LIBRARY MANAGEMENT INVENTORY

In this modern era of the internet, almost all of us rely on web-based applications from small to big tasks. Well, Library management system is one of the most popular use-cases considered by the professionals while [building applications in Java](https://www.edureka.co/blog/servlet-and-jsp-tutorial/). In this article, I will show you how to create a library management system project in Java.

The following will be the sequence of topics for this article:

1. [What is Java?](https://www.edureka.co/blog/library-management-system-project-in-java#Java)
2. [What is My SQL?](https://www.edureka.co/blog/library-management-system-project-in-java#MySQL)
3. [Library Management System Project](https://www.edureka.co/blog/library-management-system-project-in-java#LibraryManagementSystemProject)
   * + [System Requirements](https://www.edureka.co/blog/library-management-system-project-in-java#SystemRequirements)
     + [Tables Considered](https://www.edureka.co/blog/library-management-system-project-in-java#TablesConsidered)
     + [Project Code](https://www.edureka.co/blog/library-management-system-project-in-java#ProjectCode)
     + [Output](https://www.edureka.co/blog/library-management-system-project-in-java#Output)

Let’s get started.

**What is Java?**

Java is an[object-oriented language](https://www.edureka.co/blog/object-oriented-programming/) similar to[C++](https://www.edureka.co/blog/object-oriented-programming-in-cpp/), but with advanced and simplified features. Java is **free to access** and can **run** on **all platforms**.

The features of Java are as follows:

* **Simple:**Java has made life easier by removing all the complexities such as pointers, operator overloading as you see in C++ or any other programming language.
* **Object-oriented:**Everything is considered to be an “[**object**](https://www.edureka.co/blog/java-objects-and-classes/)” which possess some state, behavior and all the operations are performed using these objects.
* **Secured:**All the code is converted in **[bytecode](https://www.edureka.co/blog/java-bytecode/" \t "_blank)** after compilation, which is not readable by a human. and java does not use an explicit pointer and run the programs inside the sandbox to prevent any activities from untrusted sources. It enables to develop virus-free, tamper-free systems/applications.

**What is MySQL?**

[**MySQL**](https://www.edureka.co/blog/mysql-tutorial/) is an open-source relational database management system that works on many platforms. It provides multi-user access to support many storage engines and is backed by Oracle. So, you can buy a commercial license version from Oracle to get premium support services.

The features of My SQL are as follows:

* **Ease of Management –**The software very easily gets downloaded and also uses an event scheduler to schedule the tasks automatically.
* **Robust Transactional Support –**Holds the ACID (Atomicity, Consistency, Isolation, Durability) property, and also allows distributed multi-version support.
* **Comprehensive Application Development –**MySQL has plugin libraries to embed the database into any application. It also supports stored procedures, triggers, functions, views and many more for application development. Refer to [***RDS Tutorial***](https://www.edureka.co/blog/rds-aws-tutorial/) to understand Amazon’s RDBMS.

**Library Management System Project in Java**

Library Management System is one of the most popular projects which is created using Java. So, in this article, I will show you how to create this project using the following system requirements.

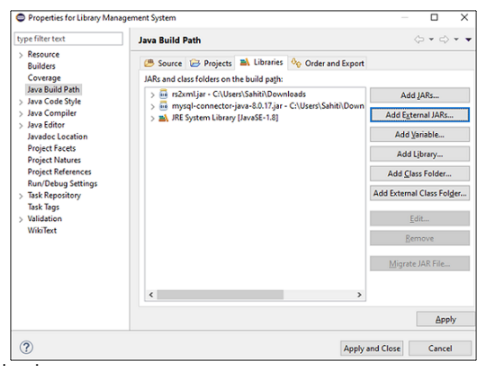
**System Requirements**

To execute the below project, you will need the following business requirements:

* [MySQL Community Server](https://www.edureka.co/blog/install-mysql/)
* MySQL JDBC Connector
* [Java](https://www.oracle.com/technetwork/java/javase/downloads/index.html)
* [Eclipse IDE](https://www.edureka.co/blog/setup-eclipse-ide/)
* rs2xml.jar

The rs2xml jar is used to display the data in a table format. So, once you create a project in Eclipse IDE, you have to import the rs2xml jar and [JDBC connector JAR](https://www.edureka.co/blog/connect-mysql-database-in-java) into the project.

To do that, **right-click on the project**, choose**Build Path** -> **Configure Build Path**. In the dialog box, which opens up, choose **Add External JARs**, and add the JAR files. Once added, click on Apply **and Close**. Refer below.



**Tables Considered**

Now, for this particular project, I have considered three tables, which are:

* **Users** -> This table consists of the columns {UID, Username, Password, Admin}
* **Books**-> The book’s table consists of the columns {BID, Book name, Price, Genre}
* **Issue** -> This table consists of the columns {IID, UID, BID, IssueDate, Period, ReturnDate, Fine}

Alright, so now that the Initial set is done, and I have told you the schema of tables, let us get started.

**Library Management System Project Code**

For your better understanding, I have divided the code into the following functions and I will be explaining you function-wise:

* [Login](https://www.edureka.co/blog/library-management-system-project-in-java#Login)
* [Connect](https://www.edureka.co/blog/library-management-system-project-in-java#Connect)
* [Create/ Reset](https://www.edureka.co/blog/library-management-system-project-in-java#CreateorReset)
* [User Menu](https://www.edureka.co/blog/library-management-system-project-in-java#UserMenu)
* [Admin Menu](https://www.edureka.co/blog/library-management-system-project-in-java#AdminMenu)

Also, to create a GUI, I will be using Swing. Swing is a library or a set of program components used to create graphical user interface components such as scroll bars, buttons, dialog boxes, etc.

