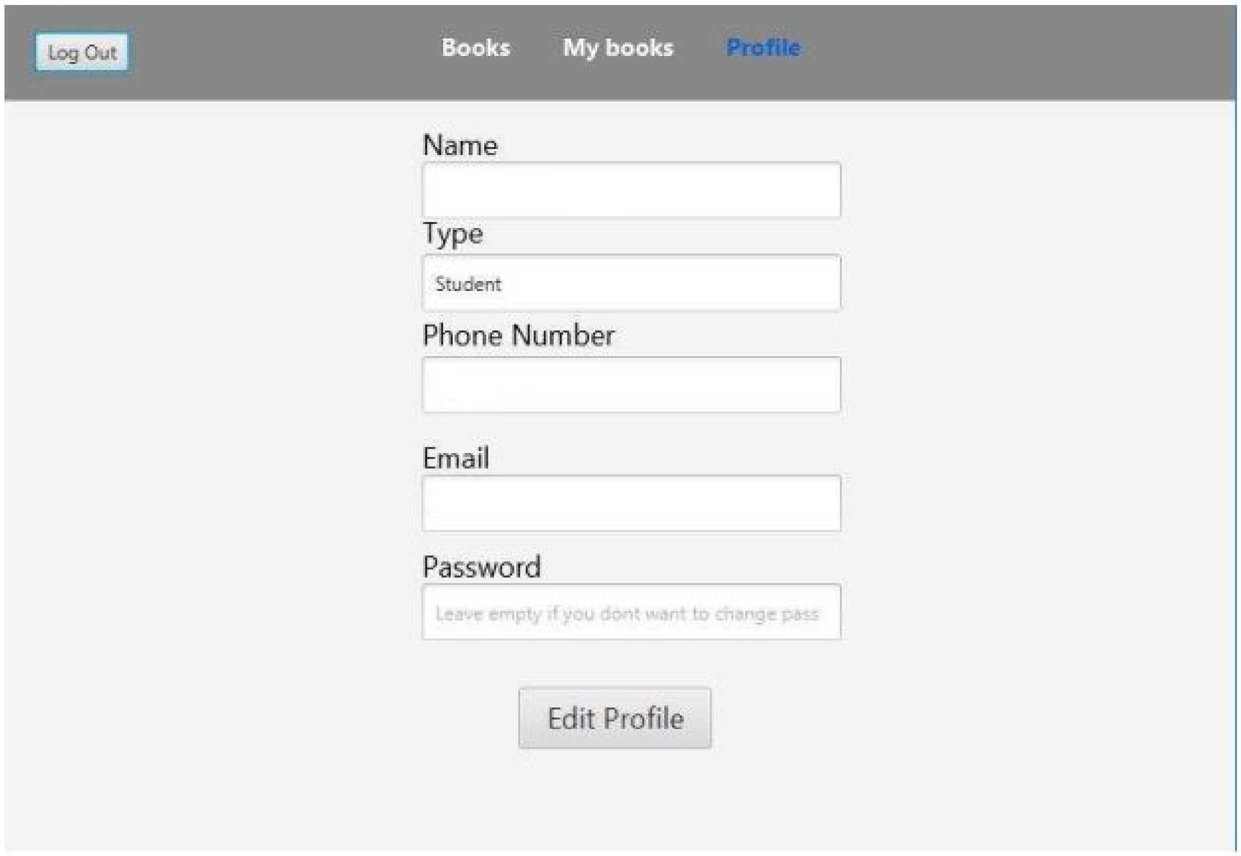

Confirmation

Confirmation

Do you want ot return this book:New book: M, 3 edition ?

Figure 8: Return Book confirmation

Inside of **Profile Tab,** we can see all the information about current user and fields that prepopulated with data which equal to user's data.



