UNIVERSITY OF THE EAST - CALOOCAN
College of Engineering
Department of Computer Studies and Systems

CCP1102 – Computer Programming 2 –JAVA

FINAL REQUIREMENTS

LEARNING SPACE INTEGRATED LIBRARY SYSTEM

Submitted by

Calibuso, Catherine Hannah S.
Dela Pena, Kristine Mae P.
Dipus, Ma. Dominique L.
Estras, Julianne France
Nicolasora, Gabriel Antonio E.
Rubic, Justine Christopher U.
Sacramed, Albert M.
Varela, Chenie B.

CCP1102 – IT1B

Submitted to

Prof. Kathleen M. Dimaano

May 24, 2021

INTRODUCTION

For the past years, libraries are seen as antiquated. As the world keeps progressing and becoming surrounded by modern technology its evolution has become beneficial to humans in numerous ways such as addressing the concerns and increasing expectations of the users' needs to access reliable and easier reading materials, as well as the demand for the increased operational efficiency of a library application. One of them is the establishment of an online library system from a standard one. Online libraries have always served as places for communities to quiet their minds, relax from stressful thoughts without the need of physically going out and they are embracing that role even more today. With this said, the use of the legacy system in libraries is diminishing and they are becoming more open to new technologies or services and positively adapting to it.

This opened an opportunity for system creators to provide a wider range of functions especially those focused on the internet. Most of the systems now offer online-based portals where the users may log on, view their accounts, and authenticate with also the help and integration of databases. This also inspired the creation of an online library system called, Learning Space Integrated Library System. The Learning Space Integrated Library System is a simple, user-friendly, and easy to operate system in which borrowers will truly enjoy the use of borrowing and returning books. The system also promotes intellectual growth and facilitates access to information and resources efficiently among other things. With its easy-to-use functionalities, users will experience and enjoy an effortless reading experience.

SYSTEM FUNCTIONALITIES AND CAPABILITIES

The Learning Space Integrated Library System has many system functionalities and features that can assist users in borrowing and returning books with ease. Some of the features are the following:

LOGIN FORM

- Register - Register button will allow the user to register with our library and receive a unique ID No., which will be used to log in to our library.
- Home - Home button will lead you to the library's home page and show a view of it.
- About us - About us button will provide you with the opportunity to learn more about our library.
- FAQ - The FAQ button will take you to the answers to the most frequently asked questions about our library by our users.
- Sign in - Once you have already registered, you can use the sign-in button to choose whether you're a librarian, a student, or a faculty member, and it will direct you to the view of your system based on your membership type.

ADMIN FORM

- Home - The Home button in the Admin Form will take you to the admin system's site, where you can also view the details of the users who have registered in the library.
- Add Librarian - The Add Librarian button in the upper right corner of the homepage allows you to add another admin or librarian that can manage the admin system.
- Books - The Book feature or button allows the librarian to add, edit, and remove books as well as browse the table of available books.
- Issue books - Issue book is the button or feature where the librarian decides whether or not the borrower may borrow a particular book.
- Records - The Record function would allow the librarian to see who borrowed the book and what sort of member they are, whether a student or a faculty member. Admin can also view the book that the user borrowed, the date it was borrowed, the due date, and the date it was returned.
- Users - The librarian has access to those who have registered in the library and can look at their records because of the User button the Admin system has.
- Logout - It will redirect the to the login form and allow the admin view system to exit.

STUDENT FORM

- Home - The Home button in the Student Form will take you to the student system's homepage, where various buttons are active to guide you to the form that you want inside the student system.
- Books Borrowed: View here - This feature in the student system allows students to see which books have already been borrowed.
- Books - This feature will provide the student with a list of the available books in our library as well as the status of the book, whether it is available or not.
- Request Book - The student may also suggest or request what book they're looking for at the bottom of the Books Form or feature.
- Book Borrow - This function allows the student to make a request to the librarian for a book that he or she wishes to borrow by providing the required details.
- Return Book - The student will be able to return the book to the library using the return book function.
- Search - In addition to checking the list in the books feature, if the student does not have enough time to scroll and find the book, he or she can search for it directly by using the search button.
- Logout - It will transfer the user to the login form and exit the student view system.

FACULTY FORM

- Home - The Faculty Form's Home button will take you to the faculty system's homepage, where numerous buttons will be active to direct you to the form you want inside the faculty system.
- Books Borrowed: View here - This feature of the faculty system allows faculty to see which books have already been borrowed.
- Books - This feature will show the faculty member a list of the books that are currently available in our library, as well as the status of the book, whether it is available or not.
- Request Book - A faculty member may also recommend or request a book at the bottom of the Books Form or function.
- Book Borrow - This function enables a faculty member to submit a request to the librarian for a book that he or she wishes to borrow by supplying the necessary information.
- Return Book - Using the return book option, the faculty member would be able to return the book to the library.

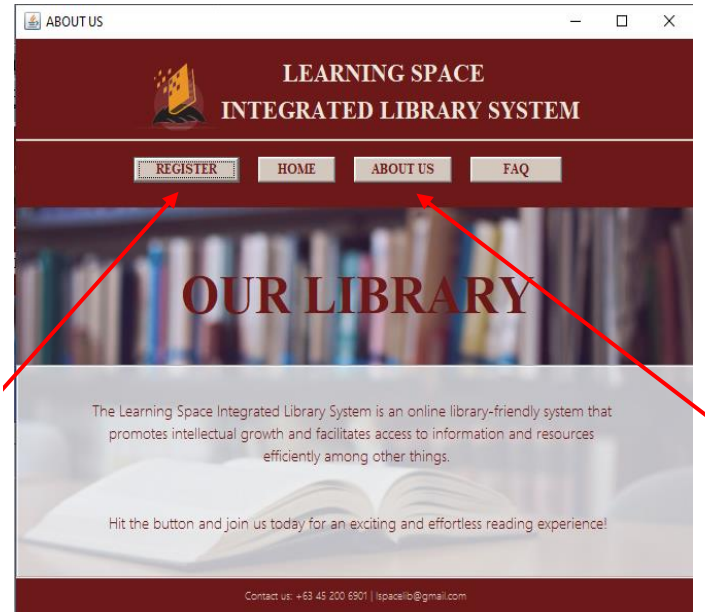
- Search - If a faculty member does not have enough time to browse through the books feature and locate the book, he or she can search for it directly by using the search button.
- Logout - It will take you to the login form and withdraw you from the faculty view system.

LOGIN MODULE

The Login page is where the user will input his or her User ID matching with a password, as well as member type. If the username, password and user type combination does not match a user account, the user is disallowed access in the library system.

Logging in with input

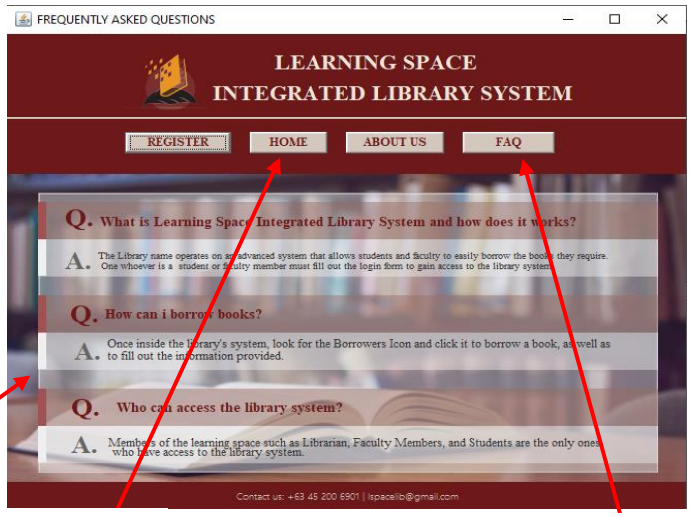
OTHER MODULES



REGISTER button

ABOUT US button

The ABOUT US page displays the brief introduction about our system where it is appropriate to all of the users even students, faculty, or librarian admin can access it. And also, it is a library friendly access because it is a system that promotes intellectual growth and facilitates access to numerous of information and resources efficiently among other things.



Questions and answers display

HOME button

FAQ button

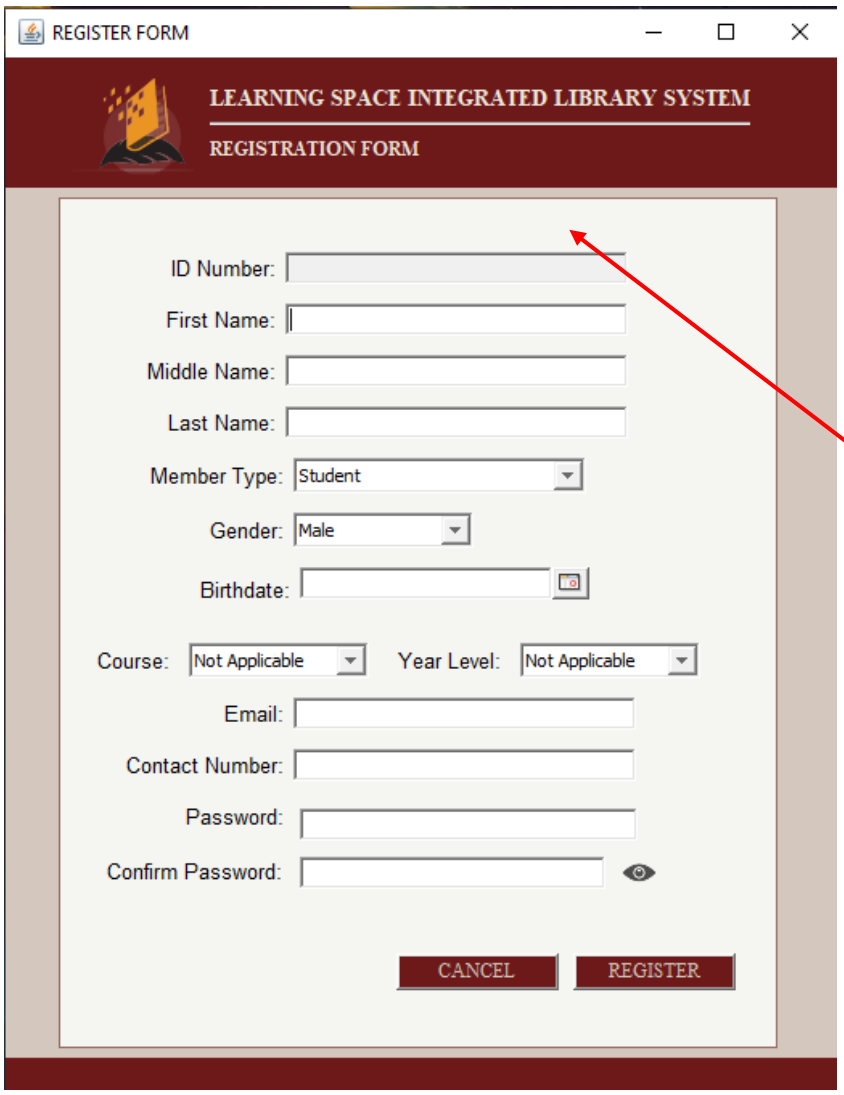
In the FAQs page of the library system, this displays the common questions that users ask including how to borrow a book and how does the library system itself operates.

CODE

```
private void signinActionPerformed(java.awt.event.ActionEvent evt) {
    user = enterUserID.getText();
    newID = Integer.parseInt(user);
    newUSERTYPE = String.valueOf(UserType.getSelectedItem());
    newPASS = String.valueOf(enterPass.getPassword());

    try{
        rs = stmt.executeQuery("SELECT * FROM USERS");
        while(rs.next()) {
            id = rs.getInt("IDNO");
            pass = rs.getString("PASSWORD");
            fn = rs.getString("FIRSTNAME");
            memtyp = rs.getString("MEMBERTYPE");

            if(newID == id) {
                if(newPASS.equals(pass)) {
                    if(newUSERTYPE.equals(memtyp)){
                        temp_user = newID;
                        temp_pass = newPASS;
                        temp_usertype = newUSERTYPE;

                        con.commit();
```

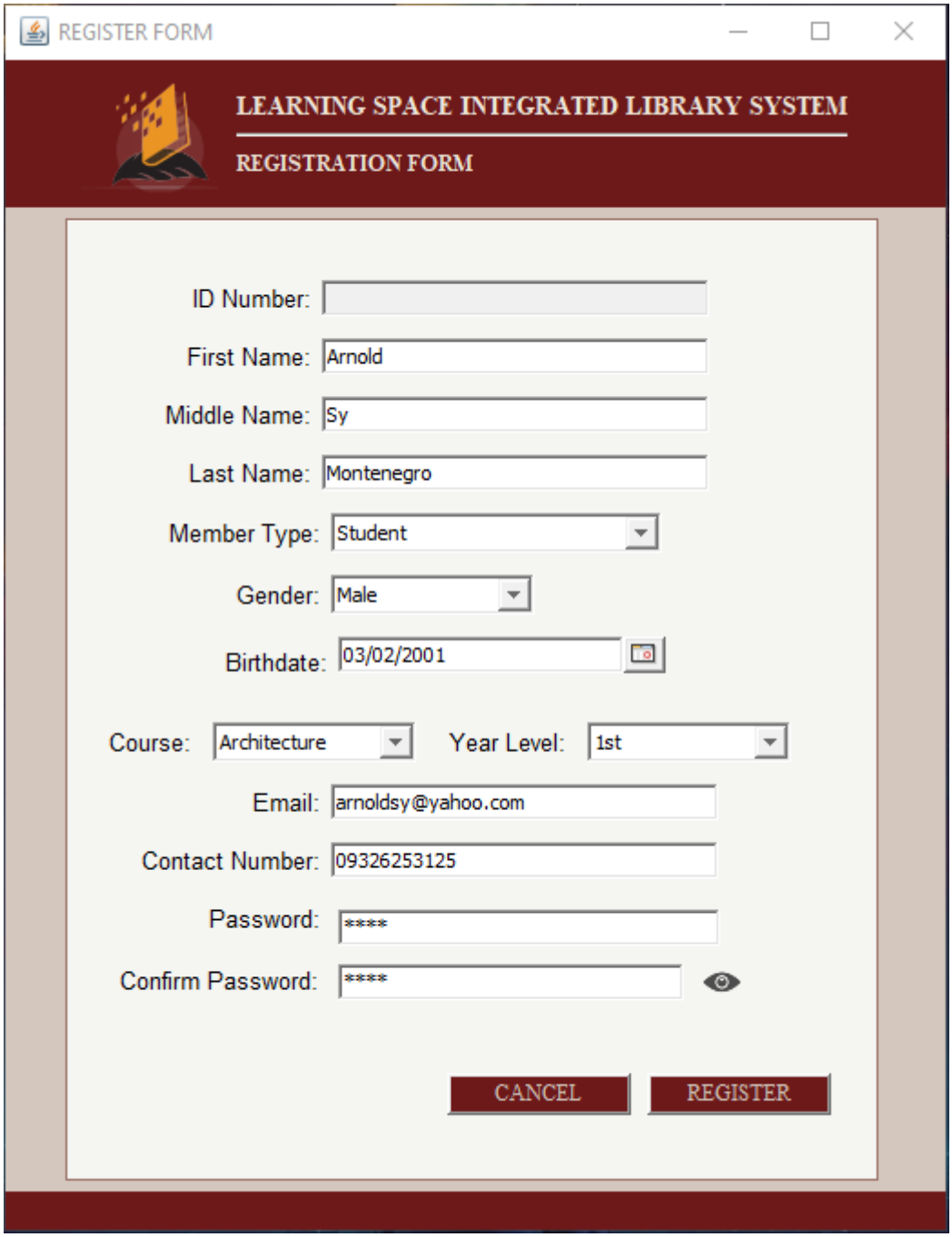
```
                        if(newUSERTYPE.equals("Librarian Admin")){
                            new AdminSys().setVisible(true);
                            LoginFormNew.this.dispose();
                            rs.close();
                        }else if(newUSERTYPE.equals("Student")){
                            new StudentSys().setVisible(true);
                            LoginFormNew.this.dispose();
                            rs.close();
                        }else{
                            new FacultySys().setVisible(true);
                            LoginFormNew.this.dispose();
                            rs.close();
                        }
                        }
                    }
                }
            }
            if(newID != temp_user) {
                if(newPASS != temp_pass) {
                    if(newUSERTYPE != temp_usertype) {
                        JOptionPane.showMessageDialog(null, "Incorrect username, password or user type!");
                        enterUserID.setText(null);
                        enterPass.setText(null);
                    }
                }
            }
        }catch(SQLException e) {
            System.out.println(e);
        }
```

