Distributed Appointment Management System (DAMS) Project

Design Document

COMP 6231 – Distributed System Design

Winter 2022

Professor: Rajagopalan Jayakumar

Teaching Assistant: Stallone Mecwan

Group 7
40190808 Mohammed Misbah Uddin Shareef
40196507 Sahran Khuwaja
40125496 Yuguo Zheng
40216098 Yvonne Lee Chooi Mei

Table of Contents

Introduction

Distributed Appointment Management System (DAMS) is a distributed system implemented using *Web Services* technology, specifically *JAX-WS (Java API for XML Web Services).* There are 3 different servers for each city, i.e. Montreal (MTL), Quebec (QUE), and Sherbrooke (SHE). This system is used by Admins and Patient, thus, there are two Clients i.e., *AdminClient* and *PatientClient.*

System Design

Web Services technology is used between the Clients and Frontend so that they can access servers by only using names, without knowing their location. In that way, this distributed system achieves location transparency. Client and Frontend communicate with each other using *SOAP (Simple Object Access Protocol)* messages. Client uses *WSDL (Web Service Description Language)* document to search for the service and its methods, parameters and URI.

The Frontend will then send the request to the Sequencer and the Sequencer will forward the message to the Replica Manager with a sequence ID. It is implemented using totally ordered reliable multicast to ensure that all Replica Managers perform the operations in the same order. The Replica Managers will then process each request and send it to the corresponding servers. Once the servers execute the request, it will directly reply to the Front end and the Front end will send the request to the requesting Client.

Inter Process Communication (IPC) is used in this system via *UDP/IP* protocol for the communication between Frontend and the Sequencer, Sequencer and the Replica Manager, Replica Manager and the Replicas, and inter-server communication.

Active Replication for Fault Tolerance and High Availability

1. Request: Front end attaches a unique ID and uses totally ordered reliable multicast to send the request to Replica Managers
2. Coordination: The multicast delivers requests to all the Replica Mangers in the same order.
3. Execution: Every Replica Manager executes the request. They are state machines and receive requests in the same order and attaches an ID in the response.
4. Agreement: No agreement is required because all Replica Managers execute the operations in the same order, using totally ordered multicast.
5. Response: Front ends collect responses from Replica Managers and compares the responses to tolerate software failure. To tolerate crash failure, it will reply to the Client with the first response.

Reliable Multicast

On initialization
Received := {};

For process p to R-multicast message m to group g
B-multicast(*g*, *m*); // $p \in g$ is included as a destination

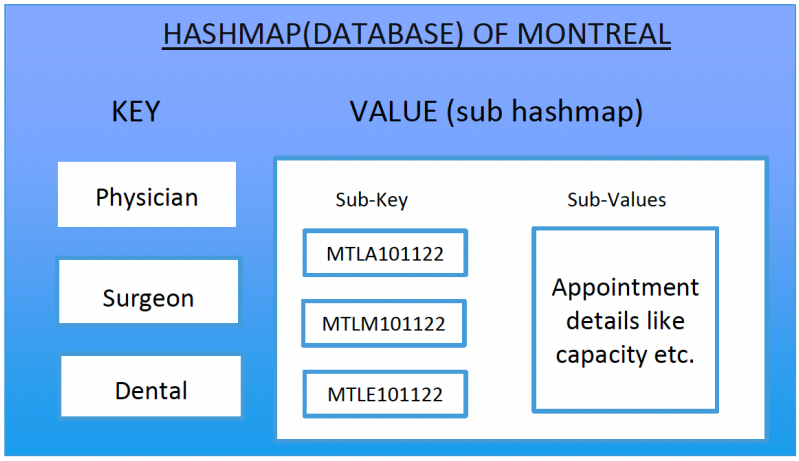
On B-deliver(*m*) *at process q with g = group*(*m*)
if ($m \notin$ *Received*)
then
Received := *Received* $\cup$ {*m*};
if ($q \neq p$) *then B-multicast*(*g*, *m*); *end if*
R-deliver m;
end if

Total Order

To multicast messages, totally ordered identifiers are attached to each message. Each receiving process makes ordering decisions based on the identifiers it receives, using the FIFO algorithm.

