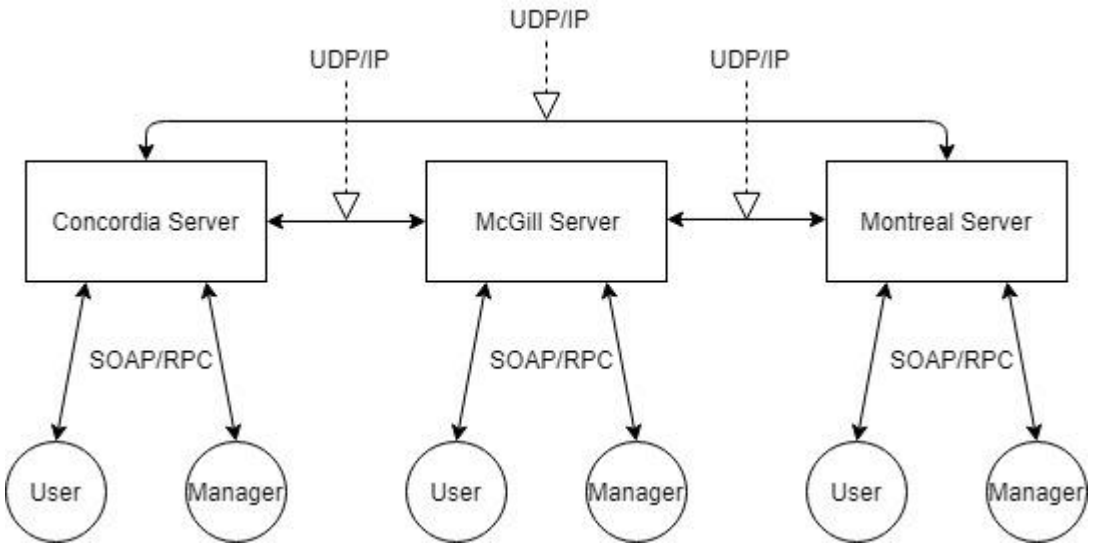
Assignment 3

Techniques:

In this assignment I have used Web Services. Web service is a standard architecture for distributed system. In web services request and reply encoded in HTML, using HTTP to issue request to the site. Web Services generalize this model so that computers can talk to computers. I have set different URL for different servers. The architecture of this design is as follows:




```
URL compURL = new URL("http://localhost:1111/comp?wsdl");
QName compQName = new QName("http://Servers/", "ConcordiaServerService");
Service compService = Service.create(compURL, compQName);
siu = compService.getPort(ServerInterface.class);

URL compURL = new URL("http://localhost:2222/comp?wsdl");
QName compQName = new QName("http://Servers/", "McGillServerService");
Service compService = Service.create(compURL, compQName);
siu = compService.getPort(ServerInterface.class);

URL compURL = new URL("http://localhost:3333/comp?wsdl");
QName compQName = new QName("http://Servers/", "MontrealServerService");
Service compService = Service.create(compURL, compQName);
siu = compService.getPort(ServerInterface.class);
```

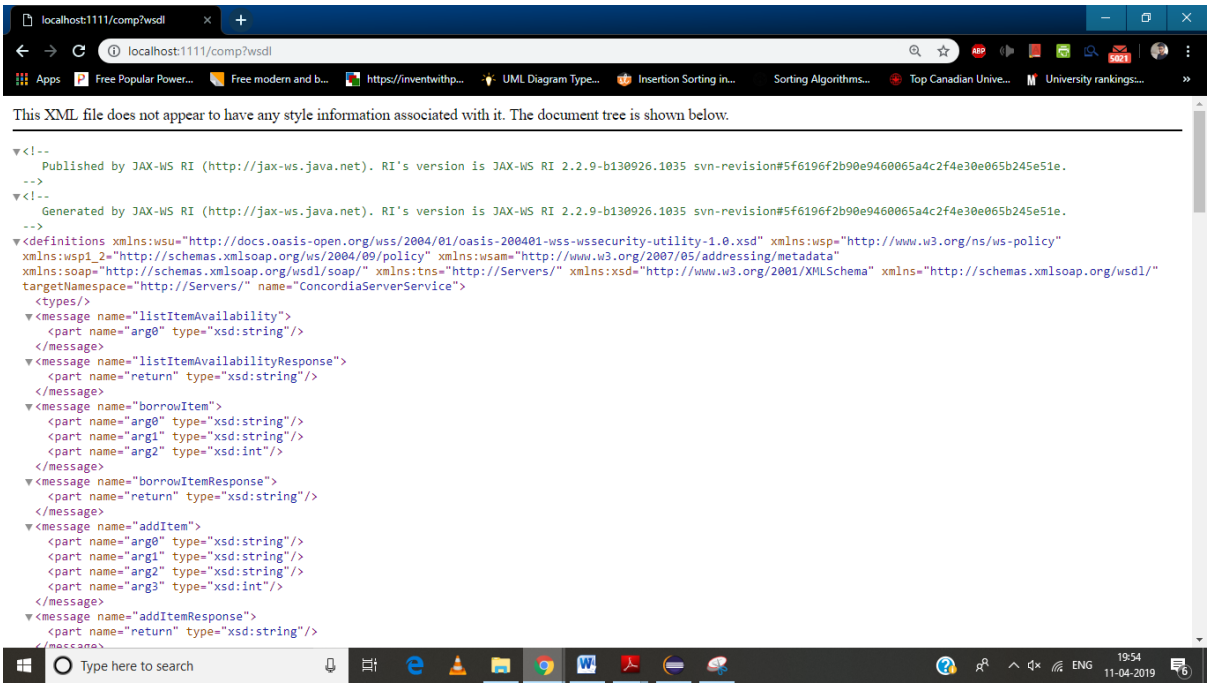
The client obtains a connection, issues a request the server processes the request, issues a response, and closes the connection thereafter. I have set different URL (SOAP address location) for different host with different port numbers like:

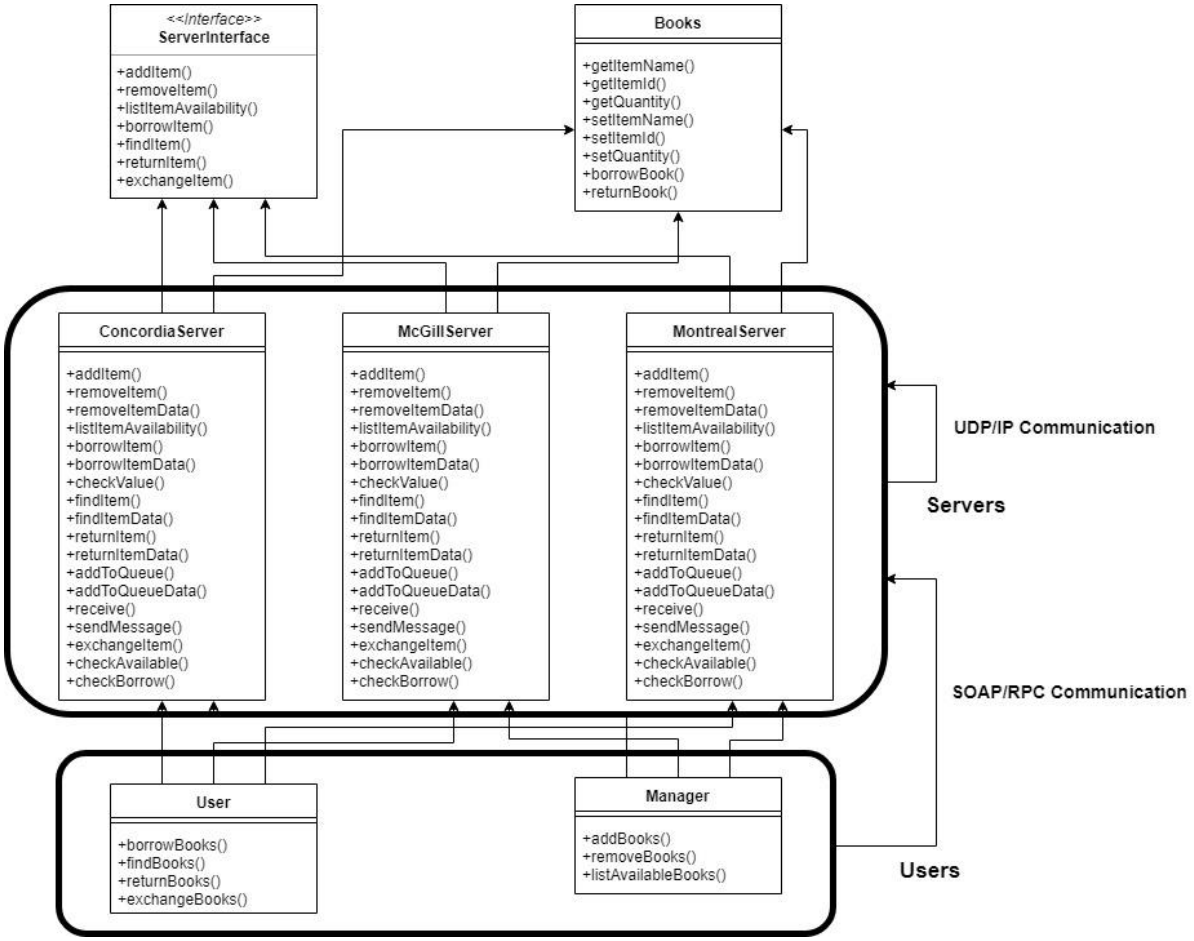
http://localhost:1111/comp?wsdl

One can go to this URL to find target namespace and name for that particular server.

```
targetNamespace="http://Servers/" name="ConcordiaServerService">
```

From there we can also see the document tree for that class. HTTP response body contains the contents of the web object requested.