USER'S ACCOUNT REGISTRATION MODULE

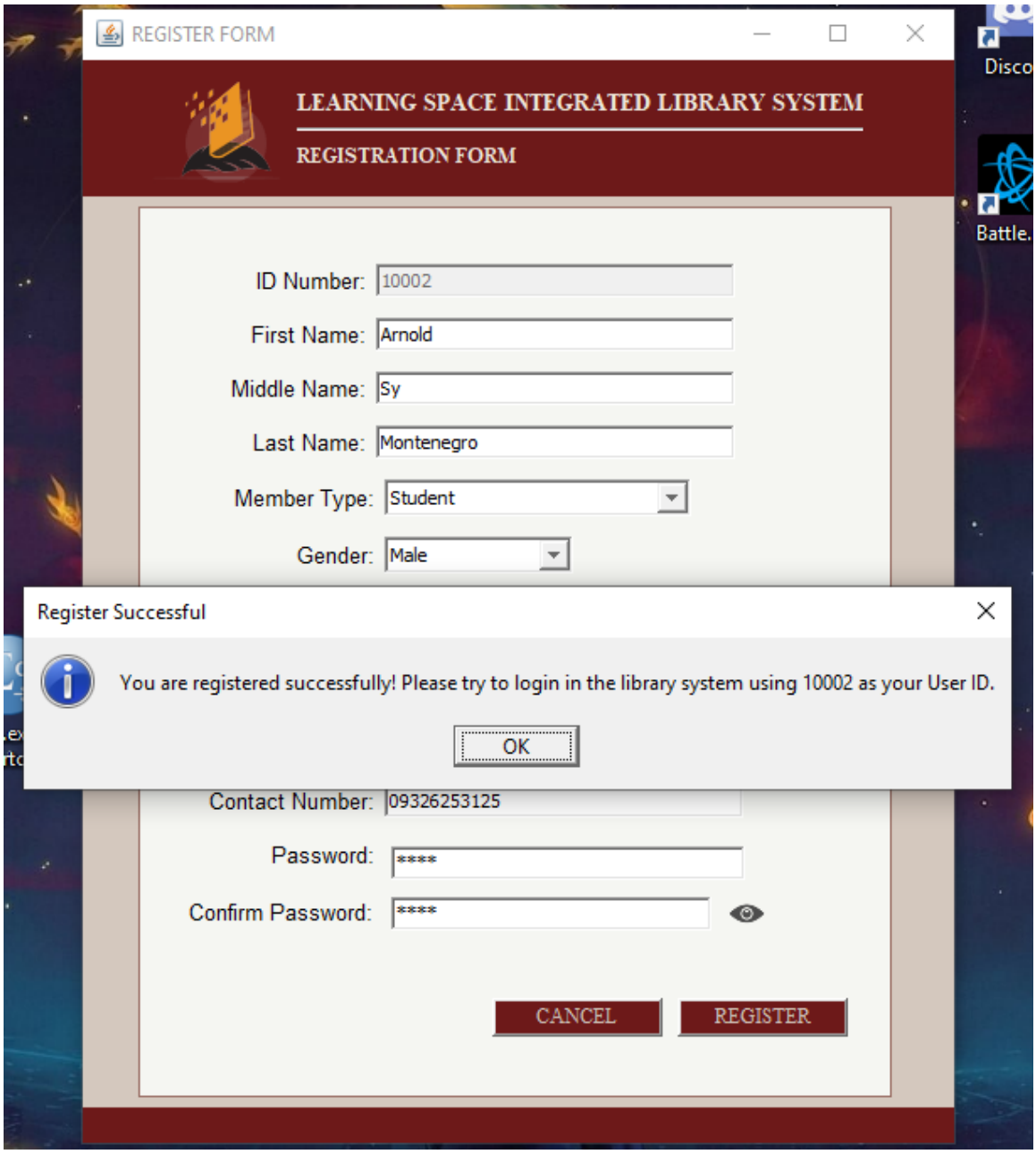


The ID Number will be automatically generated once the user has been successfully registered.

The REGISTRATION FORM the users may register here to enter to our system.



User input before registering



Once the user has been successfully registered, A JOption confirmation message will pop.

CODE

```
package LibrarySystemForms;
import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;
public class StudentRegistrationForm extends LibConnect {

    public StudentRegistrationForm() {
        initComponents();
        DoConnectUsers();

        this.eyepassclose.setVisible(false);
    }
```

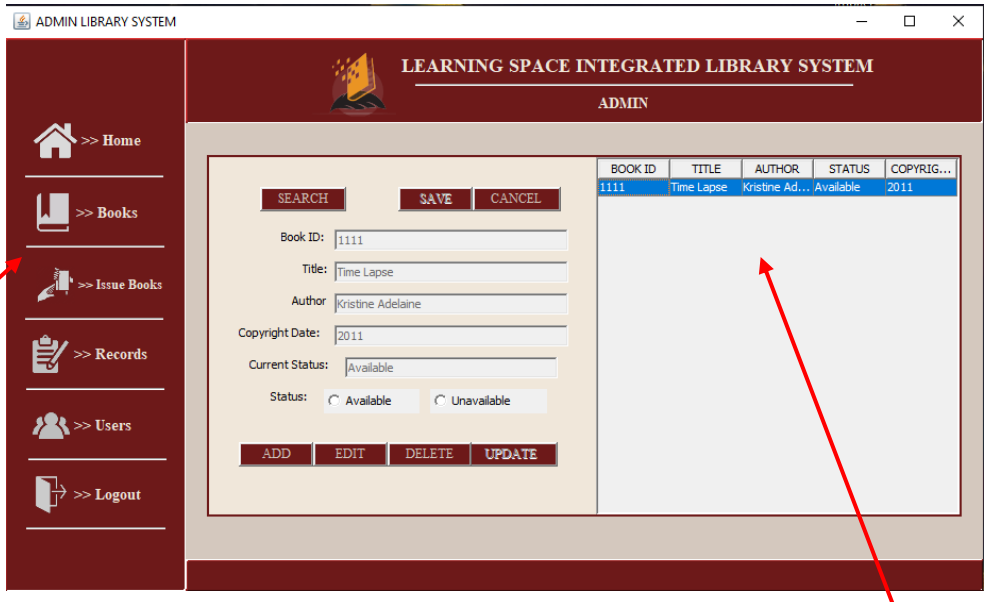
```
        //USER INPUT FILLING OUT DETAILS
        lastid = Integer.parseInt(RegID.getText());
        x = String.valueOf(lastid).trim();
        fn = RegFN.getText().trim();
        mn = RegMN.getText().trim();
        ln = RegLN.getText().trim();
        memtyp = String.valueOf(RegMT.getSelectedItem()).trim();
        gender = String.valueOf(RegGen.getSelectedItem()).trim();
        crs = String.valueOf(RegCourse.getSelectedItem()).trim();
        yrlvl = String.valueOf(RegYrLvl.getSelectedItem()).trim();
        email = RegEmail.getText().trim();
        contact = RegContact.getText();

        pw = RegPass.getText().trim();
        confirm_pass = String.valueOf(RegConfirmPass.getPassword());
        bdate = sdf.format(RegBdate.getDate()).trim();

        if(pw.equals("")) {
            JOptionPane.showMessageDialog(null, "Please input password!", "Password Required",
                    JOptionPane.INFORMATION_MESSAGE);
        }else if(!pw.equals(confirm_pass)){
             JOptionPane.showMessageDialog(null, "Please retype your password.");
        }else{
```

```
    stmt.execute("INSERT INTO USERS values ("+x+", '"+fn+"','"+mn+"', '"+ln+"', '"+pw+"', '"+memtyp+"', '"+gender+"','"
            + "'"+bdate+"','"+crs+"','"+yrlvl+"', '"+email+"', "+contact+")");

        JOptionPane.showMessageDialog(this, "You are registered successfully! Please try to login in the library system"
            + " using " + lastid + " as your User ID.",
            "Register Successful", JOptionPane.INFORMATION_MESSAGE);


    LoginFormNew login = new LoginFormNew();
    login.setVisible(true);
    this.dispose();
}



}catch (SQLException ex) {
    Logger.getLogger(StudentRegistrationForm.class.getName()).log(Level.SEVERE, null, ex);
}
```

```
private void regsucessActionPerformed(java.awt.event.ActionEvent evt) {
    if(RegFN.getText().isEmpty() || RegMN.getText().isEmpty() || RegLN.getText().isEmpty()
    || RegEmail.getText().isEmpty() || RegContact.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please fill out all information before submitting this form.");
     }

    try {
        rs = stmt.executeQuery("SELECT MAX(IDNO) FROM USERS");
        if(rs.next()) {
            lastid = rs.getInt(1);
            lastid++;
            RegID.setText(Integer.toString(lastid));

        }else{
            RegID.setText("1001");
        }
```

ADMIN MODULE

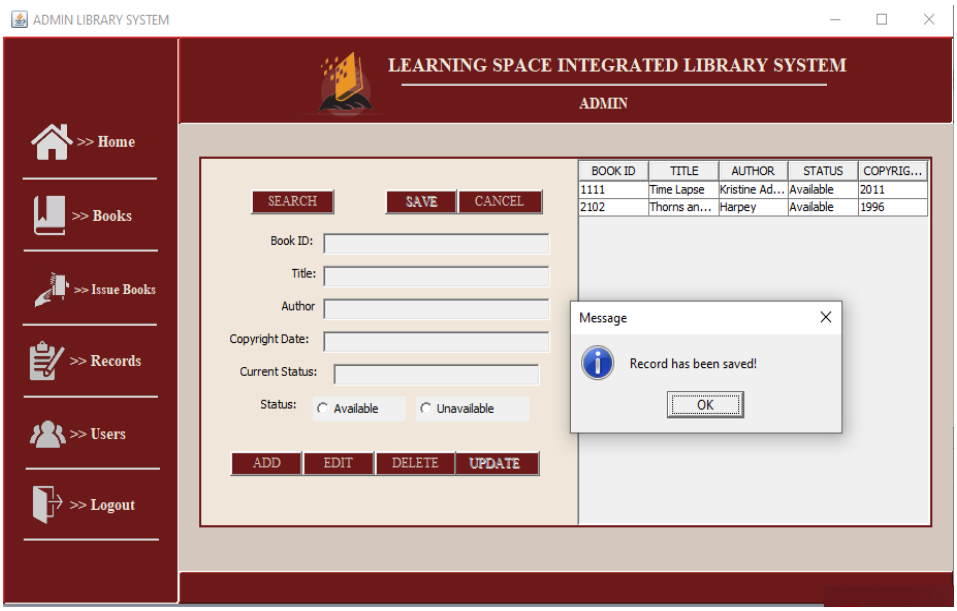
The Admin Homepage, the admin will directly go on homepage so that he/she can now edit some information, return, or borrow a book.



BOOKS button

The BOOKS page is where the ADMIN can edit, add, or delete all the necessary information about the books, including its books ID, title author as well as the status of the book.

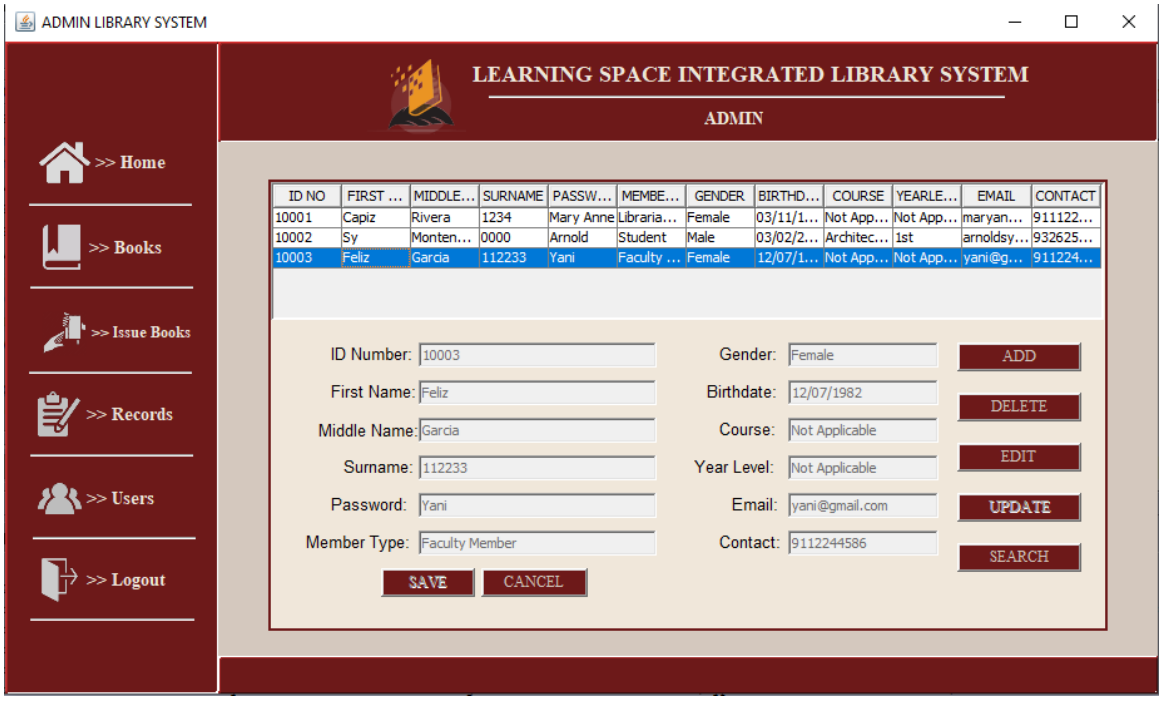
Table in which display the books.



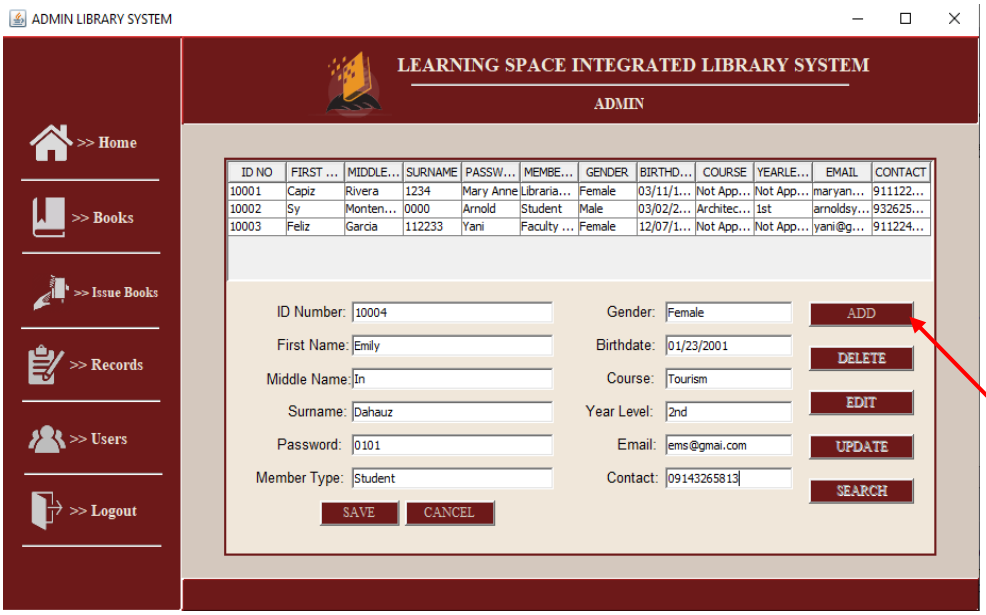
In adding books, the admin must fill up all the required details. You can set the status setting of the specific book into available or unavailable.