Before I discuss the code of functions with you, let me show you the code for the main class and the libraries to be imported:

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.Locale;

import java.util.concurrent.TimeUnit;

import javax.swing.\*;

import net.proteanit.sql.DbUtils;

public class main {

public static class ex{

public static int days=0;

}

public static void main(String[] args) {

login();

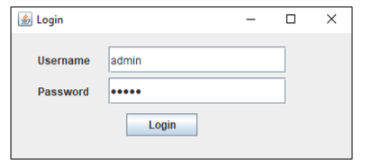
//create();

}

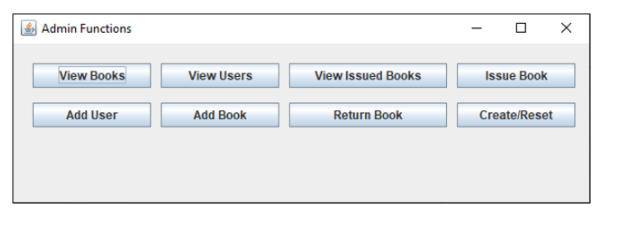
Now in this article on Library Management System in Java, let us understand the code of all the above functions.

#### ****Login****

I have created this function to enable the user and the admin login. So, initially when a user logs in for the first time, that user will be an admin by default, and the username and password will be {admin, admin}. Refer below.



For this schema, I have considered only one admin. So, once a user logs in as an admin, he or she will be redirected to the admin menu as below. I will discuss the functions of the admin in the [admin menu](https://www.edureka.co/blog/library-management-system-project-in-java#AdminMenu) section.



Coming back to the login functions, refer to the below code:

public static void login() {

JFrame f=new JFrame("Login");//creating instance of JFrame

JLabel l1,l2;

l1=new JLabel("Username"); //Create label Username

l1.setBounds(30,15, 100,30); //x axis, y axis, width, height

l2=new JLabel("Password"); //Create label Password

l2.setBounds(30,50, 100,30);

JTextField F\_user = new JTextField(); //Create text field for username

F\_user.setBounds(110, 15, 200, 30);

JPasswordField F\_pass=new JPasswordField(); //Create text field for password

F\_pass.setBounds(110, 50, 200, 30);

JButton login\_but=new JButton("Login");//creating instance of JButton for Login Button

login\_but.setBounds(130,90,80,25);//Dimensions for button

login\_but.addActionListener(new ActionListener() { //Perform action

public void actionPerformed(ActionEvent e){

String username = F\_user.getText(); //Store username entered by the user in the variable "username"

String password = F\_pass.getText(); //Store password entered by the user in the variable "password"

if(username.equals("")) //If username is null

{

JOptionPane.showMessageDialog(null,"Please enter username"); //Display dialog box with the message

}

else if(password.equals("")) //If password is null

{

JOptionPane.showMessageDialog(null,"Please enter password"); //Display dialog box with the message

}

else { //If both the fields are present then to login the user, check wether the user exists already

//System.out.println("Login connect");

Connection connection=connect(); //Connect to the database

try

{

Statement stmt = connection.createStatement();

stmt.executeUpdate("USE LIBRARY"); //Use the database with the name "Library"

String st = ("SELECT \* FROM USERS WHERE USERNAME='"+username+"' AND PASSWORD='"+password+"'"); //Retreive username and passwords from users

ResultSet rs = stmt.executeQuery(st); //Execute query

if(rs.next()==false) { //Move pointer below

System.out.print("No user");

JOptionPane.showMessageDialog(null,"Wrong Username/Password!"); //Display Message

}

else {

f.dispose();

rs.beforeFirst(); //Move the pointer above

while(rs.next())

{

String admin = rs.getString("ADMIN"); //user is admin

//System.out.println(admin);

String UID = rs.getString("UID"); //Get user ID of the user

if(admin.equals("1")) { //If boolean value 1

admin\_menu(); //redirect to admin menu

}

else{

user\_menu(UID); //redirect to user menu for that user ID

}

}

}

}

catch (Exception ex) {

ex.printStackTrace();

}

}

}

});

f.add(F\_pass); //add password

f.add(login\_but);//adding button in JFrame

f.add(F\_user); //add user

f.add(l1); // add label1 i.e. for username

f.add(l2); // add label2 i.e. for password

f.setSize(400,180);//400 width and 500 height

f.setLayout(null);//using no layout managers

f.setVisible(true);//making the frame visible

f.setLocationRelativeTo(null);

}

#### ****Connect****

The connect function is used to connect the database to the [GUI](https://www.edureka.co/blog/java-swing/). So, to do that, I have mentioned the below code:

/\*Connect

The connect function is used to connect the database to the GUI. So, to do that, I have mentioned the below code:

\*/

public static Connection connect()

{

try {

Class.forName("com.mysql.cj.jdbc.Driver");

//System.out.println("Loaded driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mysql?user=root&password=edureka");

//System.out.println("Connected to MySQL");

return con;

}

catch (Exception ex) {

ex.printStackTrace();

}

return null;

}

In the above function, we are connecting our **MySQL database** with the**username “root”** and **password “edureka”** to our application. Now, once the application is connected to the database, our next step is to create or reset the database. So, next in this article on Library Management System Project in Java, let us discuss the Create function..

#### ****Create****

The create function is used to create the database, tables and add data into these tables. So, to do that, [SQL statements](https://www.edureka.co/blog/sql-commands) will be used as below.

/\*

Create

The create function is used to create the database, tables and add data into these tables.

So, to do that, SQL statements will be used as below.

