Table of Contents

[Introduction 2](#_Toc525239245)

[Application Design 3](#_Toc525239246)

[Use Case Diagram 3](#_Toc525239247)

[Librarian 3](#_Toc525239248)

[Student 3](#_Toc525239249)

[Structure 4](#_Toc525239250)

[Server 5](#_Toc525239251)

[Client 5](#_Toc525239252)

[Interface 5](#_Toc525239253)

[Database 6](#_Toc525239254)

[Testing 7](#_Toc525239255)

[Login 7](#_Toc525239256)

[Register 7](#_Toc525239257)

[Add New Book 7](#_Toc525239258)

[Modify Book 7](#_Toc525239259)

[Books List 8](#_Toc525239260)

[Borrow Books 8](#_Toc525239261)

[Return Books 8](#_Toc525239262)

[DCOMS Technologies 9](#_Toc525239263)

[Java RMI 9](#_Toc525239264)

[DCOM/COM+ 9](#_Toc525239265)

[CORBA 10](#_Toc525239266)

[User Manual 11](#_Toc525239267)

[Login 11](#_Toc525239268)

[Register 12](#_Toc525239269)

[Librarian Home Page 13](#_Toc525239270)

[New Book 14](#_Toc525239271)

[Modify Books 15](#_Toc525239272)

[Book List 16](#_Toc525239273)

[Student Home page 17](#_Toc525239274)

[Return Book 18](#_Toc525239275)

[ICE (Internet Communications Engine) 19](#_Toc525239276)

[Conclusion 19](#_Toc525239277)

[References 20](#_Toc525239278)

# Introduction

The developed application is a library management system which has been designed for APU university, the application is designed to managed the available titles of the book in APU library. There are two types of users who can access the system, librarian and students.

Librarian has the ability to manage the books list in the library the functionalities like adding new book, assigning the number of available copies, modify and delete the existing book and also see the list of books.

Students can see the list of all books in the library with some details like author, published date and available copies, students also can borrow and return the book from library using the application.

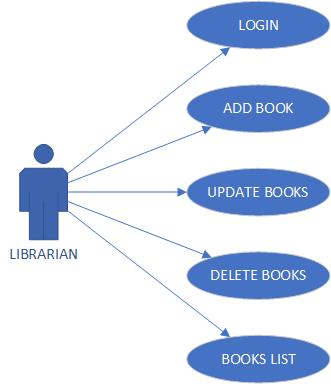
Librarian has a unique account but student can register new account by providing their details and use the account to login into the system.

The application has been developed using JAVA programming language; Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. RMI technology has been used to develop this application which will be discussed later on this document, so the system is evolving 2 different application server and client which are communicating by TCP connection.

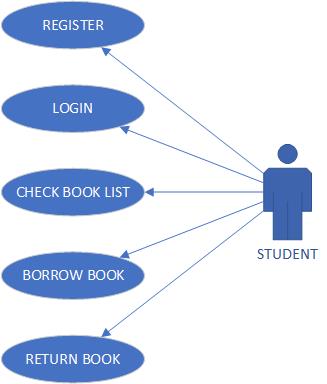
# Application Design

## Use Case Diagram

### Librarian



### Student

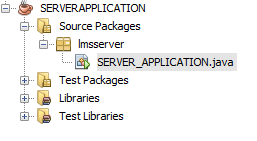


## Structure

The system contains 3 different application, client, server and interface.

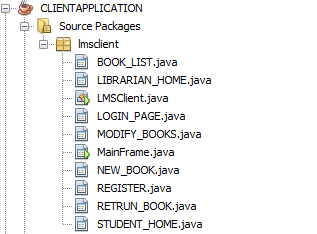
### Server

The server application is doing the main processes and has access to the database to write, read and modify data. It has one java class which is extending UnicastRemoteObject below you can see the file structure in the server.



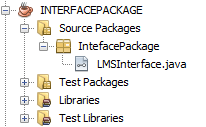
### Client

Client application is the part which is interacting with user to receive and show the information to the users, it has a main java class and different JFrames for each user form you can see the file structure in client application below



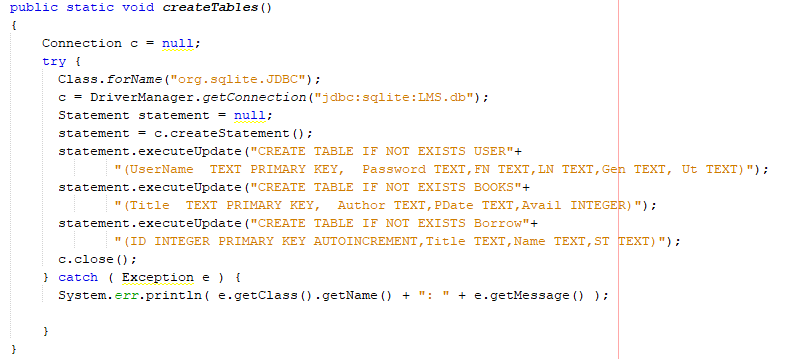
### Interface

Interface is just a java interface which is declaring the server side function to be usable inside the client, both client and server application has to include the jar file compiled by interface.