Name

Type

Student

Phone Number

Email

Password

Leave empty if you dont want to change pass

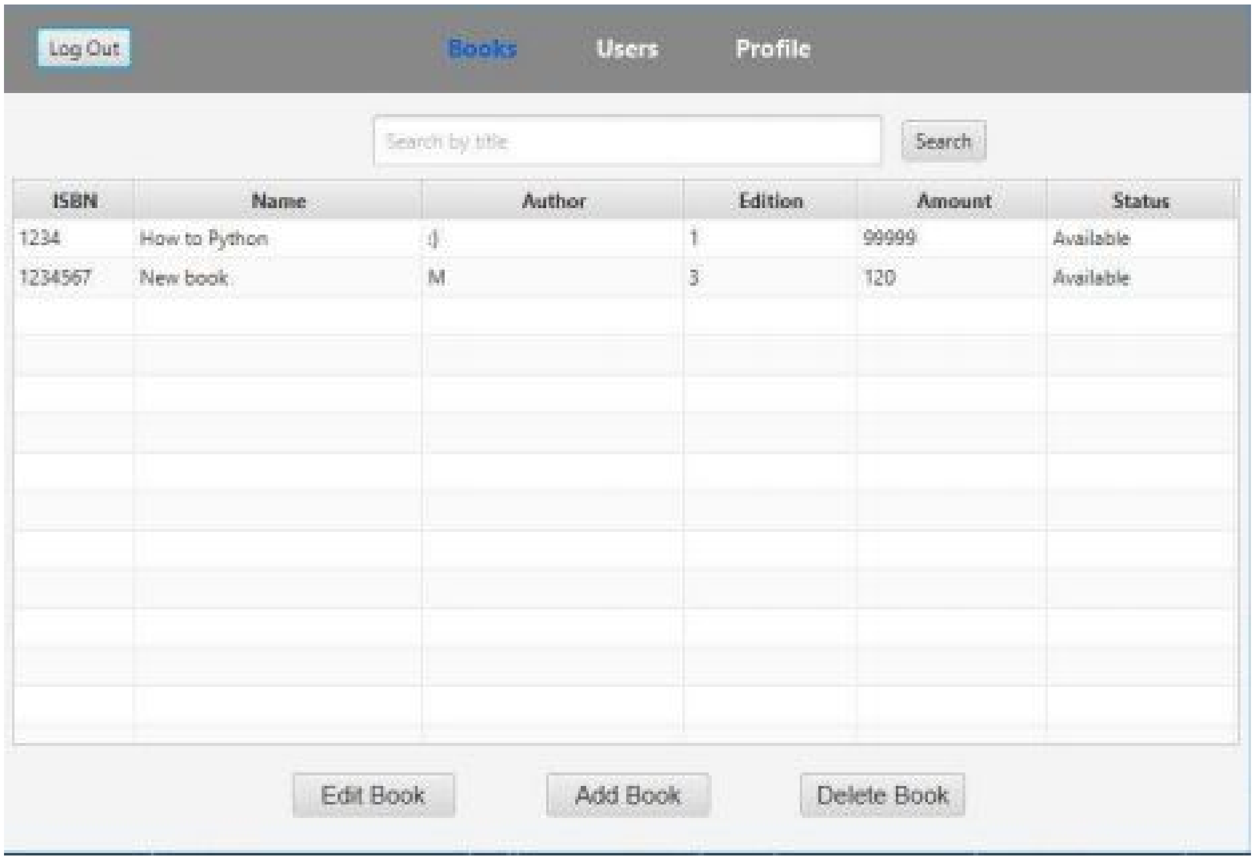
Figure 9: User Profile

Here user can edit data about himself. **None of the fields can be empty except for the password field** if the user is not willing to change his current password he can leave this field empty.

By clicking **Edit Profile** user will update their profile with the data from corresponding fields.

Librarians GUI

Now let's **Login as Librarian**. The first thing that librarian sees is the same table with all available books at LIS but with some more control. A librarian can add, delete, and edit books from this **Books** screen.



