RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT

Course Code	CSE302A
Course Name	Network Programming and Simulation
Programme	B.tech
Department	CSE
Faculty	FET

Name of the Student	Sudhanshu shekhar
Reg. No	17ETCS002213
Semester/Year	6th/3rd
Course Leader/s	Dr. Rinki Sharma

Declaration Sheet			
Student Name	SUDHANSHU SHEKHAR		
Reg. No	17ETCS002213		
Programme	B.tech	Semester/Year	6th/3rd
Course Code	CSE302A		
Course Title	Network Programming and Simulation		
Course Date		to	
Course Leader	Dr. Rinki Sharma		

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Question 1

1.1 Java Function Calls for Socket Programming

Socket programming is a way of connecting two nodes on a network to communicate with each other. One *socket* (node) listens on a particular port at an IP, while other *socket* reaches out to the other in order to form a connection.

The server forms the listener *socket while* the client reaches out to the server. Socket and Server Socket classes are used for connection-oriented socket programming.

A socket in Java is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

There are several ports of socket programming in java, various functions and calls can be used to establish connection between the client and server to do the required task. Few of the points are:

1) To connect to other machine we need a socket connection. A socket connection means the two machines have information about each other's network location (IP Address) and TCP port.The java.net.Socket class represents a Socket. To open a socket:

```
Socket socket = new Socket("127.0.0.1", 5000)
```

2) ServerSocket which waits for the client requests (when a client makes a new Socket())
3) A plain old Socket socket to use for communication with the client.
4) getOutputStream() method is used to send the output through the socket.
5) The accept() method blocks(just sits there) until a client connects to the server.

Then we take input from the socket using getInputStream() method. Our Server keeps receiving messages until the Client sends "Over".

1.2 Difference between Socket Programming using C and Java

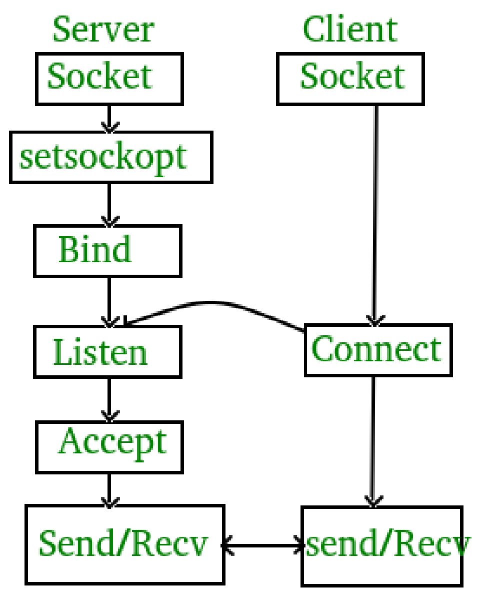
Java:

Socket programming boils down to two systems communicating with one another. Generally, network communication comes in two flavours: Transport Control Protocol (TCP) and User Datagram Protocol (UDP). TCP and UDP are used for different purposes and both have unique constraints:

- TCP is relatively simple and reliable protocol that enables a client to make a connection to a server and the two systems to communicate. In TCP, each entity knows that its communication payloads have been received.
- UDP is a *connectionless protocol* and is good for scenarios where you do not necessarily need every packet to arrive at its destination, such as media streaming.

C:

One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.



1.3 Conclusion

Execute the application by clicking on the run button.
Once you click on the **Login button**, you will see a dialog box opening up.
Here you have various options which you can explore. So, let us start with the first one:
Once, you click on View Books button, you will see the below frame displaying all the books present in the database, with their details.
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

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**.

- Suppose Server application makes a ServerSocket on a specific port which is 4123. This starts our Server listening for client requests coming in for port 4123.
- Then Server makes a new Socket to communicate with the client.

```
socket = server.accept()
```

- The accept() method blocks(just sits there) until a client connects to the server.
- Then we take input from the socket using getInputStream() method. Our Server keeps receiving messages until the Client sends "Over".
- After we're done we close the connection by closing the socket and the input stream.
- To run the Client and Server application on your machine, compile both of them. Then first run the server application and then run the Client application.

Question 2 -

1. Introduction

Here in this question we have to design the library management system using socket programming the client server method In a client/server architecture, the relationship of the computers are separated into two roles:

1 - The client, which requests specific services or resources

2 -The server, which is dedicated to fulfilling requests by responding (or attempting to respond) with requested services or resources

Here in this question, I have made a client server based library management system using socket programming in java. Here, also it is able to handle simultaneous request to issue books and once a book is reserved then the client is informed about the book details and also the date of issue and return.

2. Design specifications along with functional and non-functional requirements

Design specification along with the function and non function requirement are:

1. **Functional requirements**

   1-The system must let the new user to signup and new user to login.

   2-The system must be able to connect to the server and client and should be able to verify requests.

   3-The user should be able to borrow books from the Librarian.

   4-The system should be able to see the book requests from the server.

   5-The system must give the details all the available books for borrowing.

   6-The system must show the borrowed book ,date and return date in borrowed section

   7-The system must let the user to logout from the app

   8-The system must prepare the library database.

2.Non functional requirements

1-The system performance should be fast and accurate.

2-The system must handle unexpected and expected errors and should be able to handle large amount of data.

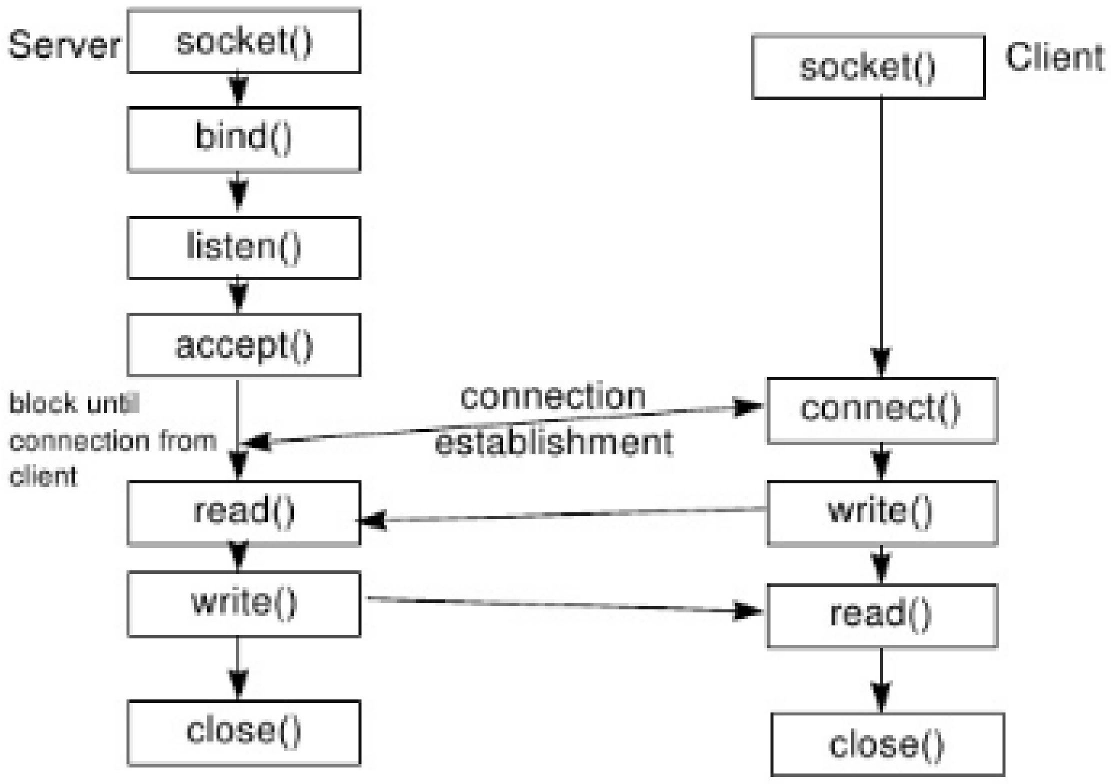
3-The system must be secure.

4-The system must be intuitive and fluid to use.

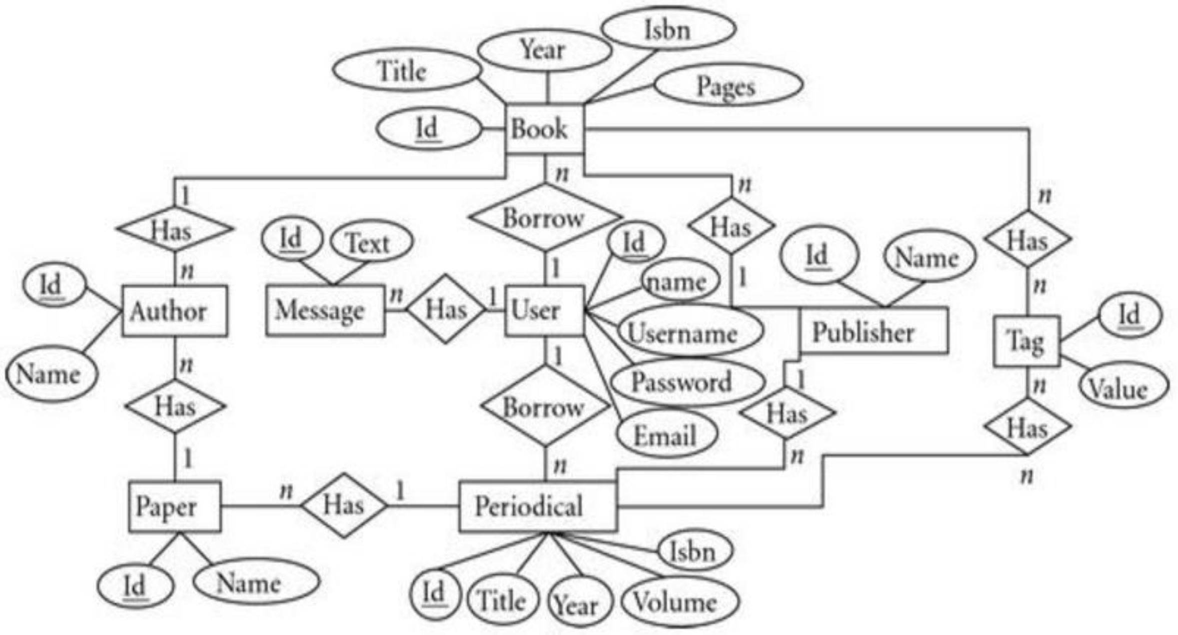
5-The system must allow User authentication and validation of members using their unique member ID. Proper accountability which include not allowing a member to see other members account. – Only administrator will see and manage all members account. – CAPTCHA words will be used for user login.