## Database

The application is using SQLite as database to store the information, to use the SQLite we are using JDBC library and the database and tables will be created on the startup of server application if they do not exist using below function



# Testing

## Login

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Type of output** | **Expected Out Put** |
| 1 | Not fill all boxes | Error | Wrong user name or password |
| 2 | Invalid credential | Error | Wrong user name or password |
| 3 | Valid credentials | Redirect | User Home Page |

## Register

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Type of output** | **Expected Out Put** |
| 1 | Not fill all boxes | Error | Please fill all required information |
| 2 | Not same password | Error | Passwords do not match |
| 3 | Invalid Data | Error | Some data are invalid |
| 4 | Exist username | Error | Choose another username |
| 5 | Complete information | Redirect | Sign In user |

## Add New Book

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Type of output** | **Expected Out Put** |
| 1 | Not fill all boxes | Error | Please fill all required information |
| 2 | Complete information | Redirect | Success Message |

## Modify Book

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Type of output** | **Expected Out Put** |
| 1 | Not fill all boxes | Error | Please fill all required information |
| 2 | Complete information | Redirect | Success Message |

## Books List

-Show all the books with details

## Borrow Books

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Type of output** | **Expected Out Put** |
| 1 | No books selected | Error | Please select book |
| 2 | No copy available | Error | No book available now |
| 3 | Complete information | Redirect | Success Message |

## Return Books

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Type of output** | **Expected Out Put** |
| 1 | No books selected | Error | Please select book |
| 2 | Complete information | Redirect | Success Message |

# DCOMS Technologies

## Java RMI

Remote Method Invokation (RMI) is an technology that permits the sharing of Java objects between Java Virtual Machines (JVM) over a system. A RMI application comprises of a server that makes remote objects that fit in with a predefined interface, which are accessible for strategy summon to customer applications that acquire a remote reference to the protest. RMI treats a remote protest uniquely in contrast to a nearby question when the protest is passed starting with one virtual machine then onto the next. Instead of making a duplicate of the execution question in the getting virtual machine, RMI passes a remote stub for a remote protest. The stub goes about as the nearby agent, or intermediary, for the remote protest and fundamentally is, to the guest, the remote reference. The guest summons a technique on the nearby stub, which is in charge of completing the strategy approach the remote question. A stub for a remote protest actualizes a similar arrangement of remote interfaces that the remote question executes. This enables a stub to be cast to any of the interfaces that the remote protest actualizes. Be that as it may, this likewise implies just those strategies characterized in a remote interface are accessible to be brought in the getting virtual machine.

## DCOM/COM+

Distributed Component Object Model (DCOM)is the distributed version of Microsoft's COM technology which allows the creation and use of binary objects/components from languages other than the one they were originally written in, it currently supports Java(J++),C++, Visual Basic, JScript, and VBScript. DCOM works over the network by using proxy's and stubs. When the client instantiates a component whose registry entry suggests that it resides outside the process space, DCOM creates a wrapper for the component and hands the client a pointer to the wrapper. This wrapper, called a proxy, simply marshals methods calls and routes them across the network. On the other end, DCOM creates another wrapper, called a stub, which unmarshals methods calls and routes them to an instance of the component.