Data Structures

1. HashMap

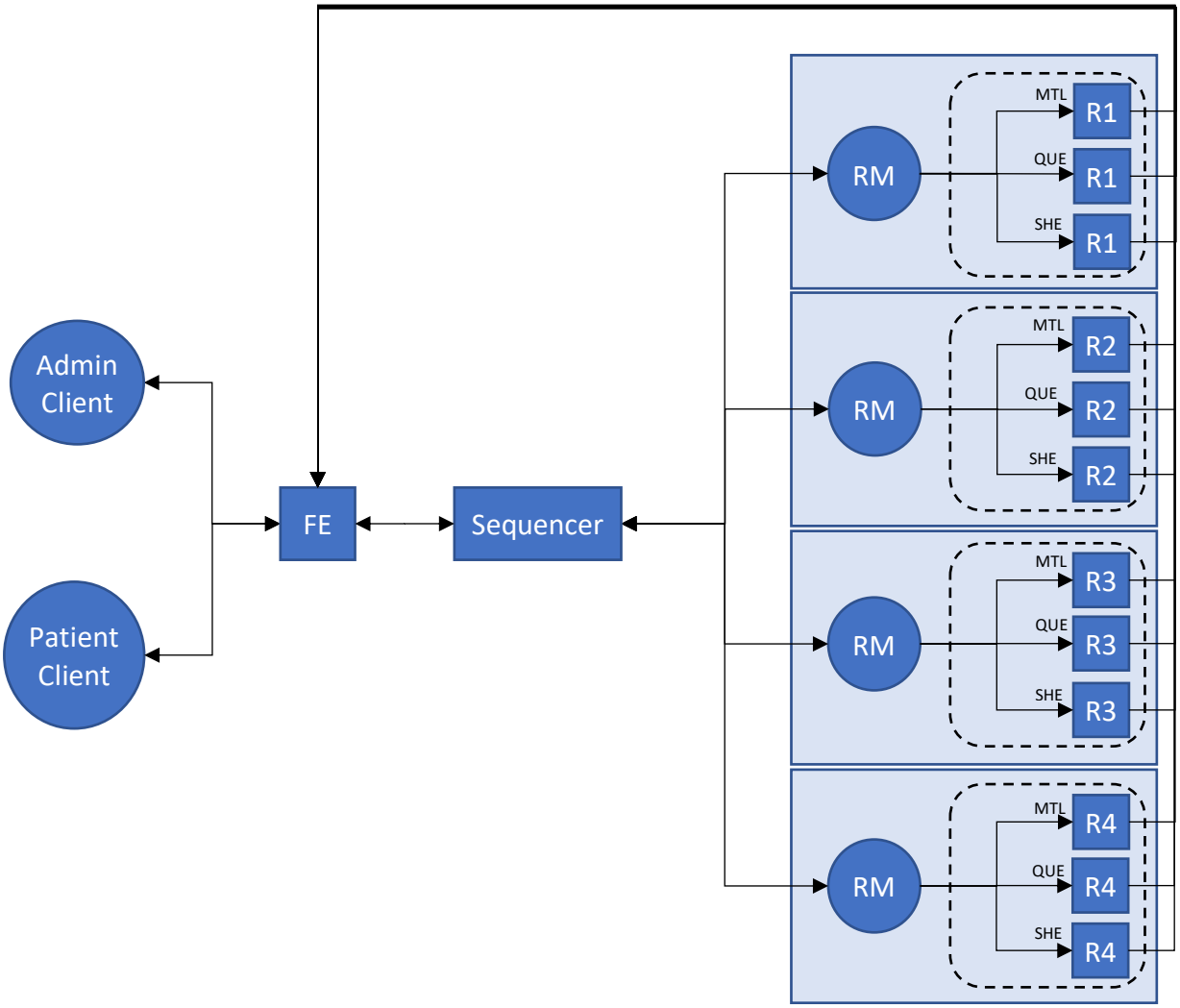


HashMaps are used to store appointments and users so that the values can be looked up efficiently as the appointments and users are often being accessed in the system.