3. LMS design, highlighting the messages between client and server along with the database structure

Client server design using socket programming:

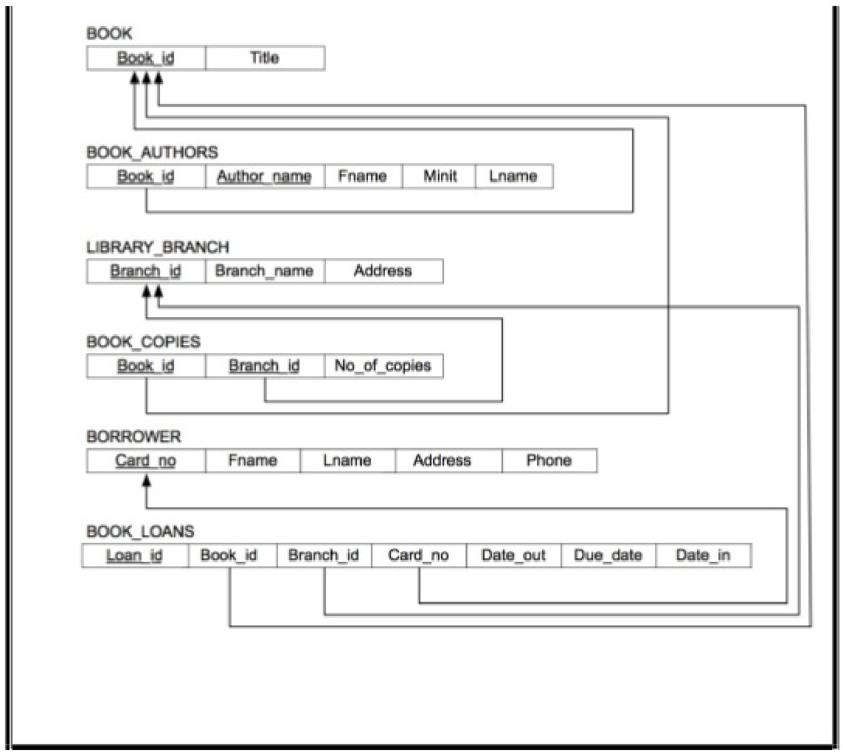


E-digram of LMS is as follows:

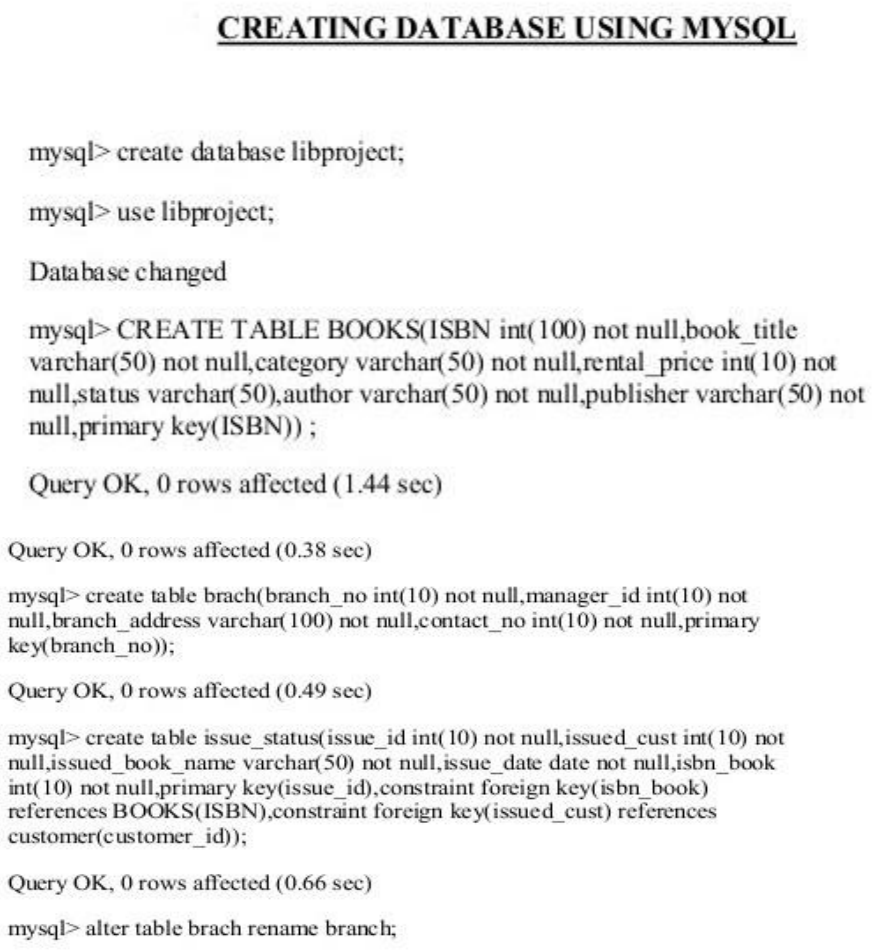


Schema digram

The schema digram of library management system is as followa:



We are using my sql for storing the data base the code for creating it is as follows:

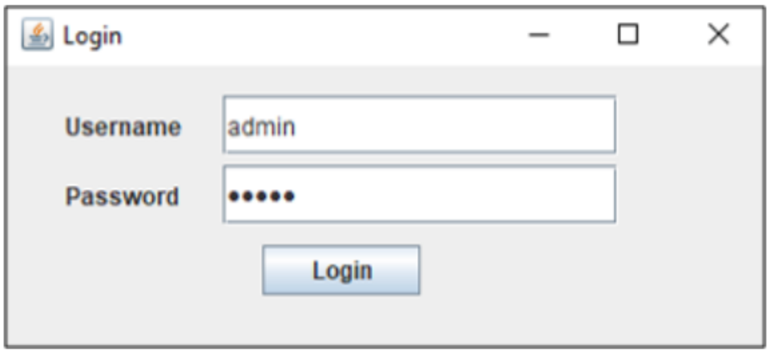


CREATING DATABASE USING MYSQL

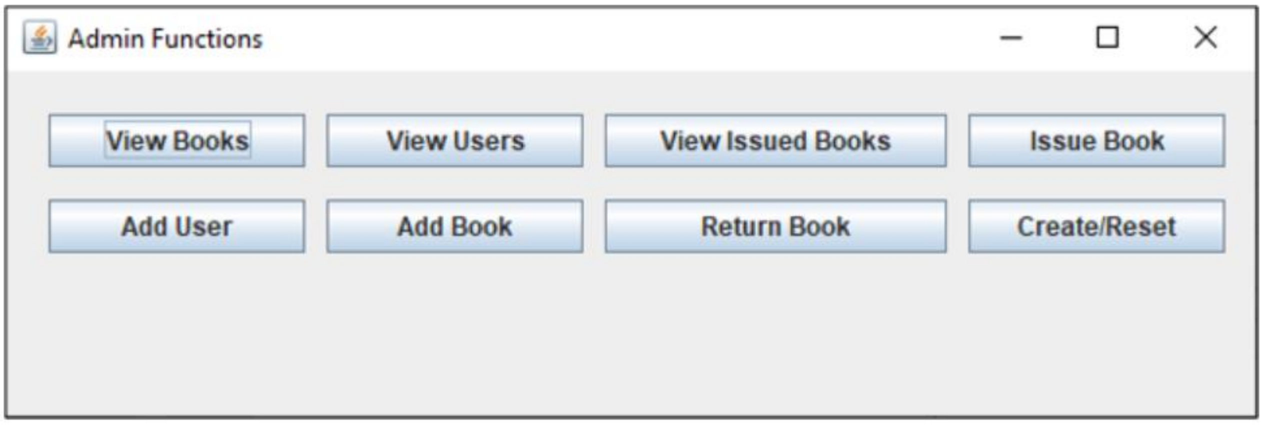
```
mysql> create database libproject;

mysql> use libproject;

Database changed

mysql> CREATE TABLE BOOKS(ISBN int(100) not null,book_title
varchar(50) not null,category varchar(50) not null,rental_price int(10) not
null,status varchar(50),author varchar(50) not null,publisher varchar(50) not
null,primary key(ISBN)) ;

Query OK, 0 rows affected (1.44 sec)

Query OK, 0 rows affected (0.38 sec)

mysql> create table brach(branch_no int(10) not null,manager_id int(10) not
null,branch_address varchar(100) not null,contact_no int(10) not null,primary
key(branch_no));

Query OK, 0 rows affected (0.49 sec)

mysql> create table issue_status(issue_id int(10) not null,issued_cust int(10) not
null,issued_book_name varchar(50) not null,issue_date date not null,isbn_book
int(10) not null,primary key(issue_id),constraint foreign key(isbn_book)
references BOOKS(ISBN),constraint foreign key(issued_cust) references
customer(customer_id));

Query OK, 0 rows affected (0.66 sec)

mysql> alter table brach rename branch;
```

