**COMP 6231 Assignment 1 Design Documentation**

**Name**: Yajing Liu **Student ID**: 40268996

This project is to develop a distributed health care management system (DHMS) for managing medical appointments. Admins can use this system to add, list, and remove appointments. And patients can use this system to book, list, and cancel appointments.

**Project Architecture**

A diagram of a computer

Description automatically generated

Figure 1 UML class diagram

As shown in Figure 1, there are nine classes in this project. The following are the details about these nine classes.

**Driver**: This class handles command line user interface. It will create new client object and call clients’ corresponding methods based on the input.

**Client**: This class is an abstract class handling all methods of clients.

**PatientClient**: This class is a subclass of Client. It calls server’s methods in bookAppointment, getAppointment and cancelAppointment. It calls printInvalidCommandMessage in the remaining methods.

**AdminClient**: This class is a subclass of Client. It calls server’s methods in all six methods.

**AppointmentRegistry**: This class is the registry of three servers. It creates local registry and rebinds three new server objects with names.

**AppointmentInterfaces**: This is an interface to define server behaviors.

**MontrealServer**: This class is the Montreal server. It implements methods in AppointmentInterfaces and uses main method to listen other servers’ requests.

**QuebecServer**: This class is the Quebec Server. It implements methods in AppointmentInterfaces and uses main method to listen other servers’ requests.

**SherbrookeServer**: This class is the Sherbrooke server. It implements methods in AppointmentInterfaces and uses main method to listen other servers’ requests.

**Major Techniques**

**RMI (in Java)**

I used RMI (in Java) for client-server communication. I create local registry and rebind three new server objects with names in AppointmentRegistry (Figure 2). I search the server objects by names (Figure 3) and call the corresponding methods in client (Figure 4). I use server to handle method calling and finish jobs.

A screen shot of a computer program

Description automatically generated

Figure 2 AppointmentRegistry code

A screen shot of a computer code

Description automatically generated

Figure 3 Get server object by name in Client

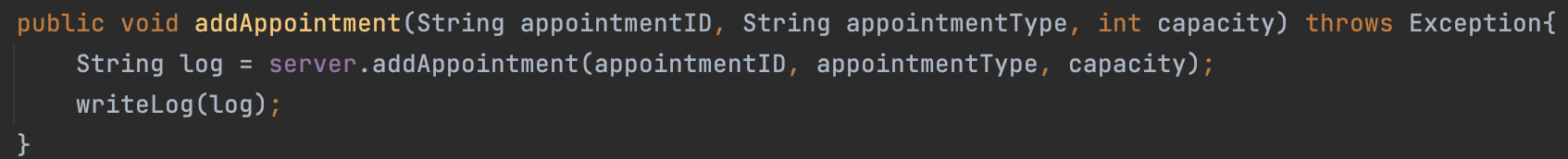


Figure 4 Call corresponding methods in AdminClient

A screen shot of a computer program

Description automatically generated

Figure 5 addAppointment in server

**UDP/IP (in Java)**

I use UDP/IP to handle server-server communication. Server can send request to get other servers’ appointments (Figure 6). And other servers can listen to the request and give response (Figure 7). The server which sends request can use the response to continue the work (Figure 6).

A screen shot of a computer program

Description automatically generated

Figure 6 Send UDP/IP request

A screen shot of a computer program

Description automatically generated

Figure 7 Receive UDP/IP request and sent response

**Synchronized**

I use synchronized keyword in bookAppointment to handle multiple users to perform operations for the same records as the same time (Figure 8).

A screen shot of a computer code

Description automatically generated

Figure 8 Synchronized

**Test Scenarios**

1. To verify that a patient can only perform a patient operation and cannot perform any admin operations, I use the following input and get the following output (Figure 9).

A screen shot of a computer

Description automatically generated

Figure 9 Invalid patient command

2. To verify that an admin can add an appointment, I use the following input and get the following output (Figure 10).

A screenshot of a computer

Description automatically generated

Figure 10 Add appointment

3. To verify that an admin cannot add same appointment again, I use the following input and get the following output (Figure 11).

A screenshot of a computer

Description automatically generated

Figure 11 Add same appointment

4. To verify that an admin can remove an appointment, I use the following input and get the following output (*Figure 12*).

A screen shot of a computer

Description automatically generated

Figure 12 Remove appointment

5. To verify that there is no deletion performed if an appointment does not exist, I use the following input and get the following output (Figure 13).

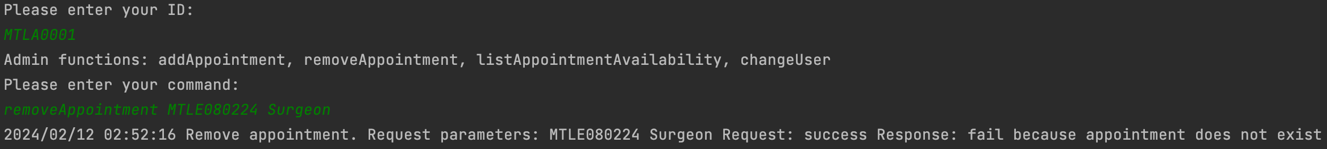


Figure 13 Remove nonexistent appointment

6. To verify that if an appointment exists and a patient has booked that appointment, delete the appointment and book the next available appointment for that patient, I use the following input and get the following output (Figure 14).