The search book allows the admin to search for a specified book information with just an input of Book ID.

This is where the admin issues a book borrower after entering the borrower's information may that person is a student or a faculty member. The date is also auto-generated when a book was borrowed and when it is due to be returned.

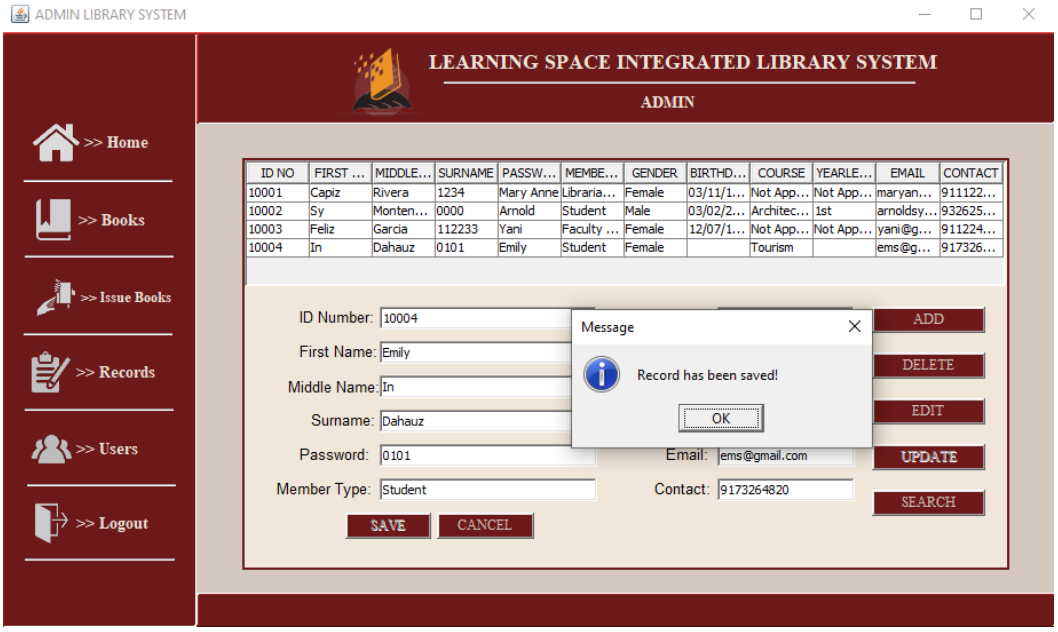


The USERS page is where the ADMIN can edit, add, update, or delete all information about the USERS.

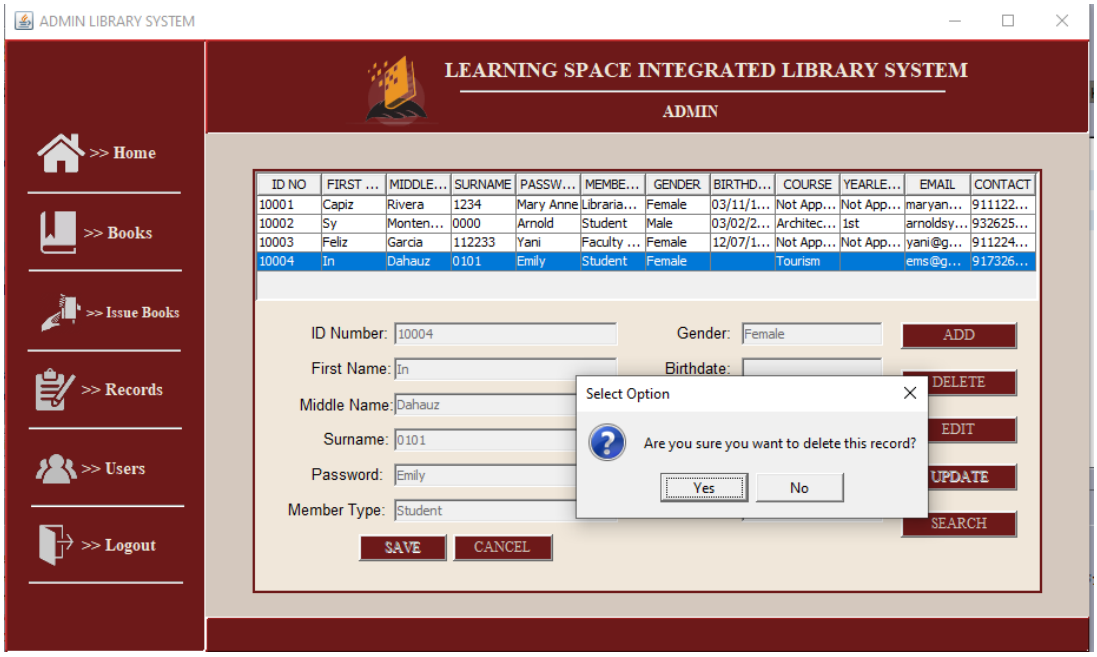


ADD Button

A sample simulation of adding a user.

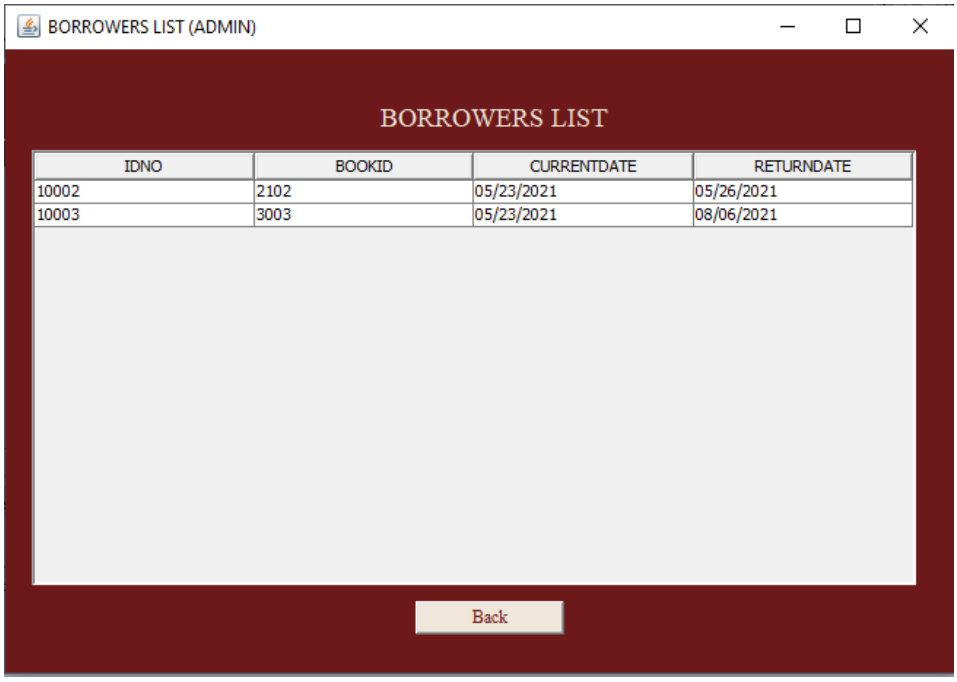


Adding a user successfully. A JOption will popup that the record has been saved and added.

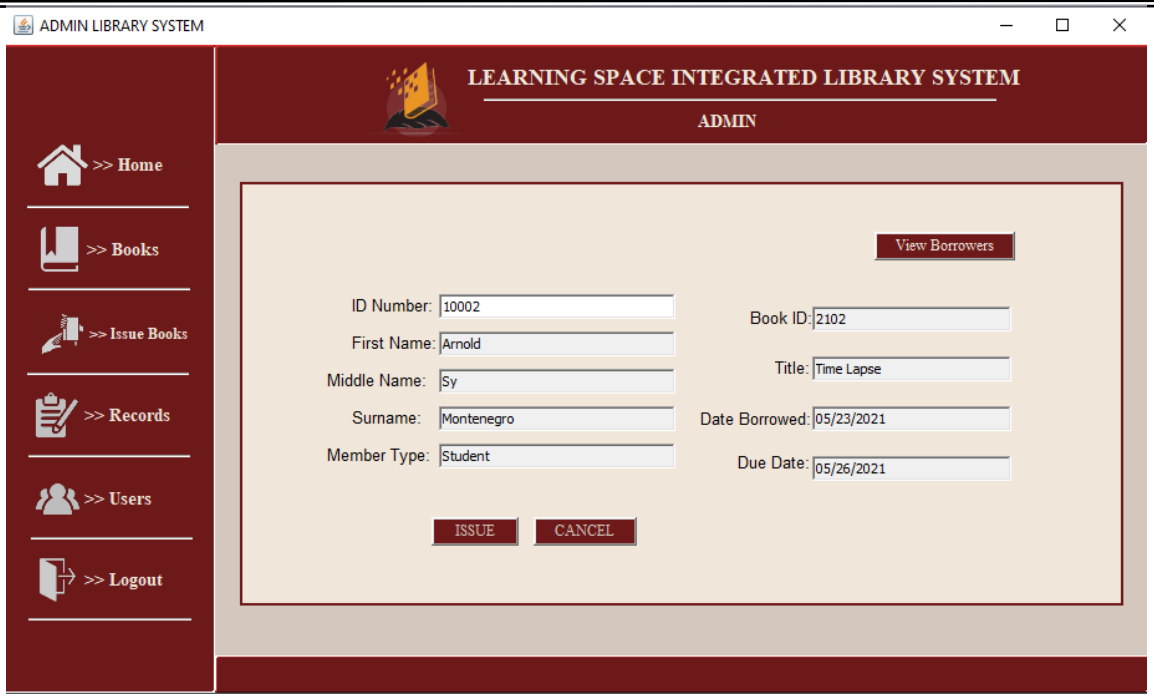


This deletes all information that belongs to a user. A JOption will pop up to choose if an admin has to delete a certain record.

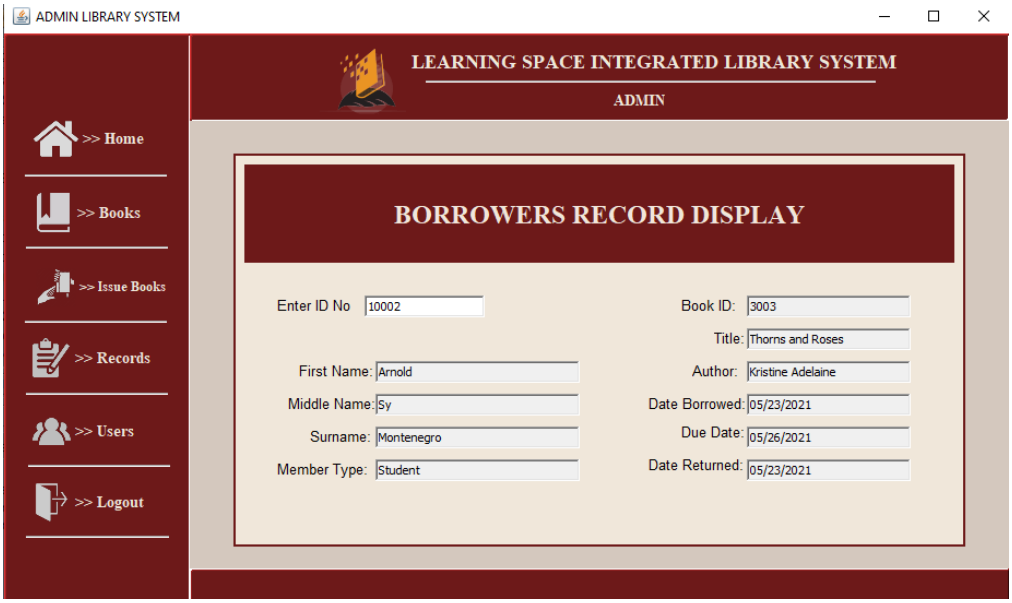
The LOGOUT button will exit the current system that a librarian admin logged in.



This BORROWERS LIST will display all the borrowed book of the faculty and student as well as the borrowed date and return date of the book.



This is where you can borrow a book after entering your student information and the details of the book you want to borrow. You can see the date the book was borrowed and when it is due to be returned.



This is where the admin can view the record display of library users that returned and borrowed.

