DBMS PROJECT

Library Management System

Guided by

Prof. Hari Seetha

TEAM MEMBERS

Parepalli sumanth 17BCE7088

ADitya jain 17BCE7066

perumalla ganesh 17bce7038

**Abstract:**

This project is developed for monitoring and controlling transactions in library in an efficient and systematic way rather than relying on the age-old system of registers and log books. This project is gaining more importance as the number of users are increasing rapidly. The project has been developed in Java and uses MySQL for backend. It has all he canned transactions which are required adding a user, adding a book, changing the book details, issue, return, renew etc. but it has also has features which are not available in other library management systems like waiting list for a book and getting notified about all transactions through email and also getting notified when the due date for a book is approaching or has been crossed.

Overall this project has been developed to help the students and librarian for smooth operation of the library and also reduce human efforts.

Contents

[Requirement Analysis: 3](#_Toc4061741)

[Database Requirements 3](#_Toc4061742)

[Functional Requirements 3](#_Toc4061743)

[ENTITY RELATIONSHIP DIAGRAM 4](#_Toc4061744)

[ER to Relational Mapping 5](#_Toc4061745)

[FOREIGN KEY RELATIONSHIPS 6](#_Toc4061746)

[Relational Database Design: Normalization 7](#_Toc4061747)

[Implementation 7](#_Toc4061748)

[Hardware Requirements 7](#_Toc4061749)

[Software Requirements 7](#_Toc4061750)

[Software/Tools Used 7](#_Toc4061751)

[Screenshots with Description 7](#_Toc4061752)

[FLOW OF CONTROL 21](#_Toc4061753)

[Screenshots of database 21](#_Toc4061754)

[Student Table 21](#_Toc4061755)

[Books Table 22](#_Toc4061756)

[Issue Table 22](#_Toc4061757)

[Returns Table 23](#_Toc4061758)

[Renew Table 23](#_Toc4061759)

[Wait Table 24](#_Toc4061760)

[Conclusion: 36](#_Toc4061761)

[Scope for future work: 36](#_Toc4061762)

[References: 36](#_Toc4061763)

**Introduction:**

We have two logins available one is for the students and other one is for the librarian. The student will have options like issue, return, renew, account details and the librarian will have the options to add a new user, add a new book, change the details of a book, delete a book and view the summary of the library.

To facilitate searching a book in the library an auto-suggestion book has been designed which allows both the students and the librarian to search for a book using the ISBN number of the book, name of the book and author of the book. It’s difficult to remember the entire name of the book or all the authors of the book. So even if the user remembers some parts of the name of the book the auto-suggestion box will filter out results on the go and suggest to the user which makes the job easy and fast.

If a particular book has high demand and all of its copies have been issued and some other students intends to issue it but since the book is not available, the student cannot do it. To help the student what the system will do it will tell the expected date to the student when the book will be available again in the library and also will provide an option to stay in the waiting list so that when the book is available again it will be reserved for him and the student will be given priority on the basis of its number on the waiting list for that book and the student need not visit the library and again and again to check whether the book is available or not once the student has opted to wait in the waiting list when the book is available he will be notified through email and the student has to do come and issue the book within 24 hrs after the student has been notified.

The student will be informed about all the transactions which he does in the library via email so that the student can also keep track of it and also check for discrepancies if any. The system also notifies students for whom the due date is just two days from current date and also it reminds the students who have crossed the due date.

## Requirement Analysis:

### Database Requirements

* Books in the library are uniquely identified by their ISBNs. Details of book like its name, author, total no of books available, total no of books issued and whether its is a reference book.
* Each student is identified uniquely by his registration number and has other attributes like name, branch, email-id and a count of the number of books issued by him/her. The student can issue a max of three books.
* Record of dates of issue and return (which is 14 days after the book is issued) should be maintained and if a student fails to return the book on or before the return date, he/she shall be penalised with certain amount of fine.
* If a student wishes to use the further, he can go for a renewal for another 7 days from the previous due date
* If a student finds a book unavailable, he/she can be on a waiting list (max 10 students) and the students who are ahead in the waiting list will be given more priority than others.

### Functional Requirements

* Whenever a student does a transaction, he/she shall be notified with an email.
* When a student fails to return the book by due date, he/she shall be penalised by certain amount of fine that is calculated as follows:

Fine=(sysdate-duedate)\*2

* If a book is not available currently, the student may join waiting list of the book and get notified when available.
* Borrower should be notified when the due date is nearby and if he/she has crossed the due date he/she shall be notified regarding that.

## ENTITY RELATIONSHIP DIAGRAM



## ER to Relational Mapping

**BOOKS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *ISBN* | *Book\_Name* | *Author* | *IQty* | *OQty* | *IsReference* |

**STUDENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Reg.No* | *Name* | *Branch* | *Book\_Cnt* | *Email\_ID* |

**ISSUE**

|  |  |  |  |
| --- | --- | --- | --- |
| *ISBN* | *Reg.No* | *Issue\_Date* | *Due\_Date* |

**RETURN**

|  |  |  |  |
| --- | --- | --- | --- |
| *ISBN* | *Reg.No* | *Fine* | *Timestamp* |

**RENEW**

|  |  |  |
| --- | --- | --- |
| *ISBN* | *Reg.No* | *Status* |

**WAIT**

|  |  |  |  |
| --- | --- | --- | --- |
| *ISBN* | *Reg.No* | *Position* | *Emsent* |

## FOREIGN KEY RELATIONSHIPS

**BOOKS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *ISBN* | *Book\_Name* | *Author* | *IQty* | *OQty* | *IsReference* |

**STUDENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Reg.No* | *Name* | *Branch* | *Book\_Cnt* | *Email\_ID* |

**ISSUE**

|  |  |  |  |
| --- | --- | --- | --- |
| *ISBN* | *Reg.No* | *Issue\_Date* | *Due\_Date* |

**RETURN**

|  |  |  |  |
| --- | --- | --- | --- |
| *ISBN* | *Reg.No* | *Fine* | *Timestamp* |

**RENEW**

|  |  |  |
| --- | --- | --- |
| *ISBN* | *Reg.No* | *Status* |

**WAIT**

|  |  |  |  |
| --- | --- | --- | --- |
| *ISBN* | *Reg.No* | *Position* | *Emsent* |

## Relational Database Design: Normalization

* All the tables are already in Boyce Codd Normal form and hence no further normalisation is required.