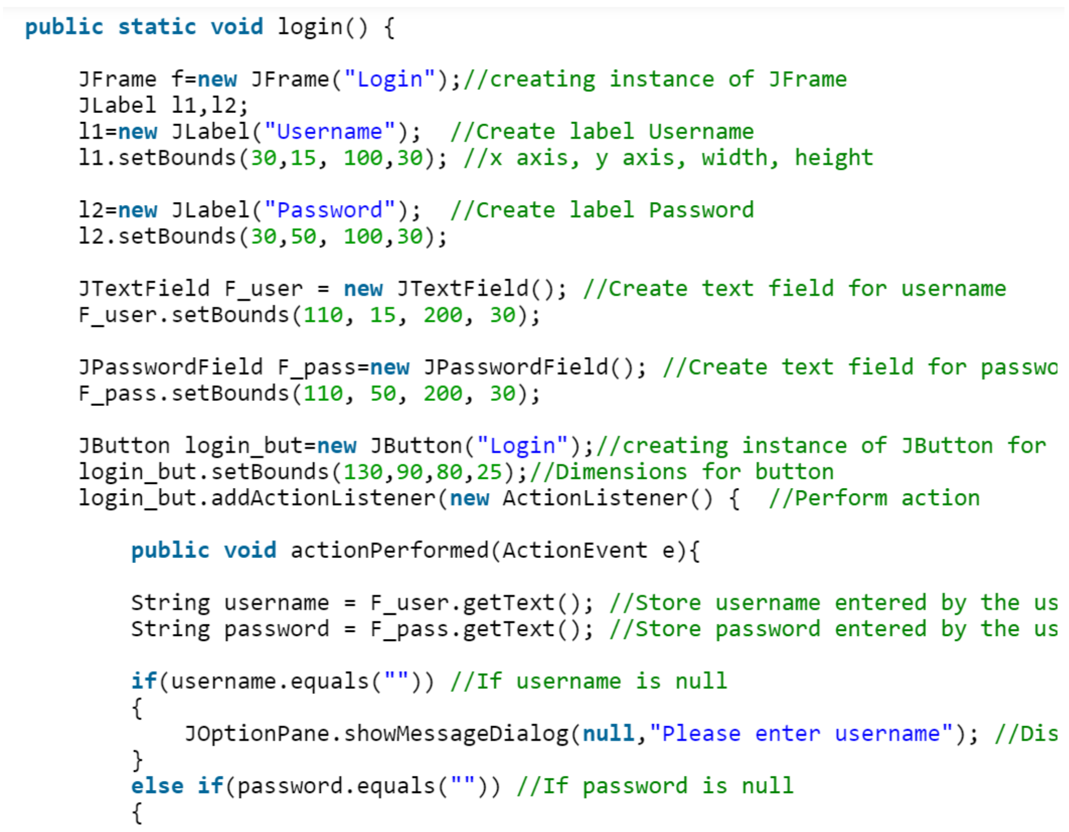
2.4 Implementation, documented using well-commented code snippets

```
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();
    }
```

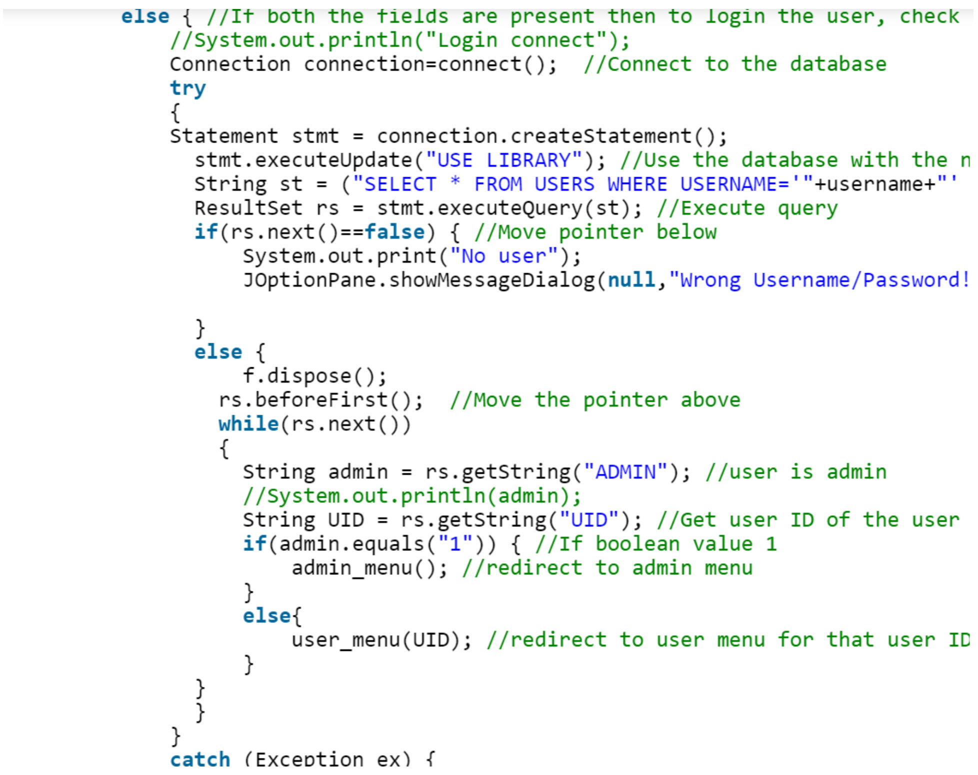


I have considered only one admin. So, once a user logs in as an admin, he or she will be redirected to the admin menu as below. I will discuss the functions of the admin in the admin menu section.



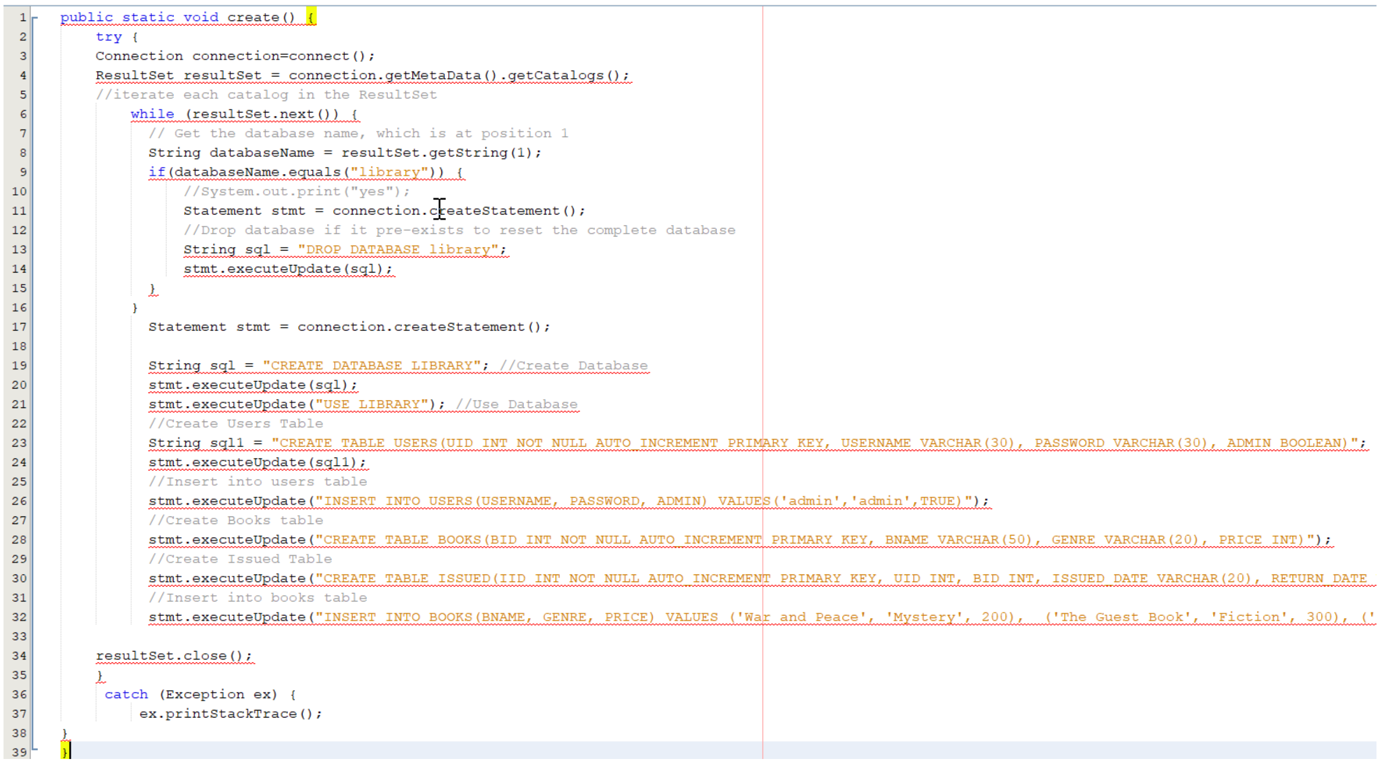


```
public static void login() {

    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 passwo
    F_pass.setBounds(110, 50, 200, 30);

    JButton login_but=new JButton("Login");//creating instance of JButton for
    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 us
        String password = F_pass.getText(); //Store password entered by the us

        if(username.equals("")) //If username is null
        {
            JOptionPane.showMessageDialog(null,"Please enter username"); //Dis
        }
        else if(password.equals("")) //If password is null
        {
```



```
        else { //If both the fields are present then to login the user, check
            //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 n
              String st = ("SELECT * FROM USERS WHERE USERNAME='"+username+"'
              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!

              }
              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) {
        }
        });


        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);

    }
```

In the above function, I am connecting my **MySQL database** with the **username “root”** and **password “sudhanshu”** to my application. Now, once the application is connected to the database, my next step is to create or reset the database.