CODE

```
// LIST FOR BOOKS
    } public ArrayList<AddClass> listbooks(){
       ArrayList<AddClass> listbooks = new ArrayList<>();
       try{
           selectbooks = "SELECT * FROM BOOKS";
           stmt = con.createStatement();
           //pst = con.prepareStatement(selectbooks);
           rs = stmt.executeQuery(selectbooks);

           AddClass books;
           while(rs.next()) {
               books = new AddClass(rs.getInt("BOOKID"), rs.getString("TITLE"), rs.getString("AUTHOR"),
               rs.getString("STATUS"), rs.getString("COPYRIGHTDATE"));
               listbooks.add(books);
           }
    }catch(Exception e) {
       JOptionPane.showMessageDialog(null, e);

    }return listbooks;
```

```
// BOOKS TABLE DISPLAY
} public void BookDisplay(){
    ArrayList<AddClass> list = listbooks();
     model = (DefaultTableModel)booktableadmin.getModel();
    Object[] row = new Object[5];
    for(int ctr = 0; ctr < list.size(); ctr++){
        row[0] = list.get(ctr).getBID();
        row[1] = list.get(ctr).getTitle();
        row[2] = list.get(ctr).getAuthor();
        row[3] = list.get(ctr).getStatus();
        row[4] = list.get(ctr).getCopyrightDate();

        model.addRow(row); {

    }
    }
```

```
private void booksearchActionPerformed(java.awt.event.ActionEvent evt) {
    i = JOptionPane.showInputDialog(null,"Enter Book ID: ","Records",JOptionPane.QUESTION_MESSAGE);
    newbookID = Integer.parseInt(i);
    try{
        RefreshBooks();
        while(rs.next()){
            bookID =rs.getInt("BOOKID");
            if(newbookID == bookID){
                temp_bid = newbookID;
                GetBookRecords();
            }
        }
        if(newbookID != temp_bid){
            showAll();
            GetBookRecords();
            JOptionPane.showMessageDialog(null,"Book not Found.");
        }
    }catch (SQLException err){
        JOptionPane.showMessageDialog(AdminSys.this, err.getMessage());
    }
```

```
private void booksaveActionPerformed(java.awt.event.ActionEvent evt) {

        bookID = Integer.parseInt(BookIDAdmin.getText());
        sbid = String.valueOf(bookID);
        title = BookTitleAdmin.getText();
        author = BookAuthorAdmin.getText();
        copyright = BookCDAdmin.getText();

        try {
        bookquery = "INSERT INTO BOOKS VALUES(?,?,?,?,?)";

        pst = con.prepareStatement(bookquery);
        pst.setString(1, sbid);
        pst.setString(2, title);
        pst.setString(3, author);
        pst.setString(5, copyright);

        if(BookAvail.isSelected()) {
            status = "Available";
            BookNotAvail.setSelected(false);
        }
        if(BookNotAvail.isSelected()) {
            status = "Unavailable";
            BookAvail.setSelected(false);
        }

        pst.setString(4, status);

                pst.executeUpdate();
```

```
                DefaultTableModel model = (DefaultTableModel)booktableadmin.getModel();
                model.setRowCount(0);

                BookDisplay();
                showAll();

                BookIDAdmin.setText("");
                BookTitleAdmin.setText("");
                BookAuthorAdmin.setText("");
                CurrentStatusAdmin.setText("");
                BookCDAdmin.setText("");

                BookIDAdmin.setEnabled(false);
                BookTitleAdmin.setEnabled(false);
                BookAuthorAdmin.setEnabled(false);
                CurrentStatusAdmin.setEnabled(false);
                BookAvail.setSelected(false);
                BookNotAvail.setSelected(false);
                BookCDAdmin.setEnabled(false);

                JOptionPane.showMessageDialog(AdminSys.this, "Record has been saved!");

        }catch(SQLException ex){
         Logger.getLogger(AdminSys.class.getName()).log(Level.SEVERE, null, ex);

    }
```

```
    private void bookdeleteActionPerformed(java.awt.event.ActionEvent evt) {
        JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this record?",
                "Select Option", JOptionPane.YES_NO_OPTION);

        try {
            row = booktableadmin.getSelectedRow();
            value  = (booktableadmin.getModel().getValueAt(row, 0).toString());

            bookquery = "DELETE FROM BOOKS WHERE BOOKID="+value;
            pst = con.prepareStatement(bookquery);
            pst.executeUpdate();

            DefaultTableModel model = (DefaultTableModel)booktableadmin.getModel();
            model.setRowCount(0);

            BookDisplay();

            BookIDAdmin.setText("");
            BookTitleAdmin.setText("");
            BookAuthorAdmin.setText("");
            CurrentStatusAdmin.setText("");
            BookCDAdmin.setText("");

            JOptionPane.showMessageDialog(null, "Record deleted!");

}catch(Exception e){
    JOptionPane.showMessageDialog(null, e);
}
```

```
    private void IDNRecordsAdminKeyReleased(java.awt.event.KeyEvent evt) {
          //SCREENSHOT
        srchbid = "SELECT BOOK_ISSUE.IDNO, USERS.FIRSTNAME, USERS.MIDDLENAME, USERS.SURNAME, "
        + "BORROW_BOOKS.BOOKID, BORROW_BOOKS.TITLE, BOOK_ISSUE.BORROWEDDATE, BORROW_BOOKS.CURRENTDATE, "
        + "USERS.MEMBERTYPE, BOOKS.AUTHOR, BOOK_ISSUE.RETURNDATE"
                + " FROM BOOKS, BORROW_BOOKS, BOOK_ISSUE, RETURN_BOOKS,"
                + "USERS WHERE USERS.IDNO="+IDNRecordsAdmin.getText()+"";
        try{
            pst = con.prepareStatement(srchbid);
            rs = pst.executeQuery();

            if(rs.next()) {
                //IDNRecordsAdmin.setText(rs.getString("IDNO"));
                FNRecordsAdmin.setText(rs.getString("FIRSTNAME"));
                MNRecordsAdmin.setText(rs.getString("MIDDLENAME"));
                SNRecordsAdmin.setText(rs.getString("SURNAME"));
                MTRecordsAdmin.setText(rs.getString("MEMBERTYPE"));
                BIDRecordsAdmin.setText(rs.getString("BOOKID"));
                TitleRecordsAdmin.setText(rs.getString("TITLE"));
                AuthorRecordsAdmin.setText(rs.getString("AUTHOR"));
                BorrDateRecordsAdmin.setText(rs.getString("BORROWEDDATE"));
                DuedateAdmin.setText(rs.getString("RETURNDATE"));
                ReturnRecordsAdmin.setText(rs.getString("CURRENTDATE"));

            }

        }catch(Exception e){
            JOptionPane.showMessageDialog(null, e);
```

```
private void IssueAdminActionPerformed(java.awt.event.ActionEvent evt) {
    UID = Integer.parseInt(IDNoissue.getText());
    uidissue = String.valueOf(UID);
    BID = Integer.parseInt(BookIDissue.getText());
    bidissue = String.valueOf(BID);
    issuedate = IssueDate.getText();
    returnbookissue = ReturnDate.getText();

    try {
      stmt = con.createStatement();
      rs = stmt.executeQuery("SELECT TITLE FROM BOOKS WHERE BOOKID ="+bidissue+"");
    if(rs.next()) {
            ResultSet rsl = stmt.executeQuery("SELECT * FROM BORROW_BOOKS WHERE IDNO="+uidissue+"");
            if(rsl.next()) {
                stmt.executeUpdate("INSERT INTO BOOK_ISSUE VALUES("+uidissue+",'"+bidissue+"', '"+issuedate+"', "
                        + "'"+returnbookissue+"')");
```

```
                JOptionPane.showMessageDialog(null, "Book issued successfully!");
                IDNoissue.setText("");
                FNissue.setText("");
                MNissue.setText("");
                SNissue.setText("");
                MTissue.setText("");
                BookIDissue.setText("");
                Titleissue.setText("");
                IssueDate.setText("");
                ReturnDate.setText("");

            }else{
                JOptionPane.showMessageDialog(null, "Incorrect ID Number!");
            }
        }else{
            JOptionPane.showMessageDialog(null, "Incorrect BookID!");
        }
    }catch(Exception e) {
        JOptionPane.showMessageDialog(null, "Book already issued to the student.");
    }
}
```

USER MODULE
(Student)

After the user logged in as 'Student' member type, there are 4 panels that a student can view and depending on what the user wants to do or what is his/her purpose.

BOOKS button

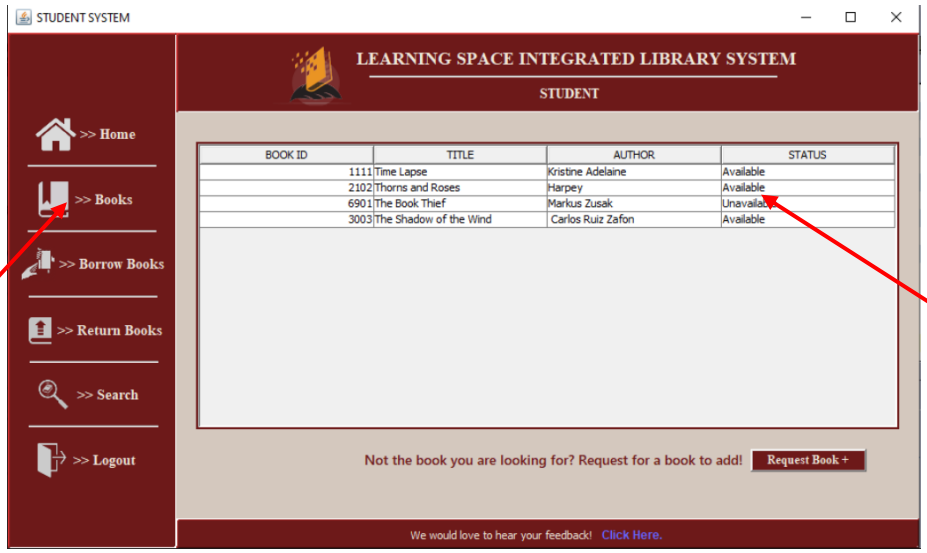
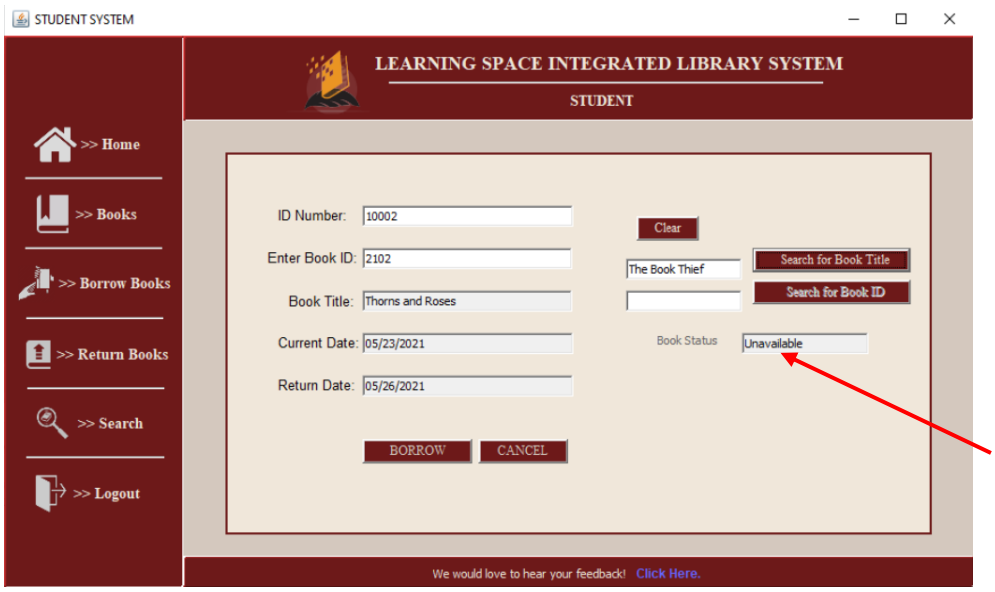


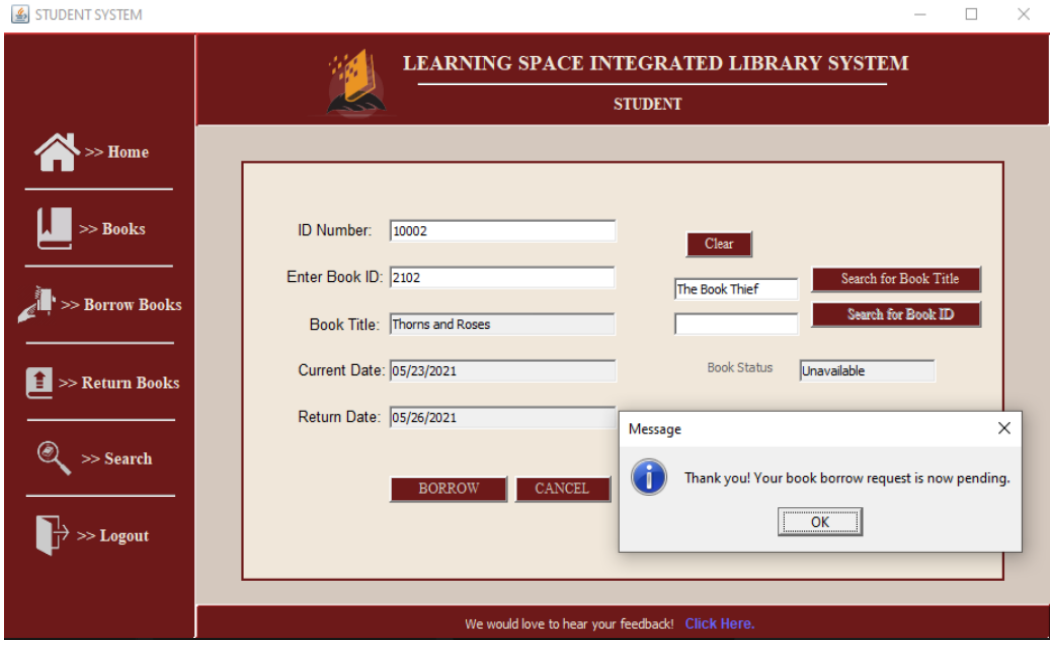
Table that contains book information

The BOOKS page is where the student can find all the necessary information about the books, including its books ID, title author as well as the status of the book.



Displays the status of a books

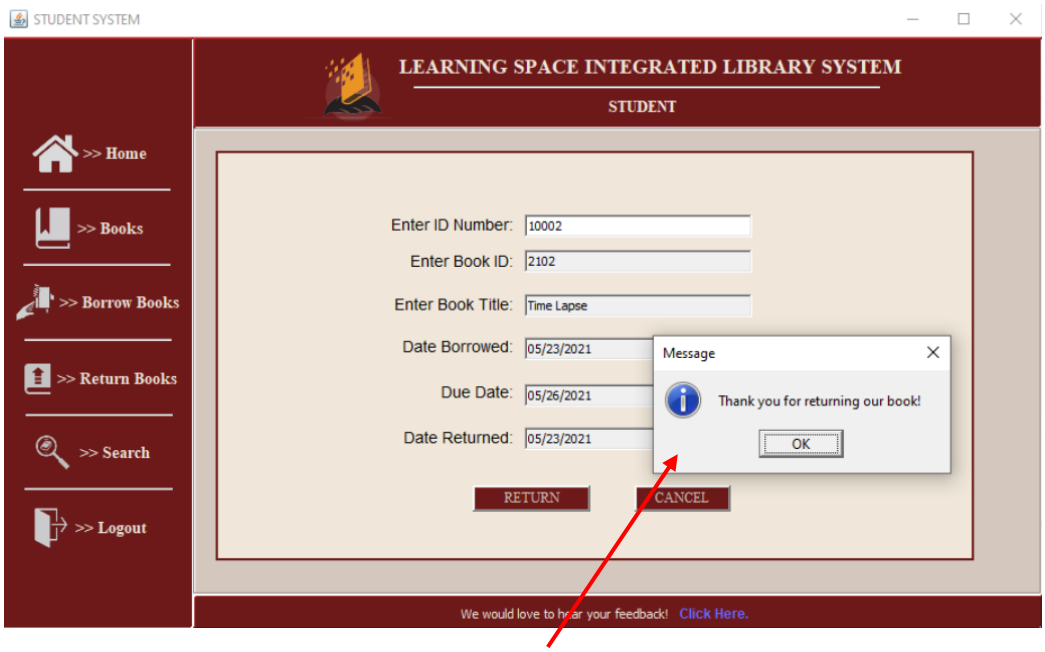
In BORROW BOOKS section this is where the student can borrow a book by entering book ID and their ID number. The book title, current date, and return date is already auto generated therefore, student doesn't have to fill that information. They can also search a book by their title and book ID in-order to see the status of the book



A JOption confirmation message will pop saying the student's book borrow request is now pending.

RETURN BOOKS button

In RETURN BOOKS, student must return the book at their own risk. Wherein the student should fill up the forms such as id number, book id, book title, date borrowed, due data and date returned.



Book returns successful

SEARCH button

In SEARCH page, student can find a book by searching the book ID and find out the status and title of the book.

User input with search result. This displays that the book ID the student's searched has a status of being available.