## Implementation

### Hardware Requirements

#### Minimum Requirements

|  |  |
| --- | --- |
| Processor | Minimum 2 cores |
| RAM | Minimum 2 GB |

#### Recommended Requirements

|  |  |
| --- | --- |
| Processor | 4 Cores or more |
| RAM | 8 GB or more |
| Disk I/O | RAID 10 or RAID 0+1 disk setup |

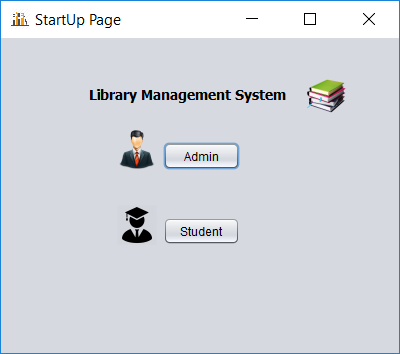
### Software Requirements

* Windows 7 or more
* Windows Vista, Server 2008 or above (64 bit)

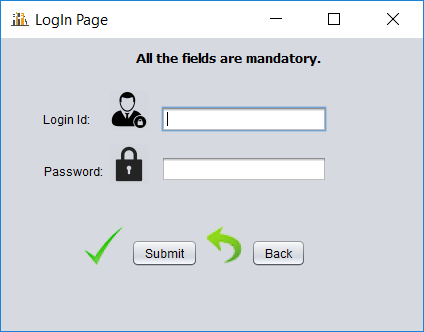
### Software/Tools Used

* NetBeans 8.2
* MySQL server 8
* JDK 8.0

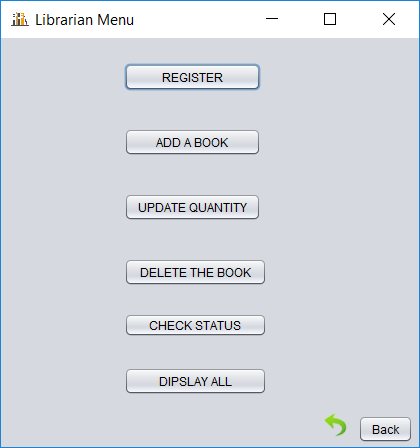
### Screenshots with Description



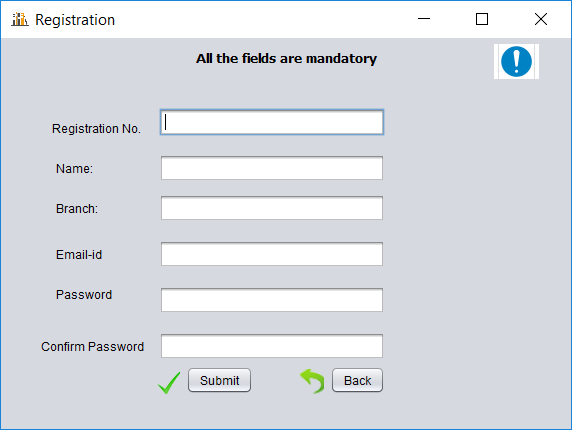
This is the page that appear on the screen as soon as the project is executed. In this frame, the user needs to select the login method.



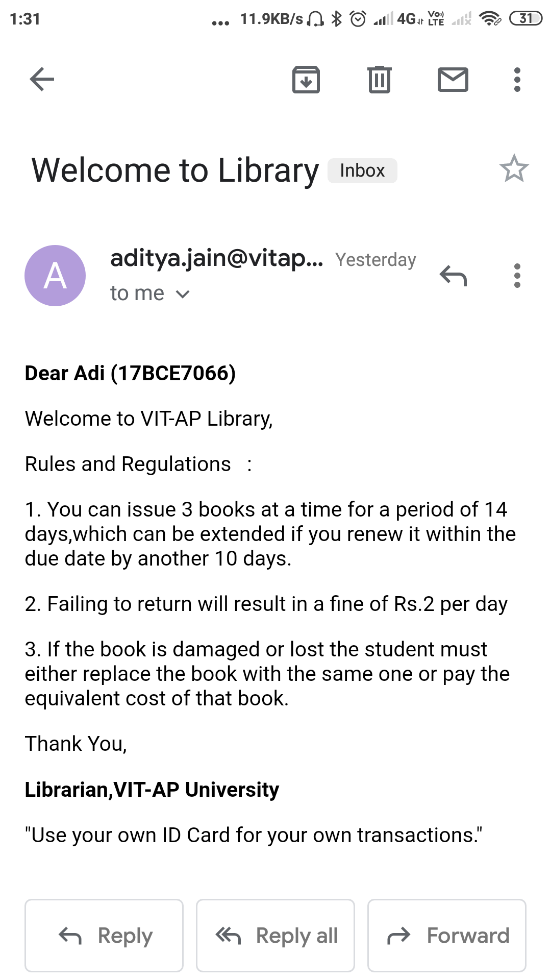
When the user clicks the admin button, it will be redirected to this page where he/she needs to enter the admin username and password to continue.

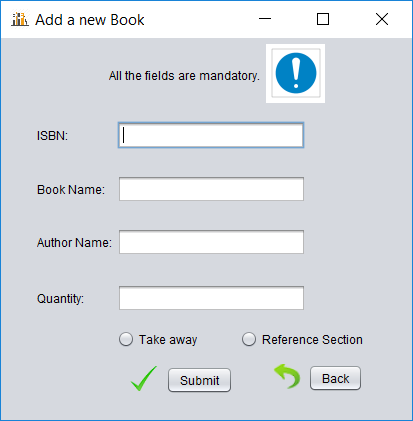


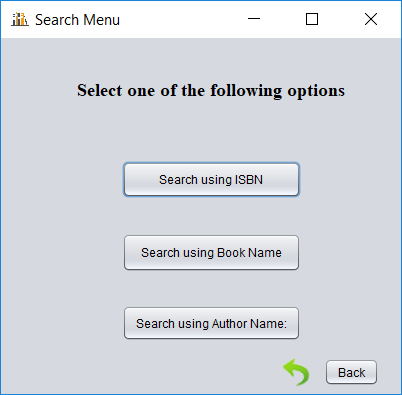
After successful login, admin can choose whatever action he wants to perform in this frame.



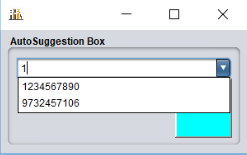
When admin chooses the register option, he/she will be redirected to this frame where he/she can add the details of the student to be registered and the student will receive an email about the registration and rules of the library.