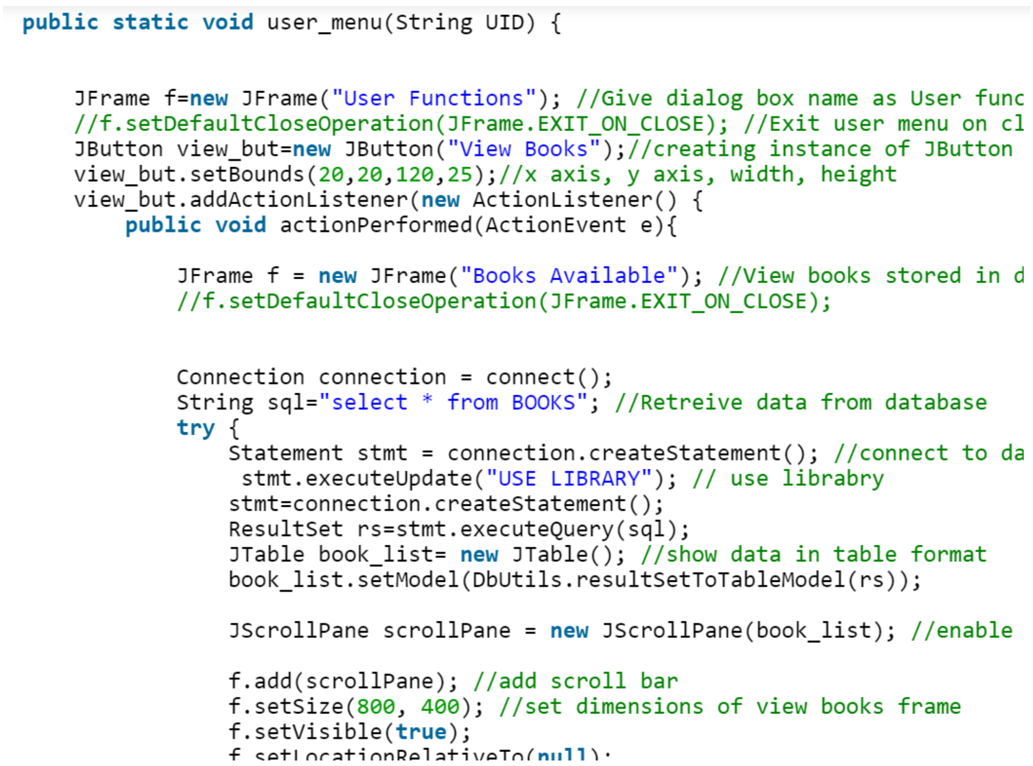


```
public static void create() {
    try {
    Connection connection=connect();
    ResultSet resultSet = connection.getMetaData().getCatalogs();
    //iterate each catalog in the ResultSet
        while (resultSet.next()) {
          // Get the database name, which is at position 1
          String 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
          //Insert into books table
          stmt.executeUpdate("INSERT INTO BOOKS(BNAME, GENRE, PRICE) VALUES ('War and Peace', 'Mystery', 200),  ('The Guest Book', 'Fiction', 300), ('

    resultSet.close();
    }
     catch (Exception ex) {
         ex.printStackTrace();
  }
}
```

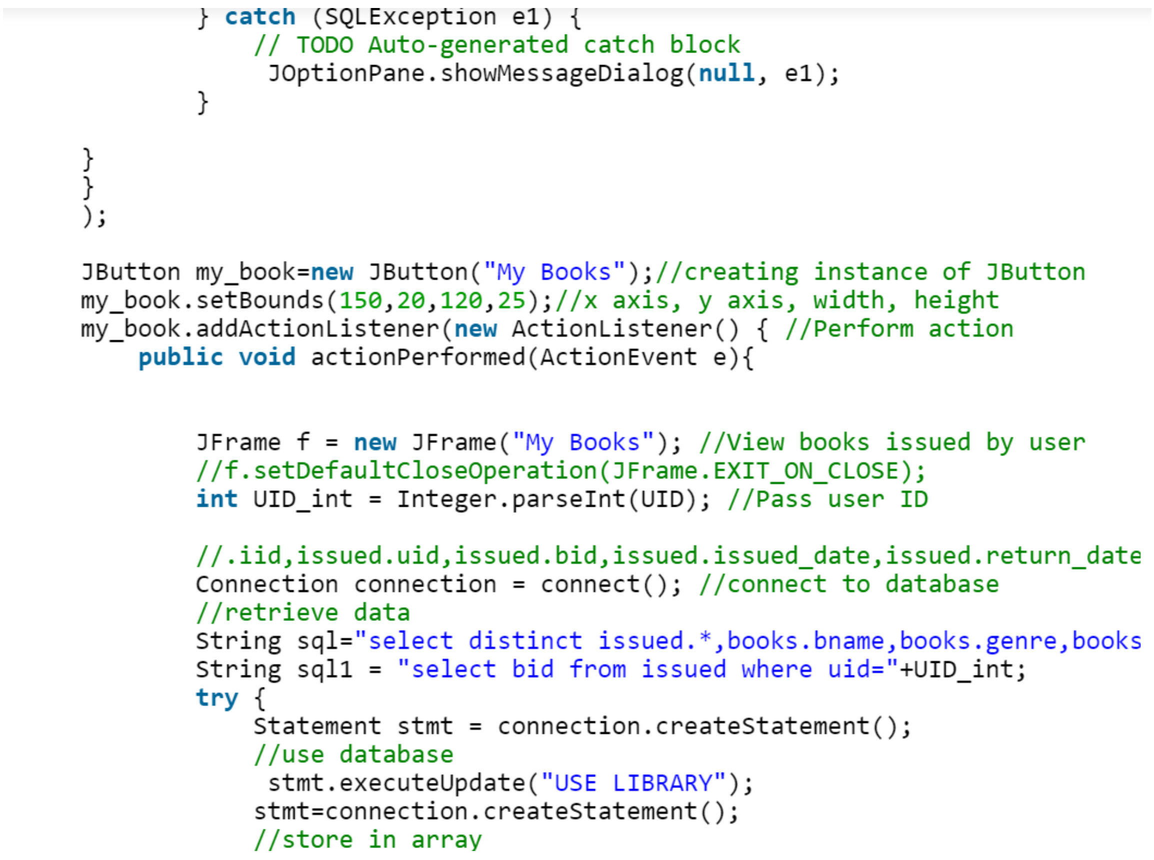
Now, that I have created the database, connected with GUI and enables the login function, next in this Library Management System Project in Java, now the functions of the User Menu.

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.



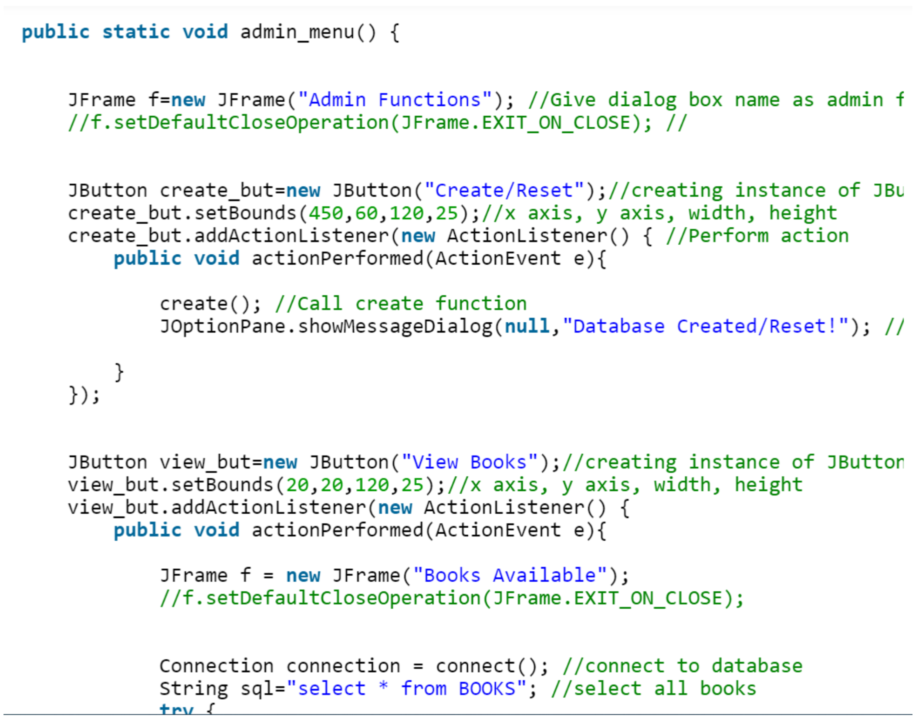
```
public static void user_menu(String UID) {


    JFrame f=new JFrame("User Functions"); //Give dialog box name as User func
    //f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Exit user menu on cl
    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 d
            //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 da
                 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

                f.add(scrollPane); //add scroll bar
                f.setSize(800, 400); //set dimensions of view books frame
                f.setVisible(true);
                f.setLocationRelativeTo(null);
```