CODE

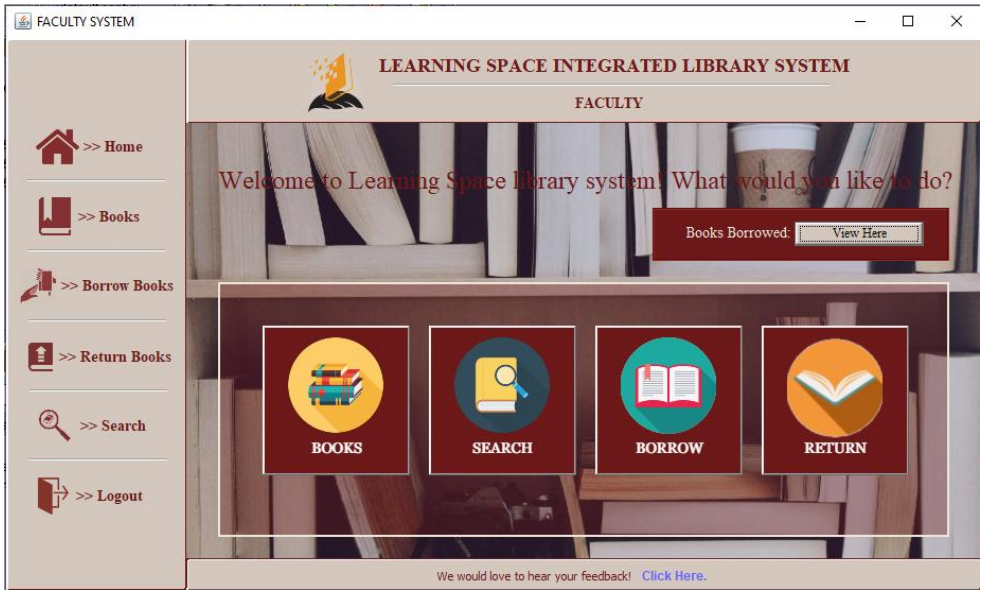
```
1   package LibrarySystemForms;
2   import java.awt.*;
3   import java.sql.ResultSet;
4   import java.sql.SQLException;
5   import java.util.logging.Level;
6   import java.util.logging.Logger;
7   import javax.swing.*;
8   import java.text.DateFormat;
9   import java.text.SimpleDateFormat;
10  import java.util.Calendar;
11  public class StudentSys extends LibConnect{
12      public StudentSys() {
13          initComponents();
            DoConnectBooks();
            DoConnectUsers();
            DoConnectIssueBooks();
17
18          CalendarReturnDateStud.setMinSelectableDate(thisDate);
19
20          TransparentStudent.setBackground(new Color(236, 240, 241, 90));
21          StudentHomePage.setVisible(true);
22          BookStudent.setVisible(false);
23          BookBorrowStudent.setVisible(false);
24          SearchStudent.setVisible(false);
25          ReturnBooksStudent.setVisible(false);
26
27      }
```

```
    private void StudentBorrowButtonActionPerformed(java.awt.event.ActionEvent evt) {
    UID = Integer.parseInt(EnterIDNoStud.getText());
        uidissue = String.valueOf(UID);
        BID = Integer.parseInt(EnterBIDStud.getText());
        bidissue = String.valueOf(BID);
        borrowdate = StudBorrow.getText();
        returndate = StudReturn.getText();

        try {
            stmt=con.createStatement();
            rs = stmt.executeQuery("SELECT * FROM BOOKS WHERE BOOKID="+bidissue+"");
            if(rs.next()) {
                ResultSet rsl = stmt.executeQuery("SELECT * FROM USERS WHERE IDNO="+uidissue+"");
                if(rsl.next()) {
                    stmt.executeUpdate("INSERT INTO BORROW_BOOKS VALUES("+uidissue+","+bidissue+", '"+borrowdate+"', "
                            + "'"+returndate+"')");
                    JOptionPane.showMessageDialog(null, "Thank you! Your book borrow request is now pending.");
                    EnterIDNoStud.setText("");
                    StudBorrow.setText("");
                    StudReturn.setText("");
                    EnterBIDStud.setText("");
                    TitleStud.setText("");
                }else{
                    JOptionPane.showMessageDialog(null, "Incorrect ID Number!");
                }
            }else{
                JOptionPane.showMessageDialog(null, "Incorrect BookID!");
            }
        }catch(Exception e) {
            JOptionPane.showMessageDialog(null, e);
```

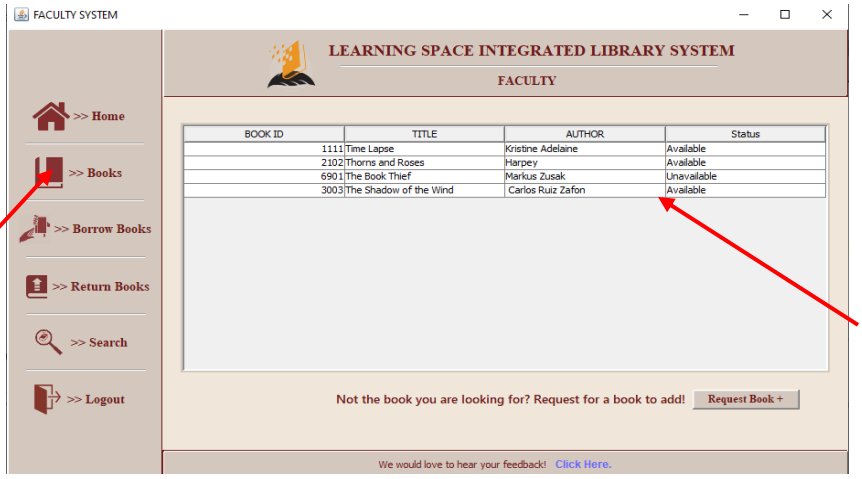
```
25              ReturnBooksStudent.setVisible(false);
26
27          }
28          public String addSubtractDate(int n){
29          DateFormat dateFormat = new SimpleDateFormat("MM/dd/Y");
30          Calendar cal = Calendar.getInstance();
31          cal.add(Calendar.DATE, n);
            String result = dateFormat.format(cal.getTime());
33          return result;
34      }
            {
36      }void GetBorrowRecords(){
                try {
38                  userID = rs.getInt("IDNO");
39                  bookID = rs.getInt("BOOKID");
40                  title = rs.getString("TITLE");
41                  curdate = rs.getString("CURRENTDATE");
42                  redate = rs.getString("RETURNDATE");
43              }catch (SQLException err){
44                  JOptionPane.showMessageDialog(StudentSys.this, err.getMessage());
45              }
46      } void CancelBtn() {
47          StudentHomePage.setVisible(true);
48          BookStudent.setVisible(false);
49          BookBorrowStudent.setVisible(false);
50          ReturnBooksStudent.setVisible(false);
51          SearchStudent.setVisible(false);
52      }
53
```

```
    private void SearchIDActionPerformed(java.awt.event.ActionEvent evt) {
    SearchTitle.setEnabled(false);
        try{
            pst = con.prepareStatement("SELECT BOOKID, STATUS FROM BOOKS WHERE BOOKID=?");
            booksrch = Integer.parseInt(BrrwsrchID.getText());
            pst.setInt(1, booksrch);
            rs = pst.executeQuery();

            if(rs.next()==false){
                JOptionPane.showMessageDialog(null, "Sorry! Record not found.");
                srchstatus.setText("");

            }else{
                srchstatus.setText(rs.getString("STATUS"));
            }

        }catch (Exception e) {
            JOptionPane.showMessageDialog(rootPane, e);
//              Logger.getLogger(StudentRegistrationForm.class.getName()).log(Level.SEVERE, null, ex);
        }
```

USER MODULE
(Faculty)



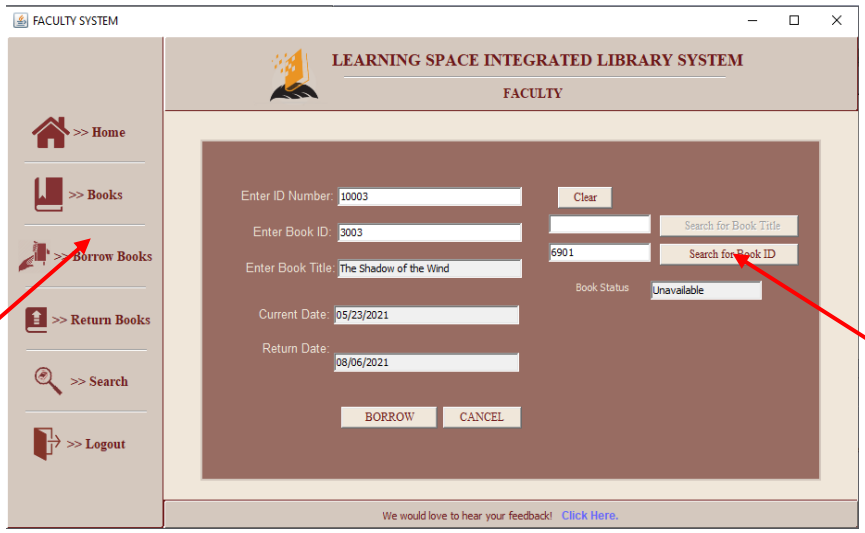
After the user logged in as 'faculty' in member type, the user will be directed to this HOME page of the faculty module. The faculty module is identical to the student module in terms of functionality. The only difference between the two modules is the color, which makes it easy to tell which sort of users are they.



BOOKS button

Table that contains book information

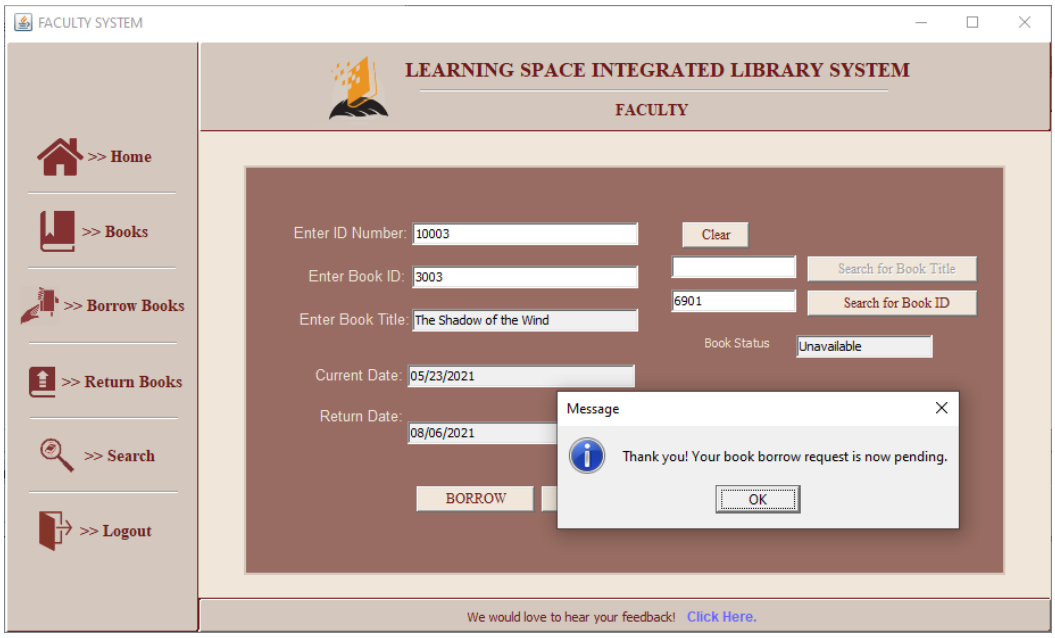
The BOOKS page is where the faculty can find all the necessary information about the books, including its books ID, title author as well as the status of the book.



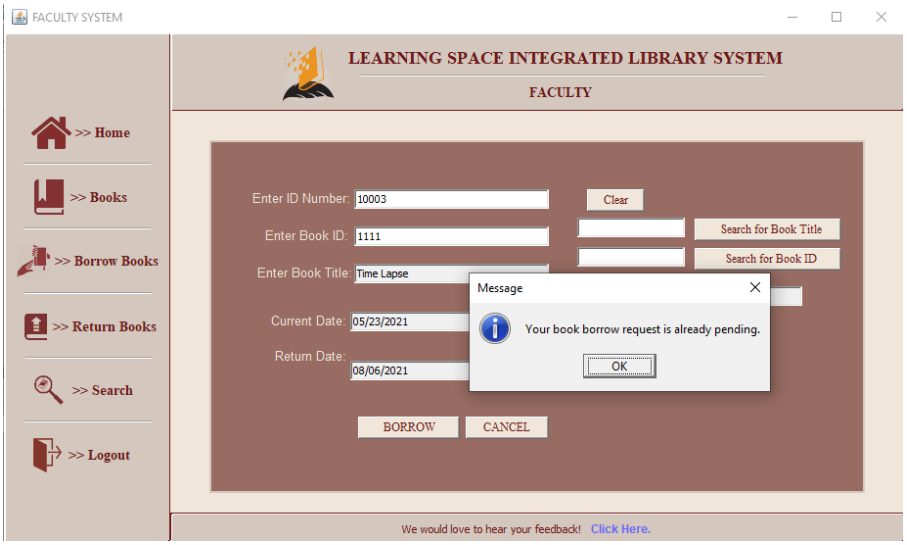
BORROW BOOKS button

Displays the status of a books

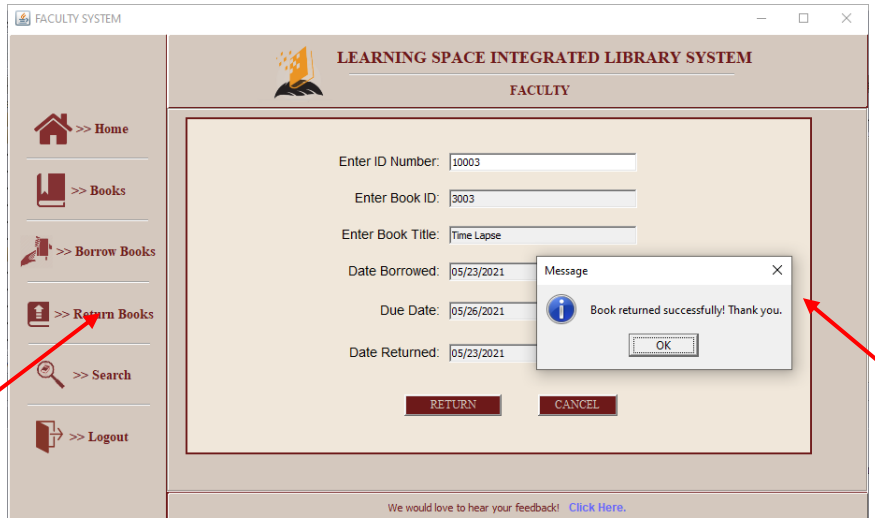
In BORROW BOOKS page this is where the faculty can borrow a book by entering book ID and their ID number. The book title, current date, and return date is already auto generated therefore faculty doesn't have to fill that information. They can also search a book by their title and book ID in-order to see the status of the book



A JOption confirmation message will pop whenever a faculty member borrows a book stating that the book borrow request is now pending.



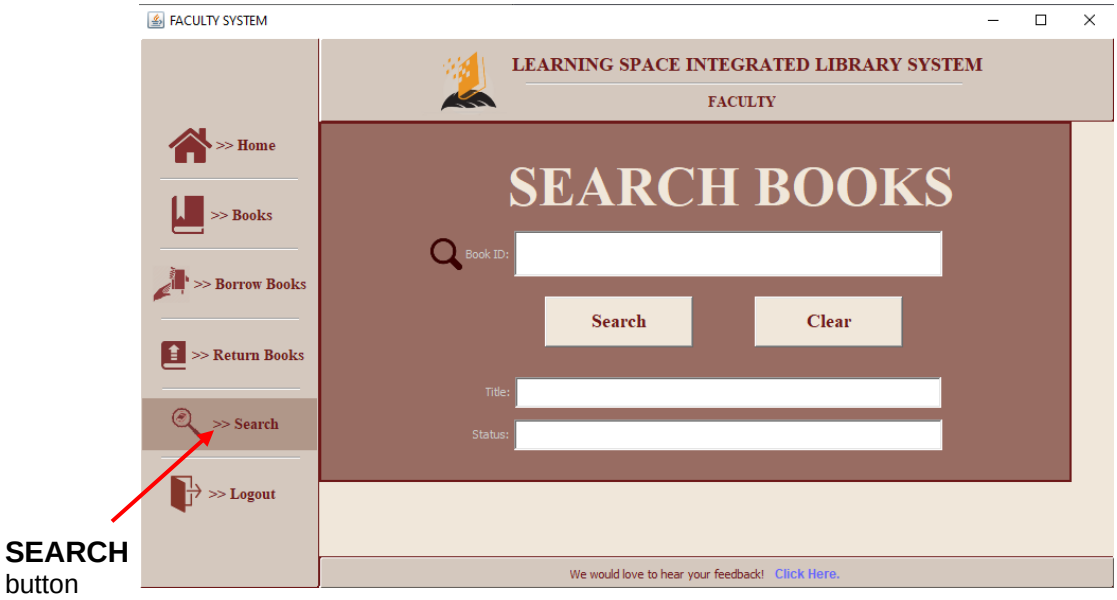
If the faculty already requested a book exact same book, the system will tell the faculty that the request is already pending to avoid duplication.