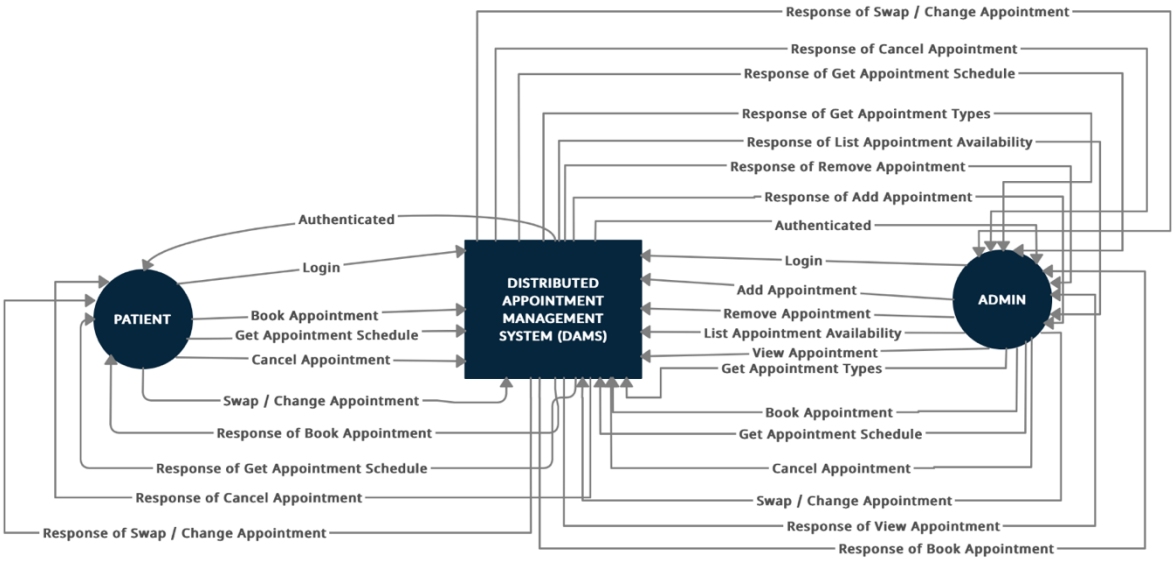
2. ArrayList
ArrayLists are used in the system to store the list of user's appointments

Architecture diagram

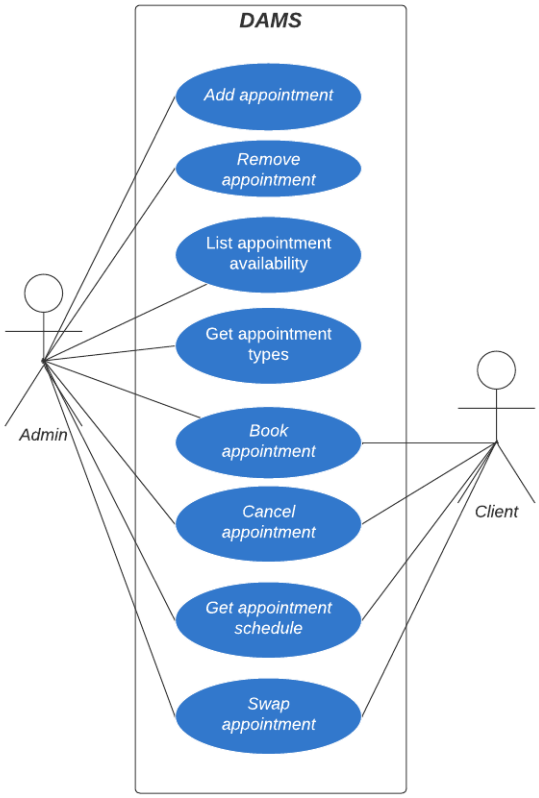
Active replication model for fault tolerance and high availability



System Context / Block Diagram



Use-Case Diagram



Implementation Details

a. Frontend
   i. Login
   ii. Parse client request
   iii. Forward request to sequencer
   iv. Compare replica results
   v. Respond to client
b. Replica Manager
   i. Send request to sequencer
   ii. Detect failure replica
   iii. Recover replica
   iv. Replace replica
   v. Check replica health
c. Sequencer
   i. Multicast request
   ii. Generate sequence number
d. Replicas
   i. Add appointment
   ii. Remove appointment
   iii. List appointment availability
   iv. Book appointment
   v. Get appointment schedule
   vi. Cancel appointment
   vii. Swap appointment