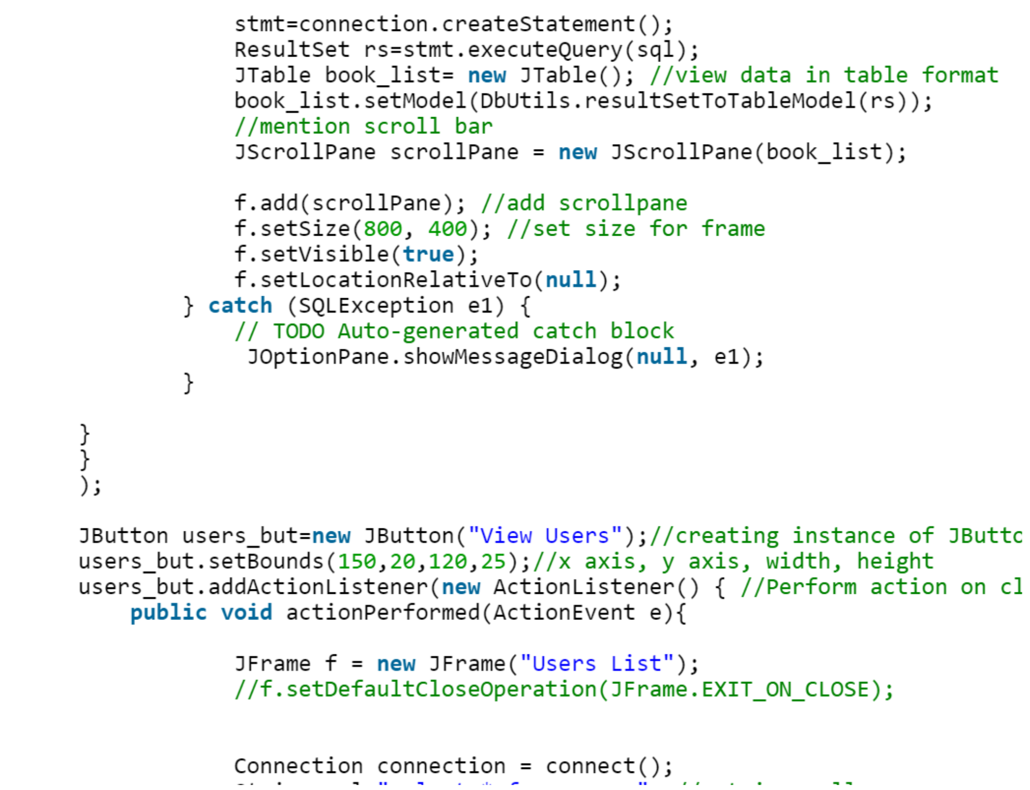
```
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                 JOptionPane.showMessageDialog(null, e1);
            }

    }
    }
    );

    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
            Connection connection = connect(); //connect to database
            //retrieve data
            String sql="select distinct issued.*,books.bname,books.genre,books
            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
```

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.



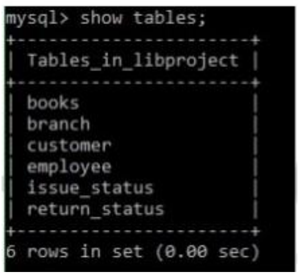
```
public static void admin_menu() {


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


    JButton create_but=new JButton("Create/Reset");//creating instance of JBu
    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!"); //

        }
    });


    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");
            //f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


            Connection connection = connect(); //connect to database
            String sql="select * from BOOKS"; //select all books
            try {
```



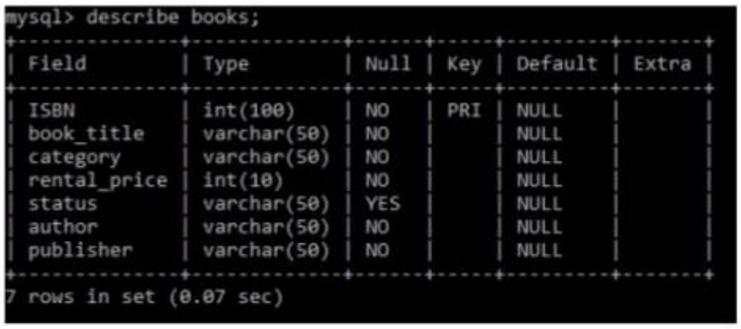
```
                stmt=connection.createStatement();
                ResultSet rs=stmt.executeQuery(sql);
                JTable book_list= new JTable(); //view data in table format
                book_list.setModel(DbUtils.resultSetToTableModel(rs));
                //mention scroll bar
                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 JButtc
    users_but.setBounds(150,20,120,25);//x axis, y axis, width, height
    users_but.addActionListener(new ActionListener() { //Perform action on cl
        public void actionPerformed(ActionEvent e){

                JFrame f = new JFrame("Users List");
                //f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


                Connection connection = connect();
                [illegible]
```

Now talking about the database in mysql. We can make the folllowing tables and store the values here too.

```
Query OK, 0 rows affected (0.50 sec)

mysql> create table return_status(return_id int(10) not null,return_cust int(10)
not null,returned_book_name varchar(50) not null,return_date date not
null,isbn_book2 int(10) not null,primary key(return_id),constraint foreign
key(isbn_book2) references BOOKS(ISBN),constraint foreign key(return_cust)
references issue_status(issued_cust));

Query OK, 0 rows affected (0.39 sec)
```

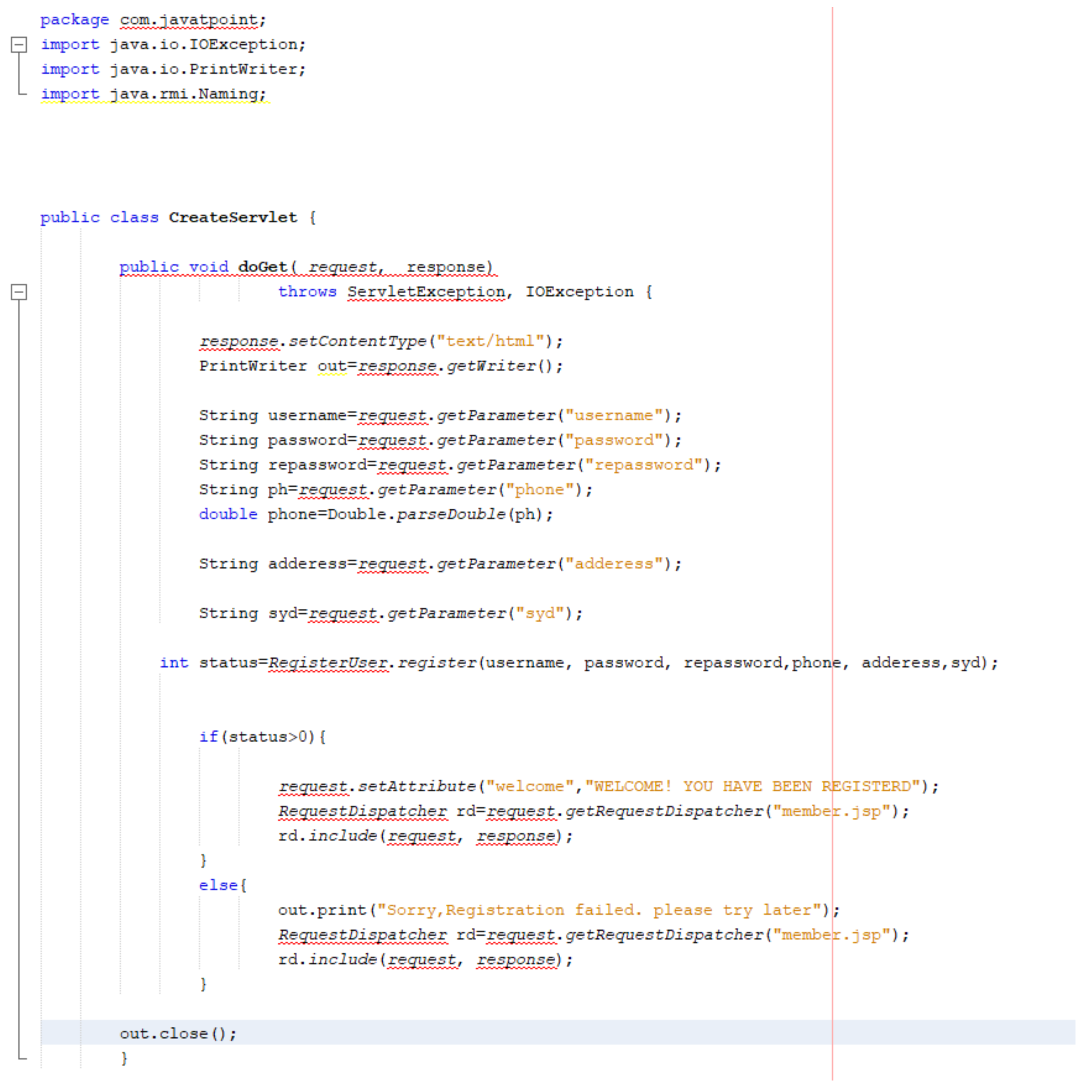


```
mysql> show tables;
+----------------------+
| Tables_in_libproject |
+----------------------+
| books                |
| branch               |
| customer             |
| employee             |
| issue_status         |
| return_status        |
+----------------------+
6 rows in set (0.00 sec)
```



```
mysql> describe books;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ISBN         | int(100)    | NO   | PRI | NULL    |       |
| book_title   | varchar(50) | NO   |     | NULL    |       |
| category     | varchar(50) | NO   |     | NULL    |       |
| rental_price | int(10)     | NO   |     | NULL    |       |
| status       | varchar(50) | YES  |     | NULL    |       |
| author       | varchar(50) | NO   |     | NULL    |       |
| publisher    | varchar(50) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
7 rows in set (0.07 sec)
```