RETURN BOOKS button

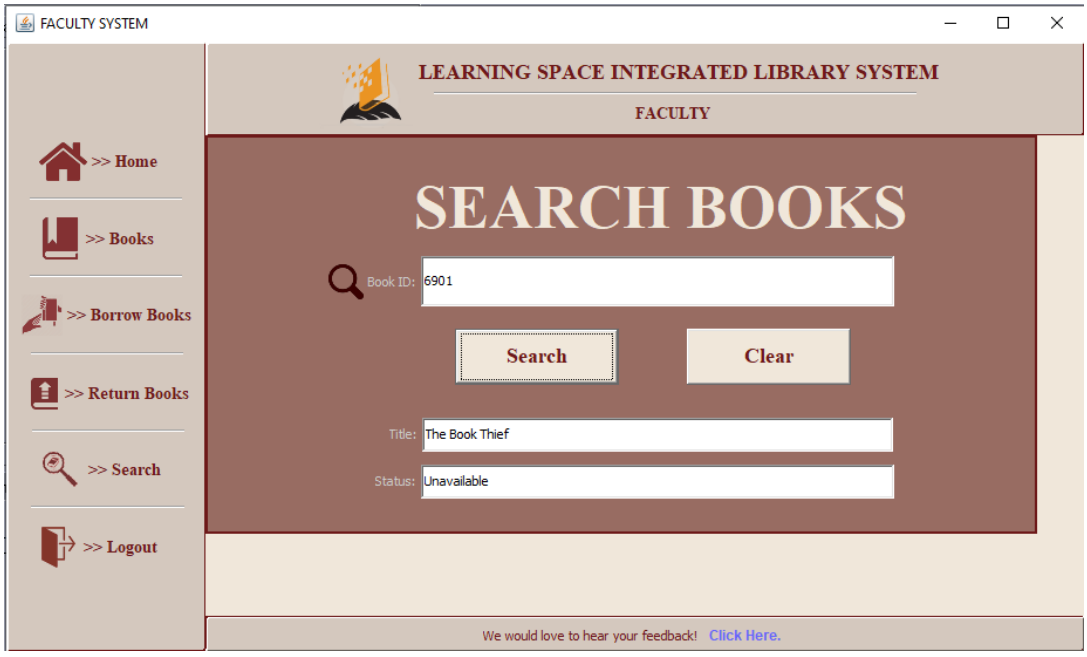
Confirmation message stating the book is returned.

In RETURN BOOK page, this is where the faculty can return the books they borrowed from the library. They only have to enter their ID number since the rest of information is auto-generated.

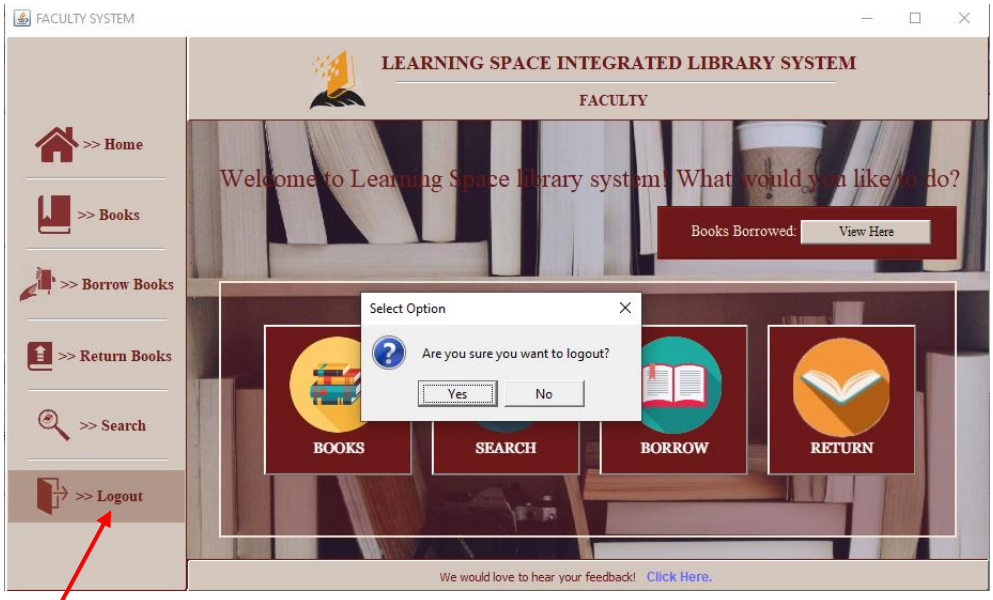


SEARCH button

In SEARCH page, the faculty can find a book by searching the book ID. The status and title of the book will be display after clicking the search button. The clear button clears all input.

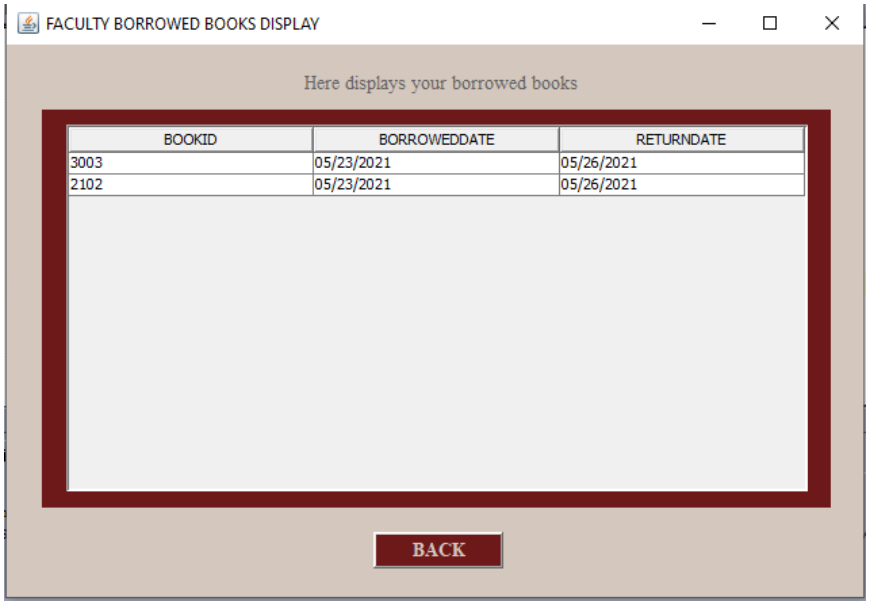


Search page with input



The LOGOUT button will redirect the faculty user to login form and the system will the faculty twice before logging out.

LOGOUT
button



This BORROWED BOOKS DISPLAY will display all the borrowed book of the faculty as well as the borrowed and return date of the book.

CODE

```
UID = Integer.parseInt(EnterIDNoFac.getText());
    uidissue = String.valueOf(UID);
    BID = Integer.parseInt(EnterBIDFac.getText());
    bidissue = String.valueOf(BID);
    borrowdate = FacBorrow.getText();
    returndate = FacReturn.getText();

    try {
        stmt=con.createStatement();
        rs = stmt.executeQuery("SELECT * FROM BOOKS WHERE BOOKID="+bidissue+"");
        if(rs.next()) {
            ResultSet rsl = stmt.executeQuery("SELECT * FROM USERS WHERE IDNO="+uidissue+"");
            if(rsl.next()) {
                stmt.executeUpdate("INSERT INTO BORROW_BOOKS VALUES("+uidissue+","+bidissue+", '"+borrowdate+"', "
                        + "'"+returndate+"')");
                JOptionPane.showMessageDialog(null, "Thank you! Your book borrow request is now pending.");
                EnterIDNoFac.setText("");
                FacBorrow.setText("");
                FacReturn.setText("");
                EnterBIDFac.setText("");
                TitleFac.setText("");
            }else{
                JOptionPane.showMessageDialog(null, "Incorrect ID Number!");
            }
        }else{
            JOptionPane.showMessageDialog(null, "Incorrect BookID!");
        }
    }catch(Exception e) {
        JOptionPane.showMessageDialog(null, e);
    }
```

```
private void BookReturnFacActionPerformed(java.awt.event.ActionEvent evt) {

        Uuid = Integer.parseInt(UIDReturnFac.getText());
        UUsuid = String.valueOf(Uuid);

        BBbid = Integer.parseInt(BIDReturnFac.getText());
        Bbsuid = String.valueOf(BBbid);

        Btitle = TitleReturnFac.getText();
        borrowd = FacCurrentDate.getText();
        duedate = FacReturnDate.getText();
        retdate = DateTodayFac.getText();
        //rdate= sdf.format (CalendarReturnDateStud.getDate());

    try {

        borrow = "INSERT INTO RETURN_BOOKS VALUES (?,?,?,?,?)";
        pst = con.prepareStatement(borrow);

        pst.setString(1, UUsuid);
        pst.setString(2, Bbsuid);
        pst.setString(3, duedate);
        pst.setString(4, borrowd);
        pst.setString(5, retdate);
        pst.execute();

        JOptionPane.showMessageDialog(null, "Book returned successfully! Thank you.");
```

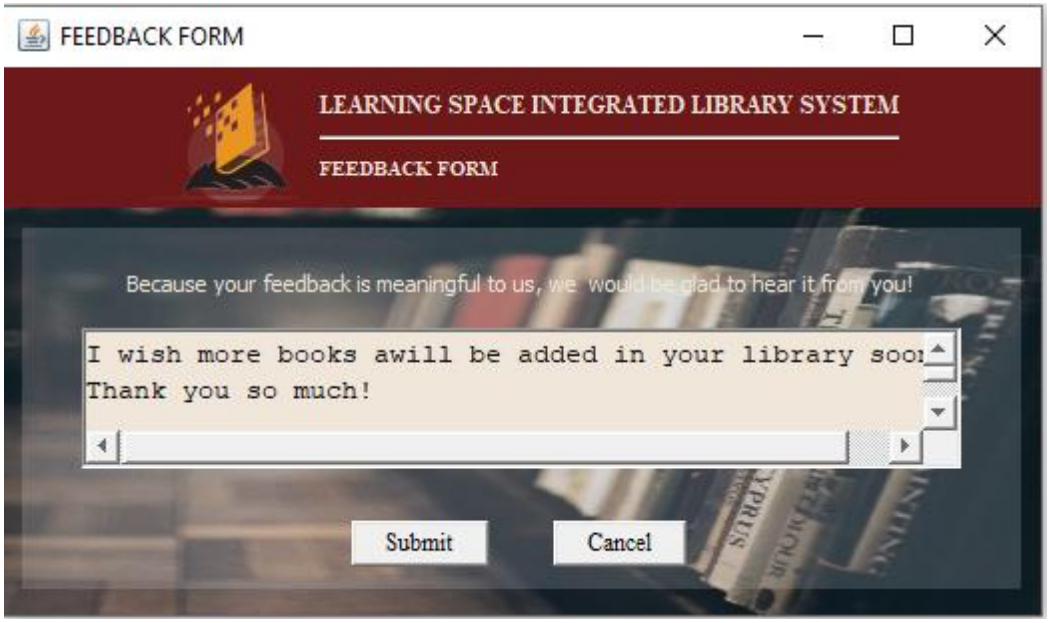
```
private void SearchIDActionPerformed(java.awt.event.ActionEvent evt) {
    SearchTitle.setEnabled(false);
    try{
        pst = con.prepareStatement("SELECT BOOKID, STATUS FROM BOOKS WHERE BOOKID=?");
        booksrch = Integer.parseInt(BrrwsrchID.getText());
        pst.setInt(1, booksrch);
        rs = pst.executeQuery();

        if(rs.next()==false){
            JOptionPane.showMessageDialog(null, "Sorry! Record not found.");
            srchstatus.setText("");

        }else{
            srchstatus.setText(rs.getString("STATUS"));
        }

    }catch (Exception e) {
        JOptionPane.showMessageDialog(rootPane, e);
        //            Logger.getLogger(StudentRegistrationForm.class.getName()).log(Level.SEVERE, null, ex);
    }
```

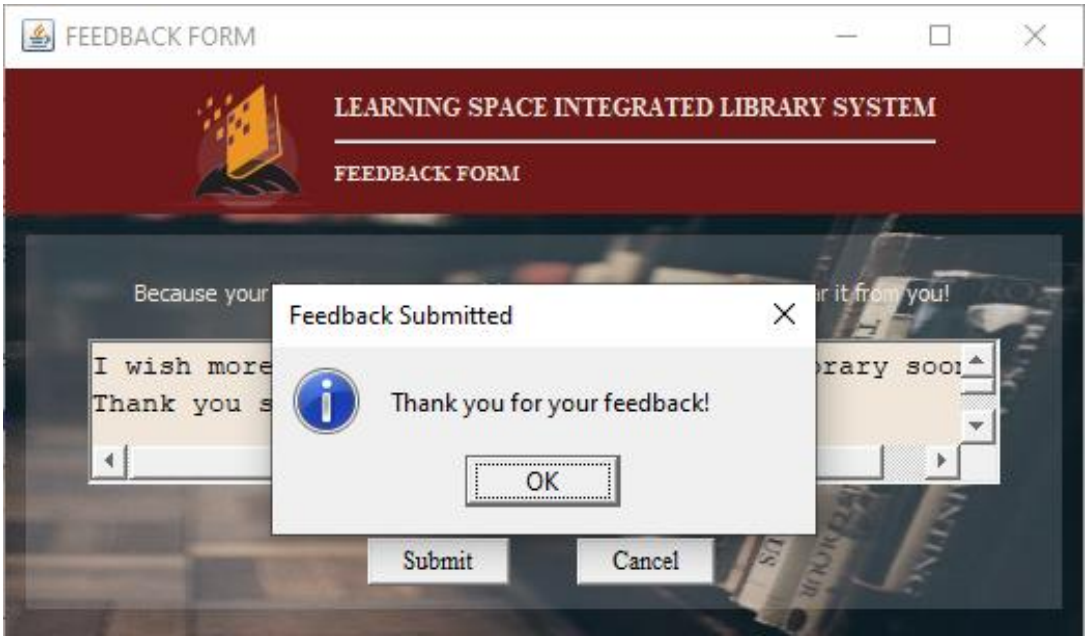
```
private void SearchTitleActionPerformed(java.awt.event.ActionEvent evt) {
    SearchID.setEnabled(false);
    try{
        pst = con.prepareStatement("SELECT TITLE, STATUS FROM BOOKS WHERE TITLE=?");
        srchbook = BrrwsrchTitle.getText();
        pst.setString(1, srchbook);
        rs = pst.executeQuery();

        if(rs.next()==false){
            JOptionPane.showMessageDialog(null, "Sorry! Record not found.");
            srchstatus.setText("");

        }else{
            srchstatus.setText(rs.getString("STATUS"));
        }

    }catch(Exception e) {
        JOptionPane.showMessageDialog(rootPane, e);
    }
}
```

```
private void clickbooksearchActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        pst = con.prepareStatement("SELECT BOOKID, TITLE, STATUS FROM BOOKS WHERE BOOKID=?");
        booksrch = Integer.parseInt(bookid.getText());
        pst.setInt(1, booksrch);
        rs = pst.executeQuery();

        if(rs.next()==false){
            JOptionPane.showMessageDialog(null, "Sorry! Record not found.");
            bookid.setText("");
            booktitle.setText("");
            bookstatus.setText("");
        }else{
            booktitle.setText(rs.getString("TITLE"));
            bookstatus.setText(rs.getString("STATUS"));
        }

    }catch (SQLException ex) {
        Logger.getLogger(StudentRegistrationForm.class.getName()).log(Level.SEVERE, null, ex);
        //JOptionPane.showMessageDialog(this, e);
    }
}
```