Test Cases

Test scenarios for fault tolerance and high availability

1. Login to Admin Client as MTLA2046
2. Get appointment types has a total count of 3.
3. List appointment availability and verify that number of dental appointments has a total count of 9.
4. Add a new Dental appointment MTLE060422 for the current city with a capacity of 2 is successful.
5. List appointment availability and verify that number of dental appointments has a total count of 10.
6. Add a new Dental appointment SHEE060422 for another city with a capacity of 2 fails as Admin is unable to add appointment for another city.
7. Manually stop one server to simulate **fault tolerance**. The replica manager will detect that the server has crashed and restart it.
8. Get appointment schedule for patient MTL2046 and verify that the number of appointments has a total count of 0.
9. Book Dental appointment MTLE100222 for patient MTL2046 is successful.
10. Get appointment schedule for patient MTL2046 and verify that the number of appointments has a total count of 1.
11. Book Dental appointment MTLE100222 for patient MTL2046 fails as Admin is unable to book the same appointment.
12. Manually restart one server which will send an incorrect response on purpose to simulate **software tolerance**.
13. Add a new Dental appointment MTLE060422 for the current city with a capacity of 2 is successful.

Test scenarios for business logic

1. Test cases for 'Add Appointment'

Test case # 1

Test scenario: Add new appointment to the system on the current server by current city server admin

Test case: Select Appointment Type and enter Appointment ID, Capacity, and Description

Pre-condition: User must be logged in and should be admin to perform this operation

Test steps:

- Login with Admin ID
- Select add appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID, Capacity, and Description

Test Data: Appointment Type: 'Surgeon', Appointment ID: MTLM020222, Capacity: 5, Description: New Appointment

Expected Results: Appointment will be added successfully to the current server

Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated

Actual Results: Appointment added successfully to the current server
Status: Pass

Test case # 2
Test scenario: Add new appointment to the system on the current server by current city server admin by foreign server Appointment ID
Test case: Select Appointment Type and enter Appointment ID, Capacity, and Description
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select add appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID, Capacity, and Description

Test Data: Appointment Type: 'Surgeon', Appointment ID: QUEM020222, Capacity: 5, Description: New Appointment (Montreal Admin using Quebec Appointment ID)
Expected Results: Error message 'Appointment ID contains invalid city code, which doesn't match current city server!'
Post condition:

- Error message will be displayed to the user
- Client and Server logs will be updated

Actual Results: Error message 'Appointment ID contains invalid city code, which doesn't match current city server!'
Status: Pass

Test case # 3
Test scenario: Add new appointment to the system on the current server by current city server admin with capacity zero
Test case: Select Appointment Type and enter Appointment ID, Capacity, and Description
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select add appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID, Capacity, and Description

Test Data: Appointment Type: ‘Surgeon’, Appointment ID: MTLM020222, Capacity: 0, Description: New Appointment
Expected Results: Error message ‘Capacity can’t be zero!’
Post condition:

- Error message will be shown to the user
- Client and Server logs will be updated

Actual Results: Error message ‘Capacity can’t be zero!’
Status: Pass

Test case # 4
Test scenario: Add new appointment to the system on the current server by current city server admin with Appointment ID containing invalid time slot
Test case: Select Appointment Type and enter Appointment ID, Capacity, and Description
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select add appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID, Capacity, and Description

Test Data: Appointment Type: ‘Surgeon’, Appointment ID: MTLP020222, Capacity: 2, Description: New Appointment
Expected Results: Error message ‘Appointment ID contains invalid time slot!’
Post condition:

- Error message will be shown to the user
- Client and Server logs will be updated

Actual Results: Error message ‘Appointment ID contains invalid time slot!’
Status: Pass

Test case # 5
Test scenario: Add new appointment to the system on the current server by current city server admin with Appointment ID containing invalid date
Test case: Select Appointment Type and enter Appointment ID, Capacity, and Description
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select add appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID, Capacity, and Description

Test Data: Appointment Type: ‘Surgeon’, Appointment ID: MTLP020KL, Capacity: 2, Description: New Appointment
Expected Results: Error message ‘Appointment ID should contain proper date in numeric format (DDMMYY)!”
Post condition:

- Error message will be shown to the user
- Client and Server logs will be updated

Actual Results: Error message ‘Appointment ID should contain proper date in numeric format (DDMMYY)!”
Status: Pass

Test case # 6
Test scenario: Add new appointment to the system on the current server by current city server admin with invalid Appointment ID
Test case: Select Appointment Type and enter Appointment ID to delete
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select remove appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Appointment Type: ‘Surgeon’, Appointment ID: MTLP02, Capacity: 2, Description: New Appointment
Expected Results: Error message ‘Appointment ID should not be less/greater than 10!”
Post condition:

