Table of Contents

[Introduction 2](#_Toc525243413)

[Use Case Diagram 3](#_Toc525243414)

[Projects and Files 4](#_Toc525243415)

[Database 5](#_Toc525243416)

[System Validation 6](#_Toc525243417)

[Unit testing 6](#_Toc525243418)

[Test parts 6](#_Toc525243419)

[Test deliverables 6](#_Toc525243420)

[DCOMS Technologies 7](#_Toc525243421)

[RMI 7](#_Toc525243422)

[Sockets 7](#_Toc525243423)

[CORBA Technology 8](#_Toc525243424)

[Screen Shots 9](#_Toc525243425)

[Login 9](#_Toc525243426)

[Register Student 10](#_Toc525243427)

[Librarian Panel 11](#_Toc525243428)

[Add Book 12](#_Toc525243429)

[Edit/Delete Books 13](#_Toc525243430)

[View Search Book 14](#_Toc525243431)

[Student Home page 15](#_Toc525243432)

[Return Book 16](#_Toc525243433)

[ICE (Internet Communications Engine) 17](#_Toc525243434)

[Conclusion 17](#_Toc525243435)

[References 18](#_Toc525243436)

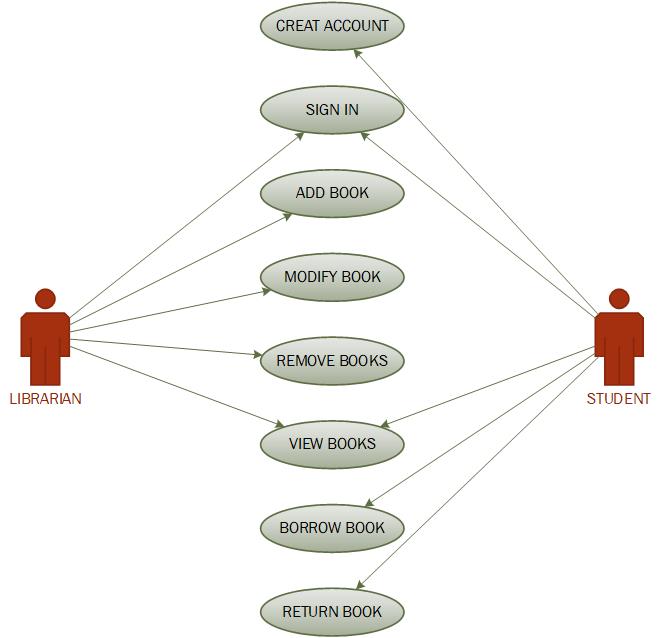
# Introduction

Asia Pacific University has a big library which needs an application to manage the list of available books and students which is borrowing the books.

The application is developed using Java RMI technology which make students able to view the book titles, details and give them ability to borrow and return the book using the system, it also has a functionality which make librarian able to manage the books inside the library like adding, removing or editing the books details.

A good solution is to distribute an application over three hosts. One that provides and saves the data, another one, which is called application server, for the logic, and the third host is the respective host of the user that provides an interface. This architecture allows to exchange each of them independent of the others. This architecture is called 3-tiers architecture. An example for this architecture is a system of a bank that provides account management for the user. It consists of a database server, a web server that uses CGI or other methods to provide dynamic content of web pages and a browser that is used by the client. To achieve the four points mentioned above the database can be mirrored to a second server that can replace the first on if it fails. The consistency of the database can be achieved by using transactions. Maintainability is already achieved by using a browser that only displays dynamic generated content so if the program changes this only has to be done on the application server and the load can be balanced by using more than one application server.

# Use Case Diagram

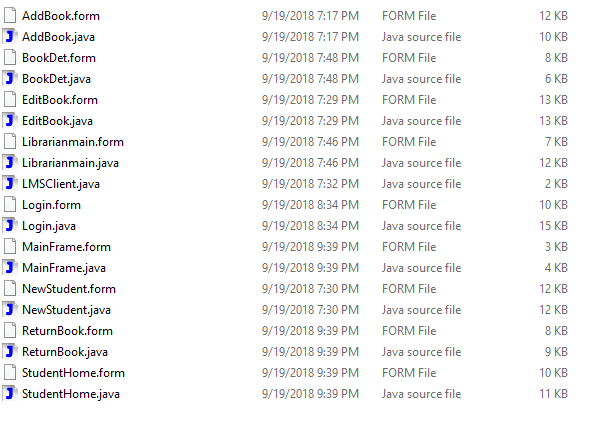


# Projects and Files

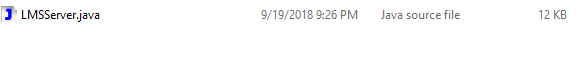
The RMI application has three different projects Client side, Server Side and interface as shown below.



