Concurrent Ticket Reservation Server in Linux

2023310570 강민석

1. Introduction

이 과제의 목적은 Linux 환경에서 동시성 및 동기화를 고려한 멀티스레드 티켓 예약 서버를 구현하는 것입니다. 서버는 최대 1024명의 클라이언트 연결을 처리하고, 좌석 관리 및 사용자 인증 기능을 제공합니다.

* 구현 환경 및 요구사항

개발 환경: Linux

사용 언어: C

요구사항:

* + 256개의 좌석 관리
  + 최대 1024 클라이언트 동시 접속
  + 사용자 인증 및 동기화 메커니즘 적용
  + 멀티스레드 기반의 thread-safe하도록 클라이언트 요청 처리

1. used liraries

stdio.h, stdlib.h: 입출력 및 메모리 관리

unistd.h: 시스템 호출 관련 기능

string.h: 문자열 처리

pthread.h: 멀티스레딩 (쓰레드 생성 및 관리)

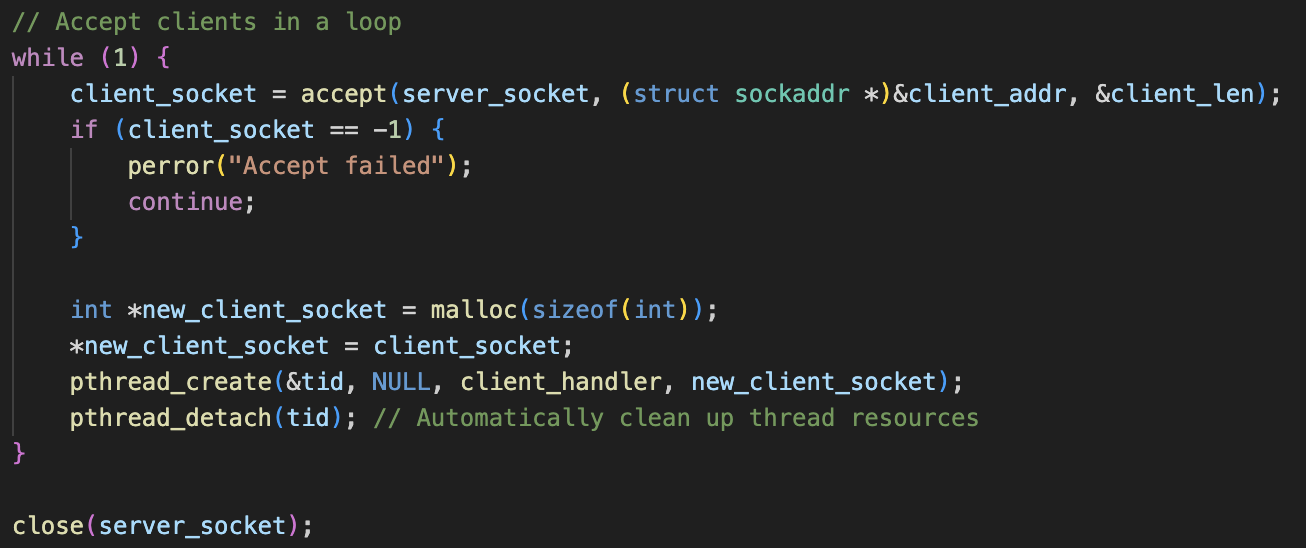
signal.h: 시그널 핸들링

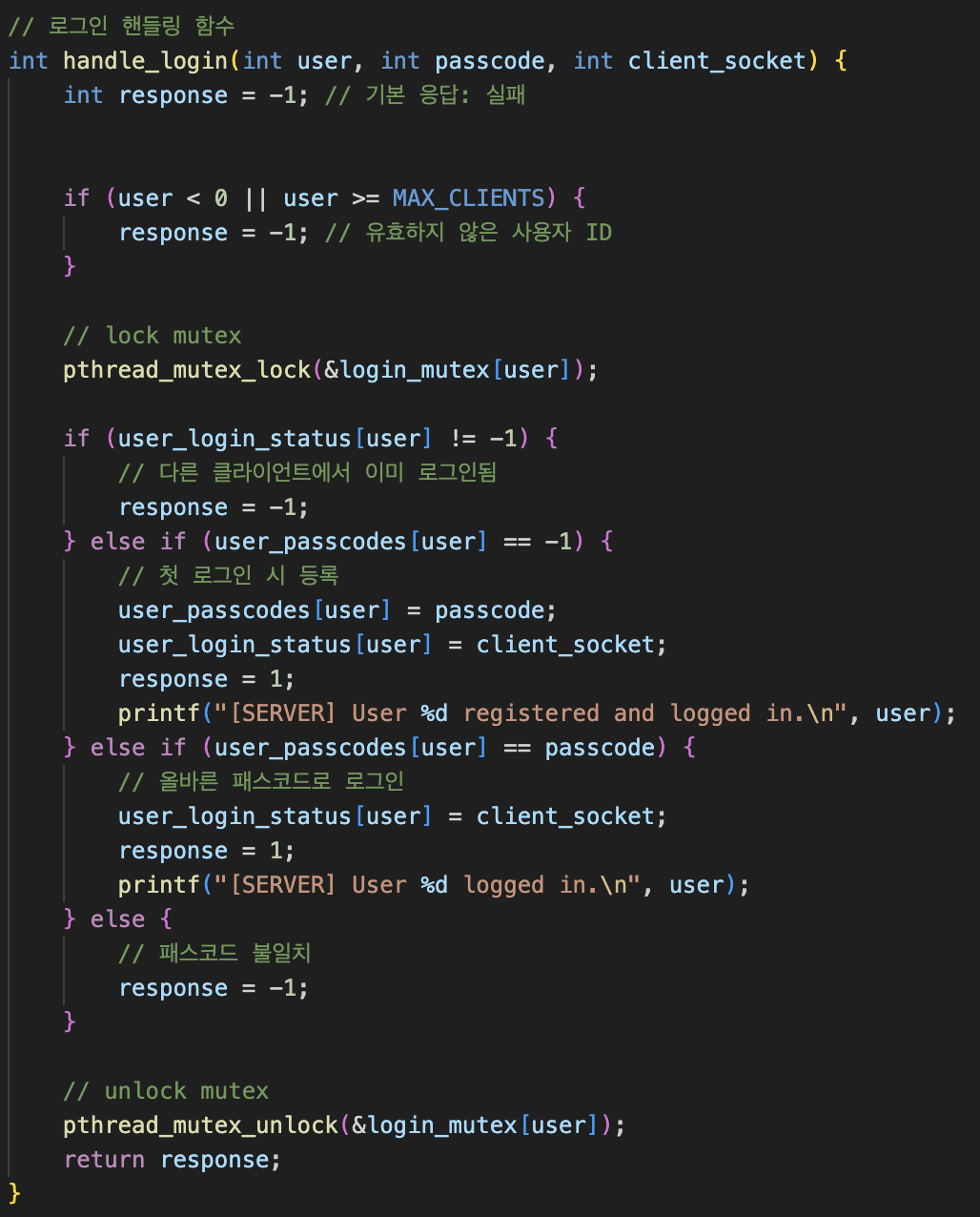
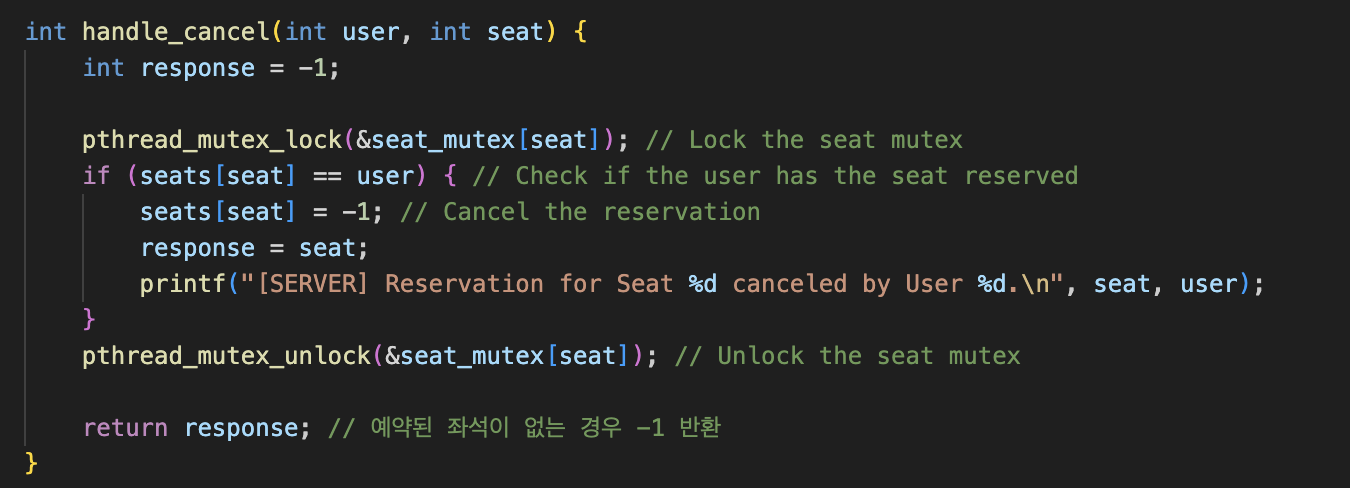
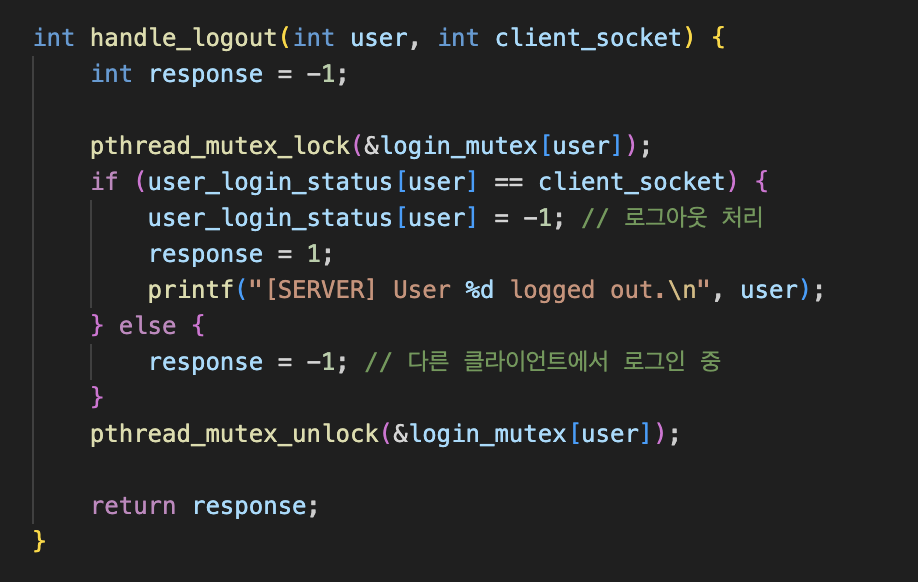
netinet/in.h, sys/socket.h, arpa/inet.h: 네트워크 프로그래밍 (소켓, 주소 설정)