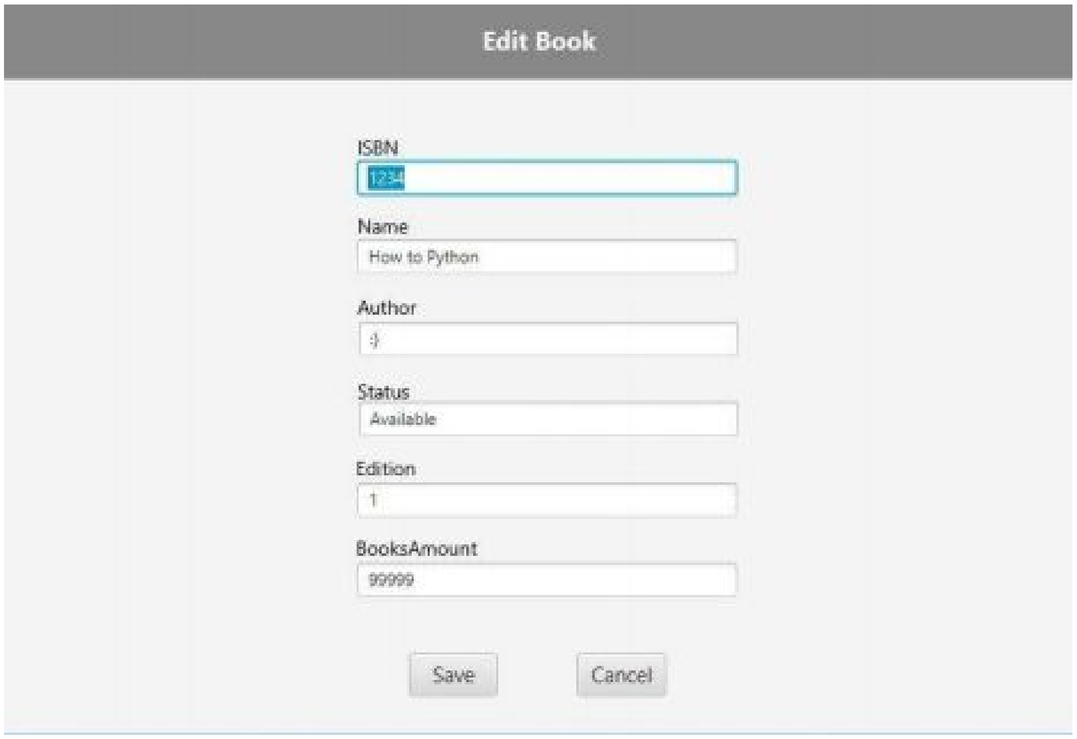
ISBN	Name	Author	Edition	Amount	Status
1234	How to Python	:)	1	99999	Available
1234567	New book	M	3	120	Available

Figure 10: Librarian Books screen

Button Below correspond to actions they describe.

By selecting a book and clicking Edit Book librarian will see another screen with book edition form if no book was selected librarian will see an alert box similar to user **Books** screen.

By Editing special book librarian can increase the number of this book, change Name, Author, Edition.


Edit Book

ISBN	1234
Name	How to Python
Author	:)
Status	Available
Edition	1
BooksAmount	99999

Figure11: Edit Book Screen

By clicking **Save** librarian will save altered data to a specific book, after successful editing librarian will return to **Books** Screen, if librarian wants to cancel the process it can be done by clicking the **Cancel** button.

Object-Oriented Explanation

In object-oriented programming, for example, an object is a self-contained entity that consists of both data and procedures to manipulate the data. In another way, object-oriented is the software engineering concept where it is represented using the "OBJECTS". Below are the objected-oriented parts we used in this "Library

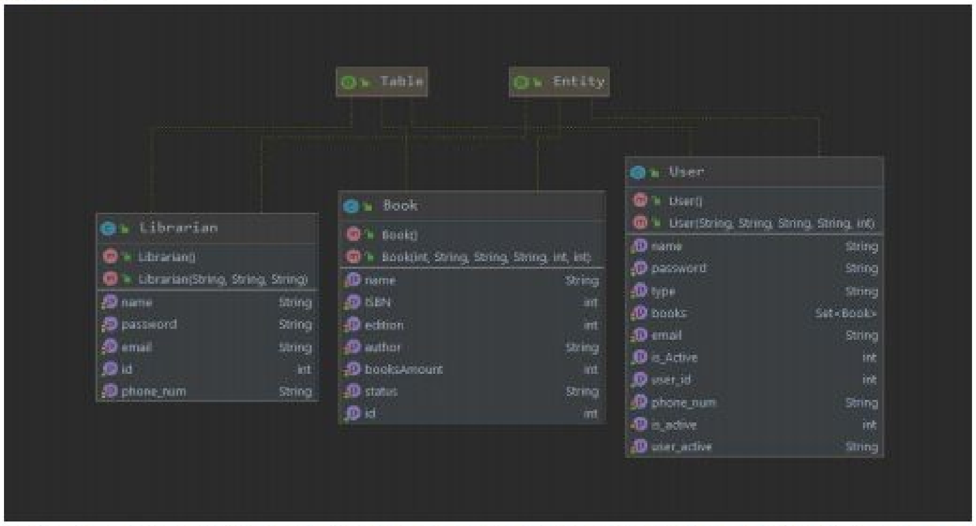
Management System”:

Database connection:

Connection to the database is made with **Hibernate ORM** which allowed to

simplify the development process and database design.