\*/

public static void create() {

try {

Connection connection=connect();

ResultSet resultSet = connection.getMetaData().getCatalogs();

//iterate each catalog in the ResultSet

while (resultSet.next()) {

// Get the database name, which is at position 1

String databaseName = resultSet.getString(1);

if(databaseName.equals("library")) {

//System.out.print("yes");

Statement stmt = connection.createStatement();

//Drop database if it pre-exists to reset the complete database

String sql = "DROP DATABASE library";

stmt.executeUpdate(sql);

}

}

Statement stmt = connection.createStatement();

String sql = "CREATE DATABASE LIBRARY"; //Create Database

stmt.executeUpdate(sql);

stmt.executeUpdate("USE LIBRARY"); //Use Database

//Create Users Table

String sql1 = "CREATE TABLE USERS(UID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY, USERNAME VARCHAR(30), PASSWORD VARCHAR(30), ADMIN BOOLEAN)";

stmt.executeUpdate(sql1);

//Insert into users table

stmt.executeUpdate("INSERT INTO USERS(USERNAME, PASSWORD, ADMIN) VALUES('admin','admin',TRUE)");

//Create Books table

stmt.executeUpdate("CREATE TABLE BOOKS(BID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY, BNAME VARCHAR(50), GENRE VARCHAR(20), PRICE INT)");

//Create Issued Table

stmt.executeUpdate("CREATE TABLE ISSUED(IID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY, UID INT, BID INT, ISSUED\_DATE VARCHAR(20), RETURN\_DATE VARCHAR(20), PERIOD INT, FINE INT)");

//Insert into books table

stmt.executeUpdate("INSERT INTO BOOKS(BNAME, GENRE, PRICE) VALUES ('War and Peace', 'Mystery', 200), ('The Guest Book', 'Fiction', 300), ('The Perfect Murder','Mystery', 150), ('Accidental Presidents', 'Biography', 250), ('The Wicked King','Fiction', 350)");

resultSet.close();

}

catch (Exception ex) {

ex.printStackTrace();

}

}

Now, that we have created the database, connected with GUI and enables the login function, next in this article on Library Management System Project in Java, let us now discuss the functions of the User Menu.

#### ****User Menu****

The User Menu is designed to show details of all the books present in the library and the books issued by the user.

/\*

User Menu

The User Menu is designed to show details of all the books present in the library and the books issued by the user.

\*/

public static void user\_menu(String UID) {

JFrame f=new JFrame("User Functions"); //Give dialog box name as User functions

//f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //Exit user menu on closing the dialog box

JButton view\_but=new JButton("View Books");//creating instance of JButton

view\_but.setBounds(20,20,120,25);//x axis, y axis, width, height

view\_but.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

JFrame f = new JFrame("Books Available"); //View books stored in database

//f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Connection connection = connect();

String sql="select \* from BOOKS"; //Retreive data from database

try {

Statement stmt = connection.createStatement(); //connect to database

stmt.executeUpdate("USE LIBRARY"); // use librabry

stmt=connection.createStatement();

ResultSet rs=stmt.executeQuery(sql);

JTable book\_list= new JTable(); //show data in table format

book\_list.setModel(DbUtils.resultSetToTableModel(rs));

JScrollPane scrollPane = new JScrollPane(book\_list); //enable scroll bar

f.add(scrollPane); //add scroll bar

f.setSize(800, 400); //set dimensions of view books frame

f.setVisible(true);

f.setLocationRelativeTo(null);

} catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

}

);

JButton my\_book=new JButton("My Books");//creating instance of JButton

my\_book.setBounds(150,20,120,25);//x axis, y axis, width, height

my\_book.addActionListener(new ActionListener() { //Perform action

public void actionPerformed(ActionEvent e){

JFrame f = new JFrame("My Books"); //View books issued by user

//f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

int UID\_int = Integer.parseInt(UID); //Pass user ID

//.iid,issued.uid,issued.bid,issued.issued\_date,issued.return\_date,issued,

Connection connection = connect(); //connect to database

//retrieve data

String sql="select distinct issued.\*,books.bname,books.genre,books.price from issued,books " + "where ((issued.uid=" + UID\_int + ") and (books.bid in (select bid from issued where issued.uid="+UID\_int+"))) group by iid";

String sql1 = "select bid from issued where uid="+UID\_int;

try {

Statement stmt = connection.createStatement();

//use database

stmt.executeUpdate("USE LIBRARY");

stmt=connection.createStatement();

//store in array

ArrayList books\_list = new ArrayList();

ResultSet rs=stmt.executeQuery(sql);

JTable book\_list= new JTable(); //store data in table format

book\_list.setModel(DbUtils.resultSetToTableModel(rs));

//enable scroll bar

JScrollPane scrollPane = new JScrollPane(book\_list);

f.add(scrollPane); //add scroll bar

f.setSize(800, 400); //set dimensions of my books frame

f.setVisible(true);

f.setLocationRelativeTo(null);

} catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

}

);

f.add(my\_book); //add my books

f.add(view\_but); // add view books

f.setSize(300,100);//400 width and 500 height

f.setLayout(null);//using no layout managers

f.setVisible(true);//making the frame visible

f.setLocationRelativeTo(null);

}

Next, in this article on Library Management System Project in Java, let us discuss the code for Admin Menu function.

#### ****Admin Menu****

The Admin Menu is designed to show details of users, books, issued books, add books, return books, add user, and create or reset the database.

/\*

Next, in this article on Library Management System Project in Java,

let us discuss the code for Admin Menu function.

Admin Menu

The Admin Menu is designed to show details of users, books, issued books,

add books, return books, add user, and create or reset the database.