Server:-



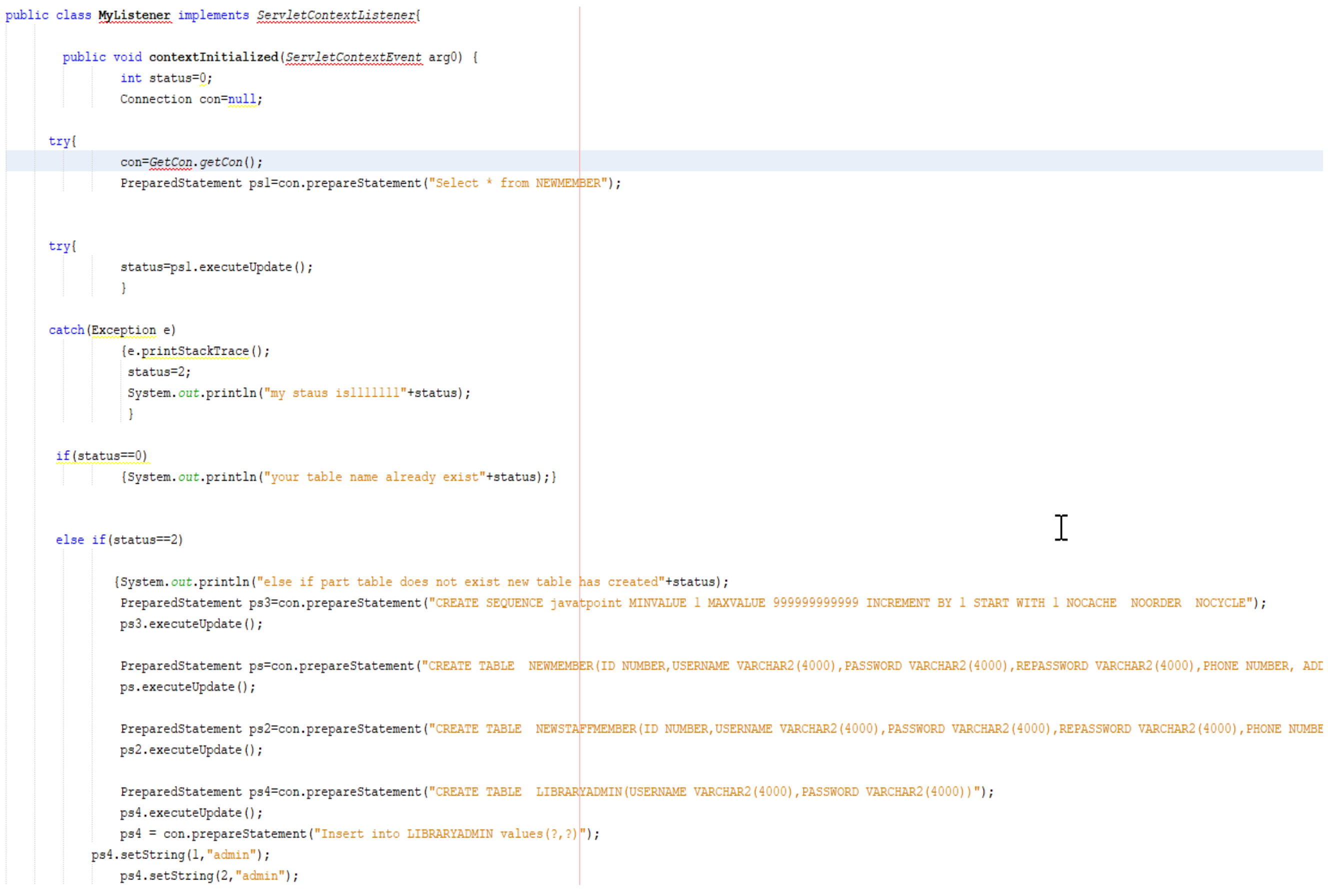
```
package com.javatpoint;
import java.io.IOException;
import java.io.PrintWriter;
import java.rmi.Naming;



public class CreateServlet {

        public void doGet( request,  response)
                throws ServletException, IOException {

            response.setContentType("text/html");
            PrintWriter out=response.getWriter();

            String username=request.getParameter("username");
            String password=request.getParameter("password");
            String repassword=request.getParameter("repassword");
            String ph=request.getParameter("phone");
            double phone=Double.parseDouble(ph);

            String adderess=request.getParameter("adderess");

            String syd=request.getParameter("syd");

        int status=RegisterUser.register(username, password, repassword,phone, adderess,syd);


            if(status>0){

                request.setAttribute("welcome","WELCOME! YOU HAVE BEEN REGISTERD");
                RequestDispatcher rd=request.getRequestDispatcher("member.jsp");
                rd.include(request, response);
            }
            else{
                out.print("Sorry,Registration failed. please try later");
                RequestDispatcher rd=request.getRequestDispatcher("member.jsp");
                rd.include(request, response);
            }

        out.close();
        }
```

```
import java.rmi.*;

public interface Details extends Remote
{
	public int open(String username,String password,double amount,String adderess,double phone) throws RemoteException;
	public String withdraw(int acno,String uname,String pwd,int amt) throws RemoteException;
	public String deposit(int acno,String uname,String pwd,int amt) throws RemoteException;
	public String transfer(int acno,String uname,String pwd,int tacno,int amt) throws RemoteException;
	public String close(int acno,String uname,String pass)	throws RemoteException;
	public String balance(int acno,String uname,String pass) throws RemoteException;
```

}

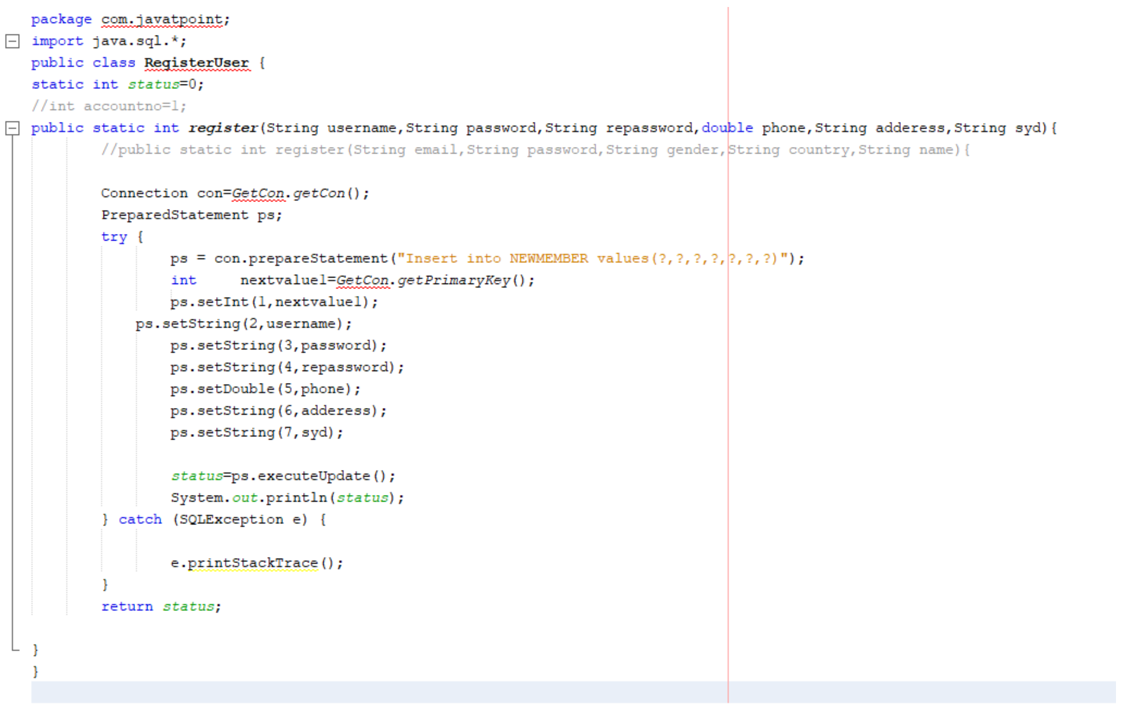
The above program GBANK IS THE INTERFACE FOR THE MAIN PROGRAM