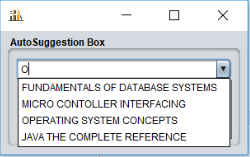
When admin selects the option to add a new book, he/she will be redirected to this page. In this page he/she can add details of the book to be added.



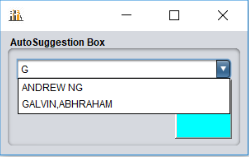
This is the search menu accessible for both admin and registered user to make the searching of the book faster and easier.



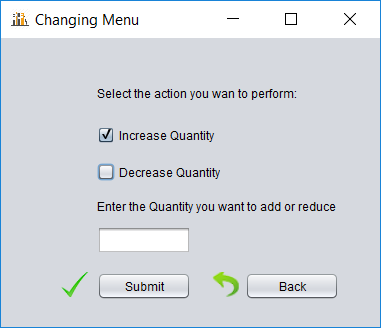
When a user selects to search by ISBN, he/she will be redirected to this frame which can help you in searching through auto suggestion feature.



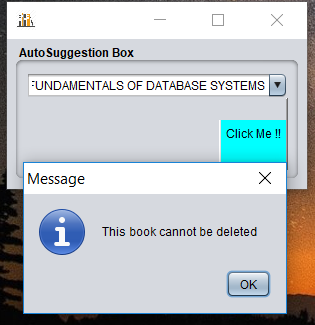
When a user selects to search by book name, he/she will be redirected to this frame which can help you in searching through auto suggestion feature.



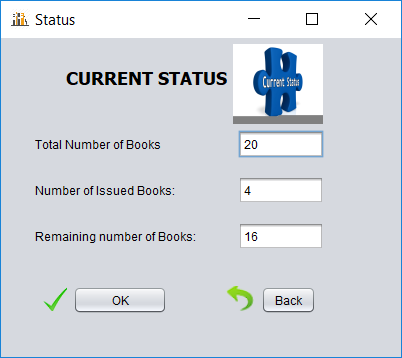
When a user selects to search by author name, he/she will be redirected to this frame which can help you in searching through auto suggestion feature.



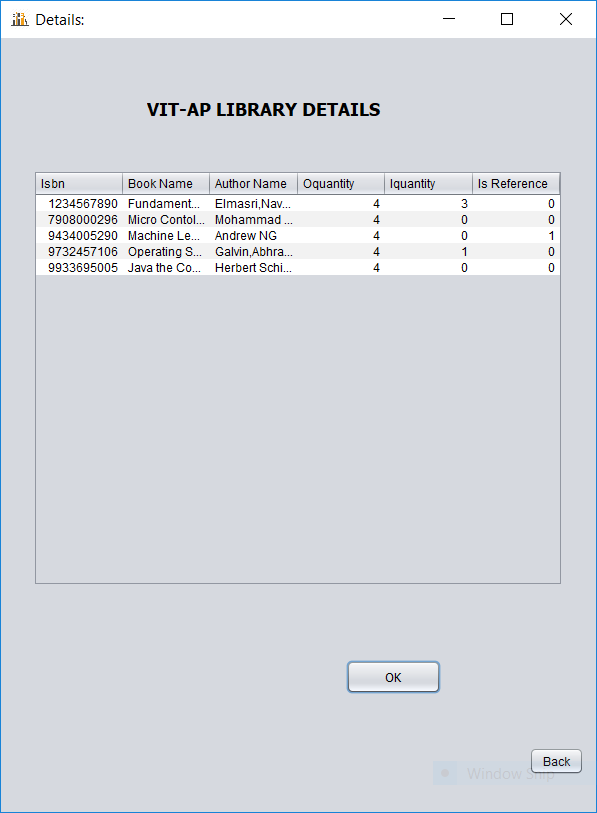
When admin selects to update the quantity, he/she will be redirected to search menu and from there he/she will be redirected to the above frame where admin can either increase or decrease quantity.



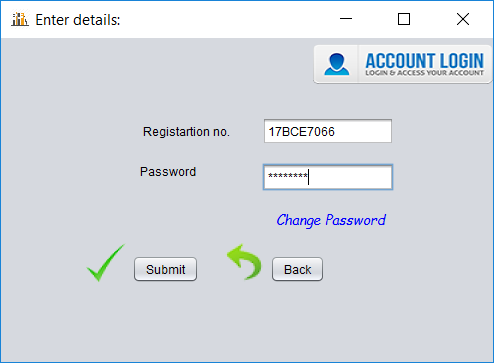
When admin selects to delete a book, he/she can use the feature of search with auto suggestion to select the book to be deleted and if the book is not issued by any of the students, the book is allowed to be deleted.



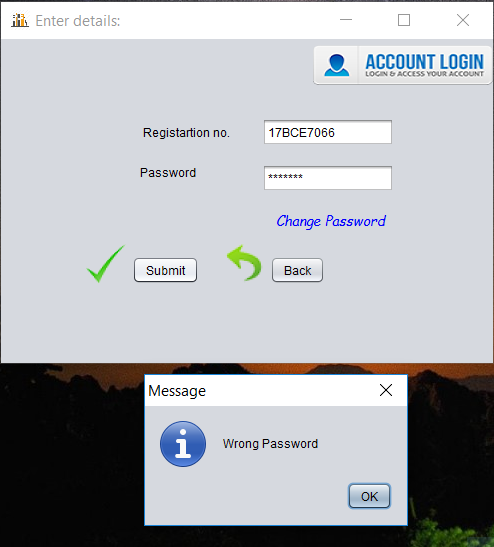
When the admin selects to check status, he/she will be redirected to the above frame where the number of books issued, available and remaining number of books in the library will be shown. This option is to give a brief summary of the library to the librarian.



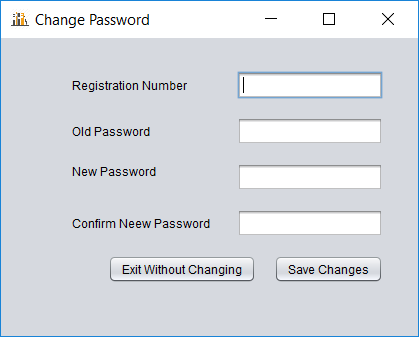
When the admin selects the Display All option in the Librarian Menu, he/she will get a detailed description of every book in the library like how many copies of that book are present in the library and how many copies of that book are issued etc.