\*/

public static void admin\_menu() {

JFrame f=new JFrame("Admin Functions"); //Give dialog box name as admin functions

//f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //

JButton create\_but=new JButton("Create/Reset");//creating instance of JButton to create or reset database

create\_but.setBounds(450,60,120,25);//x axis, y axis, width, height

create\_but.addActionListener(new ActionListener() { //Perform action

public void actionPerformed(ActionEvent e){

create(); //Call create function

JOptionPane.showMessageDialog(null,"Database Created/Reset!"); //Open a dialog box and display the message

}

});

JButton view\_but=new JButton("View Books");//creating instance of JButton to view books

view\_but.setBounds(20,20,120,25);//x axis, y axis, width, height

view\_but.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

JFrame f = new JFrame("Books Available");

//f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Connection connection = connect(); //connect to database

String sql="select \* from BOOKS"; //select all books

try {

Statement stmt = connection.createStatement();

stmt.executeUpdate("USE LIBRARY"); //use database

stmt=connection.createStatement();

ResultSet rs=stmt.executeQuery(sql);

JTable book\_list= new JTable(); //view data in table format

book\_list.setModel(DbUtils.resultSetToTableModel(rs));

//mention scroll bar

JScrollPane scrollPane = new JScrollPane(book\_list);

f.add(scrollPane); //add scrollpane

f.setSize(800, 400); //set size for frame

f.setVisible(true);

f.setLocationRelativeTo(null);

} catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

}

);

JButton users\_but=new JButton("View Users");//creating instance of JButton to view users

users\_but.setBounds(150,20,120,25);//x axis, y axis, width, height

users\_but.addActionListener(new ActionListener() { //Perform action on click button

public void actionPerformed(ActionEvent e){

JFrame f = new JFrame("Users List");

//f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Connection connection = connect();

String sql="select \* from users"; //retrieve all users

try {

Statement stmt = connection.createStatement();

stmt.executeUpdate("USE LIBRARY"); //use database

stmt=connection.createStatement();

ResultSet rs=stmt.executeQuery(sql);

JTable book\_list= new JTable();

book\_list.setModel(DbUtils.resultSetToTableModel(rs));

//mention scroll bar

JScrollPane scrollPane = new JScrollPane(book\_list);

f.add(scrollPane); //add scrollpane

f.setSize(800, 400); //set size for frame

f.setVisible(true);

f.setLocationRelativeTo(null);

} catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

}

);

JButton issued\_but=new JButton("View Issued Books");//creating instance of JButton to view the issued books

issued\_but.setBounds(280,20,160,25);//x axis, y axis, width, height

issued\_but.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

JFrame f = new JFrame("Users List");

//f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Connection connection = connect();

String sql="select \* from issued";

try {

Statement stmt = connection.createStatement();

stmt.executeUpdate("USE LIBRARY");

stmt=connection.createStatement();

ResultSet rs=stmt.executeQuery(sql);

JTable book\_list= new JTable();

book\_list.setModel(DbUtils.resultSetToTableModel(rs));

JScrollPane scrollPane = new JScrollPane(book\_list);

f.add(scrollPane);

f.setSize(800, 400);

f.setVisible(true);

f.setLocationRelativeTo(null);

} catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

}

);

JButton add\_user=new JButton("Add User"); //creating instance of JButton to add users

add\_user.setBounds(20,60,120,25); //set dimensions for button

add\_user.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

JFrame g = new JFrame("Enter User Details"); //Frame to enter user details

//g.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//Create label

JLabel l1,l2;

l1=new JLabel("Username"); //label 1 for username

l1.setBounds(30,15, 100,30);

l2=new JLabel("Password"); //label 2 for password

l2.setBounds(30,50, 100,30);

//set text field for username

JTextField F\_user = new JTextField();

F\_user.setBounds(110, 15, 200, 30);

//set text field for password

JPasswordField F\_pass=new JPasswordField();

F\_pass.setBounds(110, 50, 200, 30);

//set radio button for admin

JRadioButton a1 = new JRadioButton("Admin");

a1.setBounds(55, 80, 200,30);

//set radio button for user

JRadioButton a2 = new JRadioButton("User");

a2.setBounds(130, 80, 200,30);

//add radio buttons

ButtonGroup bg=new ButtonGroup();

bg.add(a1);bg.add(a2);

JButton create\_but=new JButton("Create");//creating instance of JButton for Create

create\_but.setBounds(130,130,80,25);//x axis, y axis, width, height

create\_but.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

String username = F\_user.getText();

String password = F\_pass.getText();

Boolean admin = false;

if(a1.isSelected()) {

admin=true;

}

Connection connection = connect();

try {

Statement stmt = connection.createStatement();

stmt.executeUpdate("USE LIBRARY");

stmt.executeUpdate("INSERT INTO USERS(USERNAME,PASSWORD,ADMIN) VALUES ('"+username+"','"+password+"',"+admin+")");

JOptionPane.showMessageDialog(null,"User added!");

g.dispose();

}

catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

});

g.add(create\_but);

g.add(a2);

g.add(a1);

g.add(l1);

g.add(l2);

g.add(F\_user);

g.add(F\_pass);

g.setSize(350,200);//400 width and 500 height

g.setLayout(null);//using no layout managers

g.setVisible(true);//making the frame visible

g.setLocationRelativeTo(null);

}

});

JButton add\_book=new JButton("Add Book"); //creating instance of JButton for adding books

add\_book.setBounds(150,60,120,25);

add\_book.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

//set frame wot enter book details

JFrame g = new JFrame("Enter Book Details");

//g.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// set labels

JLabel l1,l2,l3;

l1=new JLabel("Book Name"); //lebel 1 for book name

l1.setBounds(30,15, 100,30);

l2=new JLabel("Genre"); //label 2 for genre

l2.setBounds(30,53, 100,30);

l3=new JLabel("Price"); //label 2 for price

l3.setBounds(30,90, 100,30);

//set text field for book name

JTextField F\_bname = new JTextField();

F\_bname.setBounds(110, 15, 200, 30);

//set text field for genre

JTextField F\_genre=new JTextField();

F\_genre.setBounds(110, 53, 200, 30);

//set text field for price

JTextField F\_price=new JTextField();

F\_price.setBounds(110, 90, 200, 30);

JButton create\_but=new JButton("Submit");//creating instance of JButton to submit details

create\_but.setBounds(130,130,80,25);//x axis, y axis, width, height

create\_but.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

// assign the book name, genre, price

String bname = F\_bname.getText();

