Assignment 2

Distributed Appointment Management System

Subject: Distributed Systems Design (Comp 6231)

Submitted by: Bhargav Bhutwala

Student Id: 40196468

Instructor: R. Jayakumar

Guided by: Brijesh Lakkad

1. About System

- Appointment management is implemented as a distributed system to book and manage appointments across different branches of a corporate appointment management company. The system is built using CORBA architecture and the users can see a single system handling user requests providing location and language transparency. It also manages simultaneous requests with adequate synchronization with the help of multithreading.

Also this system has 3 servers: Montreal, Quebec and Sherbrook and they handle their events individually.

2. Methods used in this system

a. Methods for Patient

   i. **Book Appointment:** Patient can book appointments in their own city multiple time if there is enough booking capacity available for the particular appointment. They can also book appointments from the other city but for the particular customer, they can only book 3 appointments outside their own city in a given week.
   ii. **Cancel Appointment:** Patient can also cancel appointments from their booked appointments.
   iii. **Get Schedule:** Patient can also see their booked appointments in this section.

b. Methods for Admin

i. **Add Appointment:** Admin of one city can add appointments in their database with appointment Type as Physician, Surgeon and Dental shows and their respective booking capacity.
ii. **Remove Appointment:** Admin can also remove any appointment from their database at any time.
iii. **List Availability:** Admin can see the availability for any appointment Type from the all servers.

3. Protocols Followed

a. In this system, Common Object Request Broker Architecture is used to communicate between server and client. Also, the UDP Protocol is used to do inter-server communication.

4. Architecture

a. The architecture used is client-server method. Client can be manager or a customer. There are three servers for three cities: Montreal, Quebec and Sherbrook.

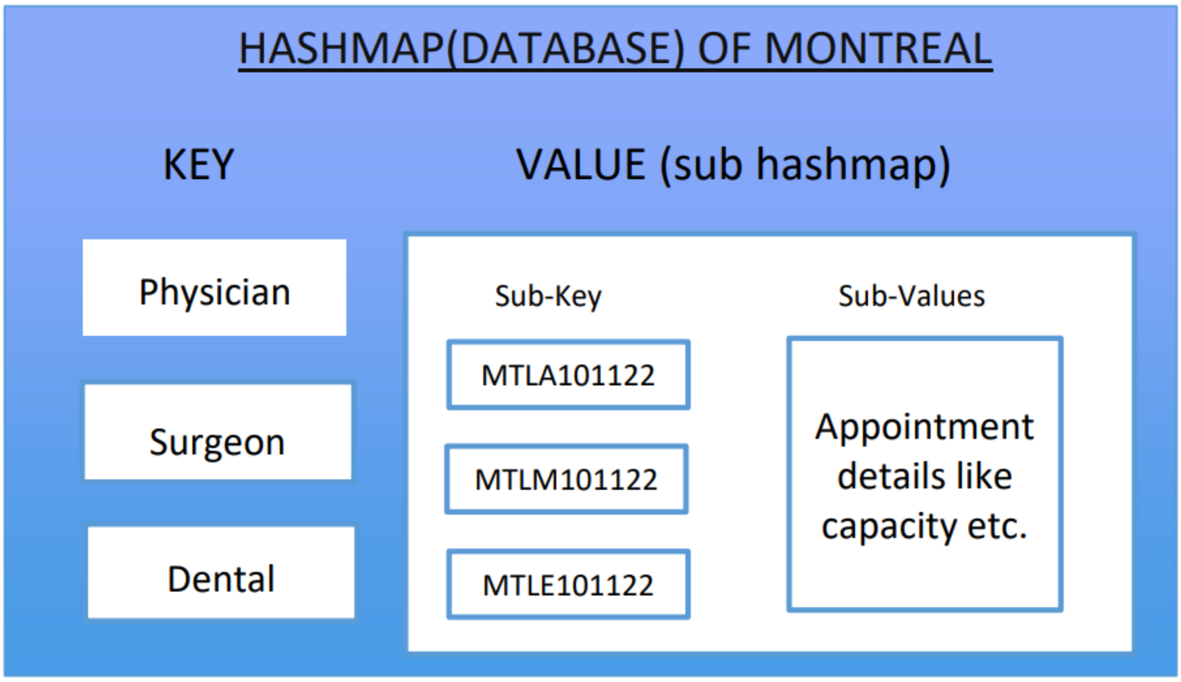
i. **Appointments.idl** contains interfaces for CORBA, it allows transparency as if client is calling its own local methods but instead client calls methods of the server through it.
ii. **Client.java** handles all the operations to be performed by customer and manager. It makes necessary calls with servers and the message from it.
iii. **Mtl_server.java** is responsible for doing all the operations related to the Montreal city and giving the final message output to the client.

iv. **Que_Server.java** is responsible for doing all the operations related to the Quebec City and giving the final message output to the client.

v. **She_Server.java** is responsible for doing all the operations related to the Sherbrook city and giving the final message output to the client.

