|  |
| --- |
|  |
|  |
|  |

**SL. NO. CONTENTS Page No**

CHAPTER-1

**1-Introduction**--------------------------------------------------------------- 3

* 1. **JAVA**------------------------------------------------------------------------------- 3
     1. Significant Language Feature-------------------------------------------- 5
     2. Java Code-------------------------------------------------------------------- 6
  2. **Swing(JAVA)**-------------------------------------------------------------------- 6
     1. A Basic Example------------------------------------------------------------ 7
     2. Component Which We used in our Project------------------------- 8
  3. **DataBase**------------------------------------------------------------------------- 9
  4. **MySQL**---------------------------------------------------------------------------- 10
  5. **WAMP Server**------------------------------------------------------------------- 11
  6. **PhpMyAdmin**------------------------------------------------------------------- 11
  7. **JTattoo**--------------------------------------------------------------------------- 11

CHAPTER-2

1. **System Analysis and database Design-------------------------- 12**
   1. **ER Diagram**--------------------------------------------------------------------- 12
   2. **Functional Requirement**--------------------------------------------------- 13
   3. **Non functional Requirement**--------------------------------------------- 13
   4. **Use Case Diagram**-------------------------------------------------------------14

CHAPTER-3

1. **System Implementation**---------------------------------------- 15
   1. **Database Design**--------------------------------------------------------- **15**
   2. **Database Connectivity**------------------------------------------------- 20
   3. **Implementation of Database operations**------------------------- 21
   4. **Stored Procedure**------------------------------------------------------- 27
   5. **Trigger**---------------------------------------------------------------------- 29

CHAPTER-4

1. **Result and Discussion**-------------------------------------------- 30

CHAPTER-5

1. **Conclusion**----------------------------------------------------------- 41

**CHAPTER 1**

**INTRODUCTION**

Library Management System is a Database application which is helpful for Librarians of college or any public library . This mini-Project is implemented using JAVA(Swings) .

Operations supported by the application are insert, delete a0nd retrieve.This will allow n number of librarian to create an account and that librarian has rights to

* Add new student
* Add new books
* Issue book to a particular student
* Taking book back from student

In the following sections, a brief introduction about the tools, languages and the databases used to develop the project are discussed.

* 1. **JAVA**

Java is a general-purpose [computer programming language](https://en.wikipedia.org/wiki/Programming_language) that is [concurrent](https://en.wikipedia.org/wiki/Concurrent_computing), [class-based](https://en.wikipedia.org/wiki/Class-based_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming)  and specifically designed to have as few implementation dependencies as possible.

It is intended to let application developers "[write once, run anywhere](https://en.wikipedia.org/wiki/Write_once,_run_anywhere)" (WORA), meaning that [compiled](https://en.wikipedia.org/wiki/Compiler) Java code can run on all platforms that support Java without the need for recompilation.

Java applications are typically compiled to [byte code](https://en.wikipedia.org/wiki/Java_bytecode) that can run on any [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) regardless of [computer architecture](https://en.wikipedia.org/wiki/Computer_architecture)

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support.

This is achieved by compiling the Java language code to an intermediate representation called [Java byte code](https://en.wikipedia.org/wiki/Java_bytecode), instead of directly to architecture-specific [machine code](https://en.wikipedia.org/wiki/Machine_code).

Java byte code instructions are analogous to machine code, but they are intended to be executed by a [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine) (VM) written specifically for the host hardware.

[End users](https://en.wikipedia.org/wiki/End_user) commonly use a [Java Runtime Environment](https://en.wikipedia.org/wiki/Java_virtual_machine) (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java [applets](https://en.wikipedia.org/wiki/Applet).

Standard libraries provide a generic way to access host-specific features such as graphics, [threading](https://en.wikipedia.org/wiki/Thread_(computing)), and [networking](https://en.wikipedia.org/wiki/Computer_network).

The use of universal byte code makes porting simple. However, the overhead of interpreting byte code into machine instructions made interpreted programs almost always run more slowly than native [executables](https://en.wikipedia.org/wiki/Executable). [Just-in-time](https://en.wikipedia.org/wiki/Just-in-time_compilation) (JIT) compilers that compile byte codes to machine code during runtime were introduced from an early stage.

Java itself is platform-independent and is adapted to the particular platform it is to run on by a [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) for it, which translates the [Java byte code](https://en.wikipedia.org/wiki/Java_bytecode) into the platform's machine language.

Java was developed to achieve 5 main goals. They are

* It should be simple, object-oriented, distributed and easy to learn.
* It should be robust and secure.
* It should be independent of a given computer architecture or platform.
* It should be possible to write an interpreter for the language. The language should also support parallelism and use dynamic typing.

## Significant Language Features

Java has significant advantages compared to other languages. Java language is easy to use and therefore easy to write, compile, debug. Moreover, it is easier to learn when compared to other programming languages. Since Java is object-oriented, it allows you to create modular programs and reusable codes. Platform-independent, Java is a robust language. The language lays more importance on early checking for errors, since Java compilers can detect many problems during the time of execution of an application. Java is multithreaded, since it has the capability for a program to perform several tasks simultaneously within the same program. Java Development has gained a significant position in the industry with programmers and developers finding it easier and more effective than many other languages.   
  
Java applications are designed to be compiled and then interpreted at runtime, unlike the conventional programming languages, which can either compile source code to native code or interpret the source code. The language itself has borrowed the syntax from C and C++. Java considers security as a part of its design. The Java language, its compiler, interpreter, and runtime environment are all developed with security. Writing network programs in Java is similar to sending and receiving data to and from a file.   
  
