COMP 6231- Distributed System Design

Instructor: R. Jayakumar

Web Service Implementation of the Distributed Movie Ticket Booking System (DMTBS)

- Assignment 3

Winter 2023

Submitted by: Antas Jain

ID: 40233532

CONTENTS

1. Overview

Distributed Movie Ticket Booking Systems have 2 types of users:

- Admin - Actions:
  - Create Movie Slots.
  - Update Movie Slots.
  - Delete Movie Slots.
  - Do Actions on behalf of a customer.
  - Get a list of movie slots for a particular movie.
- Customer - Actions:
  - Book a movie Slot.
  - Cancel a movie slot (Completely/Partially).
  - Get their booking details.
  - Exchange Movie Tickets.

There are 3 servers for movie theatres:

- ATW- Atwater
- VER- Verdun
- OUT- Outremont

There are 3 timing slots for a movie:

- A- Afternoon
- M- Morning
- E- Evening

A Client ID Consist of SERVER ID+A(for Admin)/C(for Customer)+4 unique digits.

A movie ID Consists of SERVER ID+Timing Slot(A/M/E)+Date (in DDMMYY format).

The task was to create a Distributed System for movie booking using Java Webservice

- Implementations:

  WebInterface.java: Interface for all client operations, used as a web service endpoint.

  MovieModel.java: POJO for Movie Servers (getters, setters and other methods).

  ClienModel.java: POJO for Client Servers (getters, setters and other methods)..

  Server.java: Creating Server instances.

  Client.java: Interacting with User, displaying all options for a client, sending details that forward a UDP request to the server for actions.

  ServerInst.java: Interface for Server Actions.

MovieManager.java: Middleware between Server and Client, sends UDP requests, implements endpoint interface..

Logger.java: Logs actions to a text file.

Data is stored in nested map structures as shown in (3).

- Client-Server Interaction using SOAP-based WebService

  ATWATER Server Address: http://localhost:8080/atwater?wsdl
  VERDUN Server Address: http://localhost:8080/verdun?wsdl
  OUTREMONT Server Address: http://localhost:8080/outremont?wsdl
  @WebService(endpointInterface = "com.web.service.WebInterface")

- UDP Server Ports Used:

  ATWATER: 7878
  VERDUN: 8989
  OUTREMONT: 9090

- Logs: Logs are saved for every client and every server.

  Path for SERVER Logs: /src/Logs/Server/ServerName.txt
  Path for CLIENT Logs: /src/Logs/Client/ClientId.txt

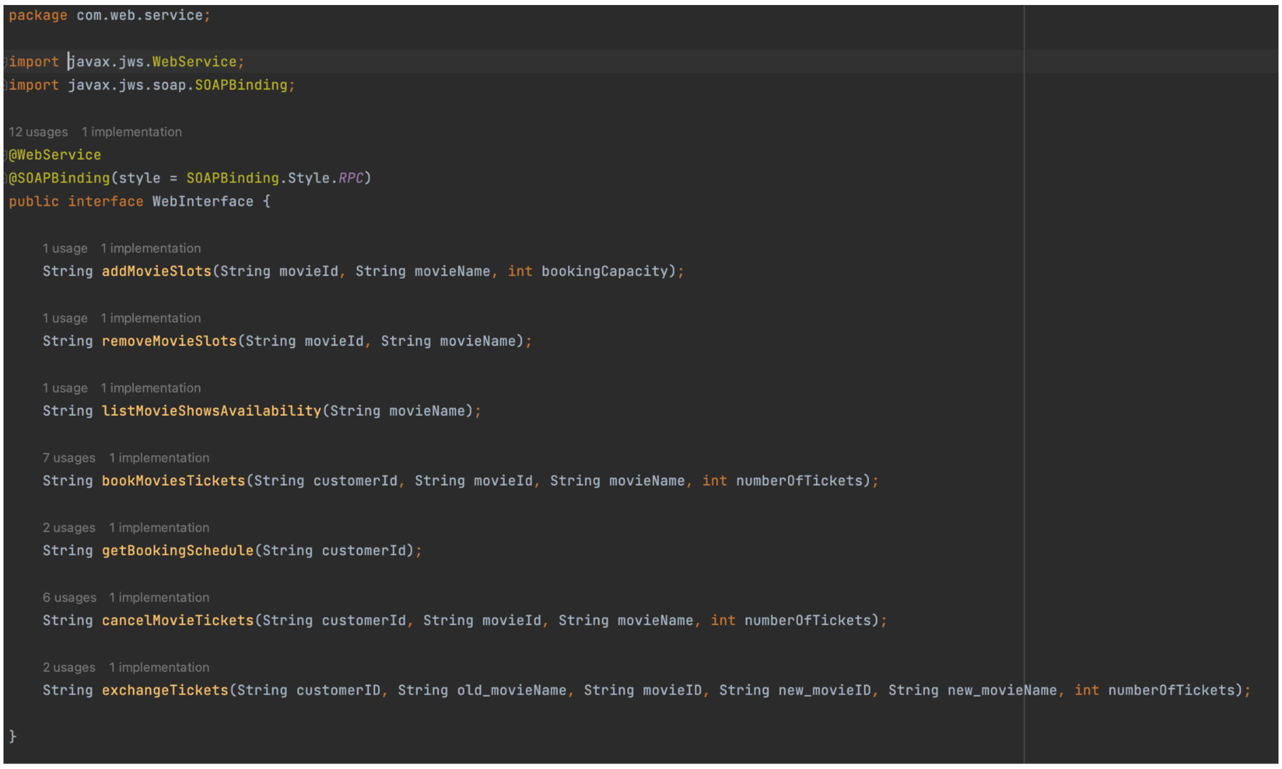
- Concurrent hashmaps were used to ensure maximum efficiency.