- Error message will be shown to the user
- Client and Server logs will be updated

Actual Results: Error message ‘Appointment ID should not be less/greater than 10!”
Status: Pass

Test case # 7
Test scenario: Add new appointment to the system on the current server by current city server admin using Appointment ID which already exists
Test case: Select Appointment Type and enter Appointment ID, Capacity, and Description
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select add appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID, Capacity, and Description

Test Data: Appointment Type: ‘Surgeon’, Appointment ID: MTLP020222, Capacity: 2, Description: New Appointment
Expected Results: Error message ‘Appointment ID already exists! Appointment ID should be unique!”
Post condition:

- Error message will be shown to the user
- Client and Server logs will be updated

Actual Results: Error message ‘Appointment ID already exists! Appointment ID should be unique!”
Status: Pass

2. Test cases for ‘Remove Appointment’

Test case # 8
Test scenario: Remove an appointment from the system on the current server by current city server admin
Test case: Select Appointment Type and enter Appointment ID
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select remove appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Appointment Type: ’Surgeon’, Appointment ID: MTLM020222
Expected Results: Appointment will be removed successfully from the current server
Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated

Actual Results: Appointment has been removed successfully from the current server
Status: Pass

Test case # 9
Test scenario: Remove an appointment from the system on the current server by current city server admin by foreign server Appointment ID
Test case: Select Appointment Type and enter Appointment ID
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select remove appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Appointment Type: 'Surgeon', Appointment ID: QUEM020222 (Montreal Admin using Quebec Appointment ID)
Expected Results: Error message: 'Appointment ID contains invalid city code, which doesn't match current city server!'
Post condition:

- Error message will be displayed to the user
- Client and Server logs will be updated

Actual Results: Error message: 'Appointment ID contains invalid city code, which doesn't match current city server!'
Status: Pass

Test case # 10
Test scenario: Remove an appointment from the system on the current server by current city server admin using Appointment ID in the specific Appointment Type which doesn't exist!
Test case: Select Appointment Type and enter Appointment ID, Capacity, and Description
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select remove appointment option from menu
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Appointment Type: 'Surgeon', Appointment ID: MTLM020223
Expected Results: Error message: 'Appointment ID doesn't exist!'
Post condition:

- Error message will be displayed to the user
- Client and Server logs will be updated

Actual Results: Error message: 'Appointment ID doesn't exist!'
Status: Pass

3. Test cases for 'List Appointment Availability'

Test case # 11
Test scenario: Retrieve and list all appointment availability from all the servers for the specific Appointment Type
Test case: Select Appointment Type
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select list appointment availability option from menu
- Select Appointment Type

Test Data: Appointment Type: ‘Surgeon’
Expected Results: Appointment availability from all the servers will be displayed to the user
Post condition:

- Data will be displayed to the user and a set of operations will be displayed to the user.
- Client and Server logs will be updated

Actual Results: Appointment availability from all the servers have been displayed to the user
Status: Pass

4. Test cases for ‘View Appointment’

Test case # 12
Test scenario: Retrieve and list all appointment availability from the current server for the specific Appointment Type
Test case: Select Appointment Type
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select view appointments option from menu
- Select Appointment Type

Test Data: Appointment Type: ‘Surgeon’
Expected Results: Appointment availability from the current server will be displayed to the user
Post condition:

- Data will be displayed to the user and a set of operations will be displayed to the user.
- Client and Server logs will be updated

Actual Results: Appointment availability from the current server has been displayed to the user
Status: Pass

5. Test cases for ‘Get Appointment Types’

Test case # 12
Test scenario: Retrieve and list all appointment types
Test case: Select Appointment types
Pre-condition: User must be logged in and should be admin to perform this operation
Test steps:

- Login with Admin ID
- Select view appointments types option from menu

Test Data: none

Expected Results: Appointment types will be displayed to the user
Post condition:

- Data will be displayed to the user and a set of operations will be displayed to the user.
- Client and Server logs will be updated

Actual Results: Appointment types have been displayed to the user
Status: Pass

6. Test cases for 'Book Appointment'

Test case # 13
Test scenario: Book an appointment on the current server by current city server patient / admin
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, Select Appointment Type and enter Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select book appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Patient ID: 'MTLP2046', Appointment Type: 'Surgeon', Appointment ID: MTLM020222
Expected Results: Appointment will be booked successfully in the current server
Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (decreased by one)