The Java programming language was developed and re-designed for use on the Internet. In the internet domain, Java’s popularity has increased tremendously, especially on the server side of the Internet. Nowadays, there are a large number of Java experts who strive for the enhancement and improvement of Java development. For beginners who are interested in learning Java, the numerous Java tutorials available online are good to start with. Java tutorials and Java tips are the best resources for learning and improvising in Java.

## 1.1.2 JAVA Code

The traditional ["Hello, world!" program](https://en.wikipedia.org/wiki/%22Hello,_world!%22_program) can be written in Java as:

**class** **HelloWorldApp** {

**public** **static** void main(String[] args) {

System.out.println("Hello World!"); *// Prints the string to the console.*

}

}

**We are using swings in our project which is GUI Widget Toolkit for JAVA.So will see about swings**

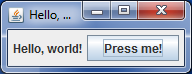
* 1. **Swing(JAVA)**

Swing was developed to provide a more sophisticated set of GUI [components](https://en.wikipedia.org/wiki/Software_component) than the earlier [Abstract Window Toolkit (AWT)](https://en.wikipedia.org/wiki/Abstract_Window_Toolkit). Swing provides a native [look and feel](https://en.wikipedia.org/wiki/Look_and_feel) that emulates the look and feel of several platforms, and also supports a [pluggable look and feel](https://en.wikipedia.org/wiki/Pluggable_look_and_feel) that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

Though Swing is intended to be replaced by [JavaFX](https://en.wikipedia.org/wiki/JavaFX), it will remain part of the Java SE specification for the foreseeable future

**1.2.1** A basic example

[](https://en.wikipedia.org/wiki/File:Swing_example_on_Windows_7.png)

The basic example code running on [Windows 7](https://en.wikipedia.org/wiki/Windows_7)

The following is a rather simple Swing-based program. It displays a window (a [JFrame](https://docs.oracle.com/javase/9/docs/api/javax/swing/JFrame.html)) containing a label and a button.

**import** **java.awt.FlowLayout**;

**import** **javax.swing.JButton**;

**import** **javax.swing.JFrame**;

**import** **javax.swing.JLabel**;

**import** **javax.swing.WindowConstants**;

**import** **javax.swing.SwingUtilities**;

**public** **class** **SwingExample** **implements** Runnable {

@Override

**public** void run() {

*// Create the window*

JFrame f = **new** JFrame("Hello, !");

*// Sets the behavior for when the window is closed*

f.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

*// Add a layout manager so that the button is not placed on top of the label*

f.setLayout(**new** FlowLayout());

*// Add a label and a button*

f.add(**new** JLabel("Hello, world!"));

f.add(**new** JButton("Press me!"));

*// Arrange the components inside the window*

f.pack();

*// By default, the window is not visible. Make it visible.*

f.setVisible(**true**);

}

**public** **static** void main(String[] args) {

SwingExample se = **new** SwingExample();

*// Schedules the application to be run at the correct time in the event queue.*

SwingUtilities.invokeLater(se);

}

}

**1.2.2 Components which we used in our project**

#### JButton

**JButton** class provides functionality of a button. JButton class has three constuctors,

**JButton**(Icon *ic*)

**JButton**(String *str*)

**JButton**(String *str*, Icon *ic*)

It allows a button to be created using icon, a string or both. JButton supports **ActionEvent**. When a button is pressed an **ActionEvent** is generated.

#### JTextField

**JTextField** is used for taking input of single line of text. It is most widely used text component. It has three constructors,

**JTextField**(int *cols*)

**JTextField**(String *str*, int *cols*)

**JTextField**(String *str*)

*cols* represent the number of columns in text field.

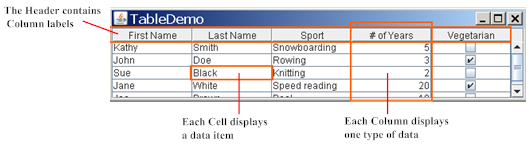
#### JComboBox

Combo box is a combination of text fields and drop-down list.**JComboBox** component is used to create a combo box in Swing. Following is the constructor for JComboBox,

**JComboBox**(String *arr[]*)

**JTable**

With the [JTable](https://docs.oracle.com/javase/8/docs/api/javax/swing/JTable.html) class you can display tables of data, optionally allowing the user to edit the data. JTable does not contain or cache data; it is simply a view of your data. Here is a picture of a typical table displayed within a scroll pane:



**1.3 Database**

A database is a collection of [information](http://searchsqlserver.techtarget.com/definition/information) that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images.**Database** software systems are programmed in SQL, and examples include Microsoft SQL Server, MySQL, Oracle SAP HANA and FoxPro.

There are different types of DBMS ranging from small systems that run on personal [computers](http://ecomputernotes.com/fundamental/introduction-to-computer/what-is-computer) to huge systems that run on mainframes.

The following are main examples of database applications:

• Computerized library systems

• Automated teller machines

• Flight reservation systems

• Computerized parts inventory systems

A DBMS system is also required to protect the integrity of data and provide its security. A database management system (**DBMS**) is system software for creating and managing databases. The **DBMS** provides users and programmers with a systematic way to create, retrieve, update and manage data.

**1.4 MYSQL**

MySql is a powerful database. It's very good and free of charge. Many developers in the world selected mysql and php for developing their website.

The MySQL database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use. It's used in more than 6 million installations ranging from large corporations to specialized embedded applications on every continent in the world. (Yes, even Antarctica!)

Not only is MySQL the world's most popular open source database, it's also become the database of choice for a new generation of applications built on the LAMP stack (Linux, Apache, MySQL, PHP / Perl / Python.) MySQL runs on more than 20 platforms including Linux, Windows, OS/X, HP-UX, AIX, Netware, giving you the kind of flexibility that puts you in control.