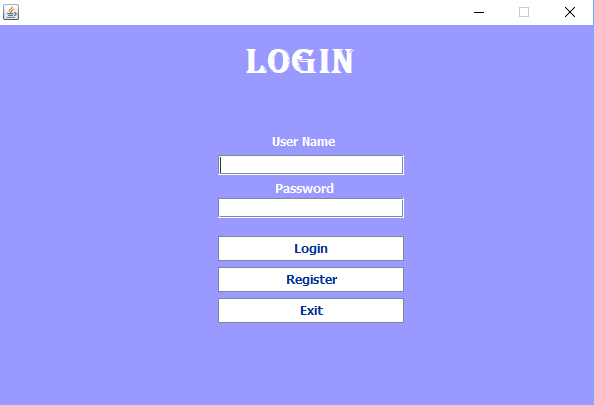
## CORBA

A CORBA application usually consists of an Object Request Broker (ORB), a client and a server. An ORB is responsible for matching a requesting client to the server that will perform the request, using an object reference to locate the target object. When the ORB examines the object reference and discovers that the target object is remote, it marshals the arguments and routes the invocation out over the network to the remote object's ORB. The remote ORB then invokes the method locally and sends the results back to the client via the network. There are many optional features that ORBs can implement besides merely sending and receiving remote method invocations including looking up objects by name, maintaining persistent objects, and supporting transaction processing. A primary feature of CORBA is its interoperability between various platforms and programming languages.

# User Manual

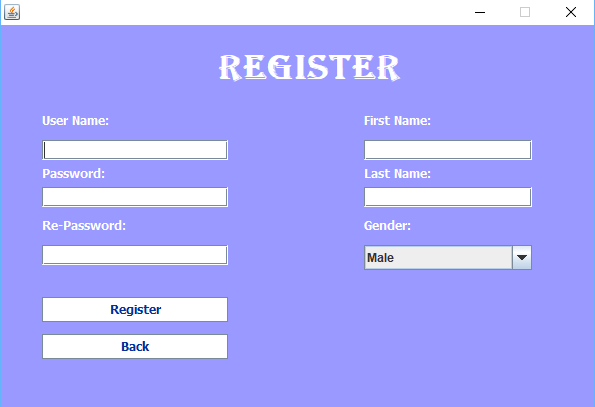
## Login

On the startup the login page will be shown which user has to insert username and password to go to the home page, student can create new account by clicking on register button.



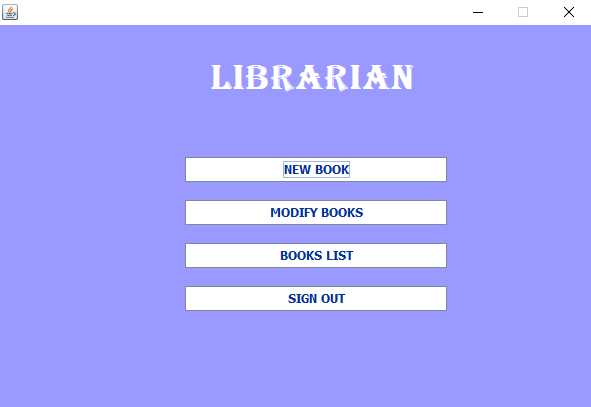
## Register

Students can create a new account by entering user name password, first name, last name and gender, if the inserted information is valid the new account will be created and can be used to login into the system.



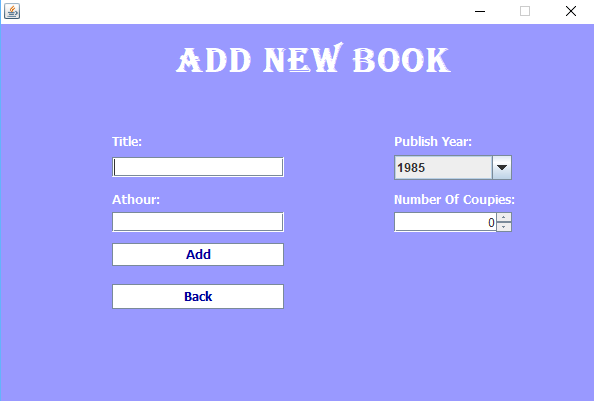
## Librarian Home Page

Librarian after signing in will see this page which can be used to go to different part of application.



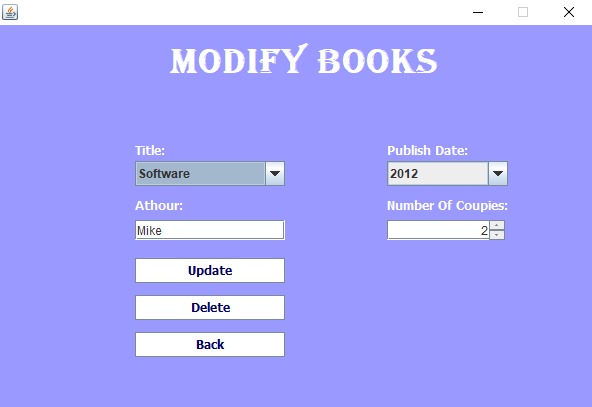
## New Book

Librarian can add new book by entering author, title, published date and number of copies.