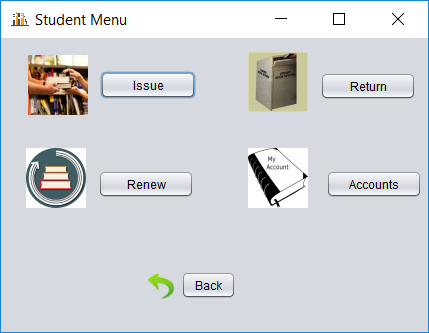
When the user selects the Student Login option in the student will be directed to the above frame. In this frame the student is required to enter the registration number and the password. On successful authentication the student will be directed to the Student Menu otherwise relevant message is displayed.



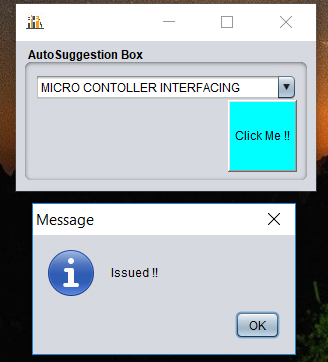
The above message is displayed if the password entered is incorrect. If the student has forgotten the password and wishes to change it then the student can choose the Change Password option.



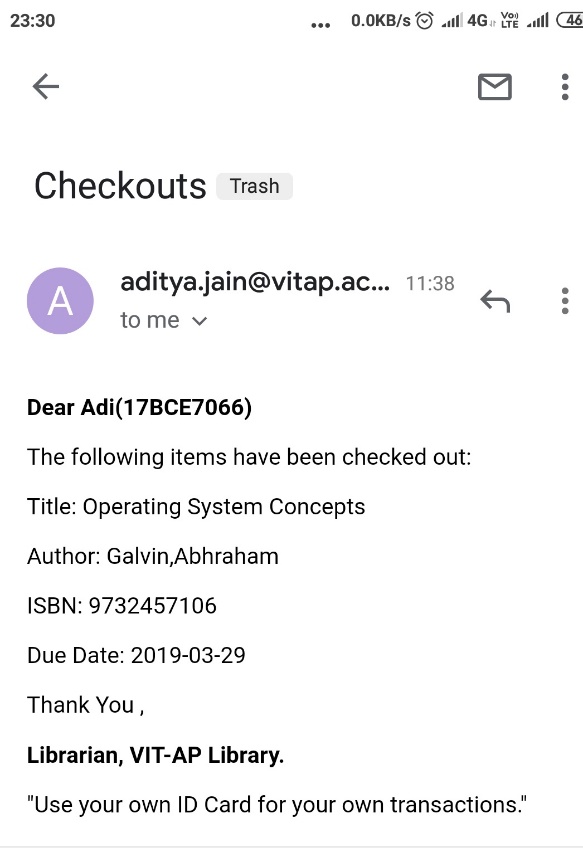
The student can fill the details in the above frame and the password will be changed

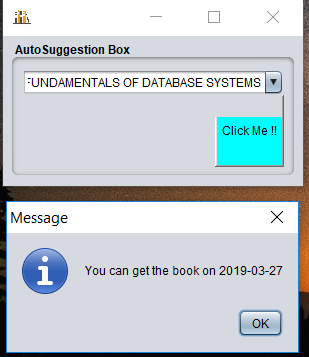


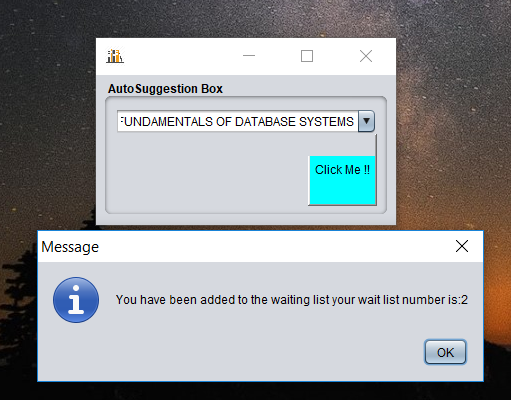
After successful authentication the above frame is shown to the student, from this frame the student can choose the required operation needed to perform.



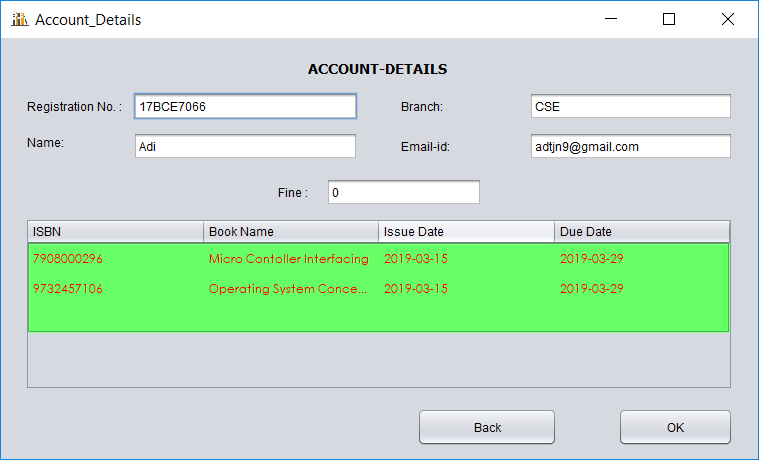
When the student selects the issue option, he/she will be re-directed to the Search Menu frame and the student can utilize the auto suggestion feature to search for the book that the student needs to issue and if the book is available the books is issued and the student will also receive an email notification about it.