String genre = F\_genre.getText();

String price = F\_price.getText();

//convert price of integer to int

int price\_int = Integer.parseInt(price);

Connection connection = connect();

try {

Statement stmt = connection.createStatement();

stmt.executeUpdate("USE LIBRARY");

stmt.executeUpdate("INSERT INTO BOOKS(BNAME,GENRE,PRICE) VALUES ('"+bname+"','"+genre+"',"+price\_int+")");

JOptionPane.showMessageDialog(null,"Book added!");

g.dispose();

}

catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

});

g.add(l3);

g.add(create\_but);

g.add(l1);

g.add(l2);

g.add(F\_bname);

g.add(F\_genre);

g.add(F\_price);

g.setSize(350,200);//400 width and 500 height

g.setLayout(null);//using no layout managers

g.setVisible(true);//making the frame visible

g.setLocationRelativeTo(null);

}

});

JButton issue\_book=new JButton("Issue Book"); //creating instance of JButton to issue books

issue\_book.setBounds(450,20,120,25);

issue\_book.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

//enter details

JFrame g = new JFrame("Enter Details");

//g.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//create labels

JLabel l1,l2,l3,l4;

l1=new JLabel("Book ID(BID)"); // Label 1 for Book ID

l1.setBounds(30,15, 100,30);

l2=new JLabel("User ID(UID)"); //Label 2 for user ID

l2.setBounds(30,53, 100,30);

l3=new JLabel("Period(days)"); //Label 3 for period

l3.setBounds(30,90, 100,30);

l4=new JLabel("Issued Date(DD-MM-YYYY)"); //Label 4 for issue date

l4.setBounds(30,127, 150,30);

JTextField F\_bid = new JTextField();

F\_bid.setBounds(110, 15, 200, 30);

JTextField F\_uid=new JTextField();

F\_uid.setBounds(110, 53, 200, 30);

JTextField F\_period=new JTextField();

F\_period.setBounds(110, 90, 200, 30);

JTextField F\_issue=new JTextField();

F\_issue.setBounds(180, 130, 130, 30);

JButton create\_but=new JButton("Submit");//creating instance of JButton

create\_but.setBounds(130,170,80,25);//x axis, y axis, width, height

create\_but.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

String uid = F\_uid.getText();

String bid = F\_bid.getText();

String period = F\_period.getText();

String issued\_date = F\_issue.getText();

int period\_int = Integer.parseInt(period);

Connection connection = connect();

try {

Statement stmt = connection.createStatement();

stmt.executeUpdate("USE LIBRARY");

stmt.executeUpdate("INSERT INTO ISSUED(UID,BID,ISSUED\_DATE,PERIOD) VALUES ('"+uid+"','"+bid+"','"+issued\_date+"',"+period\_int+")");

JOptionPane.showMessageDialog(null,"Book Issued!");

g.dispose();

}

catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

});

g.add(l3);

g.add(l4);

g.add(create\_but);

g.add(l1);

g.add(l2);

g.add(F\_uid);

g.add(F\_bid);

g.add(F\_period);

g.add(F\_issue);

g.setSize(350,250);//400 width and 500 height

g.setLayout(null);//using no layout managers

g.setVisible(true);//making the frame visible

g.setLocationRelativeTo(null);

}

});

JButton return\_book=new JButton("Return Book"); //creating instance of JButton to return books

return\_book.setBounds(280,60,160,25);

return\_book.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

JFrame g = new JFrame("Enter Details");

//g.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//set labels

JLabel l1,l2,l3,l4;

l1=new JLabel("Issue ID(IID)"); //Label 1 for Issue ID

l1.setBounds(30,15, 100,30);

l4=new JLabel("Return Date(DD-MM-YYYY)");

l4.setBounds(30,50, 150,30);

JTextField F\_iid = new JTextField();

F\_iid.setBounds(110, 15, 200, 30);

JTextField F\_return=new JTextField();

F\_return.setBounds(180, 50, 130, 30);

JButton create\_but=new JButton("Return");//creating instance of JButton to mention return date and calculcate fine

create\_but.setBounds(130,170,80,25);//x axis, y axis, width, height

create\_but.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e){

String iid = F\_iid.getText();

String return\_date = F\_return.getText();

Connection connection = connect();

try {

Statement stmt = connection.createStatement();

stmt.executeUpdate("USE LIBRARY");

//Intialize date1 with NULL value

String date1=null;

String date2=return\_date; //Intialize date2 with return date

//select issue date

ResultSet rs = stmt.executeQuery("SELECT ISSUED\_DATE FROM ISSUED WHERE IID="+iid);

while (rs.next()) {

date1 = rs.getString(1);

}

try {

Date date\_1=new SimpleDateFormat("dd-MM-yyyy").parse(date1);

Date date\_2=new SimpleDateFormat("dd-MM-yyyy").parse(date2);

//subtract the dates and store in diff

long diff = date\_2.getTime() - date\_1.getTime();

//Convert diff from milliseconds to days

ex.days=(int)(TimeUnit.DAYS.convert(diff, TimeUnit.MILLISECONDS));

} catch (ParseException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

//update return date

stmt.executeUpdate("UPDATE ISSUED SET RETURN\_DATE='"+return\_date+"' WHERE IID="+iid);

g.dispose();

Connection connection1 = connect();

Statement stmt1 = connection1.createStatement();

stmt1.executeUpdate("USE LIBRARY");

ResultSet rs1 = stmt1.executeQuery("SELECT PERIOD FROM ISSUED WHERE IID="+iid); //set period

String diff=null;

while (rs1.next()) {

diff = rs1.getString(1);

}

int diff\_int = Integer.parseInt(diff);

if(ex.days&amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;gt;diff\_int) { //If number of days are more than the period then calculcate fine

//System.out.println(ex.days);

int fine = (ex.days-diff\_int)\*10; //fine for every day after the period is Rs 10.