- To Generate WSDL Files:
  ```
   wsgen -cp . -keep -wsdl -d Resources  com.web.service.Implementation.MovieManager
  ```

- To Import WSDL Files:
  ```
  wsimport -keep -verbose src/Resources/MovieManagerService.wsdl
  ```

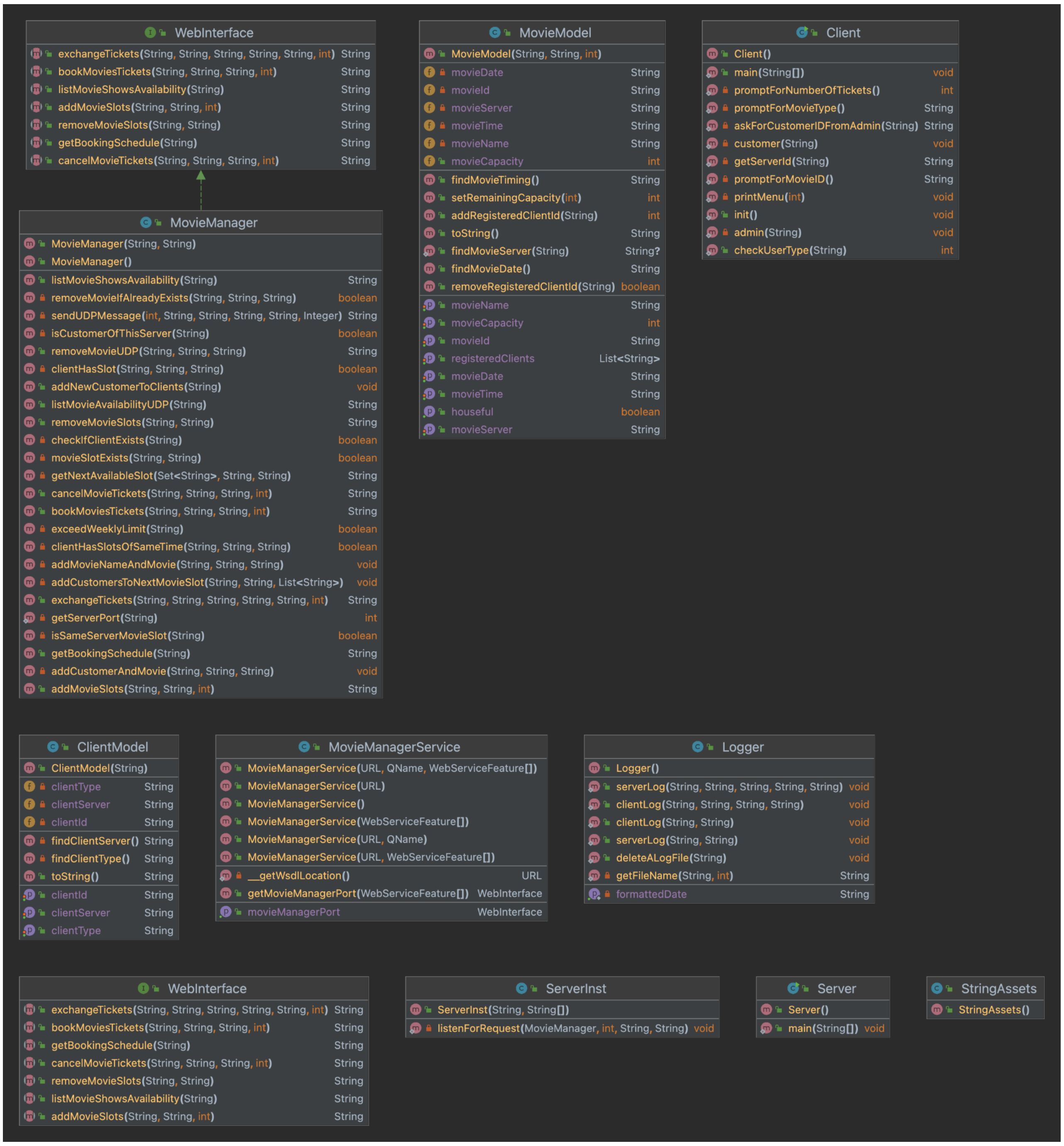
- Hardest Part of the assignment was to implement Book Movies and Cancel movies while keeping track of the number of movie tickets the user needed to book, for that I used a Concurrent hashmap with a unique id that consists of customerId+movieid+movieName. This was also the most important part according to my understanding.