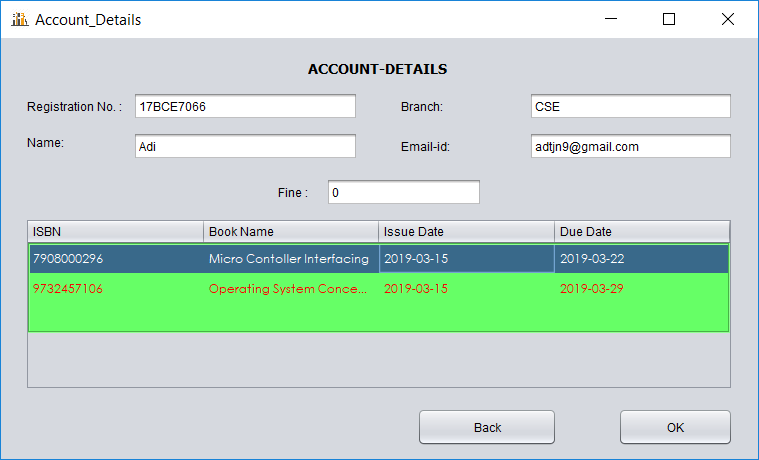


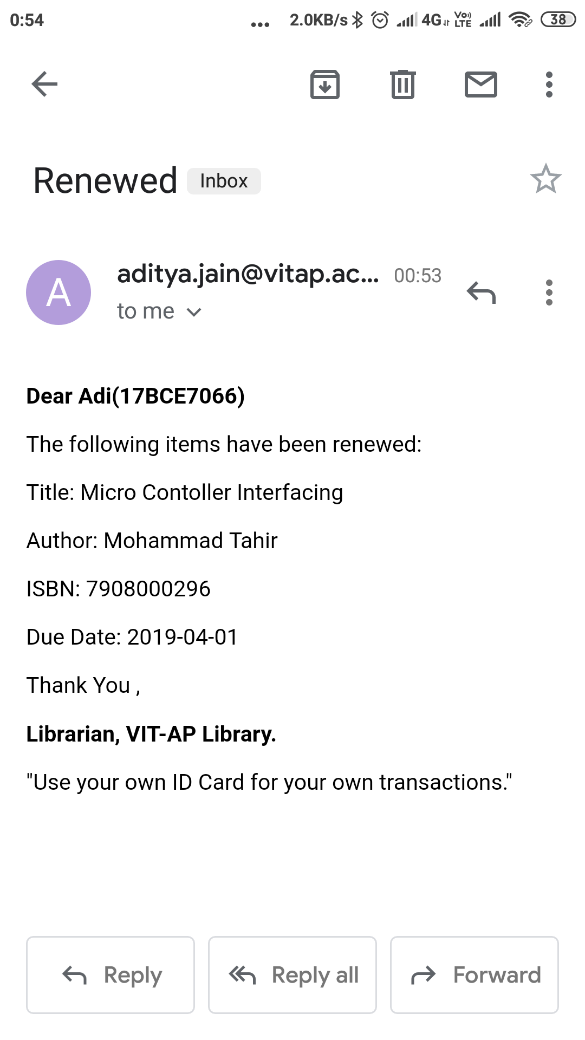


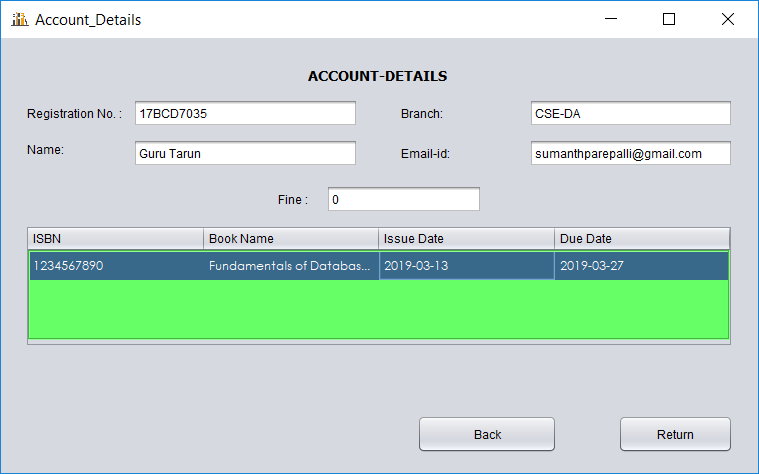
If the book which the student wants to issue is out of stock the system will show the expected date when the book will be available again and the student will also be given an option to stay on the waiting list for that book so that when that book is in stock, he/she will be given priority.



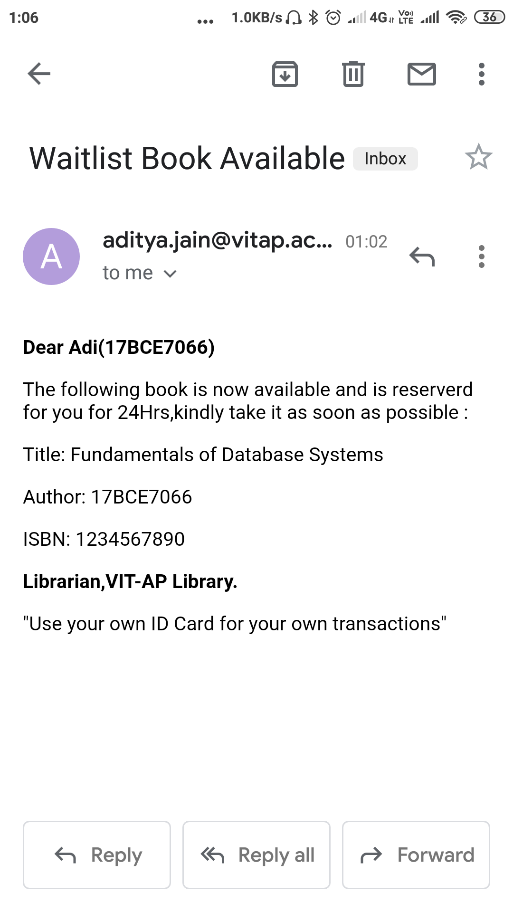
When the student selects the Account option in the Student Menu, he/she can view the information regarding the books issued, their respective due dates and fine accumulated for the student.



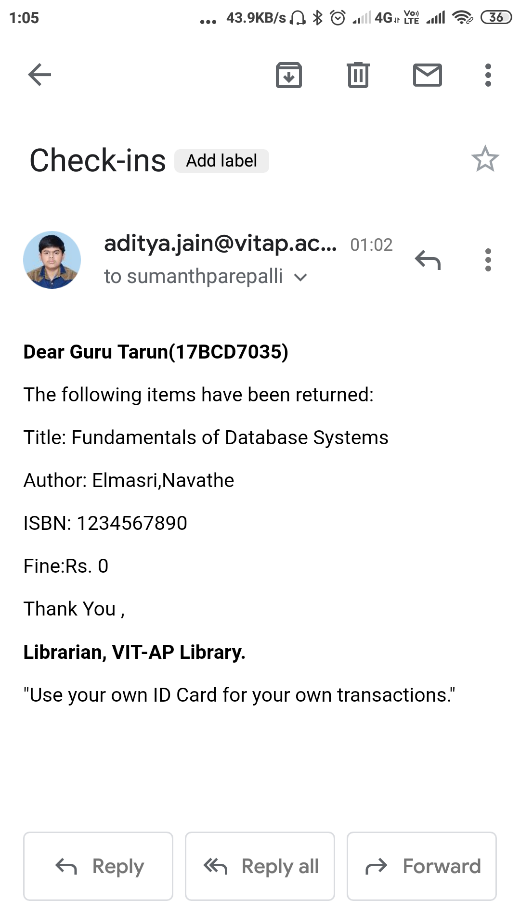
When the student selects the Renew option, a frame similar to the Account frame will appear but here the student can select the book which is to be renewed and then click on renew button and the book will be renewed and the student will be intimidated as well 