Scenarios:

Test Case 1: To test concurrent

Two users want to borrow same book which has only one quantity so only one user get the book.

```
public static void firstUser(String arg[]) throws MalformedURLException, RemoteException {
    String user_id = "CONU1111";

    URL compURL = new URL("http://localhost:1111/comp?wsdl");
    QName compQName = new QName("http://Servers/", "ConcordiaServerService");
    Service compService = Service.create(compURL, compQName);
    siu = compService.getPort(ServerInterface.class);
    System.out.println("Concordia's user");

    String item_id = "CON1122";
    int number_of_days = 26;
    String response;
    response = siu.borrowItem(user_id, item_id, number_of_days);
    System.out.println("User 1:"+response);
}

public static void secondUser(String arg[]) throws MalformedURLException, RemoteException {
    String user_id = "MCGU1111";

    URL compURL = new URL("http://localhost:2222/comp?wsdl");
    QName compQName = new QName("http://Servers/", "McGillServerService");
    Service compService = Service.create(compURL, compQName);
    siu = compService.getPort(ServerInterface.class);
    System.out.println("McGill's's user");

    String item_id = "CON1122";
    int number_of_days = 17;
    String response;
    response = siu.borrowItem(user_id, item_id, number_of_days);
    System.out.println("User 2:"+response);
}
```

Result:

```
Concordia's user
McGill's's user
User 2:CON1122 is successfully borrowed.
User 1:Book is not available to for borrowing.
```

Test Case 2: To test atomicity

It contains 4 different scenarios

Case 1: User tries already borrowed a book from other server and tries to exchange the same book from the same server. ---> Success

Case 2: User again tries to exchange the book from other server which is not available so atomicity fails. ---> Fail

Case 3: User again tries to exchange with the unborrowed book so atomicity fails. ---> Fail

Case 4: User already borrowed a book from other server and tries to exchange a book from the same server. ---> Fail

```
URL compURL = new URL("http://localhost:1111/comp?wsdl");
QName compQName = new QName("http://Servers/", "ConcordiaServerService");
Service compService = Service.create(compURL, compQName);
siu = compService.getPort(ServerInterface.class);
System.out.println("Concordia's user");

String item_id = "MCG2288";
int number_of_days = 17;
String response;
response = siu.borrowItem(user_id, item_id, number_of_days);
System.out.println(response);

String new_item_id = "MCG2222", old_item_id = "MCG2288";
response = siu.exchangeItem(user_id, new_item_id, old_item_id);  // Success
System.out.println("Case 1:"+response);

new_item_id = "CON1144";
old_item_id = "MCG2222";
response = siu.exchangeItem(user_id, new_item_id, old_item_id);  // Fail
System.out.println("Case 2:"+response);

new_item_id = "CON1188";
old_item_id = "CON1144";
response = siu.exchangeItem(user_id, new_item_id, old_item_id);  // Fail
System.out.println("Case 3:"+response);

item_id = "CON1188";
number_of_days = 26;
response = siu.borrowItem(user_id, item_id, number_of_days);
System.out.println(response);

new_item_id = "MCG2288";
old_item_id = "CON1188";
response = siu.exchangeItem(user_id, new_item_id, old_item_id);  // Fail
System.out.println("Case 4:"+response);
```

Result:

```
Concordia's user
MCG2288 is successfully borrowed.
Case 1:Successfully exchange MCG2222 with MCG2288
Case 2:Fail exchange CON1144 with MCG2222
Case 3:Fail exchange CON1188 with CON1144
CON1188 is successfully borrowed.
Case 4:Fail exchange MCG2288 with CON1188
```

Most difficult/important part:

The most difficult part in this assignment was:

[1] Problem: To make a system concurrent.

Solution:

To deal with this problem I used **synchronized** keyword in every method which allows only one request at a time in multithreaded application.

[2] Problem: To make a different test cases.

Solution:

To make a different test cases which check the concurrent and atomicity of the program.

References:

[1] https://stackoverflow.com/
[2] https://www.draw.io/
[3] https://www.geeksforgeeks.org/
[4] https://www.youtube.com/
[5] https://docs.oracle.com/
[6] https://en.wikipedia.org/
[7] Text book and the slides from Prof. M.L. Liu, California Polytechnic State University