//update fine in the system

stmt1.executeUpdate("UPDATE ISSUED SET FINE="+fine+" WHERE IID="+iid);

String fine\_str = ("Fine: Rs. "+fine);

JOptionPane.showMessageDialog(null,fine\_str);

}

JOptionPane.showMessageDialog(null,"Book Returned!");

}

catch (SQLException e1) {

// TODO Auto-generated catch block

JOptionPane.showMessageDialog(null, e1);

}

}

});

g.add(l4);

g.add(create\_but);

g.add(l1);

g.add(F\_iid);

g.add(F\_return);

g.setSize(350,250);//400 width and 500 height

g.setLayout(null);//using no layout managers

g.setVisible(true);//making the frame visible

g.setLocationRelativeTo(null);

}

});

f.add(create\_but);

f.add(return\_book);

f.add(issue\_book);

f.add(add\_book);

f.add(issued\_but);

f.add(users\_but);

f.add(view\_but);

f.add(add\_user);

f.setSize(600,200);//400 width and 500 height

f.setLayout(null);//using no layout managers

f.setVisible(true);//making the frame visible

f.setLocationRelativeTo(null);

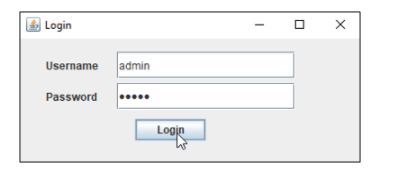
}

}

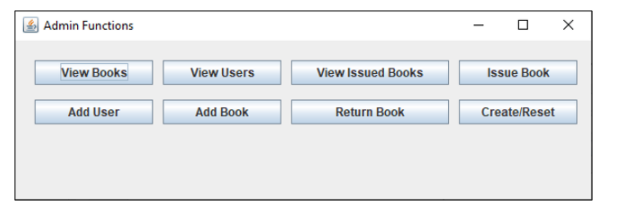
Now that you have understood all the functions, let us execute our library management system project in Java and see the outputs.

### ****Output:****

Execute the application by clicking on the run button. Once, you execute you will see the below dialog box. In the below dialog box, mention username and password as {admin, admin}. Then click on the Login button.



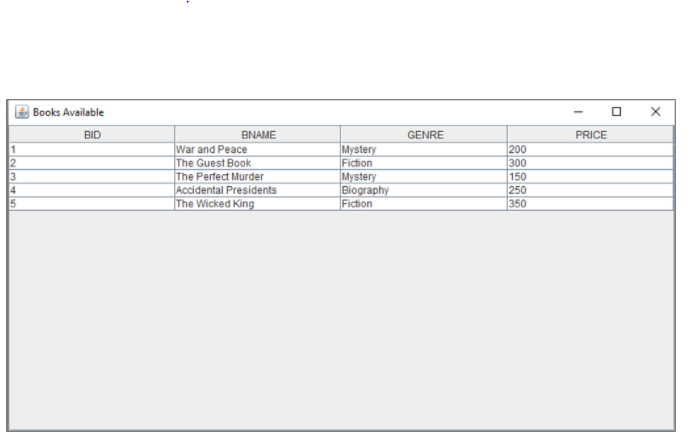
Once you click on the **Login button**, you will see the below dialog box opening up.



Here you have various options which you can explore. So, let us start with the first one:

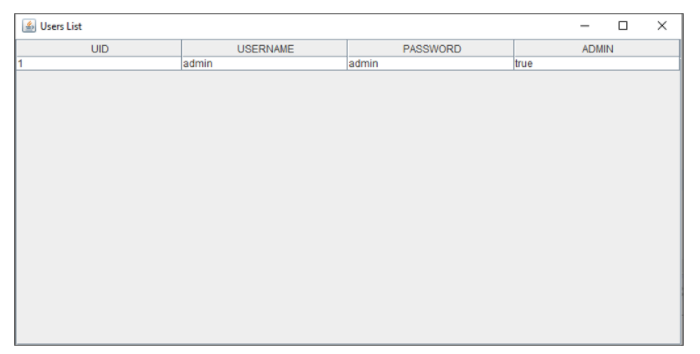
#### ****View Books****

Once, you click on View Books button, you will see the below frame displaying all the books present in the database, with their details.



#### ****View Users****

The View Users button is used to view the current users on the system. Since we just have only one user present i.e the admin, it will show you output as below:



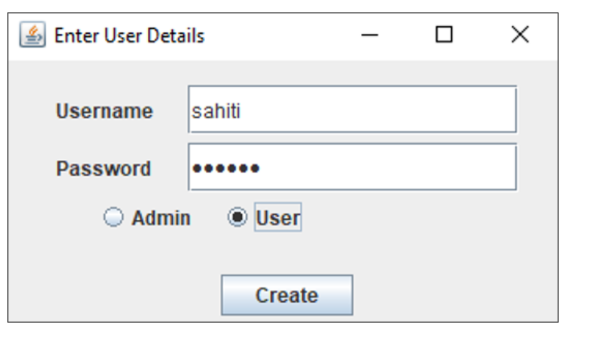
#### ****Create/Reset****

This functionality is used to create or reset a database. So, once you click on the button Create/Rest, you will see the below output:

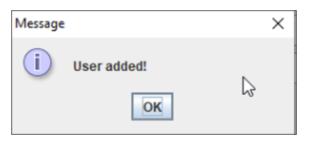


#### ****Add User****

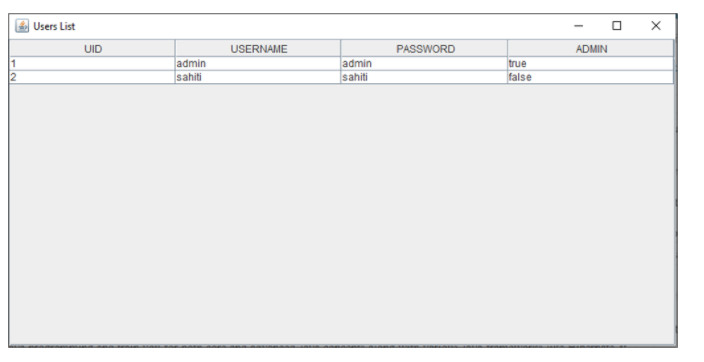
To add a user, click on the option “**Add User**” and mention details such as **username, password and choose the radio button user or admin**. By default, it will be the user. Then, click on **Create**.