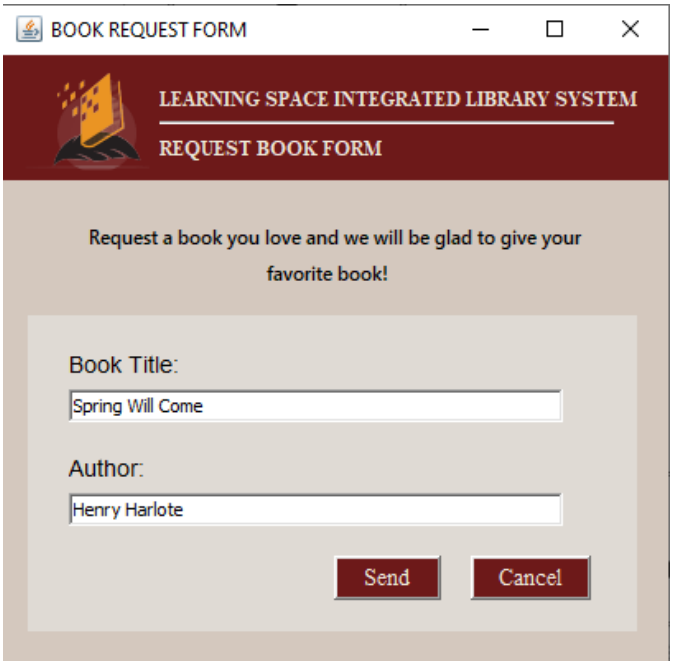
```
private void logoutfacMousePressed(java.awt.event.MouseEvent evt) {
    int response = JOptionPane.showConfirmDialog(this, "Are you sure you want to logout?", "Select Option",
            JOptionPane.YES_NO_OPTION);
    if(response == 0) {
        LoginFormNew logout = new LoginFormNew();
        logout.setVisible(true);
        this.dispose();
    }else{
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

OTHER SYSTEM MODULES



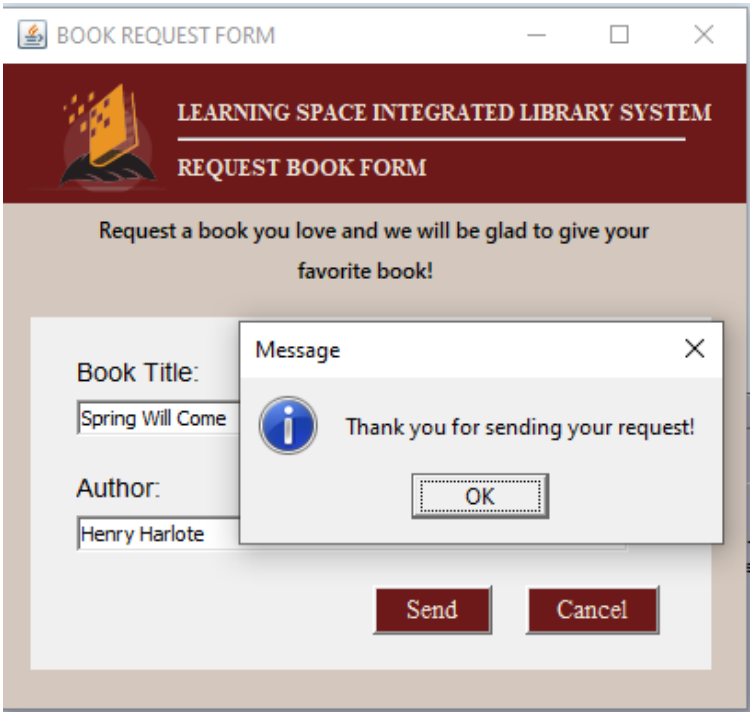


Feedback form wherein users of the library, particularly for students and faculty members are free to give their feedback about the library system. This is for design purpose only.



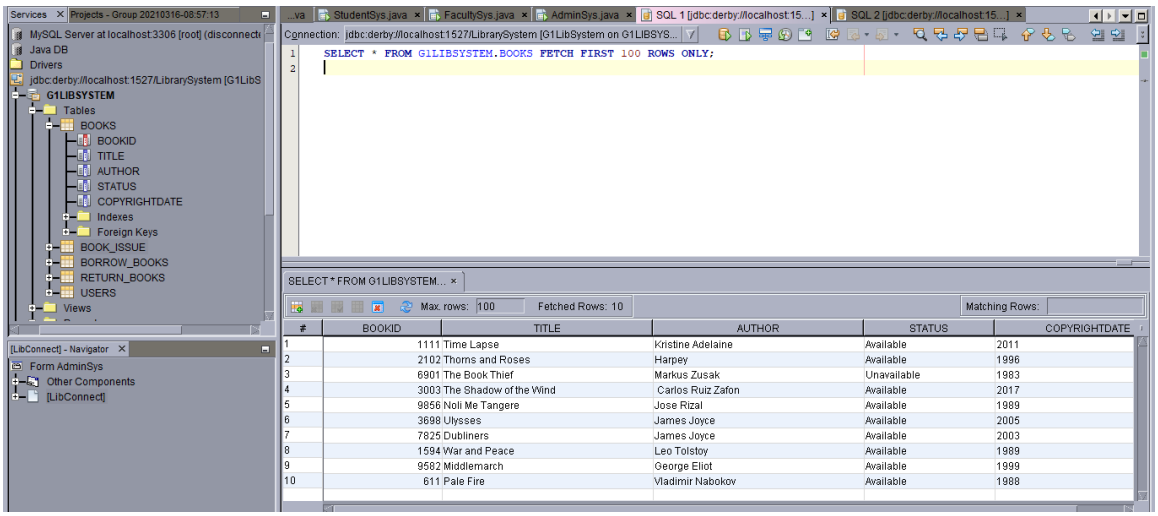
The users of the library system may request for a book they want by clicking the request book button. Users will be automatically directed to this form. particularly for This is for design purpose only.

A message will appear that the book requested has been sent. This is only for the design of the system and will not reflect onto any of the database system.

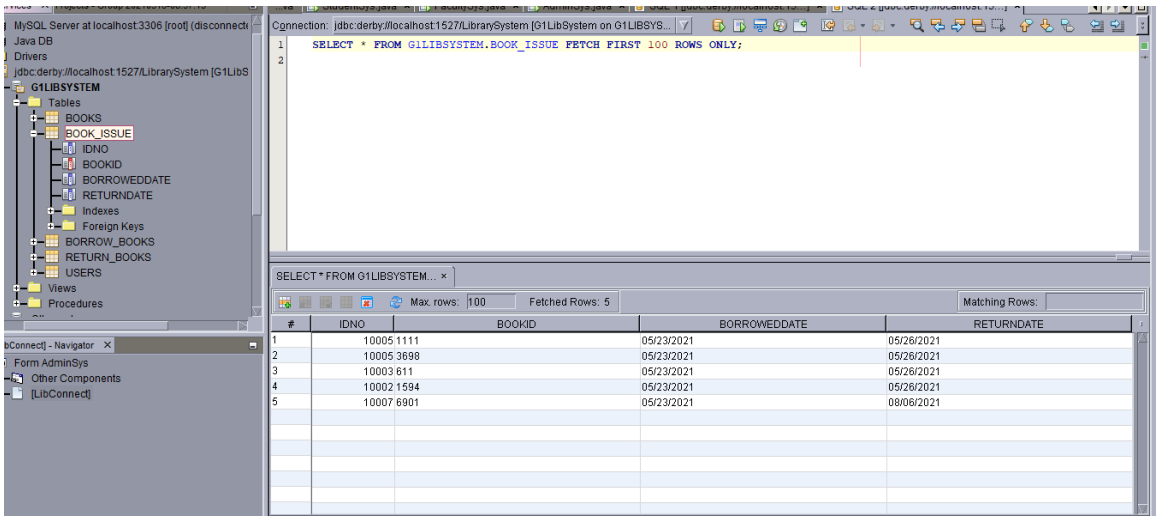


REPORT GENERATION MODULE

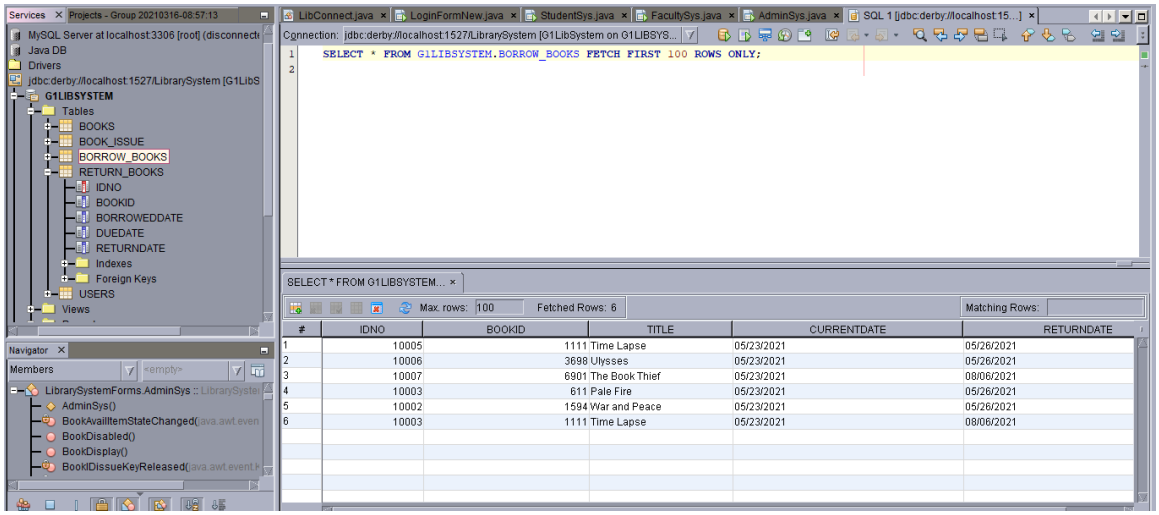
BOOK TABLE – includes the book information



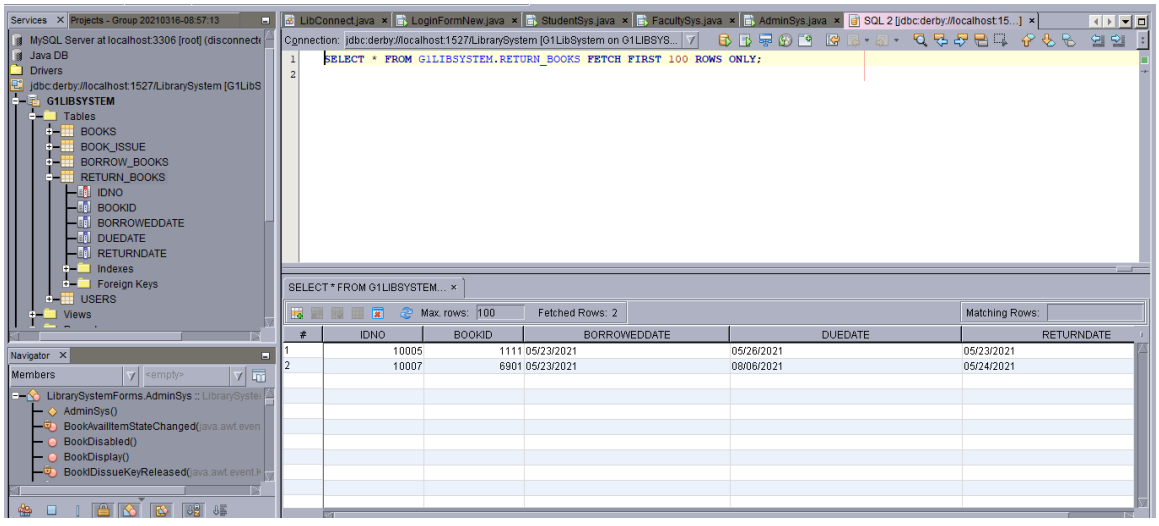
BOOK _ISSUE TABLE – this is where the issued books records will be inserted



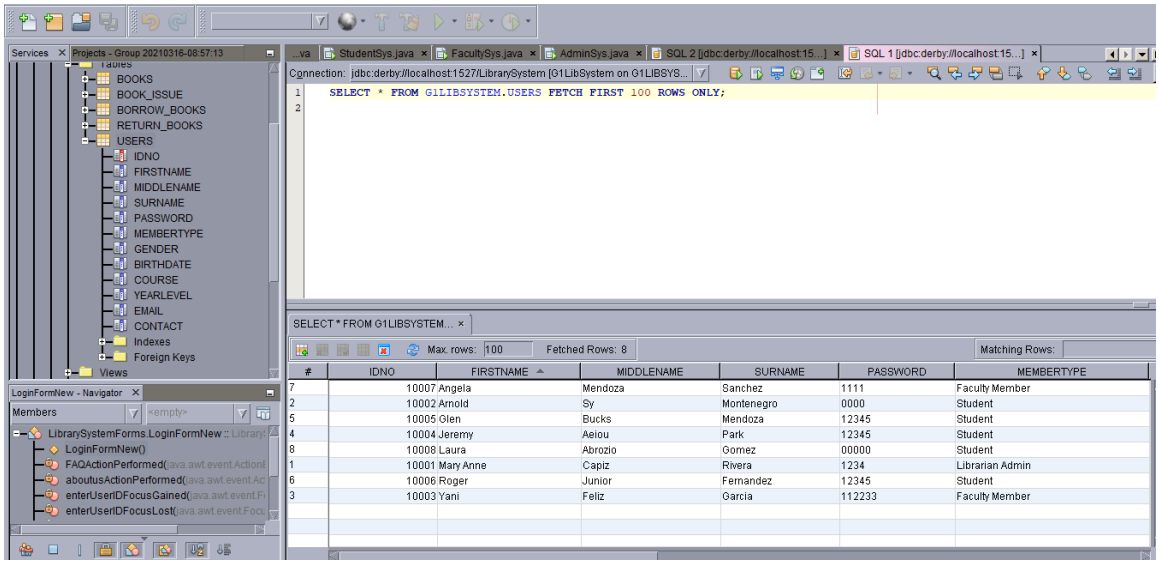
BORROW _BOOKS TABLE – this is where the borrowed books from the library users' records will be inserted

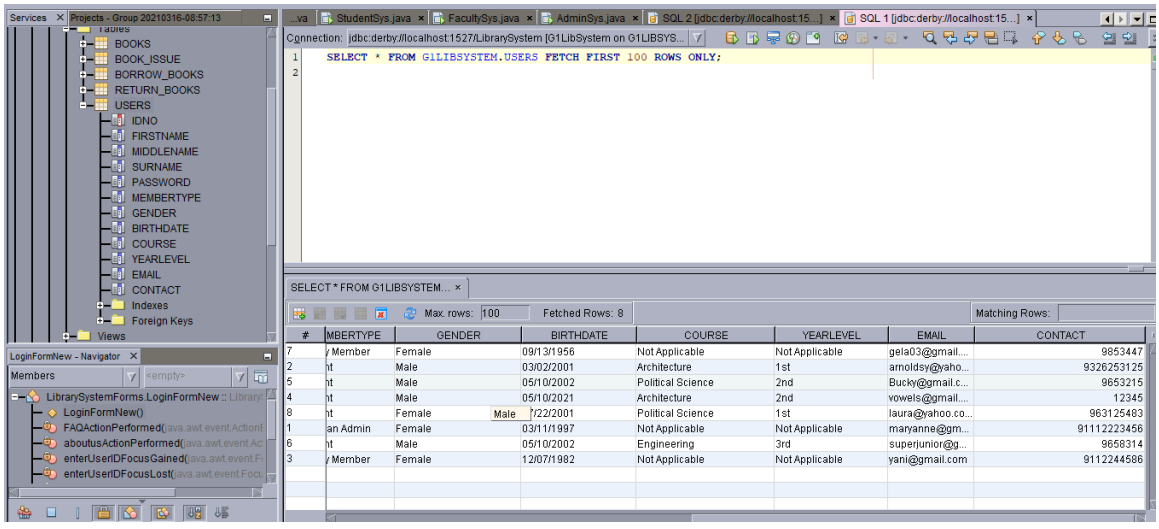


RETURN _BOOKS TABLE – this is where the books returned by the library users' records will be inserted



USERS TABLE – is where the records of the registered user will be inserted from input in the register fields in the registration form





CODE

```
package LibrarySystemForms;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
public class LibConnect extends javax.swing.JFrame {
    Connection con;
    Statement stmt;
    ResultSet rs;
    PreparedStatement pst;

    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");
    Date thisDate = new Date();
    DefaultTableModel model = new DefaultTableModel();

    //USERS LOGIN/REGISTRATION
    int newID, id, temp_user, curRow, result, lastid;
    String newPASS, newUSERTYPE, user, pass, memtyp, temp_pass, temp_usertype;
    String fn, mn, ln, bdate, gender, crs, yrlvl, email, pw, contact;
    String x, y, confirm_pass, query;
```