A screenshot of a computer

Description automatically generated

Figure 14 Delete a booked appointment

7. To verify that an admin cannot delete an appointment if patient cannot find next available slot, I use the following input and get the following output (*Figure 15*).

A screen shot of a computer

Description automatically generated

Figure 15 Delete an appointment but no next available slot for patient

8. To verify that an admin can list appointment in all cities, I use the following input and get the following output (Figure 16).

A screenshot of a computer program

Description automatically generated

Figure 16 List appointment availability

9. To verify that a patient can book an appointment, I use the following input, and get the following output (Figure 17).

A screenshot of a computer program

Description automatically generated

Figure 17 Book appointment

10. To verify that a patient cannot book a same type of appointment in a day, I use the following input and get the following output (Figure 18).

A screenshot of a computer

Description automatically generated

Figure 18 Book same type of appointment in a day

11. To verify that a patient can book an appointment in other cities, I use the following input and get the following output (Figure 19).

A screen shot of a computer

Description automatically generated

Figure 19 Book an appointment in other cities

12. To verify that a patient cannot book more than three appointments in other cities in a week, I use the following input and get the following output (Figure 20).

A screenshot of a computer

Description automatically generated

Figure 20 Book more than three appointments in other cities

13. To verify that a patient can get appointment schedule, I use the following input and get the following output (Figure 21).

A screenshot of a computer

Description automatically generated

Figure 21 Get appointment schedule

14. To verify that a patient can cancel appointment, I use the following input and get the following output (Figure 22).

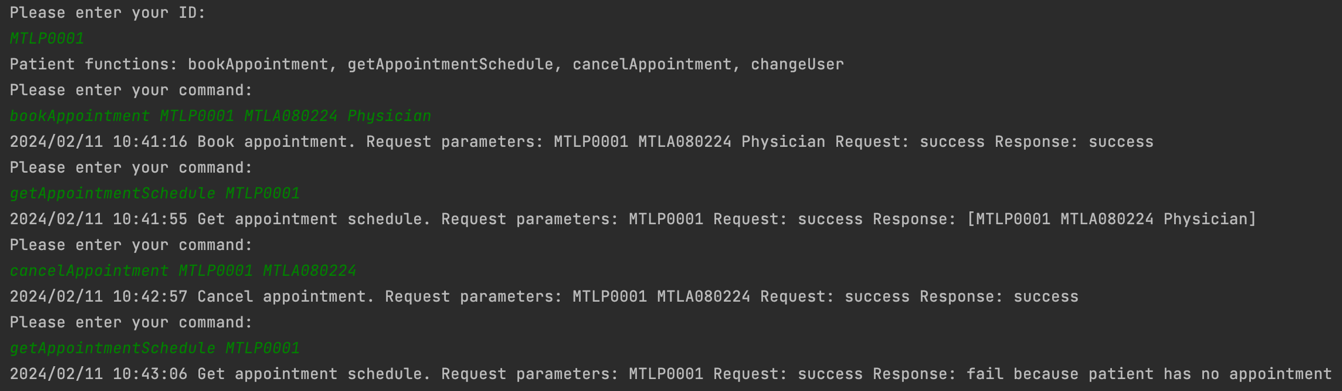


Figure 22 Cancel appointment

15. To verify that a patient can cancel appointment in other cities, I use the following input and get the following output (Figure 23).

A screenshot of a computer

Description automatically generated

Figure 23 Cancel appointment in other cities

16. To verify that a patient cannot cancel a nonexistent appointment, I use the following input and get the following output (Figure 24).

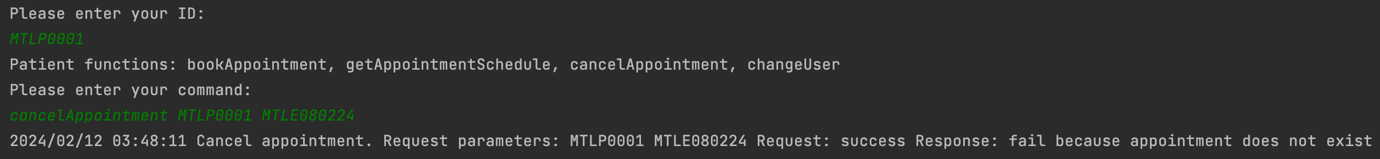


Figure 24 Cancel nonexistent appointment

**Most important parts**

The most important part is to use RMI and UDP/IP to achieve client-server communication and server-server communication. The project can be a distributed system in this way.