Actual Results: Appointment has been booked successfully on the current server
Status: Pass

Test case # 14
Test scenario: Book an appointment on the foreign server by current city server patient / admin
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, Select Appointment Type and enter Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID

- Select book appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Patient ID: ‘MTLP2046’, Appointment Type: ‘Surgeon’, Appointment ID: SHEM020222
Expected Results: Appointment will be booked successfully on the foreign server
Post condition:

- Foreign server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (decreased by one)
- Foreign booking count of the current city server user will be updated by current city server

Actual Results: Appointment has been booked successfully on the foreign server
Status: Pass

Test case # 15
Test scenario: Book an appointment on the foreign server by current city server patient / admin more than 3 times by patient / admin
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, Select Appointment Type and enter Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select book appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Patient ID: ‘MTLP2046’, Appointment Type: ‘Surgeon’, Appointment ID: SHEM050222
Expected Results: Error message ‘Can't book foreign servers' slots more than 3 times with the same Patient ID’
Post condition:

- Error message will be displayed
- Client and Server logs will be updated

Actual Results: Error message ‘Can't book foreign servers' slots more than 3 times with the same Patient ID’
Status: Pass

Test case # 16
Test scenario: Book an appointment where no capacity left for the specific Appointment ID by patient / admin
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, Select Appointment Type and enter Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select book appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Patient ID: 'MTLP2046', Appointment Type: 'Surgeon', Appointment ID: MTLM020222
Expected Results: Error message 'No empty slots available for Appointment ID for the specific Appointment Type'
Post condition:

- Error message will be displayed
- Client and Server logs will be updated

Actual Results: Error message 'No empty slots available for Appointment ID for the specific Appointment Type'
Status: Pass

Test case # 17
Test scenario: Book an appointment with same Appointment ID in the specific Appointment Type by patient / admin, which has been already booked by him/her
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, Select Appointment Type and enter Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select book appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select Appointment Type
- Provide valid Appointment ID

Test Data: Patient ID: 'MTLP2046', Appointment Type: 'Surgeon', Appointment ID: MTLM020222

Expected Results: Error message ‘Appointment ID in the specific Appointment Type has already booked! Can’t book twice with same Appointment ID in the same Appointment Type!’
Post condition:

- Error message will be displayed
- Client and Server logs will be updated

Actual Results: Error message ‘Appointment ID in the specific Appointment Type has already booked! Can’t book twice with same Appointment ID in the same Appointment Type!’
Status: Pass

7. Test cases for ‘Get Appointment Schedule’

Test case # 18
Test scenario: Get appointment schedule from all the servers by patient / admin
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select get appointment schedule from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID

Test Data: Patient ID: ‘MTLP2046’
Expected Results: Appointment schedule will be fetched from all the servers and displayed to the user (if present)
Post condition:

- Data will be displayed to the user and a set of operations will be displayed to the user.
- Client and Server logs will be updated

Actual Results: Appointment schedule has been fetched from all the servers and displayed to the user
Status: Pass

Test case # 19
Test scenario: Get appointment schedule from all the servers by patient / admin using Patient ID who hasn’t book any appointments
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select get appointment schedule from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID

Test Data: Patient ID: ‘MTLP2048’
Expected Results: Error message ‘No schedule found!’
Post condition:

- Error message will be displayed
- Client and Server logs will be updated

Actual Results: Error message ‘No schedule found!’
Status: Pass

8. Test cases for ‘Cancel Appointment’

Test case # 20
Test scenario: Cancel an appointment by patient / admin on the current city server
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select Appointment Type, and enter Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select cancel appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select Appointment Type
- Enter Appointment ID

Test Data: Patient ID: ‘MTLP2045’, Appointment Type: ‘Surgeon’, Appointment ID: ’MTLM020222’
Expected Results: Appointment will be cancelled in the specific Appointment Type
Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (increased by one)

Actual Results: Appointment will be cancelled in the specific Appointment Type
Status: Pass

Test case # 21
Test scenario: Cancel an appointment by patient / admin on the foreign city server
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select Appointment Type, and enter Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select cancel appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically

- If admin is logged in, enter Patient ID
- Select Appointment Type
- Enter Appointment ID