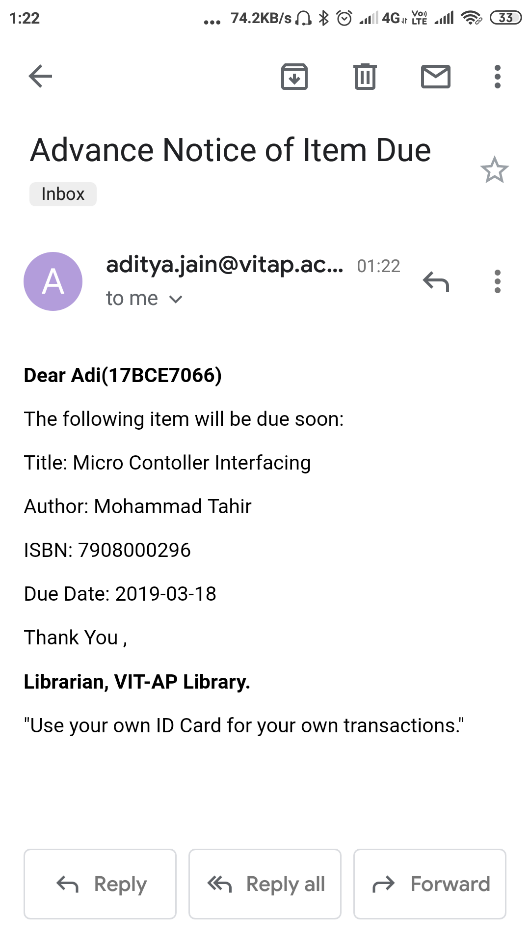
When the student selects the return option, a frame similar to the Account frame will appear but here the student can select the book which is to be returned and then click on renew button and the book will be renewed and the student will be intimidated. Here we perform another operation viz. to inform the student in the waiting list that the wait list book is now available.



This email is for the student who is at the beginning at the waiting list for that book.



This email is for the student who returns the book



When the due date of a book for a student is nearing a reminder regarding the same will be sent to the student two days prior to the due-date. The program parses through the database and sends similar email to the students for whom the above stated condition is met. It also notifies students who have crossed the due date for their issued books but have neither returned nor renewed their books

### FLOW OF CONTROL

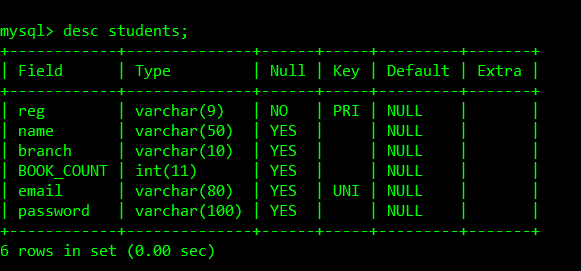


The above flow diagram represents the flow of control of the project.

## Screenshots of database

### Student Table

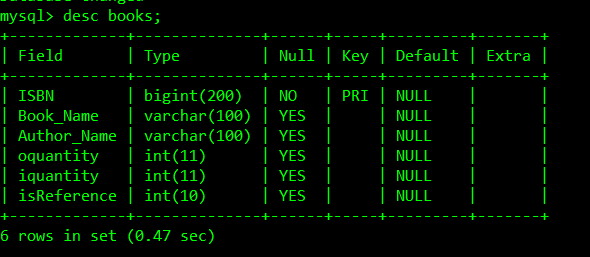
This table stores the required details of each student.



|  |  |
| --- | --- |
| reg | Registration Number (PRIMARY KEY) |
| name | Name of the student |
| branch | Branch that student belongs to |
| book\_count | No of books that the student has issued |
| email | Mail ID of the student (UNIQUE KEY) |
| password | Hash of the password required for student login |

### Books Table

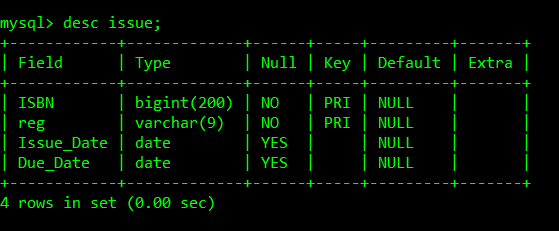
This table is to store the details of the books available



|  |  |
| --- | --- |
| ISBN | ISBN of the book (PRIMARY KEY) |
| Book\_Name | Name of the book |
| Author\_Name | Name of the author(s) |
| Oquantity | Number of the books available in the library |
| Iquantity | Number of the books that have been issued |
| isReference | Used to identify if a book is reference or not |

### Issue Table

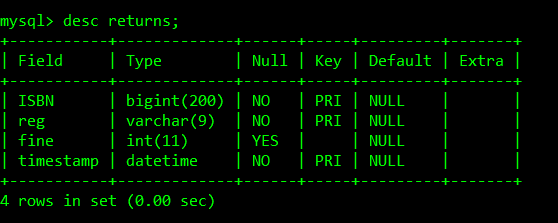
This table is to store the details of the books that are currently issued.



|  |  |
| --- | --- |
| ISBN | ISBN of the book issued (FOREIGN KEY referencing books(ISBN)) |
| reg | Registration number of the student (FOREIGN KEY referencing students(reg)) |
| Issue\_Date | Date of issue of the book |
| Due\_Date | Due date for the book to be returned |

### Returns Table

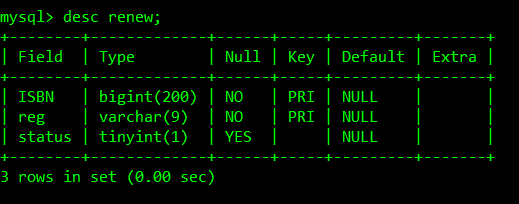
This table is to store the details of the books returned



|  |  |
| --- | --- |
| ISBN | ISBN of the book returned (FOREIGN KEY referencing books(ISBN)) |
| reg | Registration number of the student (FOREIGN KEY referencing students(reg)) |
| fine | Amount of fine the student has to pay |
| timestamp | Timestamp of the return |