5. **Data Structure**

a. We have used Hash Map to store data on server side; it stores data in form of key-value pair. We have also used Array list for storing multiple values for the same key.



6. **Logger**

a. We have also maintain log for each client (Patient as well as Admin as well as Server). This log will have all the data from client entering the database till leaving the server The log will have all the activities recorded for the server client and manager.

b. **Log Format**

   i. Log Data consist of following details:

      1. Date and type the request was sent.
      2. Request Type
      3. Request Parameter
      4. Request successful/fail
      5. Server response for the particular request

7. Flow

a. The client sends the request to the respective server.
b. Server fetches the request data.
c. It checks if there is any request for another server, if yes then it forwards the request to the request to respective server.
d. The servers of different location receive the request and creates the thread to process the request.
e. The server then responds to the client with appropriate data.

8. Challenges Faced

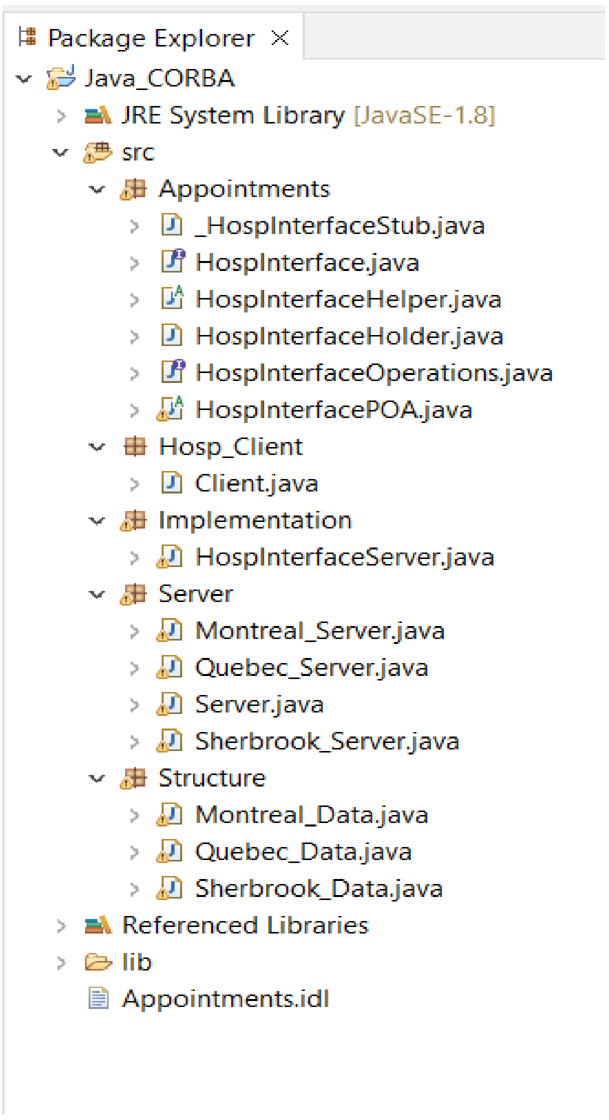
a. Implementation of synchronization while managing multiple event requests at the same time has been challenging.

9. **Test Cases**

Scenario	**Actual Output**	**Expected Output**
Patients can book any number of appointments in their own city but they cannot book more than 3 appointments outside their city.	Already booked 3 appointments outside your city	Already booked 3 appointments outside your city
If the user tries to book the appointment that is not added by the admin then the error occurs.	Enter the valid appointment type	Enter the valid appointment type
If the book is full patients cannot book the appointments.	No appointment found	No appointment found
Patient can perform only patient operations they cannot perform admin operations.	1.Book Appointment 2. List Appointment Schedule 3. Cancel Appointment 4. Swap Appointment	1.Book Appointment 2. List Appointment Schedule 3. Cancel Appointment 4. Swap Appointment
If old/new event does not exist for swap event, an	Unable to book new appointment	Unable to book new appointment

error message is shown.		
The swap event throws an error if new event add operation exceeds the month's max limit.	Cannot book appointment already reached maximum limit	Cannot book appointment already reached maximum limit

10. Code Structure



```
Package Explorer
Java_CORBA
  JRE System Library [JavaSE-1.8]
  src
    Appointments
      _HospInterfaceStub.java
      HospInterface.java
      HospInterfaceHelper.java
      HospInterfaceHolder.java
      HospInterfaceOperations.java
      HospInterfacePOA.java
    Hosp_Client
      Client.java
    Implementation
      HospInterfaceServer.java
    Server
      Montreal_Server.java
      Quebec_Server.java
      Server.java
      Sherbrook_Server.java
    Structure
      Montreal_Data.java
      Quebec_Data.java
      Sherbrook_Data.java
  Referenced Libraries
  lib
  Appointments.idl
```