Distributed System Design

COMP 6231 – Winter 2022

Concordia University

Department of Computer Science and Software Engineering

Instructor: R. Jayakumar

Distributed Appointment Management System (DAMS) using CORBA

Assignment 2

By:

Yagnik Pansuriya

Student ID:

40197002

Table of Contents

Overview

The Distributed Appointment Management System (DAMS) is a system is used for Hospital Appointment booking. Hospital admins access the system to maintain the information about the patients and their bookings. This system also allows the patients to book or cancel a medical appointment across different hospitals within the medical care system.

1. About System

There are two types of users for this system:

- Admins
- Patients

There are three servers in this system, each located in a particular city:

- Montreal (MTL)
- Sherbrooke (SHE)
- Quebec (QUE)

The system ensures that a client from Montreal can communicate with the Montreal server only and similarly for the other server clients. A server can communicate with other server using TCP/IP or UDP protocol.

2. **Operations performed by the System**

Operations performed by Admins:

- **addAppointment (appointmentId, appointmentType, capacity):** With this operation admins can add appointments with the stated parameters. An admin is allowed to add appointments in his server only.
- **removeAppointment (appointmentId, appointmentType):** With this operation admins can remove a specific appointment identified from the given appointmentID and appointmentType. Admins are allowed to do this operation on their assigned servers only.
- **listAppointmentAvailabitlity (appointmentType):** Admins by using this operation can list all the appointments with their details such as remaining capacity, booked patients, etc. Similar to above operations, Montreal admin can only perform this action with Montreal server only.

Operations performed by Patients:

- **bookAppointment (patientId, appointmentId, appointmentType):** Patient with this operation can book as many appointments in his/her own city but only at most 3 appointments from other cities overall in a week.
- **getBookingSchedule(patientId):** Patients can see all their booked appointments. This operation shows all the appointments within the server i.e, other servers communicate with each other to pass the booking details.
- **cancelAppointment(patientId,appointmentId):** With this operation patient are allowed to cancel a booked operation.
- **swapAppointment (oldAppointmentID, oldAppointmentType, newApoointmentID, newApointemntType) :** With this operation patients can swap an already booked appointment with the new appointment. We have to maintain concurrency for this operation.

To implement this operation, these steps were followed:

1. Check if the new appointment to be booked is available. If yes follow step 2 else return Failure.
2. Check if the appointment to be booked is falling under same week or not. [This helps in maintaining database correctly if rollback is required.] If yes, check if the weekly limit is not exceeded.
3. If not exceeded then cancel the old appointment and book the new appointment.
4. Else book the new appointment and then cancel the old appointment.

*****All the operations of the patients can also be performed by the admin.***

Admins and Patients are identified by a unique adminId and patientId respectively which is constructed from the acronym of their hospital's city and a 4-digit number.

Eg : AdminId :- MTLA1101 , PatientId:-MTLP1102

There are three time slots available for each appointment type on a day : Morning(M), Afternoon(A), Evening(E).