Listener:-



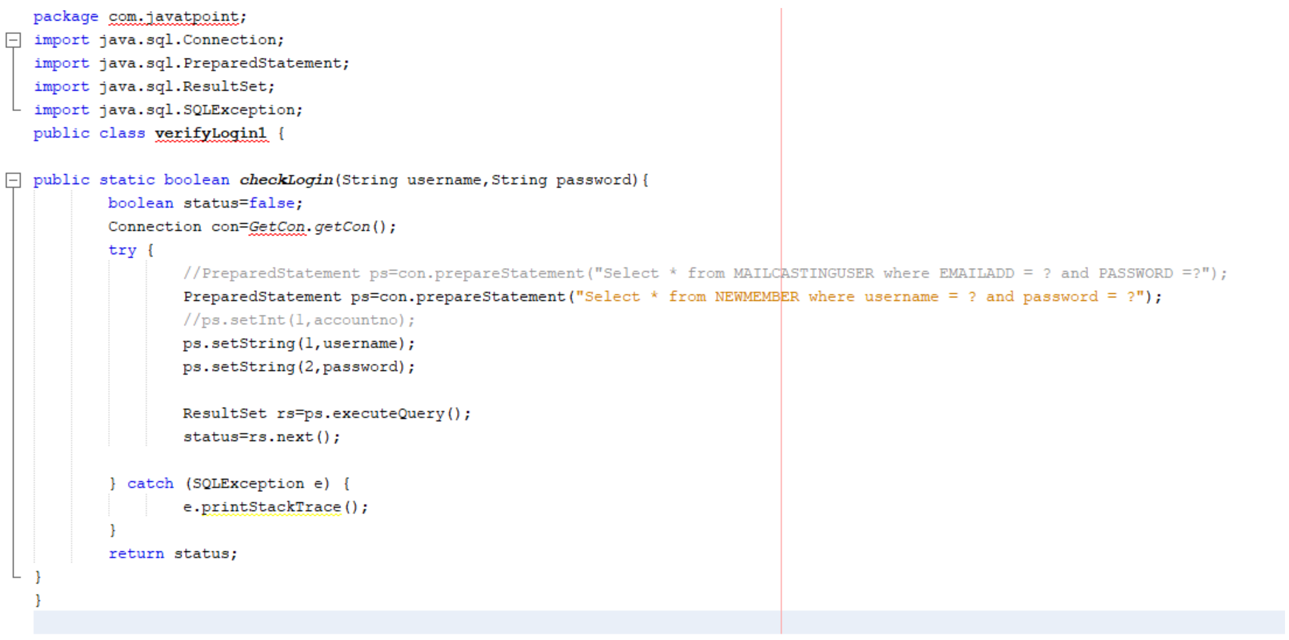
```
public class MyListener implements ServletContextListener{

        public void contextInitialized(ServletContextEvent arg0) {
                int status=0;
                Connection con=null;

      try{
                con=GetCon.getCon();
                PreparedStatement ps1=con.prepareStatement("Select * from NEWMEMBER");


      try{
                status=ps1.executeUpdate();
                }

      catch(Exception e)
                {e.printStackTrace();
                 status=2;
                 System.out.println("my staus is1111111"+status);
                 }

       if(status==0)
                {System.out.println("your table name already exist"+status);}


       else if(status==2)

               {System.out.println("else if part table does not exist new table has created"+status);
                PreparedStatement ps3=con.prepareStatement("CREATE SEQUENCE javatpoint MINVALUE 1 MAXVALUE 999999999999 INCREMENT BY 1 START WITH 1 NOCACHE  NOORDER  NOCYCLE");
                ps3.executeUpdate();

                PreparedStatement ps=con.prepareStatement("CREATE TABLE  NEWMEMBER(ID NUMBER,USERNAME VARCHAR2(4000),PASSWORD VARCHAR2(4000),REPASSWORD VARCHAR2(4000),PHONE NUMBER, ADI
                ps.executeUpdate();

                PreparedStatement ps2=con.prepareStatement("CREATE TABLE  NEWSTAFFMEMBER(ID NUMBER,USERNAME VARCHAR2(4000),PASSWORD VARCHAR2(4000),REPASSWORD VARCHAR2(4000),PHONE NUMBE
                ps2.executeUpdate();

                PreparedStatement ps4=con.prepareStatement("CREATE TABLE  LIBRARYADMIN(USERNAME VARCHAR2(4000),PASSWORD VARCHAR2(4000))");
                ps4.executeUpdate();
                ps4 = con.prepareStatement("Insert into LIBRARYADMIN values(?,?)");
            ps4.setString(1,"admin");
                ps4.setString(2,"admin");
```

Registered user



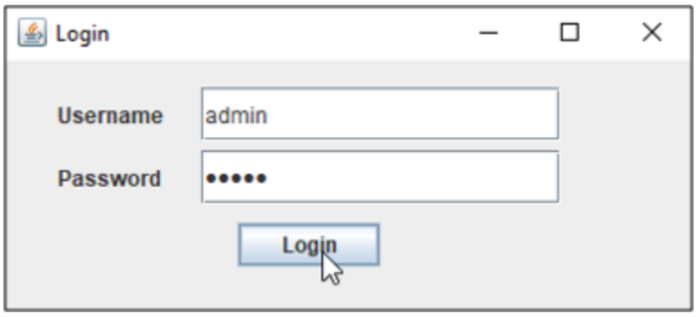
```
package com.javatpoint;
import java.sql.*;
public class RegisterUser {
static int status=0;
//int accountno=1;
public static int register(String username,String password,String repassword,double phone,String adderess,String syd){
        //public static int register(String email,String password,String gender,String country,String name){

        Connection con=GetCon.getCon();
        PreparedStatement ps;
        try {
                ps = con.prepareStatement("Insert into NEWMEMBER values(?,?,?,?,?,?,?)");
                int     nextvalue1=GetCon.getPrimaryKey();
                ps.setInt(1,nextvalue1);
            ps.setString(2,username);
                ps.setString(3,password);
                ps.setString(4,repassword);
                ps.setDouble(5,phone);
                ps.setString(6,adderess);
                ps.setString(7,syd);

                status=ps.executeUpdate();
                System.out.println(status);
        } catch (SQLException e) {

                e.printStackTrace();
        }
        return status;

}
}
```

Verify login:-



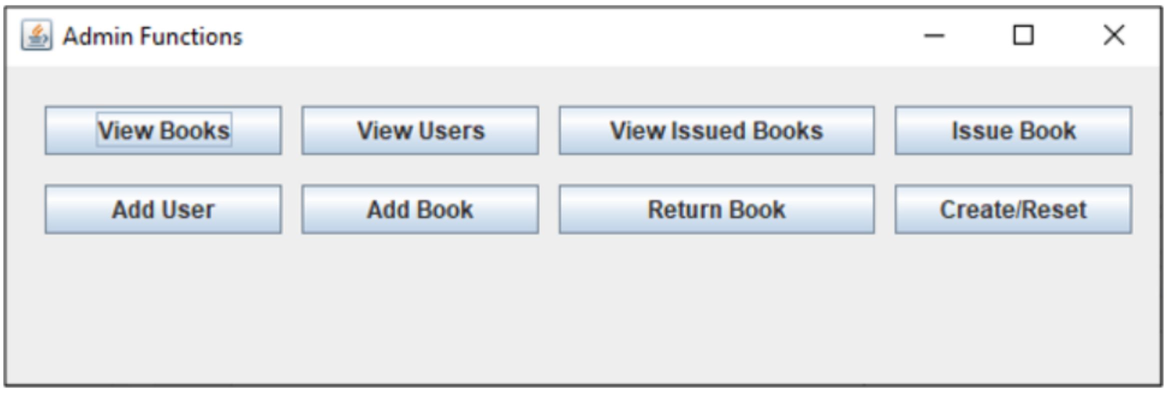
```
package com.javatpoint;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class verifyLogin1 {

public static boolean checkLogin(String username,String password){
        boolean status=false;
        Connection con=GetCon.getCon();
        try {
                //PreparedStatement ps=con.prepareStatement("Select * from MAILCASTINGUSER where EMAILADD = ? and PASSWORD =?");
                PreparedStatement ps=con.prepareStatement("Select * from NEWMEMBER where username = ? and password = ?");
                //ps.setInt(1,accountno);
                ps.setString(1,username);
                ps.setString(2,password);

                ResultSet rs=ps.executeQuery();
                status=rs.next();

        } catch (SQLException e) {
                e.printStackTrace();
        }
        return status;
}
}
```

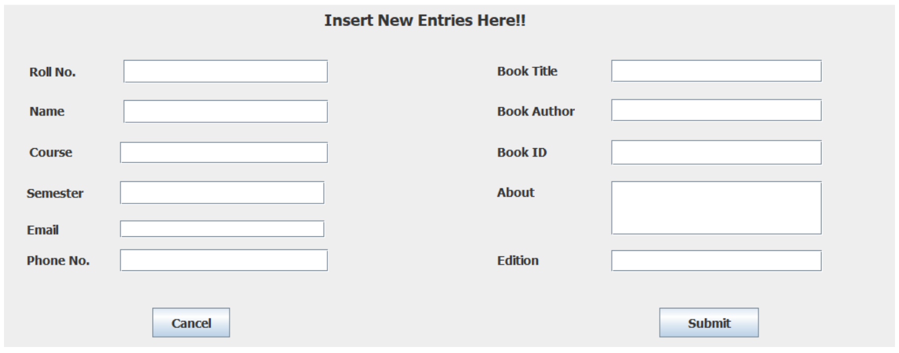
2.5 Test results, using an adequate set of test cases and screenshots