2. Web Service

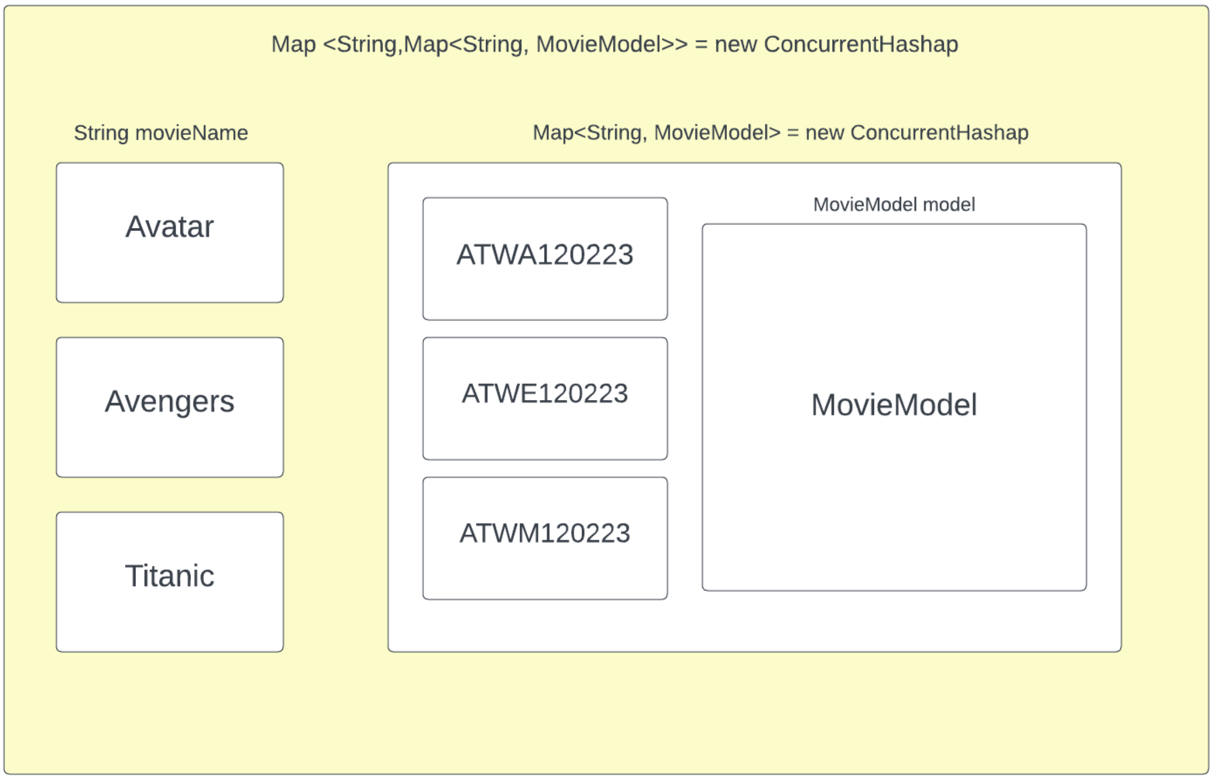


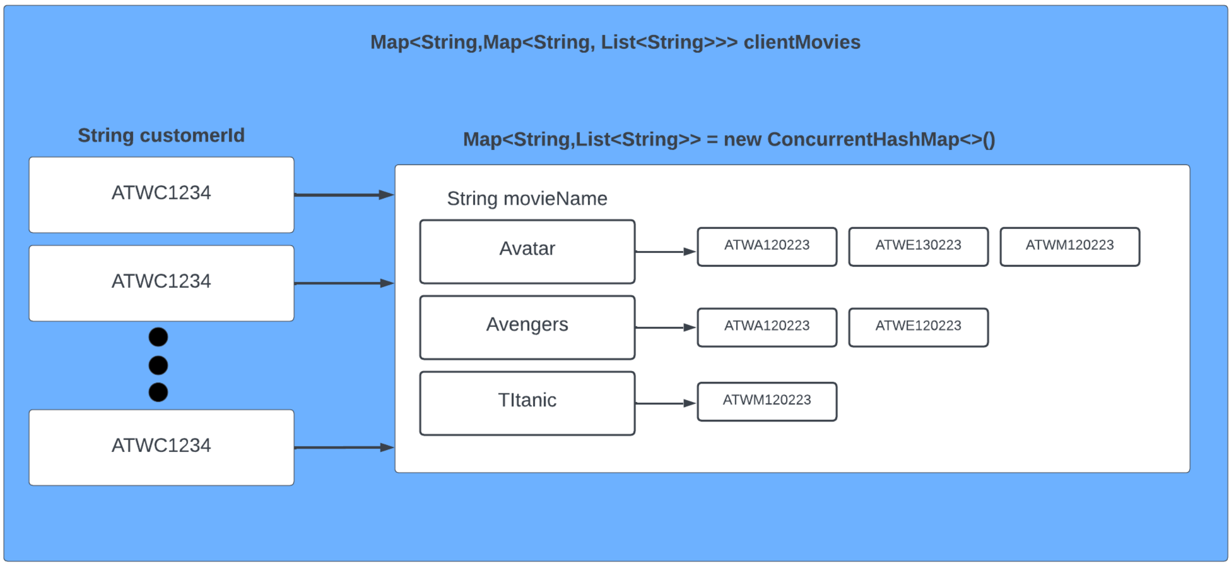
```
package com.web.service;

import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;

12 usages   1 implementation
@WebService
@SOAPBinding(style = SOAPBinding.Style.RPC)
public interface WebInterface {

    1 usage   1 implementation
    String addMovieSlots(String movieId, String movieName, int bookingCapacity);

    1 usage   1 implementation
    String removeMovieSlots(String movieId, String movieName);

    1 usage   1 implementation
    String listMovieShowsAvailability(String movieName);

    7 usages   1 implementation
    String bookMoviesTickets(String customerId, String movieId, String movieName, int numberOfTickets);

    2 usages   1 implementation
    String getBookingSchedule(String customerId);

    6 usages   1 implementation
    String cancelMovieTickets(String customerId, String movieId, String movieName, int numberOfTickets);

    2 usages   1 implementation
    String exchangeTickets(String customerID, String old_movieName, String movieID, String new_movieID, String new_movieName, int numberOfTickets);

}
```