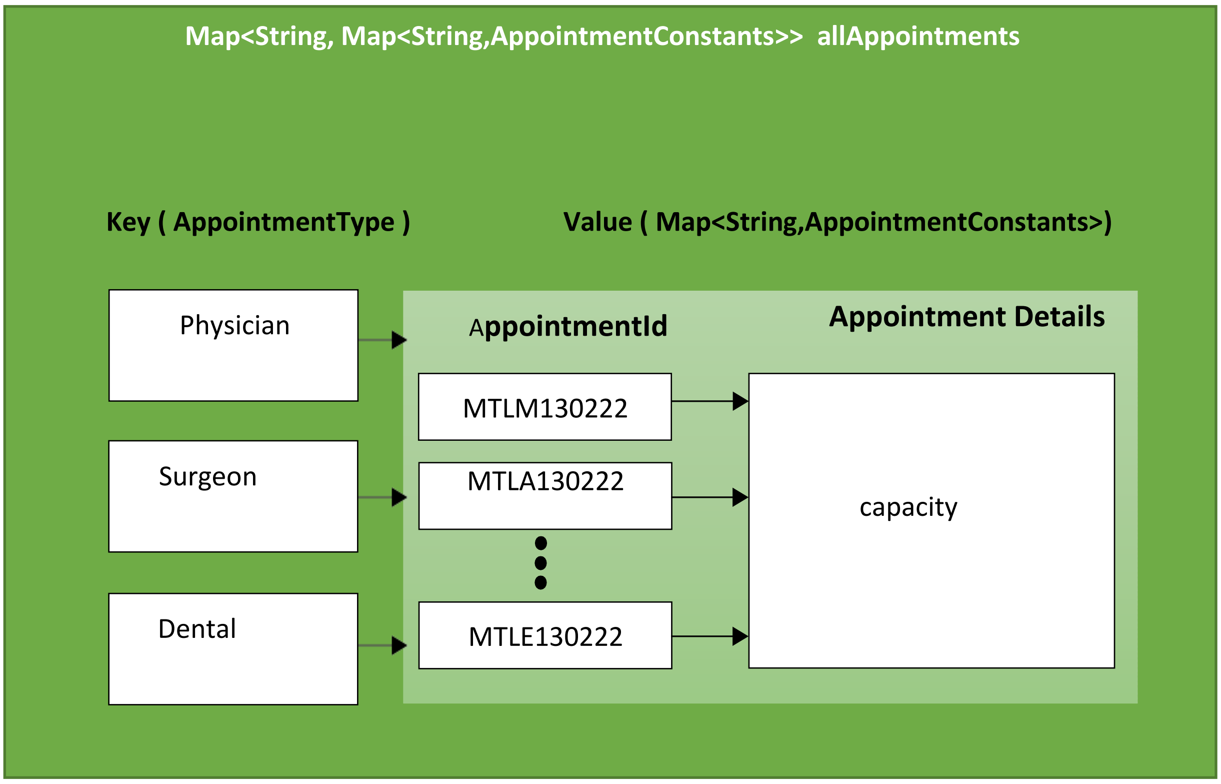
Each appointment is identified by unique appointmentId which is a combination of city, time slot and appointment date.

