Reservation System Document

Prepared By: Rayaz Shaik Date:04/29/2019

Assignment:-

Assignment is to design and write a Ticket Service that provides the

following functions:

1) Find the number of seats available within the venue

Note: available seats are seats that are neither held nor reserved.

2) Find and hold the best available seats on behalf of a customer

Note: each ticket hold should expire within a set number of seconds.

3) Reserve and commit a specific group of held seats for a customer

**Solution:-**

→ Implemented the given Service interface to provide the implementation for each function.

→ Service Implementation class uses the DAO class which is needs to interact with Persistence layer to interact with Data

→ Developed the Seat class to represents each seat object, for now we are loading for few Seat objects to the initial available seats pool. In real world this needs to be load from DB

→ Developed the SeatHold object which contains the all seat Hold details and seats which are picked from available seat pool.

Maintained the the below condition:-

Total Seats = Total available seats + seats locked and Seats Confirmed

Total available seats are which are not in Hold Seat Pool

Maintained the separate Collections to for Available seats and Locked or Confirmed Seats.

→ Used a timer and Call back in Hold Seat object where all the locked seats will be removed from Hold Seat collection and will add to available seats if the user did not confirms the locked seat within for specific number of seconds.

We took 10 seconds an example.

→ If user confirms the Seats with in Specified seconds of time then Hold Seat object state becomes the Confirmed and timer task will cancelled. Also Once this Reservation confirmed then this object needs to saved in DB.

→ To Handel the multi-threading issues we created two locks.

Read Lock: To allow multiple users to read the available seats at same time

Write Lock: To allow only one thread to hold or lock the Seat at same time.

Test Cases:

Created the **BookTicketTest** class which covers the Below Test Cases:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case Name** | **Description** | **Initial Data** | **Expected Result** |
| testNumSeatsAvailable() | Test Total number of available | Loaded the 6 Seats in DAO layer Available seats is 4  Locked seats 3 | Available seats is 6 |
| testFindAndHoldSeats() | Test Case 1:  User 1 Hold the 3 seats.  Test Case 2:  User1 confirms the seats after Hold time expire | Loaded the 6 Seats in DAO layer | Available seats is 4  Locked seats 3  Available Seats 6  Locked seats 0  Confirm Reservation status is false |
| testReserveSeats() | User1 Hold and Confirm the seats before Hold time expire | Loaded the 6 Seats in DAO layer | Available seats is 4  Confirmed seats 3  Confirm Reservation status is true |
| testFindAndHoldConcurrentUser() | Test Case 1:  User1 tries to hold 3 seats and User2 tries to hold 4 seats **concurrently**  Test Case 2:  User 3 tries to hold 3 seats | Loaded the 6 Seats in DAO layer | Available Seats 3  User1 hold seats 3  User2 hold seats 0  OR  Available Seats 2  User1 hold seats 0  User2 hold seats 4  If User1 gets seats in previous Test Case1 then  User3 hold seats 3 |