### Renew Table

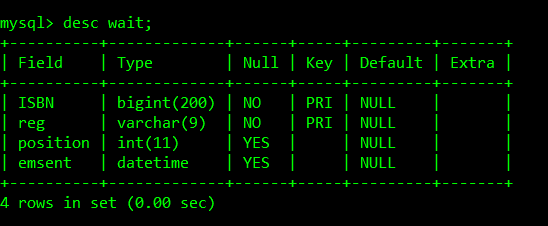
This table is to store the renewal status of the books that have been issued



|  |  |
| --- | --- |
| ISBN | ISBN of the book issued (FOREIGN KEY referencing books(ISBN)) |
| reg | Registration number of the student (FOREIGN KEY referencing students(reg)) |
| status | Status of renewal (1 – renewed | 0 – not yet renewed) |

### Wait Table

This table is to store the details of the students in waiting list



|  |  |
| --- | --- |
| ISBN | ISBN of the book issued (FOREIGN KEY referencing books(ISBN)) |
| reg | Registration number of the student (FOREIGN KEY referencing students(reg)) |
| Position | Position of the student in the waiting list (Max 10) |
| emsent | Timestamp when the student has been notified for book availability |

### Sample Code

*Uisbn.java*

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package library;

/\*\*

\*

\* @author User

\*/

import java.awt.\*;

import java.awt.event.\*;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.mail.MessagingException;

import javax.swing.\*;

public class UIsbn extends JPanel {

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/library", "root", "");

Statement st = con.createStatement();

ResultSet rs = null;

public String s = "";

public String reg = "";

private final JTextField tf;

@SuppressWarnings("rawtypes")

private final JComboBox combo = new JComboBox();

Button b;

private final Vector<String> v = new Vector<String>();

@SuppressWarnings({"rawtypes", "unchecked"})

public UIsbn(JFrame obj1, String t, String t1) throws SQLException {

super(new BorderLayout());

s = t;

reg = t1;

combo.setEditable(true);

b = new Button("Click Me !!");

b.setBounds(50, 50, 50, 20);

b.setBackground(Color.cyan);

b.addActionListener((ActionListener) new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

String x = (String) combo.getSelectedItem();

System.out.println(x);

if (s.equals("issue")) {

try {

issue(x);

} catch (SQLException ex) {

Logger.getLogger(UIsbn.class.getName()).log(Level.SEVERE, null, ex);

} catch (MessagingException ex) {

Logger.getLogger(UIsbn.class.getName()).log(Level.SEVERE, null, ex);

}

} else if (s.equals("change")) {

change(x);

} else {

try {

delete(x);

} catch (SQLException ex) {

Logger.getLogger(UIsbn.class.getName()).log(Level.SEVERE, null, ex);

}

}

obj1.setVisible(false);

}

});

tf = (JTextField) combo.getEditor().getEditorComponent();

tf.addKeyListener(new KeyAdapter() {

public void keyTyped(KeyEvent e) {

EventQueue.invokeLater(new Runnable() {

public void run() {

String text = tf.getText();

if (text.length() == 0) {

combo.hidePopup();

setModel(new DefaultComboBoxModel(v), "");

} else {

DefaultComboBoxModel m = getSuggestedModel(v, text);

if (m.getSize() == 0 || hide\_flag) {

combo.hidePopup();

hide\_flag = false;

} else {

setModel(m, text);

combo.showPopup();

}

}

}

});

}

public void keyPressed(KeyEvent e) {

String text = tf.getText();

int code = e.getKeyCode();

if (code == KeyEvent.VK\_ENTER) {

if (!v.contains(text)) {

v.addElement(text);

Collections.sort(v);

setModel(getSuggestedModel(v, text), text);

}

hide\_flag = true;

} else if (code == KeyEvent.VK\_ESCAPE) {

hide\_flag = true;

} else if (code == KeyEvent.VK\_RIGHT) {

for (int i = 0; i < v.size(); i++) {

String str = v.elementAt(i);

if (str.startsWith(text)) {

combo.setSelectedIndex(-1);

tf.setText(str);

return;

}

}

}

}

});

st = con.createStatement();

rs = st.executeQuery("select ISBN from books");

while (rs.next()) {

v.addElement(Long.toString(rs.getLong("ISBN")));

}

setModel(new DefaultComboBoxModel(v), "");

JPanel p = new JPanel(new BorderLayout());

p.setBorder(BorderFactory.createTitledBorder("AutoSuggestion Box"));

p.add(combo, BorderLayout.NORTH);

p.add(b, BorderLayout.AFTER\_LINE\_ENDS);

add(p);

setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));

setPreferredSize(new Dimension(300, 150));

}

public void delete(String x) throws SQLException {

ResultSet rs = st.executeQuery("select \* from books where ISBN=" + x);

rs.next();

int iq = rs.getInt(5);

if (iq != 0) {

JOptionPane.showMessageDialog(null, "This book cannot be deleted");

} else {

st.executeUpdate("delete from books where ISBN=" + x);

JOptionPane.showMessageDialog(null, "Book Removed");

}

Startup obj = new Startup();

obj.setTitle("StartUp");

//obj.setLocation(400, 400);

obj.setVisible(true);

}

public void change(String x) {

Change obj = new Change(x);

obj.setTitle("Changing Menu");

//obj.setLocation(400,400);

obj.setVisible(true);

}

public void issue(String x) throws SQLException, MessagingException

{

//First of all I am going to remove all the people for whom 24 Hrs have expired

ResultSet rs1;

rs=st.executeQuery("select \* from wait");

int c=0;

while(rs.next())

{

int i=rs.getInt("ISBN");

String r=rs.getString("reg");

String q=rs.getString("emsent");

int p=rs.getInt("position");

if(q!=null)

{

rs1=st.executeQuery("select timediff(now(),'"+q+"')");

rs1.next();

String diff=rs1.getString(1);

int t=Integer.parseInt(diff.substring(0, diff.indexOf(':')));

if(t>24)

{

q="delete from wait where ISBN="+i+" and reg='"+r+"'";

st.executeUpdate(q);

st.executeUpdate("update wait set position=position-1 where ISBN="+x+" and position >"+p);//updated the position for the rest of the people

}

}

c++;

rs=st.executeQuery("select \* from wait");

int cnt=c;

while(cnt>0)

{

rs.next();

cnt--;

}

}

//Now the next step is to check whether the book can be issued or not that is it a reference book or not

rs=st.executeQuery("select isReference from books where ISBN ="+x);

rs.next();

int flag=rs.getInt(1);

if(flag==0)//can be issued

{

rs=st.executeQuery("select count(\*) from wait where ISBN="+x+" group by ISBN");

if(!rs.next()) //No one is there in the waiting list

{

issue\_end(x);