The client has all the interface frame which is showing data to user and receive information from user.

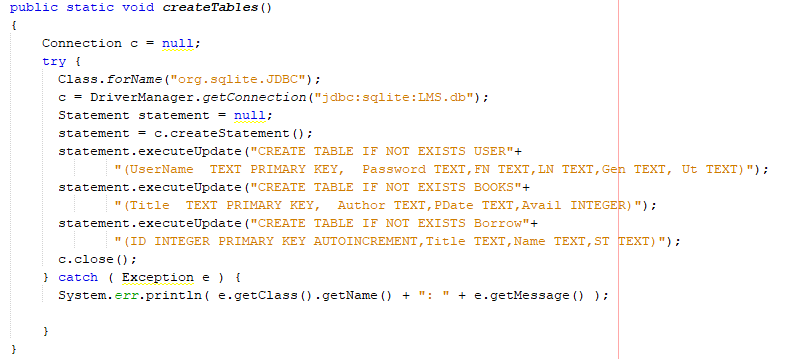


Serve side has the main class which is doing all the processed and interact with database.



## Database

SQLite is used as database to manage the data in this application, on the starting of the server the system will check for database and table existence and if any of the required table or database is not available it will create it using the blow codes.



# System Validation

## Unit testing

**Unit Testing** is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.

The developer will perform this test one time, before the first release make sure each part of system will work properly.

### Test parts

The part that will be tested for this type of testing is:

* Testing all the classes on the server side of the application
* Test all graphical interface pages in client side of application
* Test all functionality of database in the system

### Test deliverables

Following items should be delivered before the testing begins:

* The source code of each class
* The interface on each page
* The tables of database

Following items should be delivered after testing finishes:

* Any bug in each part of system
* Any issue that stops each part of the system

# DCOMS Technologies

## RMI

The main features of RMI are the following:

* method calls between different virtual machines
* transparency
* Java specific

RMI provides the call of methods on other virtual machines that are located on the same host like the caller or on machines that are connected over a network. The architecture of RMI can be seen in figure 1. The calls are transparent for the application what means that there is no difference between calling a local method and a remote method. To enable this a ”stub class” of the server object has to be compiled using the RMI compiler (rmic). This server stub has to be copied to the client. This stub acts as a proxy for the real server objects. If the client calls the methods of this stub the call is passed over the network to the server. The remote reference layer on the sever side calls the method of a skeleton of the server object that passes the call to the real object. Those skeletons are only needed in version 1.2 and lower of the JDK since newer versions have the reference of the server object already in the remote reference layer. RMI even allows to pass whole objects as parameters of methods to remote hosts transparently for the caller. Those remote objects can be used on the server side like local objects. The server then needs a stub class of the client object. With such an object callbacks to the client are enabled.

## Sockets

Sockets provide a connection oriented connection if TCP is used. A socket connection can be compared to a bidirectional pipe. After the connection is established data can be transferred byte wise in both directions. The socket itself doesn’t know anything about the meaning of the transferred data. So the data has to be encapsulated in a control structure that enables the sender and receiver to interpret the meaning of the data. Methods to compose and decompose the data have to be written by the programmer. The advantage of self-defined protocols is that they don’t have much overhead because they are specialized for the appropriate application. But the disadvantage is that it is very difficult and error prone to develop a protocol that is fast and extensible for future use. Another disadvantage is that the programmer has to handle all the possible errors. And the most significant disadvantage is that sockets don’t provide method calls. To call methods a protocol is needed, that allows interpreting data and calling the appropriate methods on a server. After the method that was called has finished the result has to be transferred back to the caller. This again requires putting the result data in a structure, transferring them over a network and decomposing the data. If a method changes the amount of parameters the protocol has to change. This reduces productivity. A great part of the work is spent to implement the protocol.

## CORBA Technology

Common Object Request Brokerage Architecture (CORBA) is an industry standard distributed object model. A key feature of CORBA is the language-neutral Interface Definition Language (IDL) to specify data types, attributes, operations, interfaces, and so on. The languages that are CORBA-compliant should have their own IDL mapping to generate the source code for their respective languages.