UML diagram:



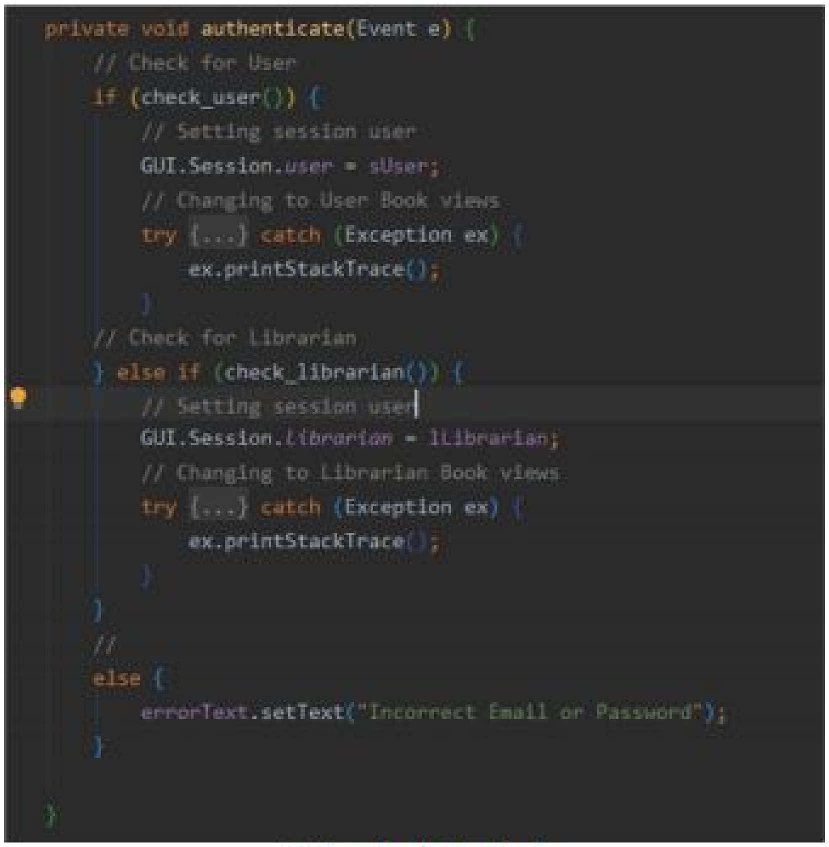
Classes in this diagram represents tables in database, because of ORM every entity of table becomes the instance of that class and its very easy to manage these entities.

Main Classes and Methods:

In library management system every functionality is divided into separate reusable code. To give detailed explanation on each function I want to start from the begining of Execution Procedure or Login Screen and give view of the thing that happening under the hood of Library Management System, this way you will get better understanding how specific functions are related and some nuances that need to covered, which will help for the future users or maintainers of this program

Login:

When user try to login with his credentials how its determined whether user is a librarian or an ordinary user. **Login process** consist of three main methods **authenticate()**, **check_librarian()**, **check_user().**



```java
private void authenticate(Event e) {
    // Check for User
    if (check_user()) {
        // Setting session user
        GUI.Session.user = sUser;
        // Changing to User Book views
        try {...} catch (Exception ex) {
            ex.printStackTrace();
        }
    // Check for Librarian
    } else if (check_librarian()) {
        // Setting session user
        GUI.Session.Librarian = lLibrarian;
        // Changing to Librarian Book views
        try {...} catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    //
    else {
        errorText.setText("Incorrect Email or Password");
    }

}
```