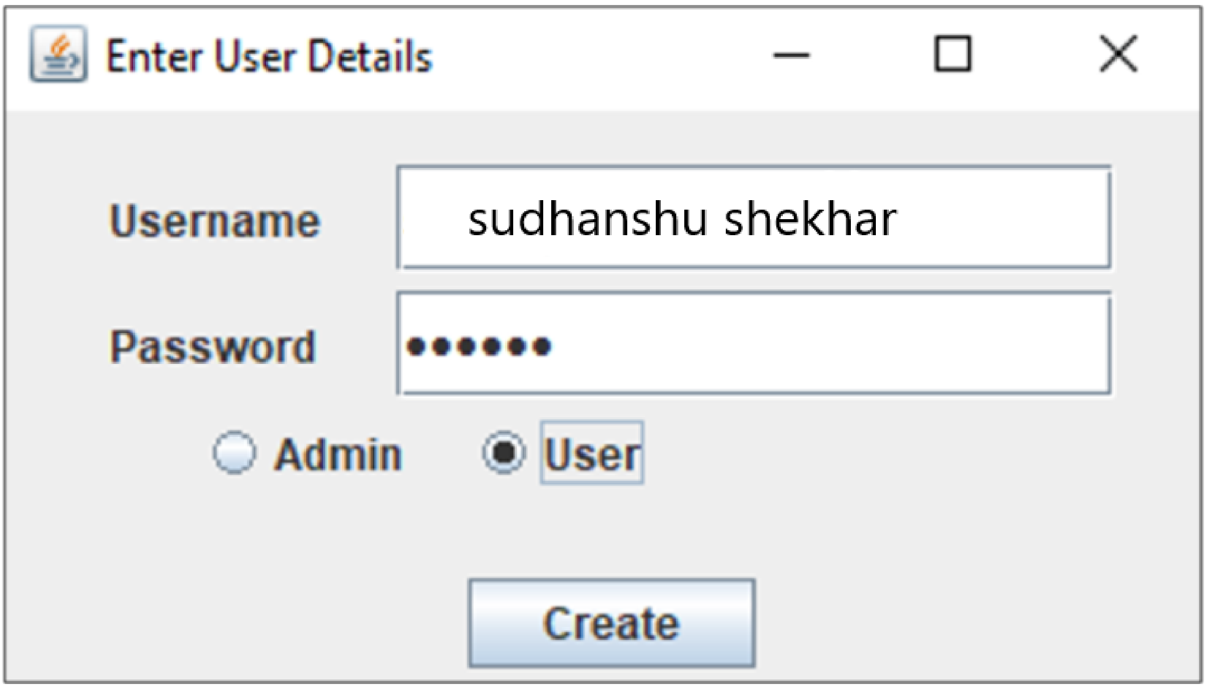


(fig 2)

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





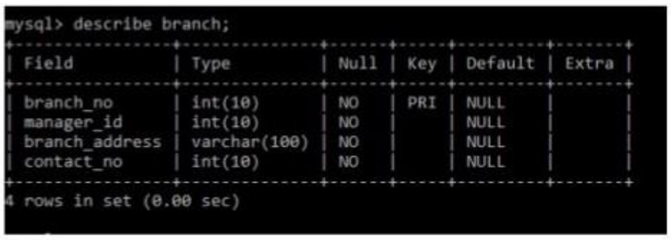

Enter User Details

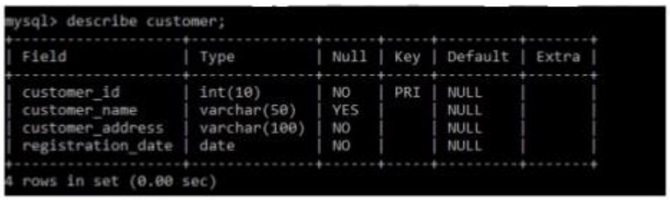
Username: sudhanshu shekhar

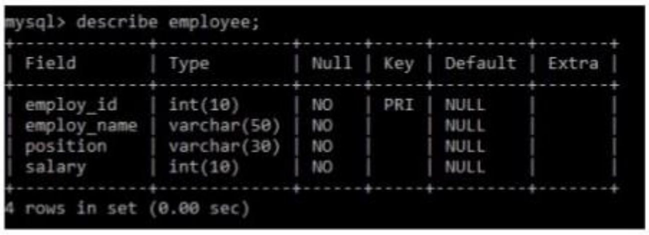
Password: ••••••

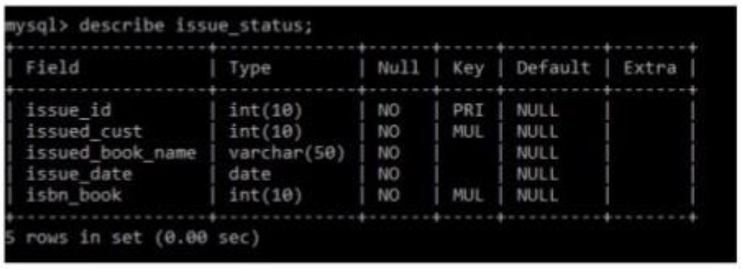
Admin [✓ User]

In SQL,

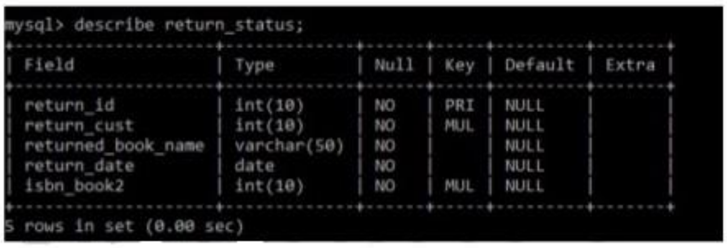

```
mysql> describe branch;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| branch_no      | int(10)      | NO   | PRI | NULL    |       |
| manager_id     | int(10)      | NO   |     | NULL    |       |
| branch_address | varchar(100) | NO   |     | NULL    |       |
| contact_no     | int(10)      | NO   |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```


```
mysql> describe customer;
+-------------------+--------------+------+-----+---------+-------+
| Field             | Type         | Null | Key | Default | Extra |
+-------------------+--------------+------+-----+---------+-------+
| customer_id       | int(10)      | NO   | PRI | NULL    |       |
| customer_name     | varchar(50)  | YES  |     | NULL    |       |
| customer_address  | varchar(100) | NO   |     | NULL    |       |
| registration_date | date         | NO   |     | NULL    |       |
+-------------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

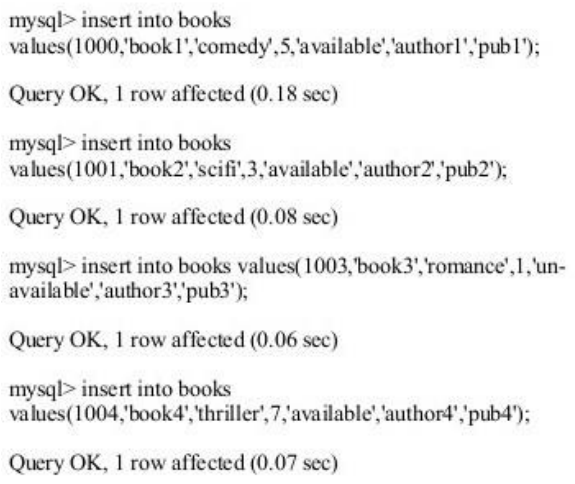

```
mysql> describe employee;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| employ_id   | int(10)     | NO   | PRI | NULL    |       |
| employ_name | varchar(50) | NO   |     | NULL    |       |
| position    | varchar(30) | NO   |     | NULL    |       |
| salary      | int(10)     | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```



```
mysql> describe issue_status;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| issue_id         | int(10)     | NO   | PRI | NULL    |       |
| issued_cust      | int(10)     | NO   | MUL | NULL    |       |
| issued_book_name | varchar(50) | NO   |     | NULL    |       |
| issue_date       | date        | NO   |     | NULL    |       |
| isbn_book        | int(10)     | NO   | MUL | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```



```
mysql> describe return_status;
+--------------------+-------------+------+-----+---------+-------+
| Field              | Type        | Null | Key | Default | Extra |
+--------------------+-------------+------+-----+---------+-------+
| return_id          | int(10)     | NO   | PRI | NULL    |       |
| return_cust        | int(10)     | NO   | MUL | NULL    |       |
| returned_book_name | varchar(50) | NO   |     | NULL    |       |
| return_date        | date        | NO   |     | NULL    |       |
| isbn_book2         | int(10)     | NO   | MUL | NULL    |       |
+--------------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```



```
mysql> insert into books
values(1000,'book1','comedy',5,'available','author1','pub1');

Query OK, 1 row affected (0.18 sec)

mysql> insert into books
values(1001,'book2','scifi',3,'available','author2','pub2');

Query OK, 1 row affected (0.08 sec)

mysql> insert into books values(1003,'book3','romance',1,'un-
available','author3','pub3');

Query OK, 1 row affected (0.06 sec)

mysql> insert into books
values(1004,'book4','thriller',7,'available','author4','pub4');

Query OK, 1 row affected (0.07 sec)
```

2.6 Conclusion:-

Execute the application by clicking on the run button.
Once you click on the **Login button**, you will see a dialog box opening up.
Here you have various options which you can explore. So, let us start with the first one:
Once, you click on View Books button, you will see the below frame displaying all the books present in the database, with their details.
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

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**.

- Suppose Server application makes a ServerSocket on a specific port which is 4123. This starts our Server listening for client requests coming in for port 4123.
- Then Server makes a new Socket to communicate with the client.

```
socket = server.accept()
```

- The accept() method blocks(just sits there) until a client connects to the server.
- Then we take input from the socket using getInputStream() method. Our Server keeps receiving messages until the Client sends “Over”.
- After we’re done we close the connection by closing the socket and the input stream.
- To run the Client and Server application on your machine, compile both of them. Then first run the server application and then run the Client application.