Whether you're new to database technology or an experienced developer or DBA, MySQL offers a comprehensive range of certified software, support, training and consulting to make you successful.

**1.5 WAMP Server**

The acronym WAMP refers to a set of free ([open source](https://www.webopedia.com/TERM/O/open_source.html)) [applications](https://www.webopedia.com/TERM/A/application.html), combined with Microsoft Windows, which are commonly used in [Web server](https://www.webopedia.com/TERM/W/Web_server.htm) environments. The WAMP stack provides developers with the four key elements of a Web server:  an [operating system](https://www.webopedia.com/TERM/O/operating_system.htm), [database](https://www.webopedia.com/TERM/D/database.html), Web server and Web scripting software. The combined usage of these programs is called a server stack. In this stack, [Microsoft Windows](https://www.webopedia.com/TERM/M/Microsoft_Windows.html) is the operating system (OS), [Apache](https://www.webopedia.com/TERM/A/Apache_Web_server.html) is the Web server, [MySQL](https://www.webopedia.com/TERM/M/MySQL.html) handles the database components, while [PHP](https://www.webopedia.com/TERM/P/PHP.html), [Python](https://www.webopedia.com/TERM/P/Python.htm), or [PERL](https://www.webopedia.com/TERM/P/Perl.html) represents the dynamic scripting languages.

**1.6 phpMyAdmin**

phpMyAdmin is a [free and open source](https://en.wikipedia.org/wiki/Free_and_open_source) administration tool for [MySQL](https://en.wikipedia.org/wiki/MySQL) and [MariaDB](https://en.wikipedia.org/wiki/MariaDB). As a portable [web application](https://en.wikipedia.org/wiki/Web_application) written primarily in [PHP](https://en.wikipedia.org/wiki/PHP), it has become one of the most popular MySQL administration tools, especially for [web hosting services](https://en.wikipedia.org/wiki/Web_hosting_service).

**1.7 JTattooo**

JTattoo consists of several different Look and Feels for Swing applications. All of them enables developers to improve their application with an excellent user interface. So JTattoo opens desktop applications the door to end users who are unfortunate with the Look and Feels shipped with the standard JDK.

**CHAPTER 2**

**System Analysis and Design**

In this chapter, a complete description of the project development are discussed. The requirement of the project identified are showcased. The database design is done Using High-Level Conceptual Data Models

**2.1 ER-Diagram**

maintains

Manages

Student

Add

Issue

ReturnB

Contains

Book

Account

**2.2 Functional Requirements**

Functional requirements of a software project that interpret the function of a part. It defines its functions, input and output. The typical functional requirements includes:

Application contains 1 modules:

* Admin module

Librarian module

* Librarian can able to add new Student.
* Librarian can able to add New Books.
* Librarian can able to see the Statistics.
* Librarian can able to issue Book.
* Librarian can able to get back the issued Books.

**2.3 Non- Functional Requirements**

Non-functional requirements it specifies the canon of the articular process not the particular judgment of the system and particular behavior of the process. Non-functional requirements define how the system work.

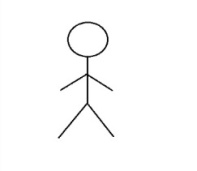
* This application is developed to make Library operation easier to Librarians so that it will save the time.
* This application work efficiently it works on all logical paths and independently
* .This application is available during all the tim.
* This application reduces all the complexity during issuing and Returning.

.

* Using of application is secure, because it display appropriate information about the Book,Student Details
* The system should capable to enhance with further technology in future to improve its features compared to the existing system such as QR Scanner etc.
* The system should be reliable and it should be related in all the condition and it should be recoverable in all the situation or condition if error occurs.

**2.4 Use Case Diagram**

The use case diagrams usually refer to behavioral diagrams helps people to understand the interaction between user and system. Use case diagram identify different users of the system.It is used to define some set of actions, which is called as use cases.Actors are the result of some valuable use cases.Use case figures are also called as unified modeling language



Librarian

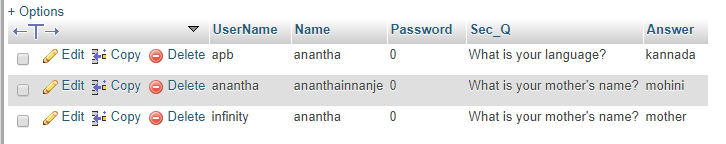
**CHAPTER 3**

**SYSTEM IMPLEMENTATION**

**3.1 Database Design**

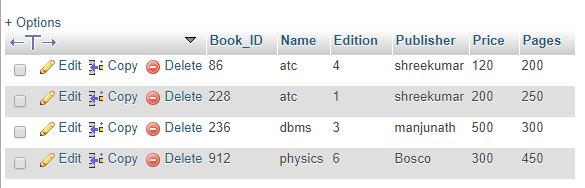
**Account Table**

CREATE TABLE `account` (  
 `UserName` varchar(100) NOT NULL,  
 `Name` varchar(100) NOT NULL,  
 `Password` varchar(100) NOT NULL,  
 `Sec\_Q` varchar(100) NOT NULL,  
 `Answer` varchar(100) NOT NULL,  
 PRIMARY KEY (`UserName`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1