```
    //BOOKS
    int bookID, newbookID, temp_bid, bookresult, BID, newbid;
    String title, author, publisher, i, newtitle,newauthor,newpub, status, curdate, redate;
    int newIDno, booksrch, duefine;;
    String s, Idno, Borrowsearch, srchstatus, bookquery, sbid;
    int row, click;
    String value, selectbooks, str_status, selectusers, selectborrowers, value1, rdate, fdate;
    String bidissue, uidissue, returndate, borrowdate, bidreturn, uidreturn, bookreturn;
    String dateissue, datereturn, viewbooks, days, srchbook, copyright,rsl2a;


    //USERS DISPLAY
    int  UID, newuserID, userID,temp_uid, newuid;
    String fname, mname, sname, passw, mt, gndr, suid, bdte, crse, yrlevel, eml, scntct, userquery,  cntct;
    String str_uid, str_bid, btitle, issuedate, returnbookissue, srchuid, srchbid, fetch_uid, fine,srhbidf;

    //STUDENT BORROW

    int IDNo, BookID, Bbid, BBbid, Uuid;
    String BUID,Bsuid,Bfname,Bmname,Bsname,Bpassw,Bbsuid,Btitle,Bauthor,duedate,borrowd,borrow,UUsuid;

    //FACULTY BORROW
    int  Fbid,Fabid;
    String FFsuid,Facsuid,Ftitle,Fdate,borrowf,fdater, retdate;
```

```
    public void DoConnectUsers(){
        try{
            //CONNECT TO THE DATABASE
            String host = "jdbc:derby://localhost:1527/LibrarySystem";
            String uName = "G1LibSystem";
            String uPass = "12345";
            con = DriverManager.getConnection(host, uName, uPass);

            //EXECUTE SOME SQL AND LOAD THE RECORDS INTO THE RESULTSET
            stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_UPDATABLE);
            String sql = "SELECT * FROM G1LIBSYSTEM.USERS";

            pst = con.prepareStatement(sql);
            rs = stmt.executeQuery(sql);
            con.commit();

        }catch(SQLException err){
            JOptionPane.showMessageDialog(LibConnect.this, err.getMessage());
        }
    }
```

```
72
73  public void RefreshUsers(){
74          try{
75              stmt.close();
76              rs.close();
77              stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
78                      ResultSet.CONCUR_UPDATABLE);
79              String sql = "SELECT * FROM G1LIBSYSTEM.USERS";
80              rs = stmt.executeQuery(sql);
81          }catch(SQLException ex) {
82              Logger.getLogger(LibConnect.class.getName()).log(Level.SEVERE, null, ex);
83          }
84  }public void DoConnectBooks(){
85          try {
86              String url = "jdbc:derby://localhost:1527/LibrarySystem";
87              String uName = "G1LibSystem";
88              String uPass = "12345";
89              con = DriverManager.getConnection(url, uName ,uPass);
90              stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
91                      ResultSet.CONCUR_UPDATABLE);
92              String sql ="SELECT * FROM G1LIBSYSTEM.BOOKS";
93              rs =stmt.executeQuery(sql);
94              pst = con.prepareStatement(sql);
95              rs.next();
96              con.commit();
97
98          }catch (SQLException err){
99             JOptionPane.showMessageDialog(LibConnect.this, err.getMessage());
100         }
101
```

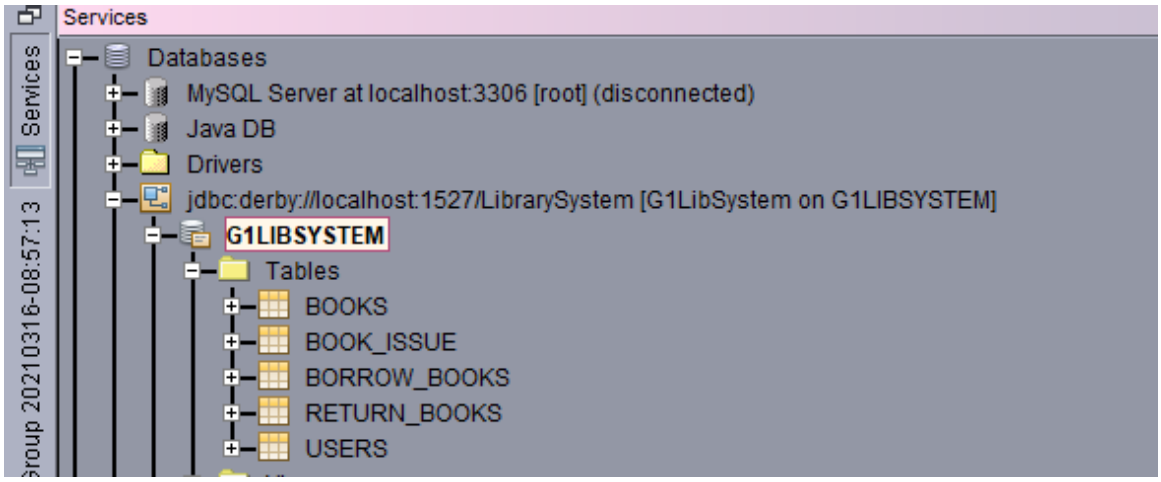
```
101
102 } public void RefreshBooks(){
103         try{
104             stmt.close();
105             rs.close();
106             stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
107                     ResultSet.CONCUR_UPDATABLE);
108             String sql ="SELECT * FROM G1LIBSYSTEM.BOOKS";
109             rs =stmt.executeQuery(sql);
110         }catch (SQLException ex){
111             Logger.getLogger(LibConnect2.class.getName()).log(Level.SEVERE, null, ex);
112         }
113
114 }    public void DoConnectIssueBooks(){
115         try{
116             //CONNECT TO THE DATABASE
117             String host = "jdbc:derby://localhost:1527/LibrarySystem";
118             String uName = "G1LibSystem";
119             String uPass = "12345";
120             con = DriverManager.getConnection(host, uName, uPass);
121
122             //EXECUTE SOME SQL AND LOAD THE RECORDS INTO THE RESULTSET
123             stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
124                     ResultSet.CONCUR_UPDATABLE);
125             String sql = "SELECT * FROM G1LIBSYSTEM.BOOK_ISSUE";
126
127             pst = con.prepareStatement(sql);
128             rs = stmt.executeQuery(sql);
129             con.commit();
130
```

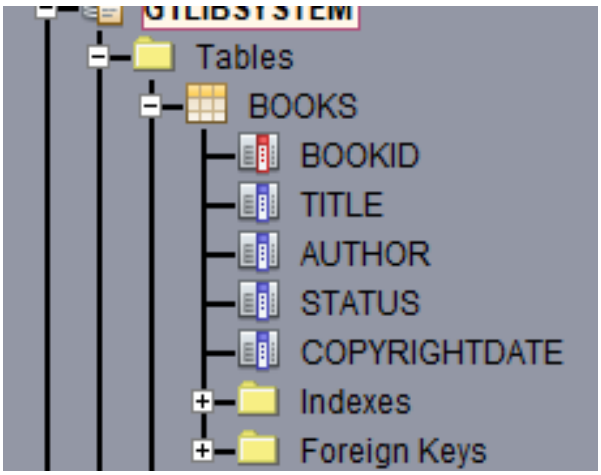
```
    }
    public void DoConnectBorrowersBooks(){
    try{
        //CONNECT TO THE DATABASE
        String host = "jdbc:derby://localhost:1527/LibrarySystem";
        String uName = "G1LibSystem";
        String uPass = "12345";
        con = DriverManager.getConnection(host, uName, uPass);

        //EXECUTE SOME SQL AND LOAD THE RECORDS INTO THE RESULTSET
        stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_UPDATABLE);
        String sql = "SELECT * FROM G1LIBSYSTEM.BORROW_BOOKS";

        pst = con.prepareStatement(sql);
        rs = stmt.executeQuery(sql);
        con.commit();

    }catch(SQLException err){
        JOptionPane.showMessageDialog(LibConnect.this, err.getMessage());
    }
```

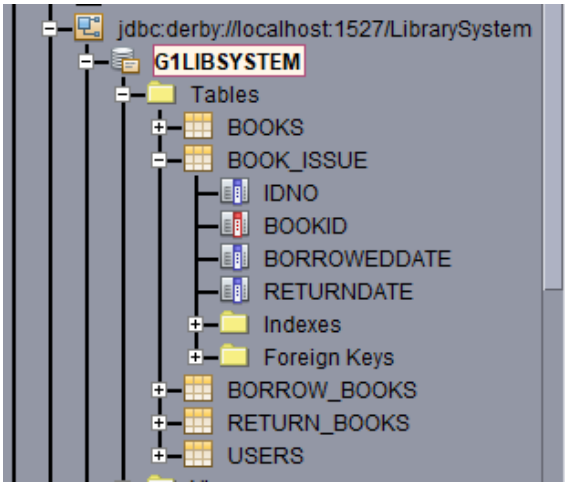
```
154             }
155             }public void DoConnectReturnBooks(){
156             try {
157                 String url = "jdbc:derby://localhost:1527/LibrarySystem";
158                 String uName = "G1LibSystem";
159                 String uPass = "12345";
160                 con = DriverManager.getConnection(url, uName ,uPass);
161                 stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
162                         ResultSet.CONCUR_UPDATABLE);
163                 String sql ="SELECT * FROM G1LIBSYSTEM.RETURN_BOOKS";
164                 rs =stmt.executeQuery(sql);
165                 pst = con.prepareStatement(sql);
166                 rs.next();
167                 con.commit();
168
169             }catch (SQLException err){
170                JOptionPane.showMessageDialog(LibConnect.this, err.getMessage());
171             }
172     }
173     }
```

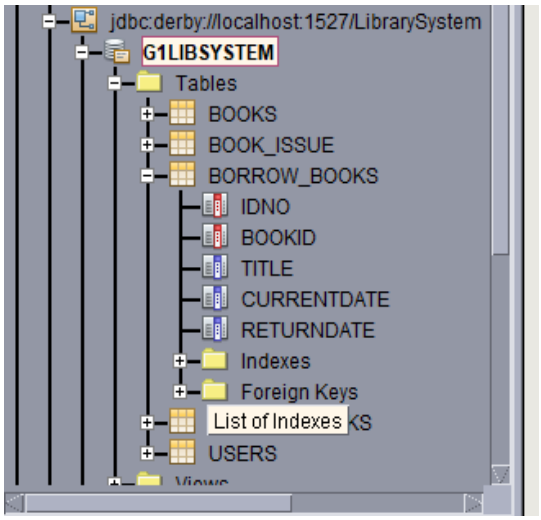
```
public void RefreshUsers(){
        try{
            stmt.close();
            rs.close();
            stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_UPDATABLE);
            String sql = "SELECT * FROM G1LIBSYSTEM.USERS";
            rs = stmt.executeQuery(sql);
        }catch(SQLException ex) {
            Logger.getLogger(LibConnect.class.getName()).log(Level.SEVERE, null, ex);
        }
```

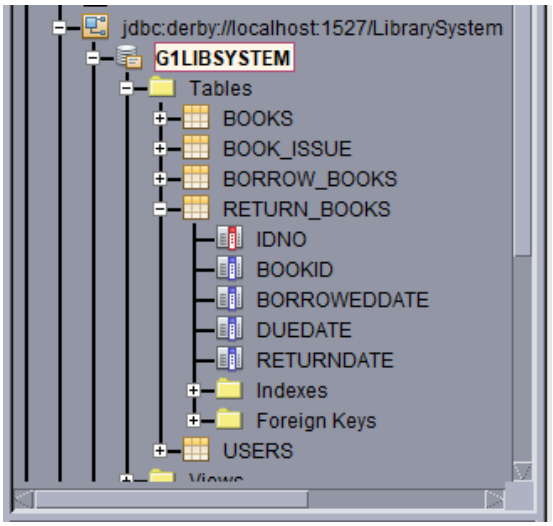
DATABASE SCREENSHOTS (database creation and tables)

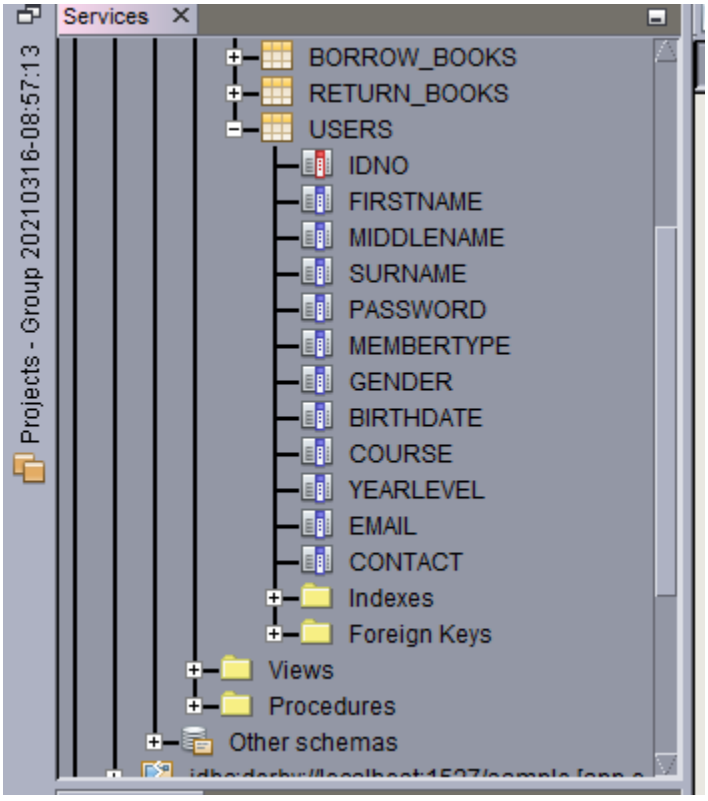







jdbc:derby://localhost:1527/LibrarySystem
G1LIBSYSTEM
Tables
BOOKS
BOOK_ISSUE
BORROW_BOOKS
IDNO
BOOKID
TITLE
CURRENTDATE
RETURNDATE
Indexes
Foreign Keys
List of Indexes
USERS


jdbc:derby://localhost:1527/LibrarySystem
G1LIBSYSTEM
Tables
BOOKS
BOOK_ISSUE
BORROW_BOOKS
RETURN_BOOKS
IDNO
BOOKID
BORROWEDDATE
DUEDATE
RETURNDATE
Indexes
Foreign Keys
USERS


Services
Projects - Group 20210316-08:57:13
BORROW_BOOKS
RETURN_BOOKS
USERS
IDNO
FIRSTNAME
MIDDLENAME
SURNAME
PASSWORD
MEMBERTYPE
GENDER
BIRTHDATE
COURSE
YEARLEVEL
EMAIL
CONTACT
Indexes
Foreign Keys
Views
Procedures
Other schemas

SYSTEM FINDINGS

This section outlines the group's accomplishments in terms of improving the system. The highlighted **YELLOW** indicates a successful change in the system while **RED** represents a failed system modification.

- Admin System should have a list of books that have been borrowed by the users(a tabula in the admin system).

- Add a copyright date in the 'book' information.

- The book borrowed date and return date should be auto-generated.

- When borrowing (and returning) a book, the date should be the current date.

Limit the books that have been borrowed per user (Add number of books).

- Limit the number of days that a user can borrow a book.
  For the student: 3-4 days (as suggested)
  For faculty: per semester (as suggested)

- After requesting a book, the user should be redirected to the student or faculty system.

- When sending a feedback, the user should not be logged-out or be redirected to the log-in page.

- The ID number should be auto-generated after you logged in to the student or faculty system. This should not require for the user to re-type his/her ID number again when borrowing.

- When searching a book in the borrow book menu, the book title should also be searchable. In other words, add a filter for book search.

- Replace 'active' and 'inactive' status with 'available' and 'unavailable' (inactive and active status are for users only.)

- Notify the librarian if a user borrowed and returned a book (a tabula in the admin system).

- In the registration form, hide both ‘password’ and ‘confirm password'.

- Book status’ should be automatically updated (after borrowing a book.)

- Create an approval option/click list or table of borrow information (who requested, book title, book id. Etc.) **The admin must not re-type the information of the borrower.**