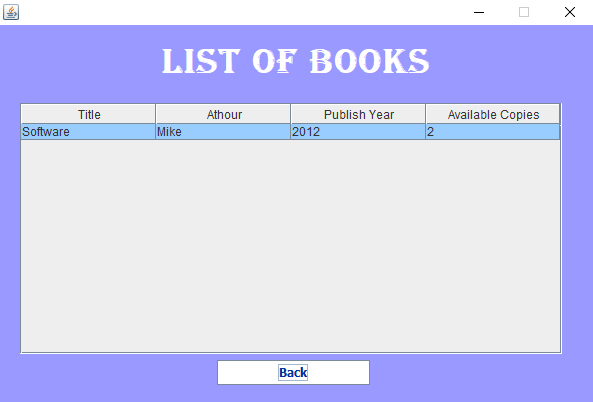
## Modify Books

Librarian can update and delete a book in this page.



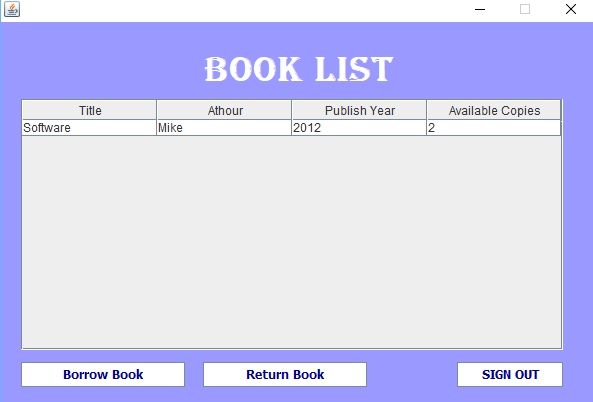
## Book List

The list of the books and details will be shown.



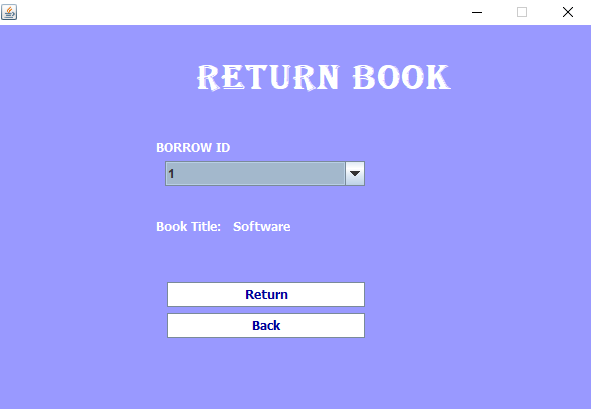
## Student Home page

Student will see this page after login which is showing the books details and they can borrow the selected book by clicking on the button.



## Return Book

Student can return the book in this page by choosing the ID and clicking on return button.



# ICE (Internet Communications Engine)

The Internet Communications Engine, or Ice, is an open source RPC framework developed by ZeroC. It provides SDKs for C++, C#, Java, JavaScript, MATLAB, Objective-C, PHP, Python, and Ruby, and can run on various operating systems, including Linux, Windows, macOS, iOS and Android.

Ice implements a proprietary communications protocol, called the Ice protocol that can run over TCP, TLS, UDP, and Web Socket. As its name indicates, Ice can be suitable for applications that communicate over the Internet, and include functionality for traversing firewalls.

# Conclusion

RMI is a very good way to implement distributed systems if they can be builds in pure Java. The programmer doesn’t have to care about the distribution during development. So the objects can be distributed dependent on runtime. The protocol of the communication is checked by the Java compiler so the programmer doesn’t have to care about this. The disadvantage is the fact that RMI is only available for Java. If server or client cannot be built in Java another method has to be used. A possibility can be CORBA or if the communication has to be fast, sockets have to be used, to reduce overhead, that is inevitable if higher level protocols are used.

# References

1880-1954, H. (2013). *Java head*. [Place of publication not identified]: Hardpress Ltd.

Java Concept Of The Day. (2017). *Inheritance In Java - Java Concept Of The Day*. [online] Available at: http://javaconceptoftheday.com/inheritance-in-java/ [Accessed 9 Jul. 2017].

Object-Oriented Programming Languages: Tools for Effective Communication on Application. (2016). *International Journal of Science and Research (IJSR)*, 5(8), pp.406-411.

Sabharwal, C. (1998). Java, Java, Java. *IEEE Potentials*, 17(3), pp.33-37.