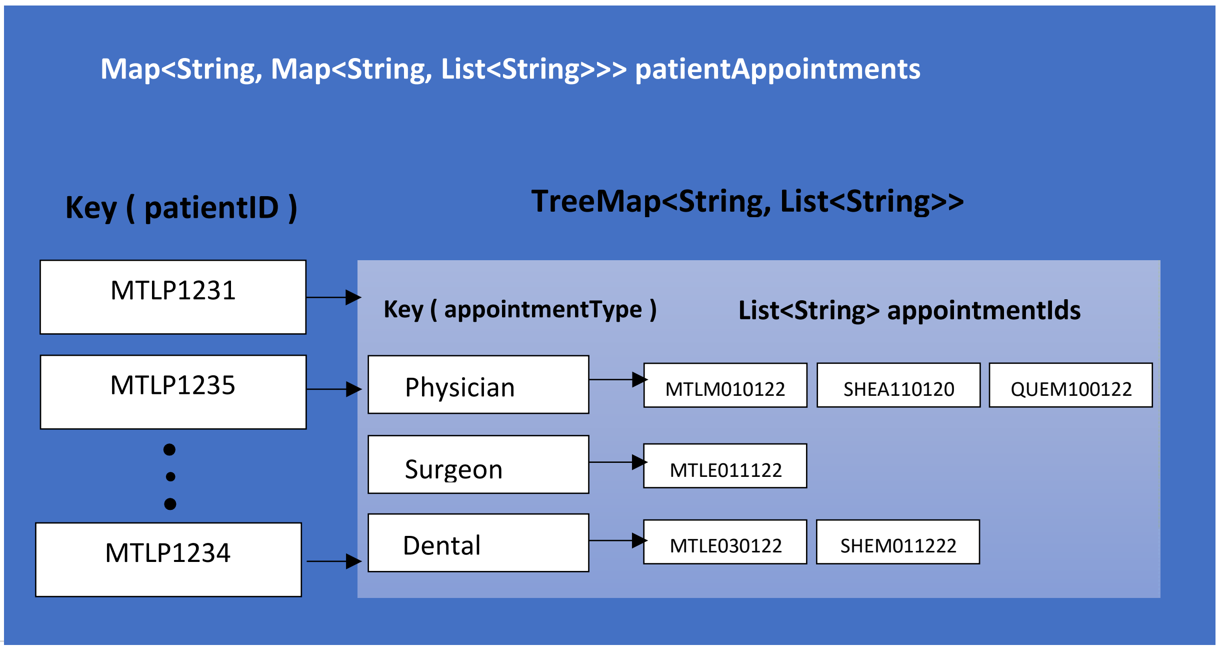
Eg. AppointmentId:- MTLE150222

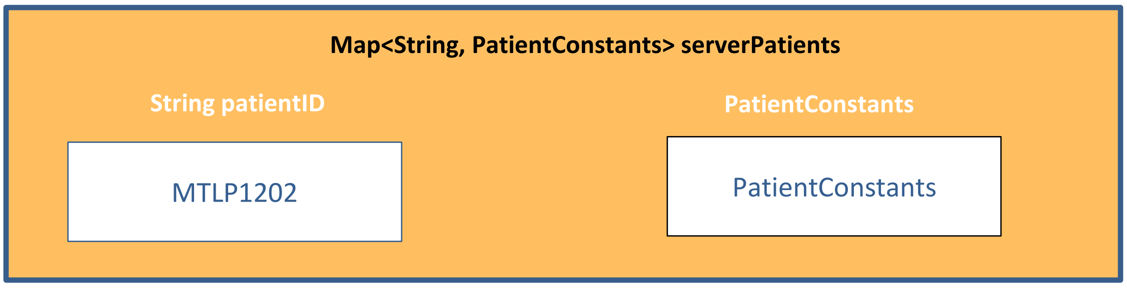
Implementation

Data Structures

All the data is maintained within each server, using three Map structures shown in the figure below.







- Both Server and Client maintain separate logfiles
- Log files for each server are located under directory : \Logs\server\
- Log files for each client are located under directory : \Logs\client\
- ConcurrentHashMap are used to store the data, because it is thread safe and lookup time in O(1).
- Inter server communication for operations such as listAvailabilityUDP() and removeAppointmentUDP() are using UDP protocol to transfer data between servers.
- CORBA is used for all type of communication in this system.