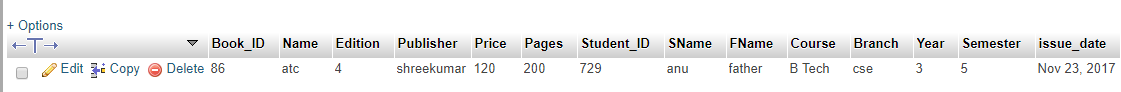
**Book Table**

CREATE TABLE `book` (  
 `Book\_ID` varchar(100) NOT NULL,  
 `Name` varchar(100) NOT NULL,  
 `Edition` varchar(100) NOT NULL,  
 `Publisher` varchar(100) NOT NULL,  
 `Price` varchar(100) NOT NULL,  
 `Pages` varchar(100) NOT NULL,  
 PRIMARY KEY (`Book\_ID`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1



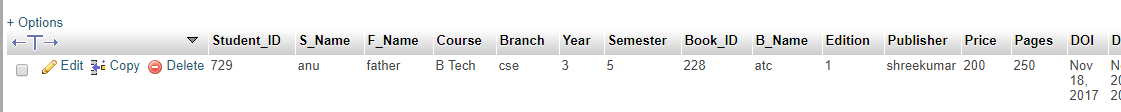
**Issue Table**

CREATE TABLE `issue` (  
 `Book\_ID` varchar(100) NOT NULL,  
 `Name` varchar(100) NOT NULL,  
 `Edition` varchar(100) NOT NULL,  
 `Publisher` varchar(100) NOT NULL,  
 `Price` varchar(100) NOT NULL,  
 `Pages` varchar(100) NOT NULL,  
 `Student\_ID` varchar(100) NOT NULL,  
 `SName` varchar(100) NOT NULL,  
 `FName` varchar(100) NOT NULL,  
 `Course` varchar(100) NOT NULL,  
 `Branch` varchar(100) NOT NULL,  
 `Year` varchar(100) NOT NULL,  
 `Semester` varchar(100) NOT NULL,  
 `issue\_date` varchar(100) NOT NULL,  
 PRIMARY KEY (`Book\_ID`,`Student\_ID`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1



**ReturnB Table**

CREATE TABLE `returnb` (  
 `Student\_ID` varchar(100) NOT NULL,  
 `S\_Name` varchar(100) NOT NULL,  
 `F\_Name` varchar(100) NOT NULL,  
 `Course` varchar(100) NOT NULL,  
 `Branch` varchar(100) NOT NULL,  
 `Year` varchar(100) NOT NULL,  
 `Semester` varchar(100) NOT NULL,  
 `Book\_ID` varchar(100) NOT NULL,  
 `B\_Name` varchar(100) NOT NULL,  
 `Edition` varchar(100) NOT NULL,  
 `Publisher` varchar(100) NOT NULL,  
 `Price` varchar(100) NOT NULL,  
 `Pages` varchar(100) NOT NULL,  
 `DOI` varchar(100) NOT NULL,  
 `DOR` varchar(100) NOT NULL  
) ENGINE=MyISAM DEFAULT CHARSET=latin1

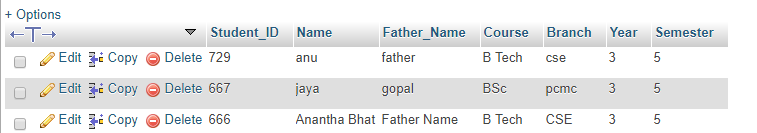
****

**oldIssuedata Table**

CREATE TABLE `oldissuedata` (  
 `BName` varchar(100) NOT NULL,  
 `Book\_ID` varchar(100) NOT NULL,  
 `SName` varchar(100) NOT NULL,  
 `Student\_ID` varchar(100) NOT NULL,  
 `issued` varchar(100) NOT NULL  
) ENGINE=MyISAM DEFAULT CHARSET=latin1

**Student Table**

CREATE TABLE `student` (  
 `Student\_ID` varchar(100) NOT NULL,  
 `Name` varchar(100) NOT NULL,  
 `Father\_Name` varchar(100) NOT NULL,  
 `Course` varchar(100) NOT NULL,  
 `Branch` varchar(100) NOT NULL,  
 `Year` varchar(100) NOT NULL,  
 `Semester` varchar(100) NOT NULL,  
 PRIMARY KEY (`Student\_ID`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1



**3.2 Database Connectivity**

import java.sql.Connection;

import java.sql.DriverManager;

import javax.swing.JOptionPane;

public class DbConnection {

public static Connection openConnection() {

try {

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

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/lib", "root", "");

return conn;

} catch (Exception e) {

JOptionPane.showMessageDialog(null, e);

return null;

}

}

}

**3.3 Implementation of Database Operations**

**To add New Student**

String sql="insert into student(Student\_ID,Name,Father\_Name,Course,Branch,Year,Semester) values (?,?,?,?,?,?,?)";

try{

ps=conn.prepareStatement(sql);

ps.setString(1, jTextField1.getText());

ps.setString(2, jTextField2.getText());

ps.setString(3, jTextField3.getText());

ps.setString(4, (String)jComboBox1.getSelectedItem());

ps.setString(5, jTextField5.getText());

ps.setString(6, (String)jComboBox2.getSelectedItem());

ps.setString(7, (String)jComboBox3.getSelectedItem());

ps.execute();

JOptionPane.showMessageDialog(null, "New Student Registered !!!");

setVisible(false);

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

**To add New Book**

String sql = "insert into book(Book\_ID,Name,Edition,Publisher,Price,Pages) values (?,?,?,?,?,?)";

try{

ps= conn.prepareStatement(sql);

ps.setString(1, jTextField1.getText());

ps.setString(2, jTextField2.getText());

ps.setString(3, (String)jComboBox1.getSelectedItem());

ps.setString(4, jTextField3.getText());

ps.setString(5, jTextField5.getText());

ps.setString(6, jTextField6.getText());

ps.execute();

JOptionPane.showMessageDialog(null,"New Book Added!!!!");

setVisible(false);