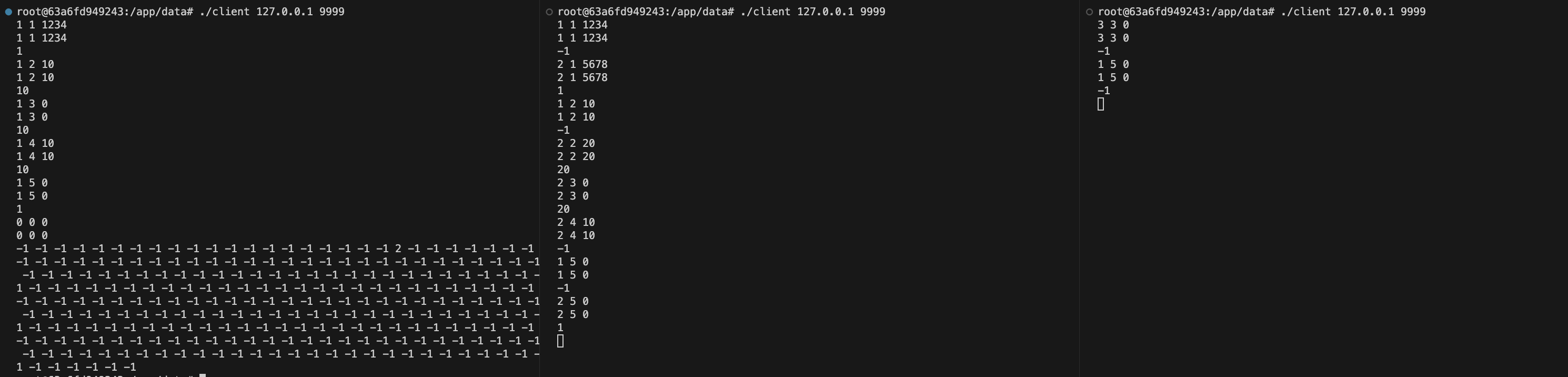
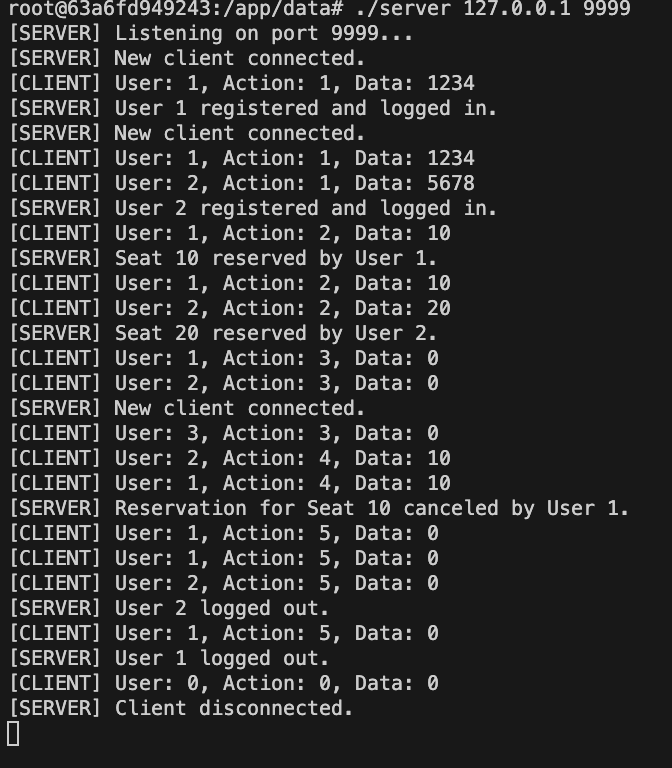
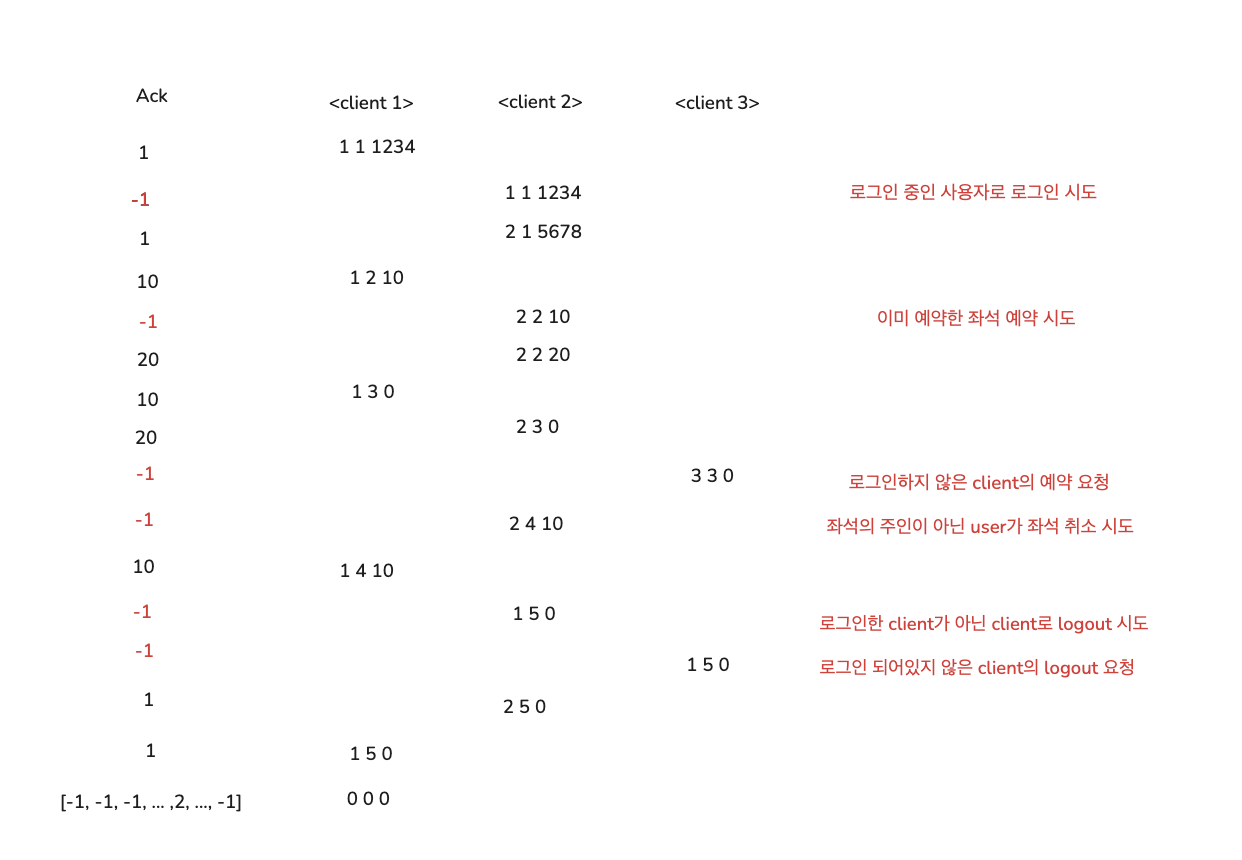
1. implementation logic  
   client의 요청을 받으면 thread를 만들어 client\_handler 함수 내부에서 action을 수행하도록 구현하였습니다.  
   모든 action은 내부에서 예외 처리를 진행하게 되는데, 유저 로그인을 확인해야하는 2, 3, 4, 5번 action에 대해서는 따로 validate 함수를 만들어 검증을 진행하였습니다. 자세한 사항은 아래에 넣어 두었습니다. 각 함수에서 작업의 수행상태 (성공 시 요구사항에 나와있는 값, 실패 시 -1)를 반환하게 됩니다.  
   