Distributed objects have a server and the client. The server usually provides a remote interface and the client calls the methods defined in the remote interface after receiving a remote interface handle. CORBA also has a concept similar to the RMI technology. However, in RMI the remote objects are available only for client applications written in Java language. On the other hand, in CORBA the remote objects are available for any application written in a CORBA-compliant language. This is made possible using the CORBA IDL.

A CORBA Object Request Broker (ORB) connects a client application with the objects it wishes to use. Unlike traditional client/server technologies, the client application need not know where the object resides. It needs to know only the object's name and the parameters that are used to send and receive messages. ORB locates the object, routes the request, and returns the result.

# Screen Shots

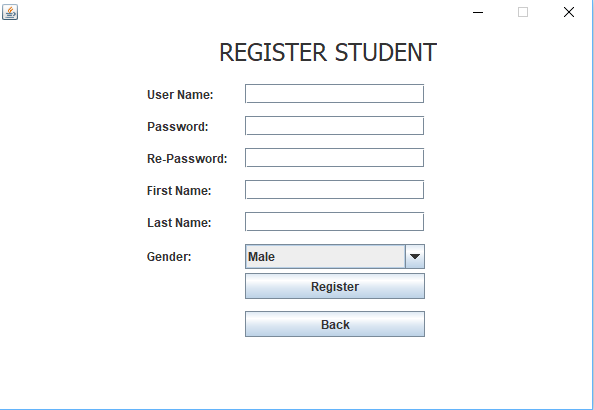
## Login

Librarian and students should enter their credential in this page to login into the page if the credential is invalid error message will be shown.



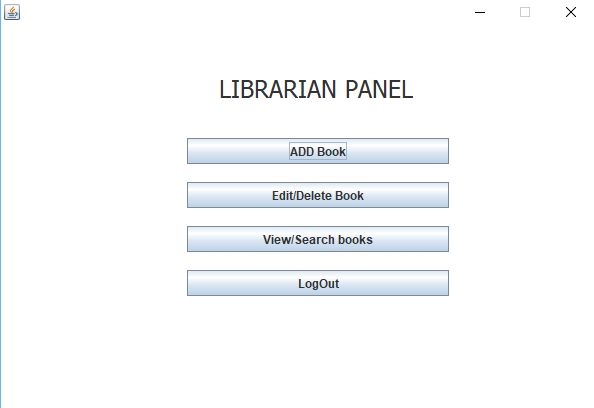
## Register Student

Student should create a new account before login in this page by providing the required information to be able to login into the system.



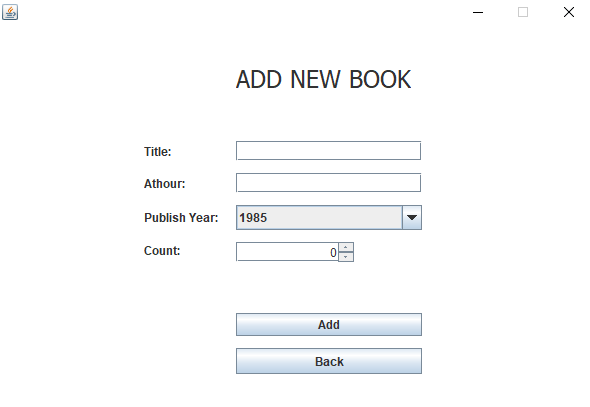
## Librarian Panel

Librarian can access to different functionality by using their main panel.



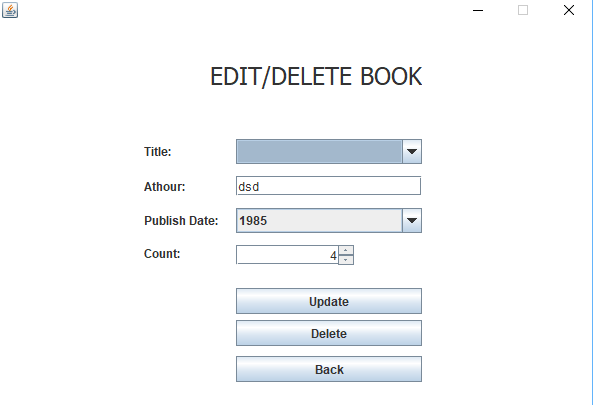
## Add Book

Librarian can add new book by entering all the required information.