}catch(Exception e){

JOptionPane.showMessageDialog(null,e);

}

**To Issue New Book**

String sql = "select \* from book where Book\_ID=?";

try{

ps=conn.prepareStatement(sql);

ps.setString(1, jTextField1.getText());

rs=ps.executeQuery();

if(rs.next()){

String add1=rs.getString("Name");

jTextField2.setText(add1);

String add2=rs.getString("Edition");

jTextField3.setText(add2);

String add3=rs.getString("Publisher");

jTextField4.setText(add3);

String add4=rs.getString("Price");

jTextField5.setText(add4);

String add5=rs.getString("Pages");

jTextField6.setText(add5);

rs.close();

ps.close();

}else{

JOptionPane.showMessageDialog(null, "Book Id Not Found");

}

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

finally{

try{

rs.close();

ps.close();

}catch(Exception e){

}

}

String sql="select \* from student where Student\_ID=?";

try{

ps=conn.prepareStatement(sql);

ps.setString(1, jTextField7.getText());

rs=ps.executeQuery();

if(rs.next()){

String add1 = rs.getString("Name");

jTextField8.setText(add1);

String add2 = rs.getString("Father\_Name");

jTextField9.setText(add2);

String add3 = rs.getString("Course");

jTextField10.setText(add3);

String add4 = rs.getString("Branch");

jTextField11.setText(add4);

String add5 = rs.getString("Year");

jTextField12.setText(add5);

String add6 = rs.getString("Semester");

jTextField13.setText(add6);

rs.close();

ps.close();

}else{

JOptionPane.showMessageDialog(null, "Student Id Not Found");

}

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}finally{

try{

rs.close();

ps.close();

}catch(Exception e){

}

}

**To getback issued Book**

String sql="delete from issue where Student\_ID=?";

try{

ps=conn.prepareStatement(sql);

ps.setString(1, jTextField1.getText());

ps.execute();

}catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}

public void ReturnUpdate(){

String sql = "insert into returnb(Student\_ID,S\_Name,F\_Name,Course,Branch,Year,Semester,Book\_ID,B\_Name,Edition,Publisher,Price,Pages,DOI,DOR) values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";

try {

ps = conn.prepareStatement(sql);

ps.setString(1, jTextField1.getText());

ps.setString(2, jTextField2.getText());

ps.setString(3, jTextField3.getText());

ps.setString(4, jTextField4.getText());

ps.setString(5, jTextField5.getText());

ps.setString(6, jTextField6.getText());

ps.setString(7, jTextField7.getText());

ps.setString(8, jTextField14.getText());

ps.setString(9, jTextField13.getText());

ps.setString(10, jTextField12.getText());

ps.setString(11, jTextField11.getText());

ps.setString(12, jTextField10.getText());

ps.setString(13, jTextField9.getText());

ps.setString(14, jTextField8.getText());

ps.setString(15, ((JTextField)jDateChooser1.getDateEditor().getUiComponent()).getText());

ps.execute();

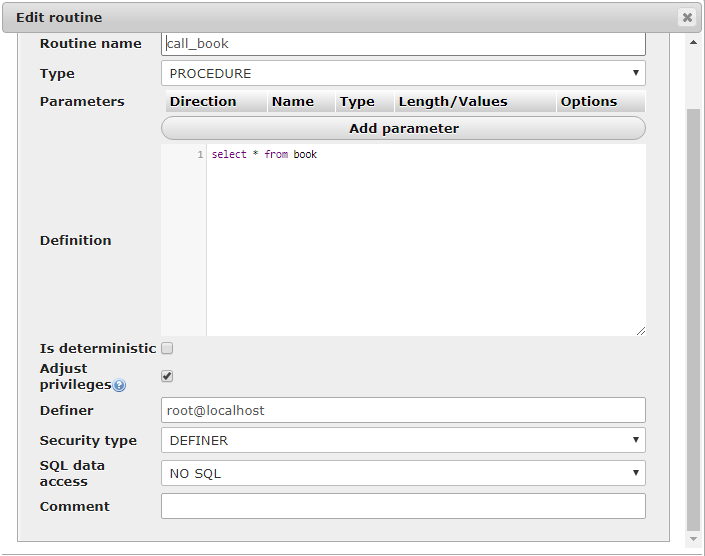
JOptionPane.showMessageDialog(null, "Book returned!!!");

} catch (Exception e) {

JOptionPane.showMessageDialog(null, e);

}

**3.4 Stored Procedure**



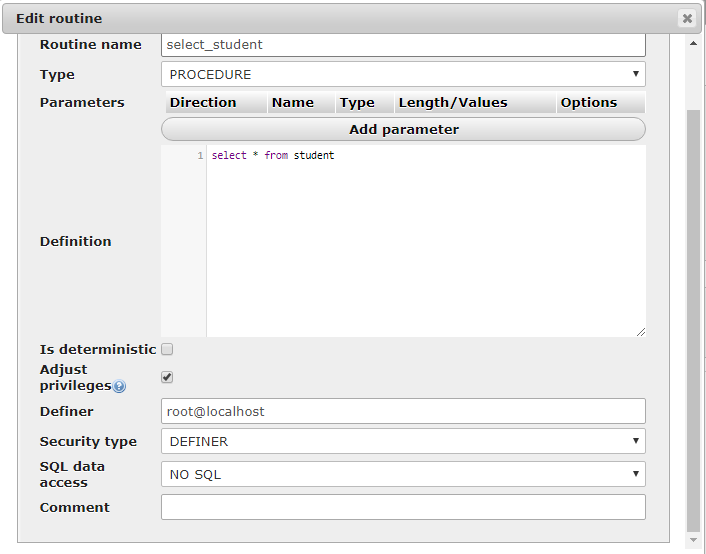
This Stored Procedure helps to display book details in statistics table.

