Assignment - 1

Student ID : 40230634
Name : Mayurkumar Kanbhai Jodhani
Subject : COMP 6231 Distributed Systems Design

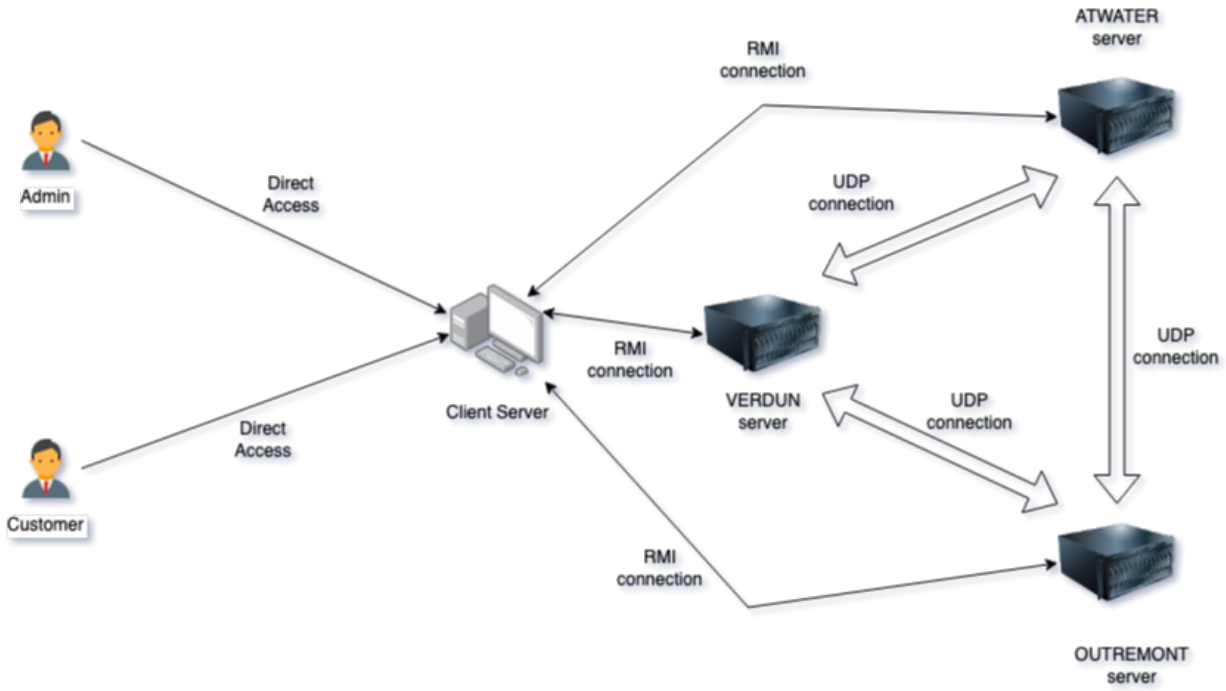
Overview

This system is for managing movie ticket bookings for three theaters in different areas: Atwater (ATW), Verdun (VER), and Outremont (OUT). There are two types of users: Admins and Customers, both identified by a unique ID. Admins are responsible for creating movie slots with movie type and booking capacity, and customers can book movie tickets if the movie is available. The system identifies the server to which the user belongs by looking at the prefix of their ID. Each user has a log of their actions on the system.

Each server will be running on different hosts.On the other side, client will connect with the associated server directly through RMI service while server will be able to communicate with other servers through UDP message system to provide local information about the tickets and to perform operations such as canceling a ticket or booking one.

To provide efficiency and scalability, the threading is used to execute requests while having UDP connection, whereas RMI provides a default service to handle multi-threading.

Architecture



Here, different users such as admin and customer can directly access the client server to perform different operation such as

- Add movie slot
- Remove movie slot
- List movie slots availability
- Book a movie ticket
- Cancel movie tickets
- Get schedule of customer

The client server can connect to three different servers using RMI (Remote Method Invocation) service provided by the associated server.If user tries to perform any actions with the associated server, only RMI connection is required to fulfilled the need.While fetching movie slot availability and getting schedule for a customer will require inter UDP connection.

UDP connection is required if the user tries to perform the action other than the associated server such as a customer of Atwater tries to book a movie ticket for Verdun.

Implementation

Input.java

This class will provide a way to handle all the validation such as valid movie ID, user ID and the exceptions such as InputMismatchException.This will provide different functions such as getString, getIntegerInRange, getMovieID, getUserID, etc.

Query.java

This class will handle all the UDP requests and execute them in a different thread to increase server efficiency. After receiving UDP requests, it will decipher the code, and execute it and send back the response with the same packet address.

Admin.java

This interface will provide a RMI object definition for admin actions to be used by the client side. This interface will be implemented by the AdminImpl to define the logic behind all the actions.

Customer.java

This interface will provide a RMI object definition for the customer to be used by the client side. This interface will be implemented by the CustomerImpl to define the logic behind all the actions.

Logger.java

This class will provide a method to get an instance of Logger by implementing factory design pattern.It will simply add the file handler to be used based on the provided argument.

Commands.java

This class will provide action codes for all the UDP requests to be sent between different servers.While executing the request and generating the response, this class will be used.

