Q. Design and implement a metro station entry gate.

Boolean isCardValid(Card card)

Card: Car\_id, User\_id, Balance

User: User\_Id, Card\_Id, Govt\_Id,

Wallet : Id,

Transactions:

Authentication

Enum {IDLE, OPEN,CLOSE};

Class ErrorMessages {

String CardNotDetedted() {

Return “Card not detected”;

}

String UserAlreadyOccupied() {

Return “You are already using the gate. Wait for your time”;

}

String GateOccupied() {

Return “Someone is already using the gate. Wait for your time”;

}

}

Class Authentication {

Authentication {

Gate gate = new Gate();

}

Boolean Authentication(Card card) {

if(card == null) {

ErrorMessage.CardNotDetected ;

Return false;

}

if(queue.contains(card) {

ErrorMessage.UserOccupied();

Return false;

}

if( gate.state != IDLE ) {

ErrorMessage.GateOccupied()

Return false;

}

gate.state = IDLE;

Return true;

}

}

Class GateAction {

GateAction() {

If gate.shouldOpenGate(card) {

gate.OpenGate(card);

Wait()

gate..CloseGate(card);

}

}

}

Class Gate {

State state;

ReadWriteLock rwLock = new ReadWriteLock();

Set queue = new ConcurrentHAshSet<Card>();

Gate() {

This.state = IDLE;

}

Boolean shouldOpenGate( Card card) {

Bool valid = false;

rwLock.read.lock();

Valid = isCardValid(card);

rwLock.read.unlock();

Return valid;

}

Void OpenGate(Card card) {

rwLock.write.lock();

print “Gate is open now”

queue.add(card);

This.state = OPEN;

rwLock.write.unlock();

}

Void CloseGate(Card card) {

rwLock.write.lock();

print “Gate is closed now”

This.state = CLOSE;

queue.remove(card);

rwLock.write.unlock();

}

}

Q. Design a movie ticket booking system -

Given Movie, Venue

1. Select time slot
2. Select seats
3. Make payment
4. Confirm booking

***Public APIs (Gateway)***

GetTimeSlots(Movie, Venue)

GetSeatsByTimeSlot(Time, Movie, Venue)

SelectSeats(Time, Movie, Venue)

MakePayment(Seat,Time, Movie Venue)

GetTickets (User)=>

[ (Movie, Venue, Time, Seat[]).

(Movie, Venue, Time, Seat[]),

(Movie, Venue, Time, Seat[]) ]

**Backend Microservices**

User and Venue Authentication

User and Tickets: 1:n

Tickets and seats: 1:1

Booking System

Movie Venue Time

Seat

BlockSeat(Seat,Time, Movie, Venue)

TimeoutBlockSeat(Seat,Time, Movie, Venue) // 10 mins

ConfirmSeat(Seat,Time, Movie Venue)

Movie and Venue => 1:n

movie and time => 1:n

time and seat => 1:n

Time and money => 1:1

DisplayTicket() =>

(Movie, Venue, Time, Seat[])

Payment System

MakePayment(Seat[] , Time, Movie, Venue, Amount)

User and payment

***Databases***

Venue and movie => 1:n

movie and time => 1:n

Time and money => 1:1

time and seat => 1:n

User and Tickets: 1:n

Tickets and seats: 1:1