String sql="{call call\_book}";

java.sql.CallableStatement cst=conn.prepareCall(sql);

rs=cst.executeQuery();

jTable2.setModel(DbUtils.resultSetToTableModel(rs));

****

This Stored Procedure helps to display student details in statistics table.

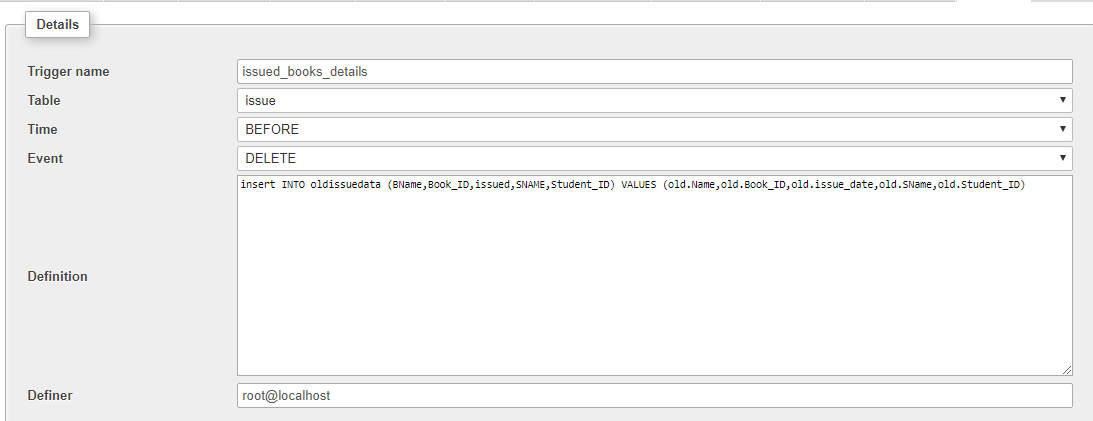
String sql="{call select\_student}";

java.sql.CallableStatement cst=conn.prepareCall(sql);

rs=cst.executeQuery();

jTable1.setModel(DbUtils.resultSetToTableModel(rs));

**3.5 Trigger**

****

This trigger helps to store the issued books details like Book ID,Issue date,Student Name and Student ID in the new Table oldissuedata

insert INTO oldissuedata (BName,Book\_ID,issued,SNAME,Student\_ID) VALUES (old.Name,old.Book\_ID,old.issue\_date,old.SName,old.Student\_ID)

After Returning Book the data will be deleted from issue table;so that details we are storing with this trigger

**CHAPTER 4**

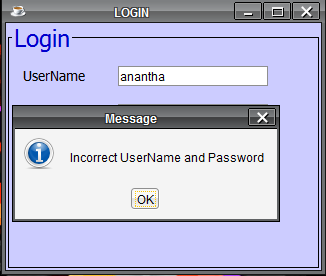
**Results and Discussion**

In this chapter the results of the project are described. The snapshot of of the project showing various functionalities like insert, delete, and retrieval are showcased.

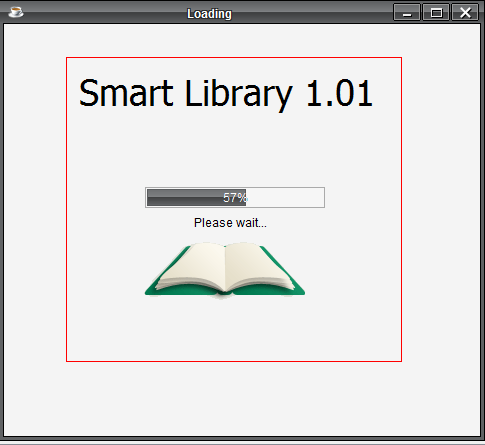
**Login Frame**



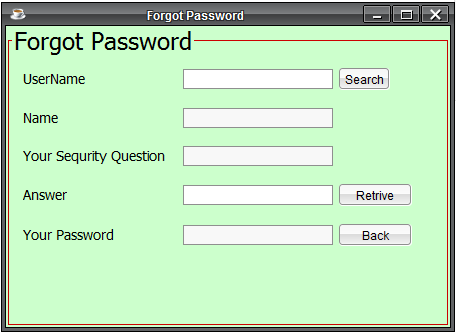
* This is the login form for the library management system.
* If you have an account you can simply enter your user name & password.
* Then click the login button.
* If you don’t have an account you can sign up
* If you forgot your password then click on Forgot password



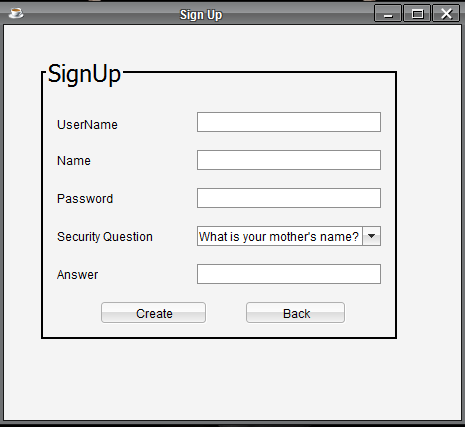
* If you enter your user name or password incorrect, you can see the alert message like this.