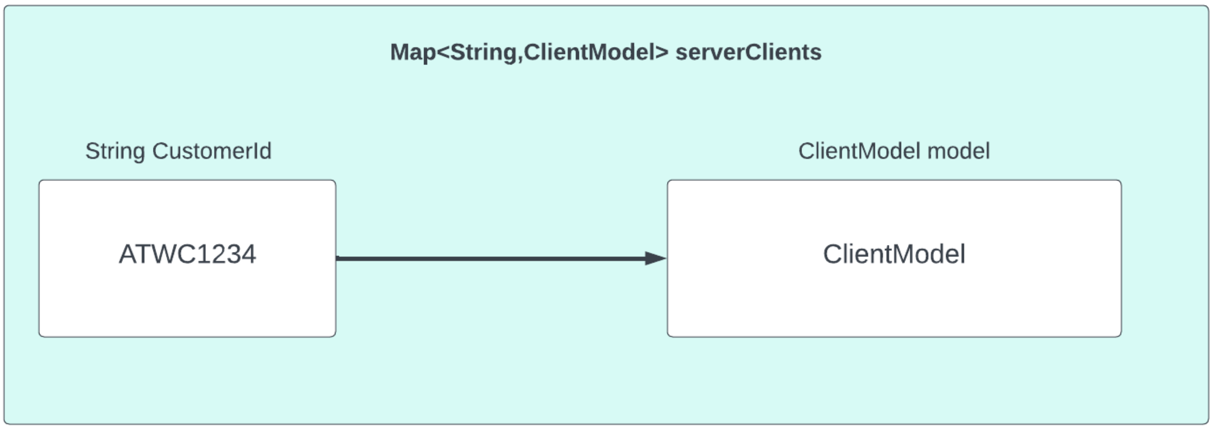
3. Class Diagram

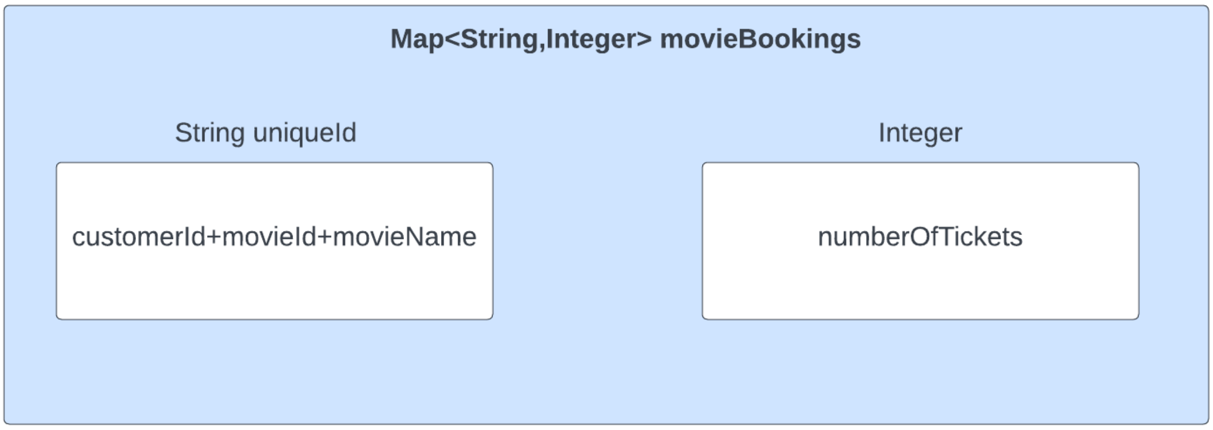

WebInterface
exchangeTickets(String, String, String, String, String, int) String
bookMoviesTickets(String, String, String, int) String
listMovieShowsAvailability(String) String
addMovieSlots(String, String, int) String
removeMovieSlots(String, String) String
getBookingSchedule(String) String
cancelMovieTickets(String, String, String, int) String
MovieModel
MovieModel(String, String, int)
movieDate String
movieId String
movieServer String
movieTime String
movieName String
movieCapacity int
findMovieTiming() String
setRemainingCapacity(int) int
addRegisteredClientId(String) int
toString() String
findMovieServer(String) String?
findMovieDate() String
removeRegisteredClientId(String) boolean
movieName String
movieCapacity int
movieId String
registeredClients List<String>
movieDate String
movieTime String
houseful boolean
movieServer String
Client
Client()
main(String[]) void
promptForNumberOfTickets() int
promptForMovieType() String
askForCustomerIDFromAdmin(String) String
customer(String) void
getServerId(String) String
promptForMovieID() String
printMenu(int) void
init() void
admin(String) void
checkUserType(String) int
MovieManager
MovieManager(String, String)
MovieManager()
listMovieShowsAvailability(String) String
removeMovieIfAlreadyExists(String, String, String) boolean
sendUDPMessage(int, String, String, String, String, Integer) String
isCustomerOfThisServer(String) boolean
removeMovieUDP(String, String, String) String
clientHasSlot(String, String, String) boolean
addNewCustomerToClients(String) void
listMovieAvailabilityUDP(String) String
removeMovieSlots(String, String) String
checkIfClientExists(String) boolean
movieSlotExists(String, String) boolean
getNextAvailableSlot(Set<String>, String, String) String
cancelMovieTickets(String, String, String, int) String
bookMoviesTickets(String, String, String, int) String
exceedWeeklyLimit(String) boolean
clientHasSlotsOfSameTime(String, String, String) boolean
addMovieNameAndMovie(String, String, String) void
addCustomersToNextMovieSlot(String, String, List<String>) void
exchangeTickets(String, String, String, String, String, int) String
getServerPort(String) int
isSameServerMovieSlot(String) boolean
getBookingSchedule(String) String
addCustomerAndMovie(String, String, String) void
addMovieSlots(String, String, int) String
ClientModel
ClientModel(String)
clientType String
clientServer String
clientId String
findClientServer() String
findClientType() String
toString() String
clientId String
clientServer String
clientType String
MovieManagerService
MovieManagerService(URL, QName, WebServiceFeature[])
MovieManagerService(URL)
MovieManagerService()
MovieManagerService(WebServiceFeature[])
MovieManagerService(URL, QName)