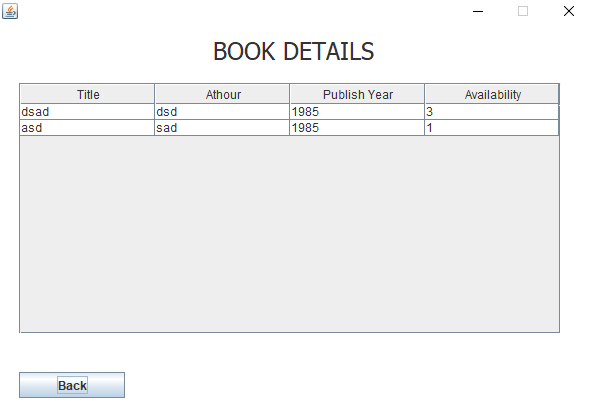
## Edit/Delete Books

The book details can be updated by librarian, also they can delete a book title.



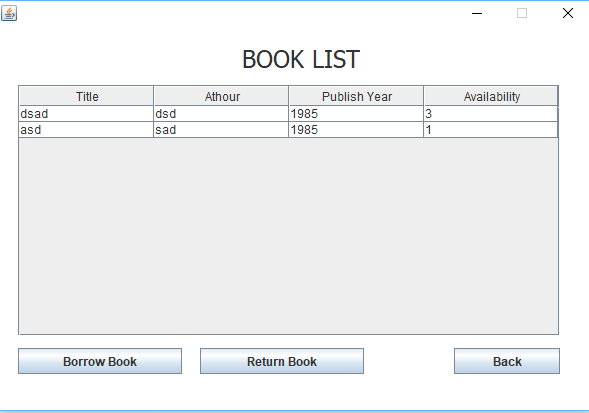
## View Search Book

The list of the books and details will be shown.



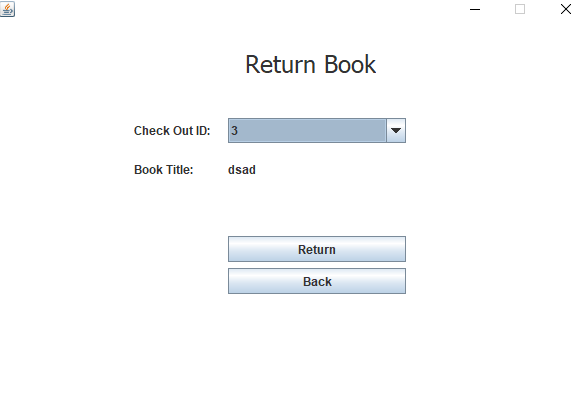
## Student Home page

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, often simply referred to as ICE, is a platform enabling distributed computing. It’s both free and open source, and simple to get started with. Distributed computing is useful to delegate a large task to several machines, or even to a single machine better equipped to handle a task. For example, perhaps you might want to assign a task with lots of floating point operations to a machine with specialized hardware designed to execute floating point operations more efficiently, but do the rest of the work on a machine that otherwise better fits your needs.

# Conclusion

ne of the central and unique features of RMI is its ability to download the definition of an object's class if the class is not defined in the receiver's Java virtual machine. All of the types and behavior of an object, previously available only in a single Java virtual machine can be transmitted to another, possibly remote, Java virtual machine. RMI passes objects by their actual classes, so the behavior of the objects is not changed when they are sent to another Java virtual machine. This capability enables new types and behaviors to be introduced into a remote Java virtual machine, thus dynamically extending the behavior of an application. The compute engine example in this trail uses this capability to introduce new behavior to a distributed program.

# References

"Checking The Kernel Memory Usage With Process Explorer". *Esupport.trendmicro.com*. N.p., 2016. Web. 9 May 2016.

Horstmann, Cay S, Gary Cornell, and Cay S Horstmann. *Core Java*. Upper Saddle River, NJ: Prentice Hall/Sun Microsystems Press, 2008. Print.

Hutton, Peter et al. *Java*. [Hong Kong]: APA Publication, 1993. Print.

"Java Input Using Java Scanner". *Java-made-easy.com*. N.p., 2016. Web. 9 May 2016.

"Java.Util.Scanner Class". *www.tutorialspoint.com*. N.p., 2016. Web. 9 May 2016.