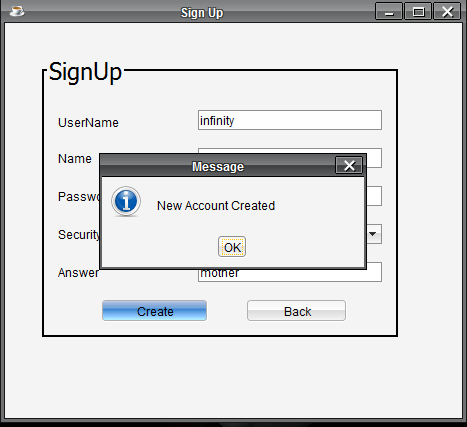
* If you enter your username and password correct, you can see this alert message



* If you forgot your password you can retrieve by giving user name and then click search.After giving answer to the secret question you will get your password



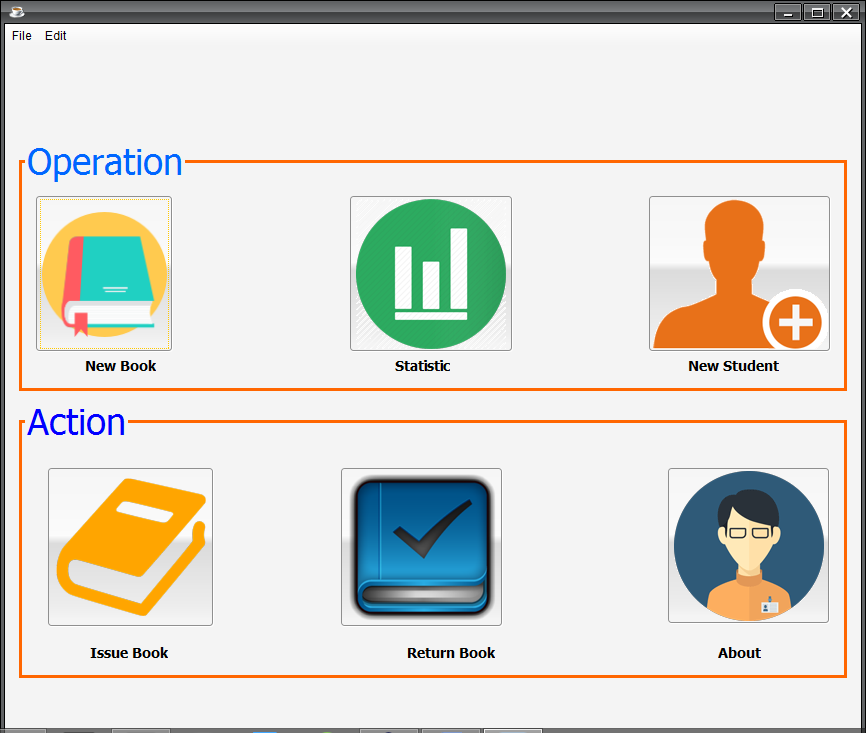
* Here New Librarian can create his New Account



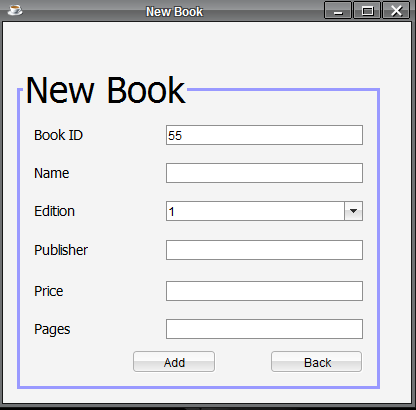
* After Creating New Account You will get message Like this

**Home Frame**

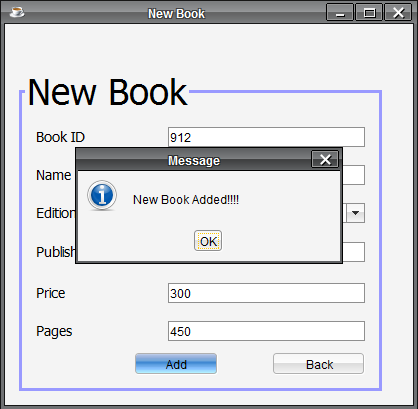
* This is the home frame for my library management system.
* There are 6 buttons
* New Book
* New Member
* Statistics
* Issue a Book
* Return a Book
* About



**New Book**

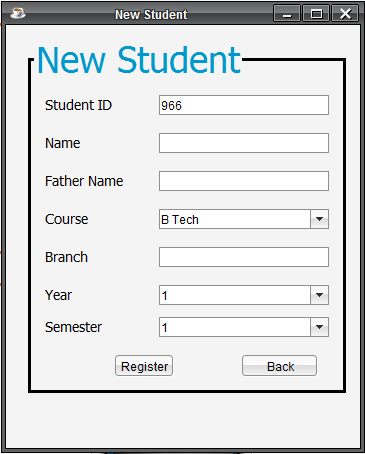


* This frame is use to add a new book.
* Using this you can add any no of new books to the library.
* Book ID is the primary key.
* Book ID is randomly genereated