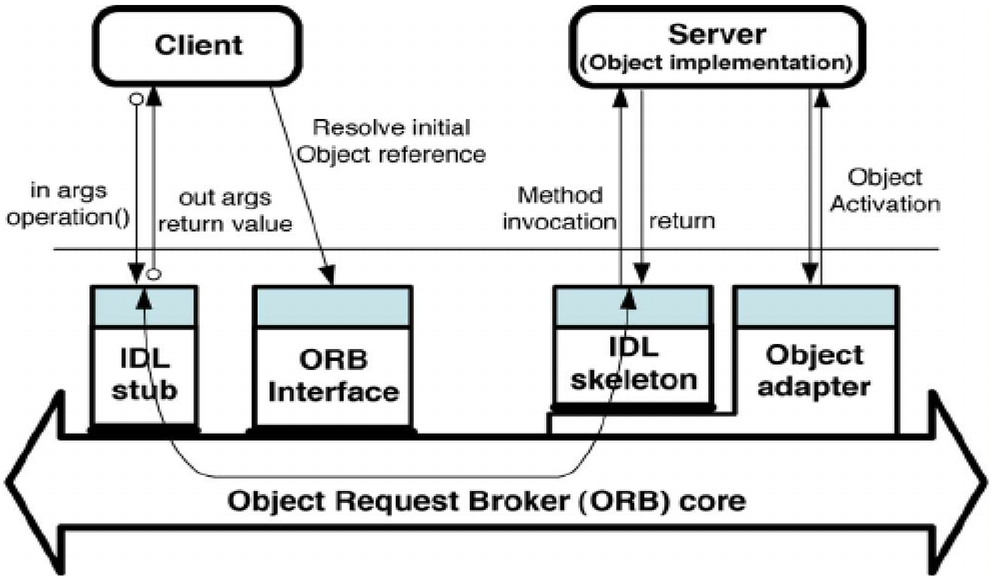


Figure-1 CORBA Architecture

Class Diagram

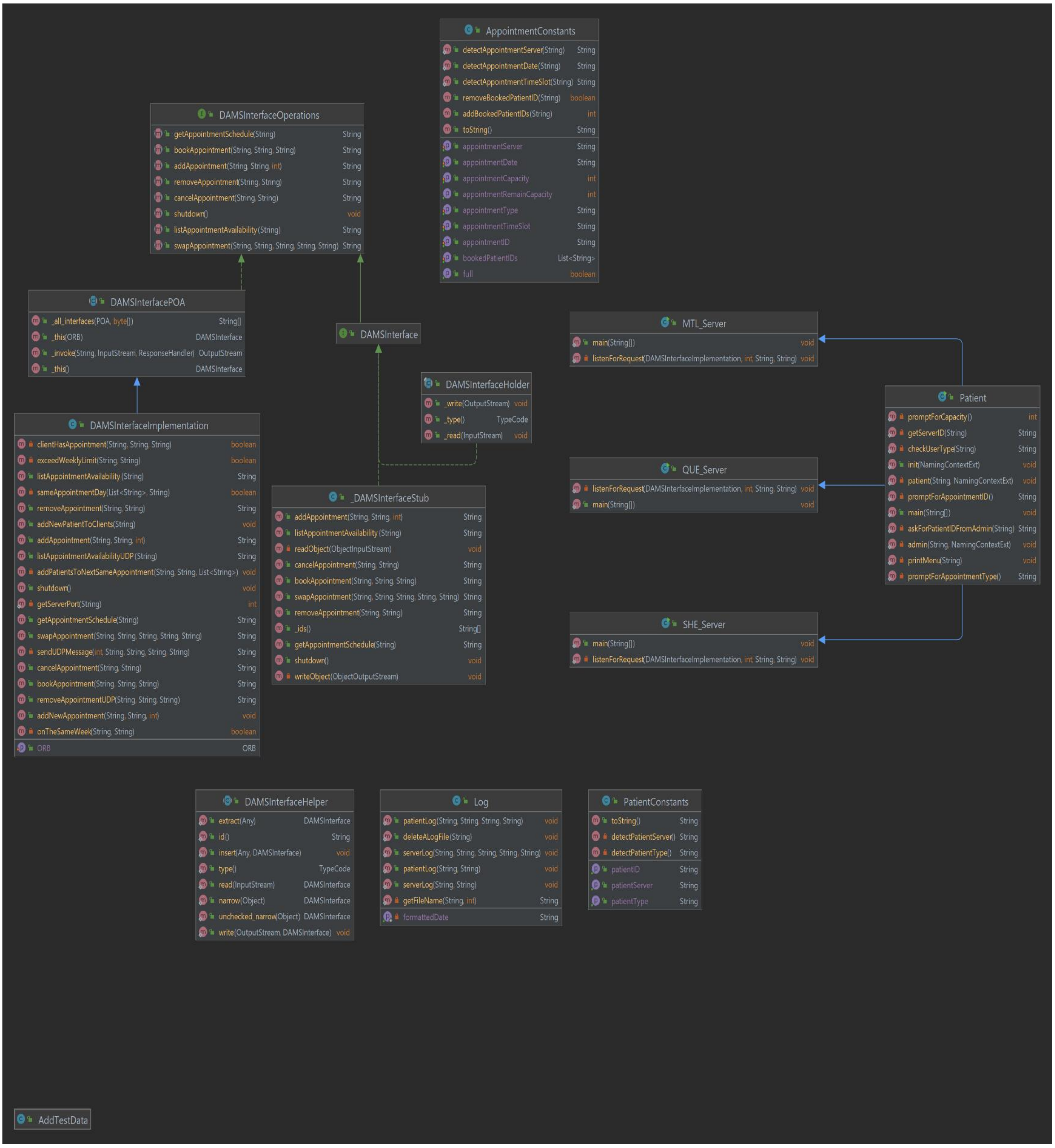

AppointmentConstants
detectAppointmentServer(String) String
detectAppointmentDate(String) String
detectAppointmentTimeSlot(String) String
removeBookedPatientID(String) boolean
addBookedPatientIDs(String) int
toString() String
appointmentServer String
appointmentDate String
appointmentCapacity int
appointmentRemainCapacity int
appointmentType String
appointmentTimeSlot String
appointmentID String
bookedPatientIDs List<String>
full boolean
DAMSInterfaceOperations
getAppointmentSchedule(String) String
bookAppointment(String, String, String) String
addAppointment(String, String, int) String
removeAppointment(String, String) String
cancelAppointment(String, String) String
shutdown() void
listAppointmentAvailability(String) String
swapAppointment(String, String, String, String, String) String
DAMSInterfacePOA
_all_interfaces(POA, byte[]) String[]
_this(ORB) DAMSInterface
_invoke(String, InputStream, ResponseHandler) OutputStream
_this() DAMSInterface
DAMSInterface
MTL_Server
main(String[]) void
listenForRequest(DAMSInterfaceImplementation, int, String, String) void
DAMSInterfaceHolder
_write(OutputStream) void
_type() TypeCode
_read(InputStream) void
Patient
promptForCapacity() int
getServerID(String) String
checkUserType(String) String
init(NamingContextExt) void
patient(String, NamingContextExt) void
promptForAppointmentID() String
main(String[]) void
askForPatientIDFromAdmin(String) String
admin(String, NamingContextExt) void
printMenu(String) void
promptForAppointmentType() String
DAMSInterfaceImplementation
clientHasAppointment(String, String, String) boolean
exceedWeeklyLimit(String, String) boolean
listAppointmentAvailability(String) String
sameAppointmentDay(List<String>, String) boolean
removeAppointment(String, String) String
addNewPatientToClients(String) void
addAppointment(String, String, int) String
listAppointmentAvailabilityUDP(String) String
addPatientsToNextSameAppointment(String, String, List<String>) void
shutdown() void
getServerPort(String) int
getAppointmentSchedule(String) String
swapAppointment(String, String, String, String, String) String
sendUDPMessage(int, String, String, String, String) String
cancelAppointment(String, String) String
bookAppointment(String, String, String) String
removeAppointmentUDP(String, String, String) String
addNewAppointment(String, String, int) void
onTheSameWeek(String, String) boolean
ORB ORB
_DAMSInterfaceStub
addAppointment(String, String, int) String
listAppointmentAvailability(String) String
readObject(ObjectInputStream) void
cancelAppointment(String, String) String
bookAppointment(String, String, String) String
swapAppointment(String, String, String, String, String) String
removeAppointment(String, String) String
_ids() String[]
getAppointmentSchedule(String) String
shutdown() void
writeObject(ObjectOutputStream) void
QUE_Server
listenForRequest(DAMSInterfaceImplementation, int, String, String) void