2. 서버 초기화  
   서버는 지정된 IP와 포트에 바인딩하고 클라이언트의 연결을 대기합니다.



1. 클라이언트 연결 및 멀티스레딩  
   새로운 클라이언트가 연결될 때마다 쓰레드를 생성하여 요청을 처리합니다.



1. 로그인 (Action 1)  
   1. 예외처리  
    - 유효한 id인지 확인.  
    - 로그인 되어있는지 확인.  
   2. 첫 로그인: 패스코드를 등록하고, 로그인 처리합니다.  
   3. 재로그인: 패스코드 검증 후 로그인 처리합니다.  
   4. 중복 로그인 방지: 사용자별 뮤텍스를 활용하여 하나의 사용자에 여러 쓰레드가 동시에 접근하는 것을 방지하였고, login status를 활용하여 하나의 사용자에 여러 쓰레드가 로그인할 수 없도록 하였습니다.  
   
2. 좌석 예약 (Action 2)  
   각 좌석에 대한 개별적인mutex를 사용하여 좌석에 대한 동기화를 보장하고 성능을 향상하였습니다.  
   - 예외처리  
    - 유효한 좌석 번호인지 확인.   
    - 좌석이 이미 예약 되어있는지 확인  
   
3. 예약 확인 (Action 3)  
   좌석 배열을 순회하여 해당 사용자가 예약한 좌석을 찾아 이를 반환합니다.  
   - 예외처리  
    - 유저 id가 유효한지 확인  
    - 로그인 되어있는지 확인  
   만약 모든 좌석을 순회했을 때 seats배열에 value가 user id와 일치하는 index가 없다면, 예약한 좌석이 없다는 의미이므로 -1을 return  
   
4. 예약 취소 (Action 4)  
   사용자가 예약한 좌석을 취소합니다. mutex를 활용하여 변경중인 좌석에 접근할 수 없도록 합니다.  
   - 예외처리  
    - 유저관련 예외처리  
    - 요청한 좌석의 주인이 유저인지 확인  
   
5. 로그아웃 (Action 5)  
   요청한 사용자를 로그아웃합니다.  
   - 예외처리  
    - 유저 id 관련 예외 확인  
   
6. execution examples



제작한 server의 기능들을 테스트하기 위한 가벼운 테스트 코드입니다.

이미지는 순서대로 테스트 시나리오, server의 output, client 1, 2, 3의 output입니다.

1. Additional Implementations

* User validate

요청한 유저가 만약 해당 유저 id로 로그인이 되어있지 않은 경우, 문제가 발생할 수 있습니다. 예를 들면, client B가 user id 2로 로그인하고, user id 1에 대한 요청을 보냈을 때 기존 과제에 명시 되어있던 요구사항으로는 해당 문제를 해결할 수 없다고 생각하였습니다.

따라서, 이 함수를 만들고 적용함으로써 보다 자연스러운 티켓팅 서버를 구현할 수 있었습니다.