Once the user is created, you will see an output as below:



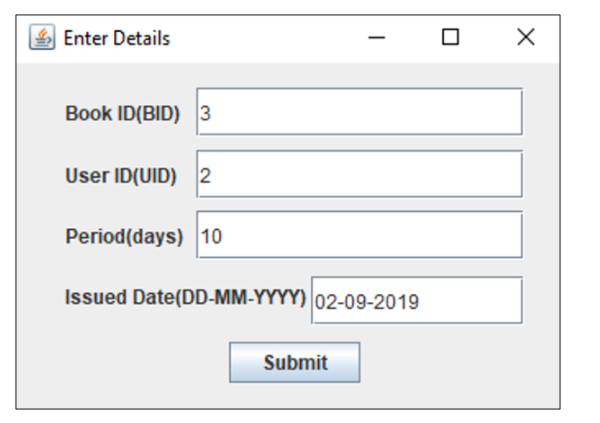
Now, again if you click on **View Users button**, you will see the below output:

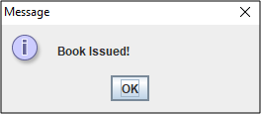


Alright, so now that we have added a user. Let us say, that particular user wants to issue books. To do that, the user has to choose the option of Issue Book.

#### ****Issue Book****

Suppose, if you are the user, once you click on the**Issue Book button**, you have to mention the **Book ID, User ID, Period(Number of days for issuing the book)**, and the **Issue Date** as follows:

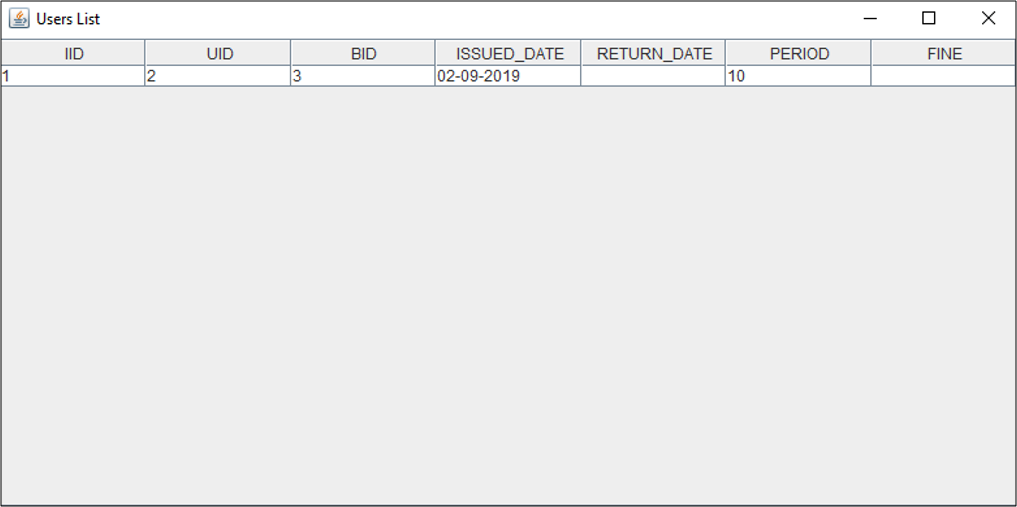


Then click on **Submit**. Once, you click on **Submit**, you will see the below dialog box: 

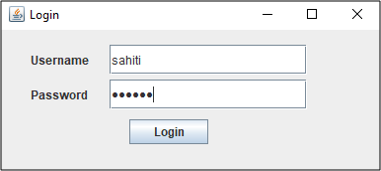
Now, if you want to see the issued books details, you can use the View Issued Books functionality.

#### ****View Issued Books****

Once you click on this button, you will see the following output:



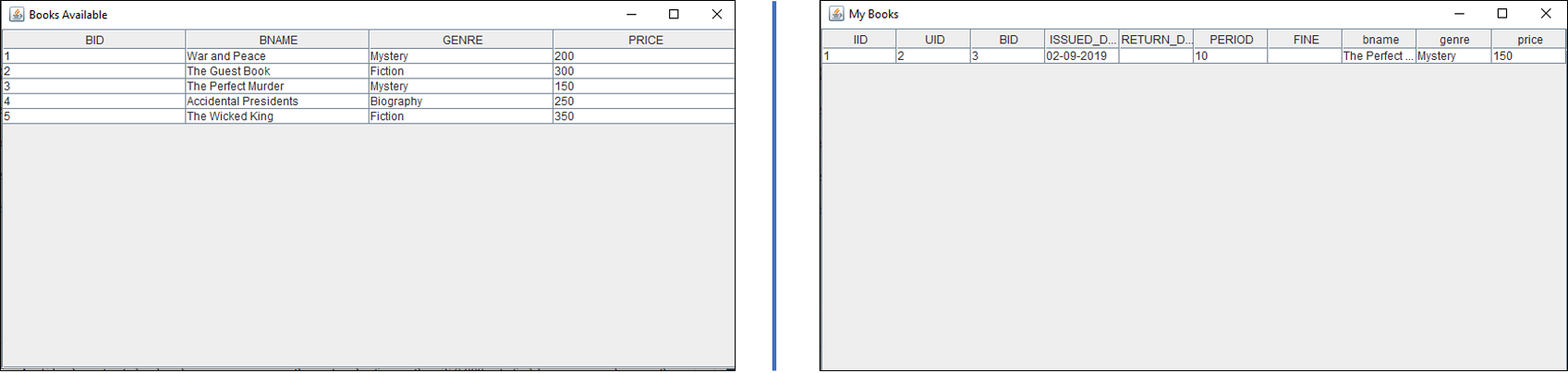
Alright, so, now **if the user logs in to the system**, using the login function, as below:



Then the user will see the below User Menu.