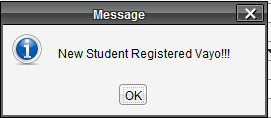


* After you complete the new book form you can see the alert message like this

**New Student**

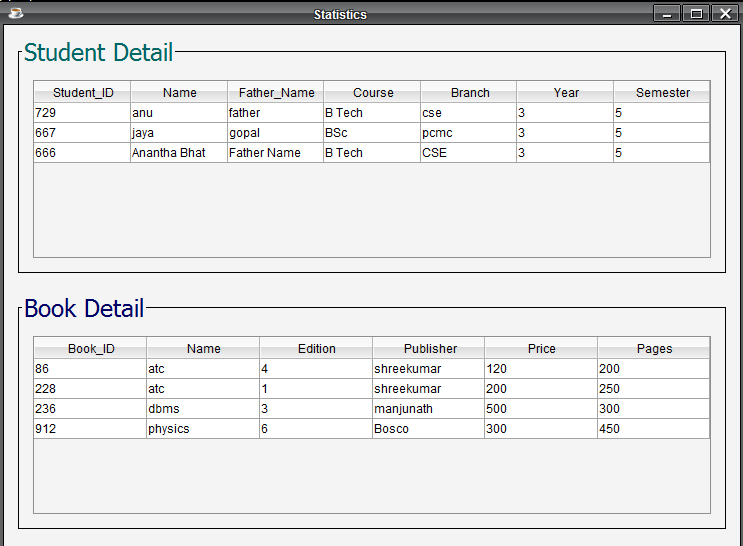


* This frame is use to add a new Student.
* Using this you can add any no of new Student to the library.
* Student ID is the primary key.
* Student ID is randomly generated



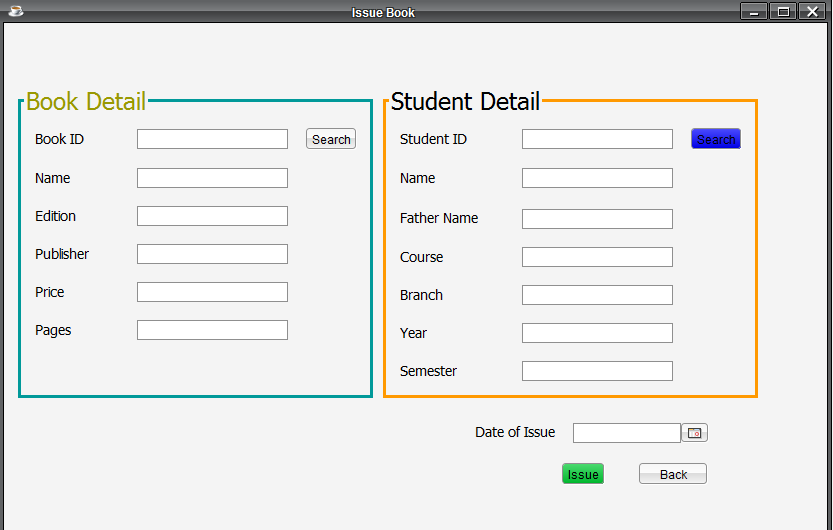
* After you complete the new member form you can see the alert message like this.

**Statistics Frame**

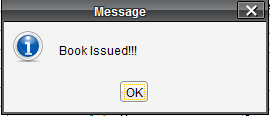


* This frame shows all the student and book data of library

**Book Issue Frame**

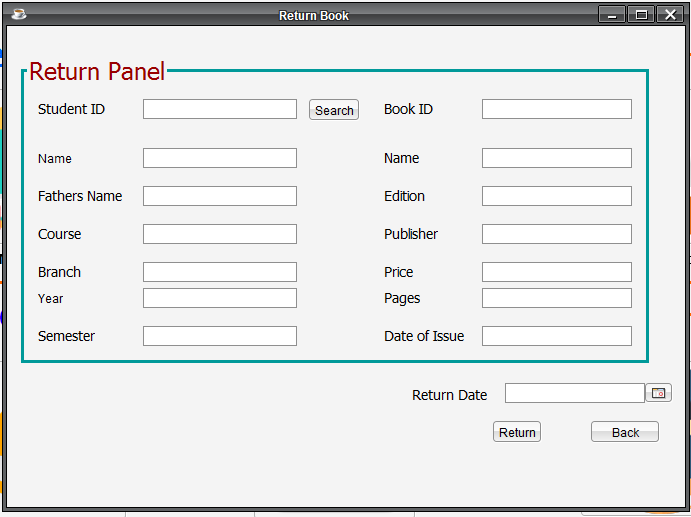


* This frame is use to enter the data of the issued book and the person.
* You just have to enter the student id and click the search button. If it is a valid student id, other fields will automatically fill.
* Do the same thing for book.
* By clicking issue button the data of the book and person will be save in the database



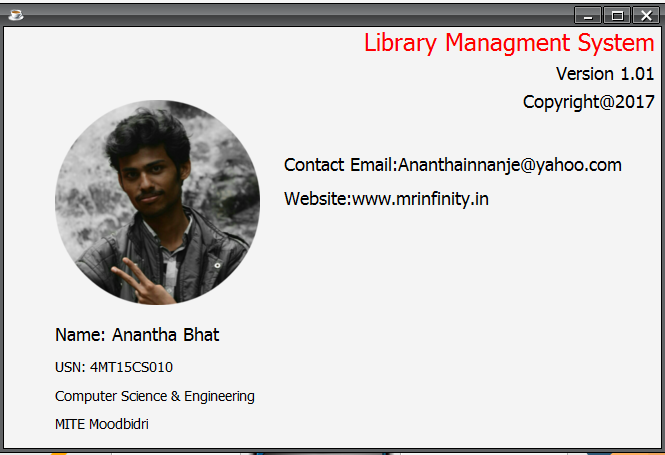
* Book issued to the particular Student

**Book Return Frame**



* This frame is use to enter the data of the returned book and the person.
* You just have to enter the student id and click the search button. If it is a valid student id, other fields will automatically fill.
* By clicking issue button the data of the book and person will be save in the database

**About Frame**



* This Frame is Bout the Developer and Version of the System

**CHAPTER 5**

**CONCLUSION**

Library database management System helps us in centralizing the data used for managing the tasks performed in a Library. The theoritical process involved in database design have been practically implemented. The project provides user friendly interface for the users to interact with the database. All database operations including insertion, deletion, and Retrievals are supported along with support for trigger and stored procedure.