main(String[]) void
SHE_Server
main(String[]) void
listenForRequest(DAMSInterfaceImplementation, int, String, String) void
DAMSInterfaceHelper
extract(Any) DAMSInterface
id() String
insert(Any, DAMSInterface) void
type() TypeCode
read(InputStream) DAMSInterface
narrow(Object) DAMSInterface
unchecked_narrow(Object) DAMSInterface
write(OutputStream, DAMSInterface) void
Log
patientLog(String, String, String, String) void
deleteALogFile(String) void
serverLog(String, String, String, String, String) void
patientLog(String, String) void
serverLog(String, String) void
getFileName(String, int) String
formattedDate String
PatientConstants
toString() String
detectPatientServer() String
detectPatientType() String
patientID String
patientServer String
patientType String
AddTestData

Figure 2-Class Diagram

Test Scenarios

No	Test Cases	Method	Cases
1.	Login	Id(admin, patient)	1.admin ID →authenticate admin ID 2.Patient ID→ authenticate patient ID
Admin operations			
2.		addAppointment()	1. appointment ID→authenticate appointment ID 2. valid appointment ID→add successfully 3. Existing appointment ID→Cannot add existing appointment 4. Add appointment to another server→ not allowed.
3.		removeAppointment()	1. appointment ID→authenticate appointment ID 2. valid appointment ID→ ➔ if patient is not booked ➔ remove successfully 3. valid appointment ID→ ➔ if patient is booked ➔ add patient to next available appointment. ➔ Remove appointment. 4. Add appointment to another server →not allowed
4.		listAppointmentAvailibilty()	1. appointment type→authenticate appointment type 2. List appointment for particular appointment type from all the server. 3. Inter-server communication required.
Patient Operations			
5.		BookAppointment()	1. appointment type→authenticate appointment type 2. appointment ID→authenticate appointment ID 3. Book on own server→ ➔ Allowed ➔ Cannot book same appointment type in a day. 4. Book on other server ➔ Allowed(Max 3 in a week) 5. Book same appointment ➔ not Allowed
6.		getBookingSchedule()	1. Show booking schedule of customer

7.		CancelAppointment()	1. appointment ID→ →authenticate appointment ID 2. cancel on own server ➔ Allowed 3. Cancel not booked Appointment ➔ Not done successfully 4. cancel on other server→ ➔ Allowed(Inter server communication required)
8.		swapAppointment()	1. Check if oldAppointmentID is booked→ ➔ if booked →procced ➔ if not booked→ return failure ➔ swapping fail. 2. Check if newAppointmentID to be book is available ➔ if valid and available • check same week and weekly limit • cancel old appointment • book new appointment • swapping done. ➔ If invalid and not available • Cannot book new appointment • Swapping cannot done. 3. Booking Success and cancel success ➔ Swapping done successfully. 4. Booking fail and cancel success ➔ Book old appointment id ➔ Swapping fail 5. Booking success and cancel fail ➔ Cancel new appointment id ➔ Swapping fail 6. Booking fail and success fail ➔ Cannot do any thing in this case ➔ Swapping fail.