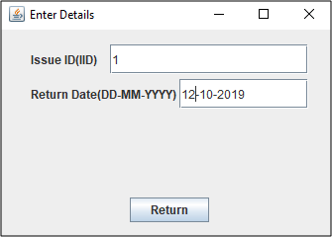
Here, the user can **view all the books** in the database by **using the View Books option** and the **books issued by the user** in the **My Books section** as below:



Now, if you wish to return the book, then you have to choose the option of Return Book.

#### ****Return Book****

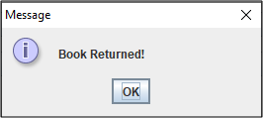
Once, you click on the Return Book, mention the **Issue ID and the return date** as below. Then click on**Return**.



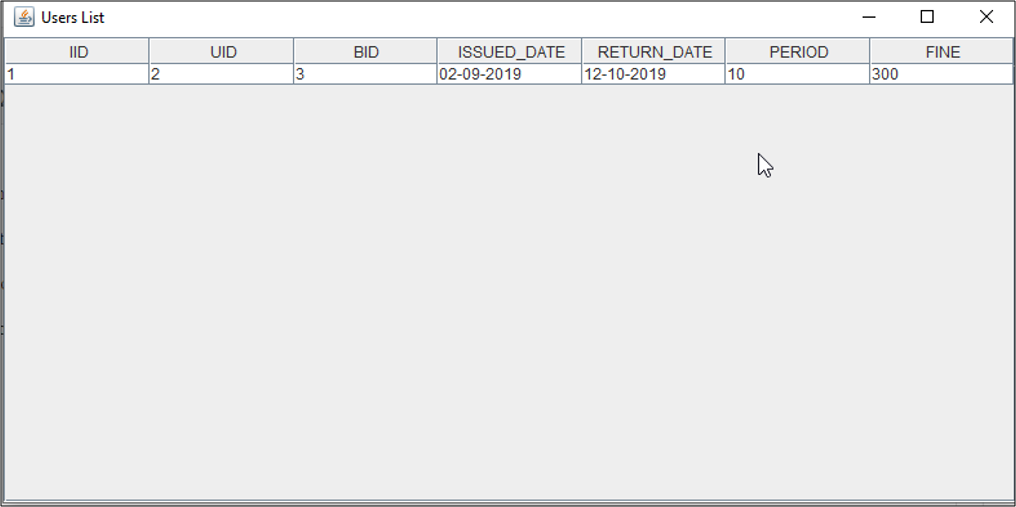
Then, you see a message box displaying the fine.



After that, you again see a dialog box, showing the message “**Book Returned**“. Refer below.



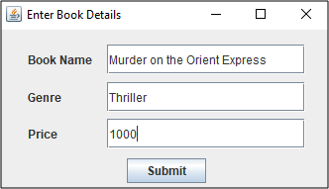
Now, if you click on the **View Issued Books**, you will see the below output:



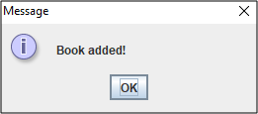
Lastly, if you wish to add a book, you can use the option of Add Book.

#### ****Add Book****

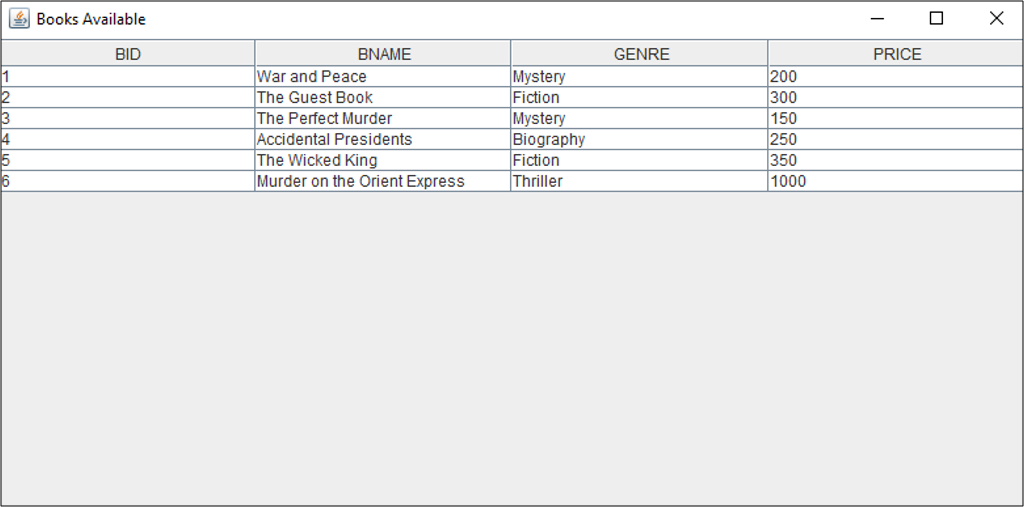
Click on the **Add Book button**, and mention the **book name, genre and price**. Then, click on the **Submit button**. Refer below.



You will see a dialog box displaying the below message:



Apart from this, you can also, see the added books in the **View Books** section as below:



This brings us to the end of our article on Library Management System Project in Java. I hope you found this article informative and added value to your knowledge.