Project3: seating booking system with multiple servers

Use Java RMI to implement a seat reservation system.

- There are at least one server and many seat reservation clients in the system.
- If there are many servers in the system, the first server plays the primary server role. We can assume the primary server is always on duty. The primary server maintains a list of all other servers.
- When a client starts, it connects to the primary server; then get a list of all servers from the primary server. Then we can randomly choose a server for processing seat reservations.
- A client can keep sending seat reservation requests to the server.
- Assume there are totally 50 seats numbered from 1 to 50.
- A client needs to be able to show all current available seats before asking a customer to choose a seat.
- Once a customer selects a seat, the client program sends a seat reservation request to the server. The request includes the customer's name. If the seat is still available, the server can complete the reservation. If the selected seat is not available any more, the server can reject the request. The server cannot assign the same seat to more than one customer.
- The primary server needs to save all reservation information.
- There are many servers are processing seat reservations simultaneously, the system needs to guarantee that no seat can be issued to more than one customer.
- We can assume that the primary server is always on. However, other servers can be down or disconnected. Make the system fault tolerant, in other words, when a client cannot get respond from a server, the system should let it connect to another server for seat reservation. The client cannot be failed just because the connected server is down or failed.

Note: you need to submit a separate document to explain your design of solving race condition (avoid issue one seat to many customers) and how to make the system fault tolerant.