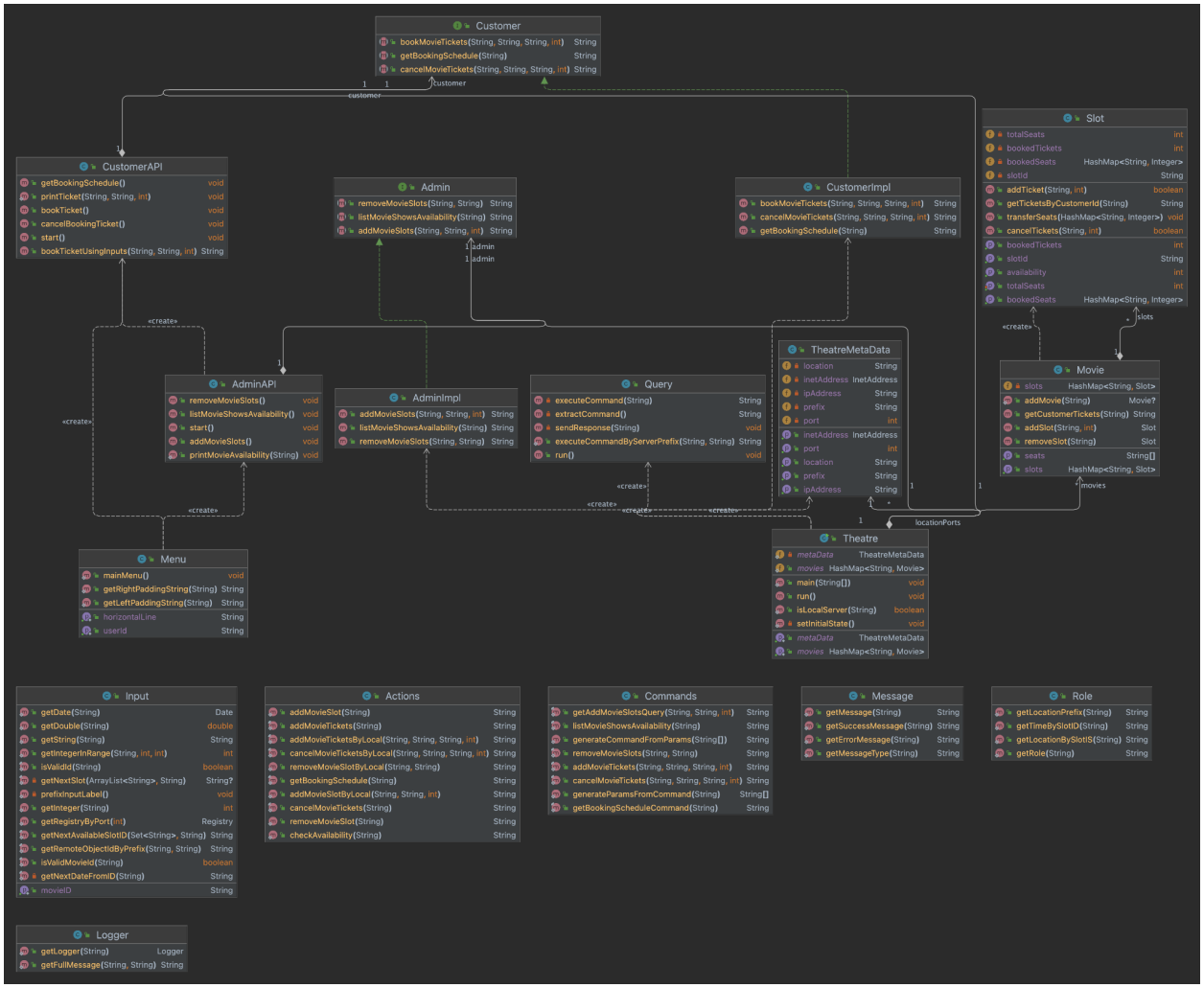
AdminImpl.java

This class is implemented from the Admin interface which has all the logic of the methods defined in the Admin interface.

CustomerImpl.java

This class is implemented from the Customer interface which has all the logic of the methods defined in the Admin interface.

Class Diagram



Test Cases

For all the testcases, the assumption is the userID is valid for either customer or admin, also if either customerID, adminID or movieID is not valid, it will give an error message to the user that the given ID is not valid.

Admin Use Cases

Base Scenario	Specification	Result
Add movie slot	slot does not exist	**will add a new** slot with the given capacity for particular movie at given location
	slot already exist by slotID	**will upgrade** the existing slot with the given capacity for particular movie at given location
cancel movie slot	booked seats are available for the next movie slot to transfer	**will remove** the identified slot and all the tickets for the same slot will be transferred to the next possible slot available at the location.
	booked seats are not available for the next movie slot to transfer	**gives an error** message that the next slot is not available to transfer all the booked seats for a given slot.
List movie shows availability	-	Show all the available seats for a given movie name.

Customer Use Cases

Base Scenario	Specification	Result
Book movie tickets	seats are **available** to be booked for a movie slot.	book provided number of tickets and store information at associated server
	seats are **not available** to be booked for a movie slot.	gives an error message that the seats are not available at the moment.
	the customer tries to book a ticket at a **different server** than the server associated with the customer.	the ticket will be booked by a different server than the server associated with the customer using the inter UDP connection.
	The customer has already booked movie tickets more than three times, and tried to book another ticket at a different location than the associated one.	the error message will be sent that the customer has already booked the limited number of tickets at a remote location.
Get booking schedule	the movie name is **valid**	it will give all the available seat counts for all the slots in different locations.
	the movie name is **not valid**	it will give an error message that the movie does not exist for a given name.
Cancel movie tickets	customer tries to cancel the number of tickets **less than or equal** to the booked tickets by him/her.	It will update the booked ticket count associated with the customer and also remote the entry at a given location.
	customer tries to cancel the number of tickets **more than** the booked tickets	It will give an error message that the given number of tickets not booked by the associated customer.

Summary

Here the most important element of the system is the UDP connection between different servers to provide architecture of distributed systems. While RMI connection is easy to setup between client machine and server machine, the most difficult task in my perspective is how to handle all the data to be store within the server and shared across the network.In the given assignment I have implemented the nested hashmap while using encapsulation to store slot details such as booked tickets and associated booking capacity.