MovieManagerService(URL, WebServiceFeature[])
__getWsdlLocation() URL
getMovieManagerPort(WebServiceFeature[]) WebInterface
movieManagerPort WebInterface
Logger
Logger()
serverLog(String, String, String, String, String) void
clientLog(String, String, String, String) void
clientLog(String, String) void
serverLog(String, String) void
deleteALogFile(String) void
getFileName(String, int) String
formattedDate String
WebInterface
exchangeTickets(String, String, String, String, String, int) String
bookMoviesTickets(String, String, String, int) String
getBookingSchedule(String) String
cancelMovieTickets(String, String, String, int) String
removeMovieSlots(String, String) String
listMovieShowsAvailability(String) String
addMovieSlots(String, String, int) String
ServerInst
ServerInst(String, String[])
listenForRequest(MovieManager, int, String, String) void
Server
Server()
main(String[]) void
StringAssets
StringAssets()

4. Data Structure


Map <String,Map<String, MovieModel>> = new ConcurrentHashap
String movieName
Map<String, MovieModel> = new ConcurrentHashap
Avatar
Avengers
Titanic
ATWA120223
ATWE120223
ATWM120223
MovieModel model
MovieModel




Map<String,ClientModel> serverClients
String CustomerId
ClientModel model
ATWC1234
ClientModel


Map<String,Integer> movieBookings
String uniqueId
Integer
customerId+movieId+movieName
numberOfTickets

5. Test Cases

1. Verify UserId:

Input	Output	Fail/Pass
ATWC1234	Display Menu	Pass
ATWA1000	Display Menu	Pass
ATWB1211	Invalid UserId	Fail

2. Admin Add Slots:

Input	Output	Fail/Pass
1 100 ATWA100223	Slot added	Pass
1 100 ATWA090223	Enter Valid Date	Fail
1 100 ATWA190223	Enter valid date	Fail

3. Book Tickets

Input	Output	Fail/Pass
1 10 ATWA100223	Success	Pass
1 110 ATWA090223	Number of Tickets exceeds total tickets	Fail
1 10 ABCA190223	Wrong server name.	Fail

4. Get Booking Schedule:

Input	Output	Fail/Pass
ATWC1234	Displays all Bookings	Pass

5. Cancel Movie Tickets:

Input	Output	Fail/Pass
1 10 ATWA120223	Cancelled Success	Pass
1 10 AAAA090223	Enter Valid id	Fail

6. Exchange Tickets:

Input	Output	Fail/Pass
Old MovieSlot: capacity 100 New MovieSlot: capacity 200 Customer trying to exchange 20 tickets to both booked slots.	Success	Pass
Old MovieSlot: 200 capacity. New MovieSlot: capacity 100 Customer trying to exchange 150 tickets to new slot.	Failure: Number of Tickets exceeds than capacity.	Fail

Old MovieSlot: 20 capacity. New MovieSlot: 20 capacity. Customer trying to exchange slot which is not booked.	Failure: Slot isn't booked.	Fail
Old MovieSlot in Verdun server: cap-20 New MovieSlot in Atwater server: cap-20 Customer trying to exchange 5 tickets.	Success	Pass. (But only 3 times a week).
Old MovieSlot not equal to user's area New MovieSlot not equals to user's area. Not happening in same week and >3	Failure	Fail (As limit is 3 for shows outside of user's area)