Test Data: Patient ID: ‘MTLP2045’, Appointment Type: ‘Surgeon’, Appointment ID: ’SHEM020222’
Expected Results: Appointment will be cancelled in the specific Appointment Type on the foreign server
Post condition:

- Foreign server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (increased by one)
- Foreign booking count of the current city server user will be updated by current city server

Actual Results: Appointment has been cancelled in the specific Appointment Type on the foreign server
Status: Pass

Test case # 22
Test scenario: Cancel an appointment by patient / admin who either hasn’t booked the appointments or hasn’t booked the appointment with specific Appointment ID of the specific Appointment Type
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select Appointment Type, and enter Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select cancel appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select Appointment Type
- Enter Appointment ID

Test Data: Patient ID: ‘MTLP2045’, Appointment Type: ‘Surgeon’, Appointment ID: ’MTLM020322’
Expected Results: Error message ‘No booking found with this Appointment ID in the specific Appointment Type to cancel!’
Post condition:

- Error message will be displayed to the user
- Client and Server logs will be updated

Actual Results: Error message ‘No booking found with this Appointment ID in the specific Appointment Type to cancel!’

Status: Pass

9. Test cases for 'Swap Appointment'

Test case # 23

Test scenario: Swap / Change an appointment by patient / admin with current city server old and new Appointment ID of the specific old and new Appointment Type respectively

Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select old Appointment Type, and enter old Appointment ID (already booked). Then select the new Appointment Type, and enter new Appointment ID

Pre-condition: User must be logged in to perform this operation

Test steps:

- Login with Patient / Admin ID
- Select swap appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select old Appointment Type
- Enter old Appointment ID
- Select new Appointment Type
- Enter new Appointment ID

Test Data: Patient ID: 'MTLP2045', Old Appointment Type: 'Surgeon', Old Appointment ID: 'MTLM020222', New Appointment Type: 'Physician', New Appointment ID: 'MTLE020222'

Expected Results: Old Appointment ID of the specific old Appointment Type will be swapped with new Appointment ID of the specific new Appointment Type

Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (increased/decreased by one)

Actual Results: Old Appointment ID of the specific old Appointment Type has been swapped with new Appointment ID of the specific new Appointment Type

Status: Pass

Test case # 24

Test scenario: Swap / Change an appointment by patient / admin with current city server old Appointment ID of the specific old Appointment Type and foreign city server new Appointment ID of the specific new Appointment Type

Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select old Appointment Type, and enter old Appointment ID (already booked). Then select the new Appointment Type, and enter new Appointment ID

Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select swap appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select an old Appointment Type
- Enter Appointment ID
- Select new Appointment Type
- Enter new Appointment ID

Test Data: Patient ID: 'MTLP2045', Old Appointment Type: 'Physician', Old Appointment ID: 'MTLE020222', New Appointment Type: 'Dental', New Appointment ID: 'SHEE100222'
Expected Results: Current city server old Appointment ID of the specific old Appointment Type will be swapped with foreign city server new Appointment ID of the specific new Appointment Type
Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (increased/decreased by one)

Actual Results: Current city server old Appointment ID of the specific old Appointment Type has been swapped with foreign city server new Appointment ID of the specific new Appointment Type
Status: Pass

Test case # 25
Test scenario: Swap / Change an appointment by patient / admin with foreign city server city server old Appointment ID of the specific old Appointment Type and current city server new Appointment ID of the specific new Appointment Type
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select old Appointment Type, and enter old Appointment ID (already booked). Then select the new Appointment Type, and enter new Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select swap appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select an old Appointment Type
- Enter Appointment ID
- Select new Appointment Type

- Enter new Appointment ID

Test Data: Patient ID: ‘MTLP2045’, Old Appointment Type: ‘Dental’, Old Appointment ID: ’SHEE100222’, New Appointment Type: ‘Physician’, New Appointment ID: ’MTLM100222’
Expected Results: Foreign city server old Appointment ID of the specific old Appointment Type will be swapped with current city server new Appointment ID of the specific new Appointment Type
Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (increased/decreased by one)

Actual Results: Foreign city server old Appointment ID of the specific old Appointment Type has been swapped with current city server new Appointment ID of the specific new Appointment Type
Status: Pass

Test case # 26
Test scenario: Swap / Change an appointment by patient / admin with foreign city server city server old and new Appointment ID
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select old Appointment Type, and enter old Appointment ID (already booked). Then select the new Appointment Type, and enter new Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select swap appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select an old Appointment Type
- Enter Appointment ID
- Select new Appointment Type
- Enter new Appointment ID

Test Data: Patient ID: ‘MTLP2045’, Old Appointment Type: ‘Dental’, Old Appointment ID: ’QUEE100222’, New Appointment Type: ‘Physician’, New Appointment ID: ’SHEA100222’
Expected Results: Foreign city server old Appointment ID of the specific old Appointment Type will be swapped with another foreign city server new Appointment ID of the specific new Appointment Type
Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.

- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (increased/decreased by one)

Actual Results: Foreign city server old Appointment ID of the specific old Appointment Type has been swapped with another foreign city server new Appointment ID of the specific new Appointment Type
Status: Pass

Test case # 27
Test scenario: Swap / Change an appointment by patient / admin who either hasn't booked the appointments or hasn't booked the appointment with specific old Appointment ID of the specific Appointment Type
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select old Appointment Type, and enter old Appointment ID (already booked). Then select the new Appointment Type, and enter new Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select swap appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select an old Appointment Type
- Enter Appointment ID
- Select new Appointment Type
- Enter new Appointment ID

Test Data: Patient ID: 'MTLP2045', Old Appointment Type: 'Dental', Old Appointment ID: 'QUEA100222', New Appointment Type: 'Physician', New Appointment ID: 'MTLE100222'
Expected Results: Error message 'No booking found with this old Appointment ID in the specific old Appointment Type to swap with new Appointment ID of the specific new Appointment Type!'
Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (increased/decreased by one)

Actual Results: Error message 'No booking found with this old Appointment ID in the specific old Appointment Type to swap with new Appointment ID of the specific new Appointment Type!'
Status: Pass

Test case # 28

Test scenario: Swap / Change an appointment by patient / admin with old and new Appointment ID of the specific old and new Appointment Type respectively
Test case: Provide Patient ID (if admin is performing), otherwise Patient ID will automatically be fetched from current session, select old Appointment Type, and enter old Appointment ID (already booked). Then select the new Appointment Type, and enter new Appointment ID
Pre-condition: User must be logged in to perform this operation
Test steps:

- Login with Patient / Admin ID
- Select swap appointment from main menu
- If patient is logged in, the Patient ID will be stored automatically
- If admin is logged in, enter Patient ID
- Select an old Appointment Type
- Enter Appointment ID
- Select new Appointment Type
- Enter new Appointment ID

Test Data: Patient ID: 'MTLP2045', Old Appointment Type: 'Dental', Old Appointment ID: 'QUEA100222', New Appointment Type: 'Physician', New Appointment ID: 'MTLE100222'
Expected Results: Error message 'No booking found with this old Appointment ID in the specific old Appointment Type to swap with new Appointment ID of the specific new Appointment Type!'
Post condition:

- Server success response will be displayed to the user and a set of operations will be displayed to the user.
- Database will be updated
- Client and Server logs will be updated
- Capacity of the slot will be updated (increased/decreased by one)

Actual Results: Error message 'No booking found with this old Appointment ID in the specific old Appointment Type to swap with new Appointment ID of the specific new Appointment Type!'
Status: Pass

10. Test cases for 'Concurrency'

Test case # 29
Test scenario: Concurrent access of shared data
Test case: Two or more clients performing operations on the shared data (concurrency)
Pre-condition: Users must be logged in at the same time to perform this operation
Test steps:

- Login with Patient / Admin IDs
- Select operations from main menu
- Provide required data

Test Data: Depend on the operation
Expected Results: Users will be able to access and update shared data concurrently with proper lock and edit mechanisms. Users will also be able to access the system concurrently and get replies from servers concurrently
Post condition:

- Server success response of specific operation will be displayed to the user

Actual Results: Users have been able to access and update shared data concurrently with proper lock and edit mechanisms. Users have also been able to access the system concurrently and get replies from servers concurrently
Status: Pass

Task Distribution

The project is a collaboration between all group members. The following are the individual tasks for each member:

- Sequencer - 40190808 Mohammed Misbah Uddin Shareef
- Front end - 40196507 Sahran Khuwaja
- Replica Manager - 40125496 Yuguo Zheng
- Test cases - 40216098 Yvonne Lee Chooi Mei