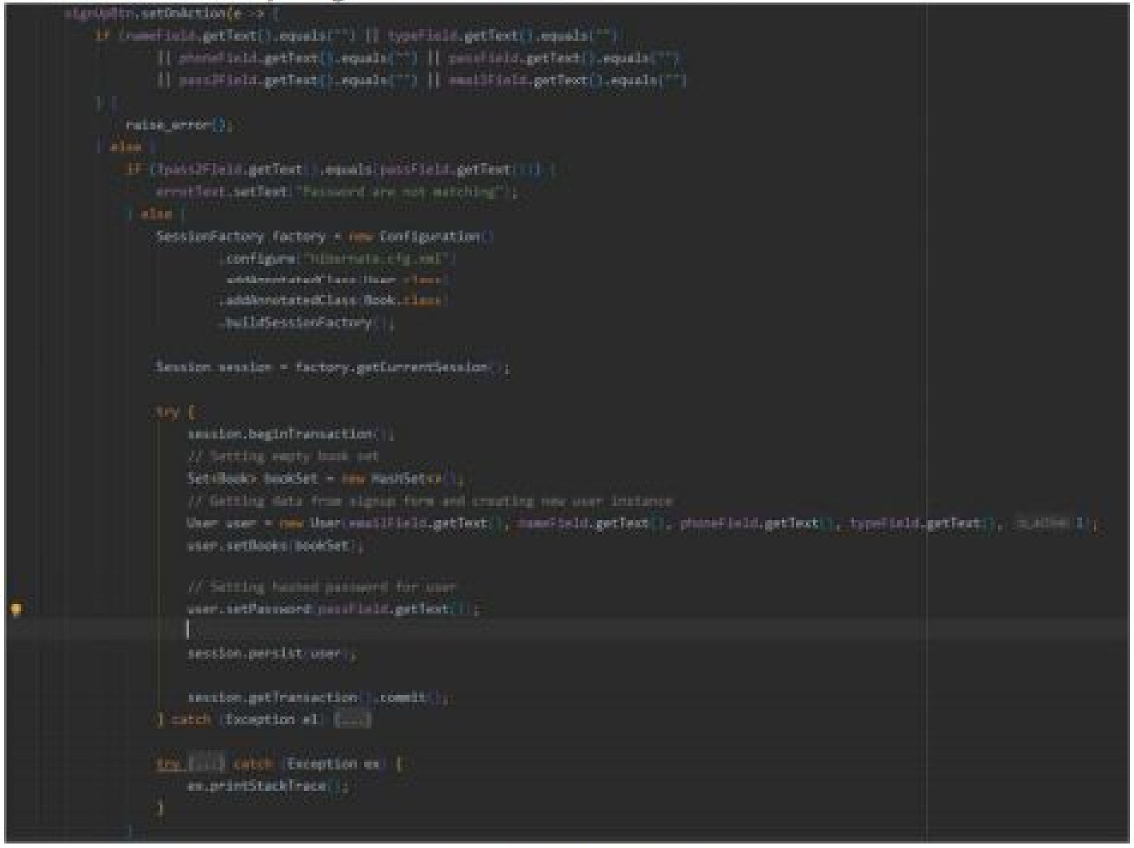
Authenticate Method

authenticate() executes two methods mentioned above if one them return True then users belongs to that category, after it sets User for this session and changes the view to Books view of each user category, if none of them is matching displays an error.

check_user() and **check_librarian()** are almost the same function but with difference in table that they make query and that **user can be active or inactive**.

Signup:

If user wants to create account in LIS he or she must fill the signup form and submit it. Signup process is bound to **Signup Button.** Every time user clicks it function take data from fields validates it and if every thing is correct saves user to database.



```java
    signUpBtn.setOnAction(e -> {
        if (nameField.getText().equals("") || typeField.getText().equals("")
                || phoneField.getText().equals("") || passField.getText().equals("")
                || pass2Field.getText().equals("") || email3Field.getText().equals("")
        ) {
            raise_error();
        } else {
            if (!pass2Field.getText().equals(passField.getText())) {
                errorText.setText("Password are not matching");
            } else {
                SessionFactory factory = new Configuration()
                        .configure("hibernate.cfg.xml")
                        .addAnnotatedClass(User.class)
                        .addAnnotatedClass(Book.class)
                        .buildSessionFactory();

                Session session = factory.getCurrentSession();

                try {
                    session.beginTransaction();
                    // Setting empty book set
                    Set<Book> bookSet = new HashSet<>();
                    // Getting data from signup form and creating new user instance
                    User user = new User(emailField.getText(), nameField.getText(), phoneField.getText(), typeField.getText(), is_admin: 1);
                    user.setBooks(bookSet);

                    // Setting hashed password for user
                    user.setPassword(passField.getText());

                    session.persist(user);

                    session.getTransaction().commit();
                } catch (Exception e1) {...}

                try {...} catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
```

Signup on button click