}

else

{

int cnt=rs.getInt(1);//No. of people in the waiting list for that book

rs=st.executeQuery("select \* from wait where ISBN="+x+" and reg='"+reg+"'");//checking is the current reg in wait list or not

if(rs.next()) //If he is there in the wait list

{

String e=rs.getString("emsent");

if(e==null) //If he was not called

{

JOptionPane.showMessageDialog(null, "Sorry the book is not yet available. You will notifed when it is available");

}

else //If he was called

{

int p=rs.getInt("position");

st.executeUpdate("delete from wait where ISBN="+x+" and reg='"+reg+"' and position="+p); //removed the person from the wait list

st.executeUpdate("update wait set position=position-1 where ISBN="+x+" and position >"+p);//updated the position for the rest of the people

issue\_end(x); //Now after this it is simple issue

}

}

else //If he is not there in the wait list

{

rs=st.executeQuery("select \* from books");

rs.next();

int i=rs.getInt("iquantity");

int o=rs.getInt("oquantity");

if(i+cnt<o) //If book can be issued normally

{

issue\_end(x);

}

else //Book cannot be issued normally

{

rs=st.executeQuery("select min(Due\_Date) from issue where ISBN="+x);

rs.next();

String d=rs.getString(1);

JOptionPane.showMessageDialog(null, "You can get the book on " + d);

if(cnt<10)

{

if (JOptionPane.showConfirmDialog(null, "Do you want to stay on the waiting list ?") == 0)

{

st.executeUpdate("insert into wait(ISBN,reg,position) values("+x+",'"+reg+"',"+(cnt+1)+")");

JOptionPane.showMessageDialog(null, "You have been added to the waiting list your wait list number is:" + (cnt + 1));

}

}

else // If already 10 people are there on the wait list

{

JOptionPane.showMessageDialog(null, "Sorry 10 people are already there on the wait list");

}

}

}

}

}

else

{

JOptionPane.showMessageDialog(null,"Sorry this book cannot be issued as this is a Reference book");

}

Startup obj = new Startup();

obj.setTitle("StartUp");

// obj.setLocation(400, 400);

obj.setVisible(true);

this.setVisible(false);

}

public void issue\_end(String x) throws SQLException, MessagingException

{

rs=st.executeQuery("select \* from books where ISBN="+x);

rs.next();

int i=rs.getInt("iquantity");

int o=rs.getInt("oquantity");

String ti,au;

ti=rs.getString("Book\_Name"); //Book-Name

au=rs.getString("Author\_Name");//Author-Name

if(i<o) //Book can be issued because it is available

{

System.out.println("Hello");

String n,due,e;

st.executeUpdate("insert into renew values("+x+",'"+reg+"',0)");

st.executeUpdate("update students set BOOK\_COUNT=BOOK\_COUNT+1 where reg='"+reg+"'");

st.executeUpdate("update books set iquantity=iquantity+1 where ISBN="+x);

st.executeUpdate("insert into issue(ISBN,reg) values ("+x+",'"+reg+"')");

st.executeUpdate("update issue set Issue\_Date=CURDATE() where ISBN="+x+" and reg='"+reg+"'");

st.executeUpdate("update issue set Due\_Date=DATE\_ADD(Issue\_Date,INTERVAL 14 DAY) where ISBN="+x+" and reg='"+reg+"'");

rs=st.executeQuery("select \* from students where reg='"+reg+"'");

rs.next();

n=rs.getString("name"); //Student-Name

e=rs.getString("email");//Student-email id

rs=st.executeQuery("select Due\_Date from issue where ISBN="+x+" and reg='"+reg+"'");

rs.next();

due=rs.getString(1);//Book Due-Date

JavaEmail ob = new JavaEmail(e, "Checkouts", "<p><strong>Dear " + n + "(" + reg + ")</strong></p>\n"

+ "<p>The following items have been checked out:</p>\n"

+ "<p>Title: " + ti + "</p>\n"

+ "<p>Author: " + au + "</p>\n"

+ "<p>ISBN: " + x + "</p>\n"

+ "<p>Due Date: " + due + "</p>\n"

+ "<p>Thank You ,</p>\n"

+ "<p><strong>Librarian, VIT-AP Library.</strong></p>"

+ "<p>\"Use your own ID Card for your own transactions.\"</p>");

ob.start();

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

return;

}

else //book cannot be issued //this part will only be executed for the first person on the wait list for that book

{

rs=st.executeQuery("select count(\*) from wait where ISBN="+x+" group by ISBN");

rs.next();

int cnt=rs.getInt(1);//No. of people in the waiting list for that book

rs=st.executeQuery("select min(Due\_Date) from issue where ISBN="+x);

rs.next();

String d=rs.getString(1);

JOptionPane.showMessageDialog(null, "You can get the book on " + d);

if(cnt<10)

{

if (JOptionPane.showConfirmDialog(null, "Do you want to stay on the waiting list ?") == 0)

{

st.executeUpdate("insert into wait(ISBN,reg,position) values("+x+",'"+reg+"',"+(cnt+1)+")");

JOptionPane.showMessageDialog(null, "You have been added to the waiting list your wait list number is:" + (cnt + 1));

}

}

else // If already 10 people are there on the wait list

{

JOptionPane.showMessageDialog(null, "Sorry 10 people are already there on the wait list");

}

return;

}

}

private boolean hide\_flag = false;

private void setModel(DefaultComboBoxModel mdl, String str) {

combo.setModel(mdl);

combo.setSelectedIndex(-1);

tf.setText(str);

}

@SuppressWarnings("rawtypes")

private static DefaultComboBoxModel getSuggestedModel(java.util.List<String> list, String text) {

DefaultComboBoxModel m = new DefaultComboBoxModel();

for (String s : list) {

if (s.contains(text)) {

m.addElement(s);

}

}

return m;

}

}

**Complete code in GitHub** [https://github.com/adtjn9/Library-Management-System](https://github.com/adtjn9/Library-Management-System%20)

## Conclusion:

We can conclude that this project offers better features than the traditional library management systems and satisfies all the functional requirements stated and provides the user with a simple and user-friendly environment use. Thus, the system overcomes the problems effectively without any corrupted data or information and reduces human effort.

## Scope for future work:

This project can be improved by hardware implementation like RFID tags and a drop box for returning books which completely removes the need to search for a book or entering anything manually thus making the system more efficient and faster and automate the process in return.

## References:

* <https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>
* <https://www.javatpoint.com/example-to-connect-to-the-mysql-database>