Please **solve one** of these two problem.

problem 1

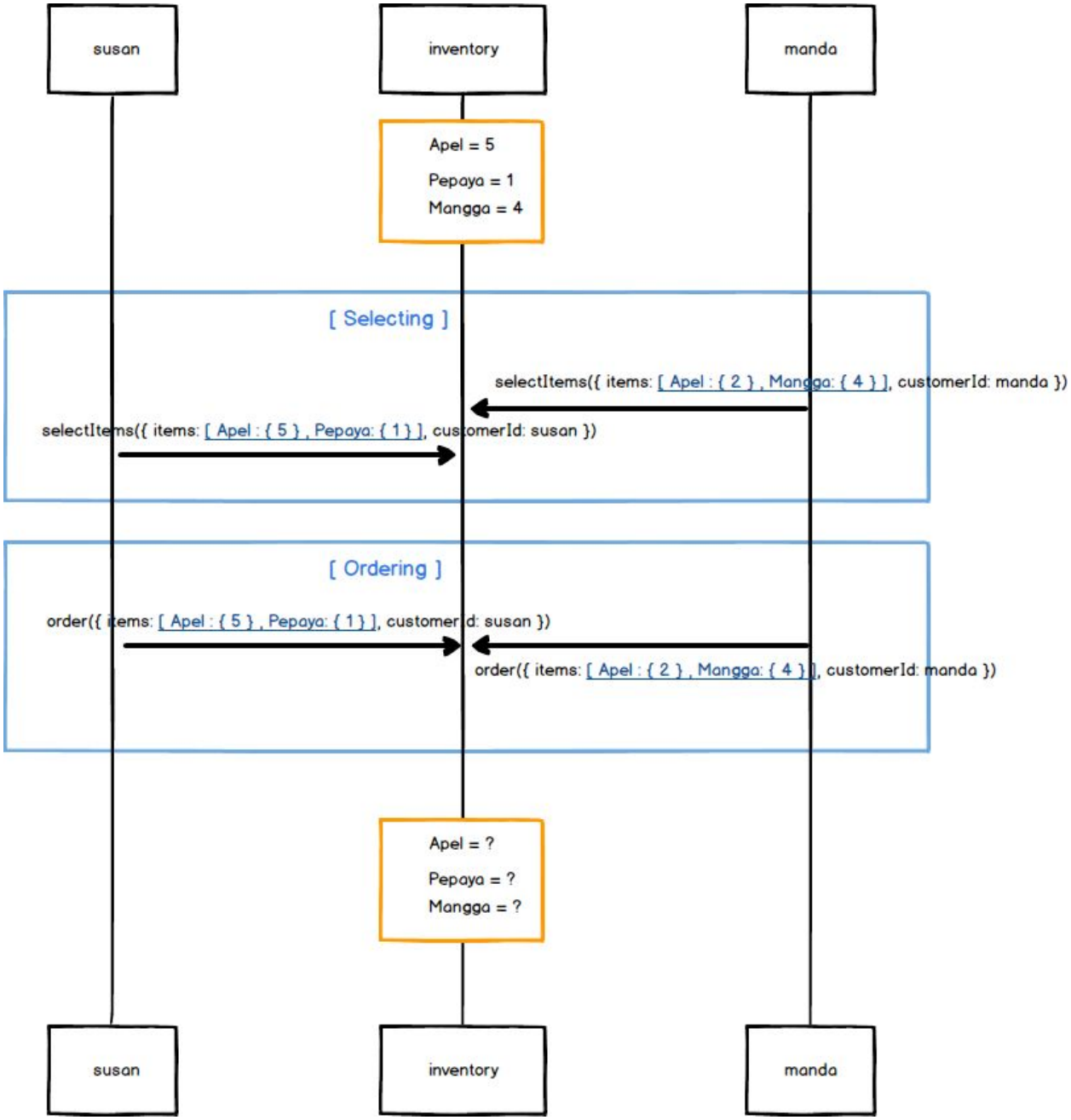
Below are the list of user requirements that are need to be implemented as the technical test for backend engineer. It should be implemented for a week effectively from [DATETIME]. Please send the github link or bitbucket link that tried to solve those problem to us, **don't forget the readme**, so we can run and test your code.

Please implement it using one of the following languages, Java, or NodeJs, or if you want to **implement with another languages, give us the explanation/benchmark why you choose that language**. If you need to ask something, feel free to contact [CONTACTPERSON]

1. (Selecting) As a customer I can select item which I want, if stock is available
2. (Ordering) As a customer I can order items which I've already selected is stock available

Situations which must to be handled

1. Susan and Manda is ordering apple concurrently
2. Apple stock = 5, How do you serve Susan and Manda order?

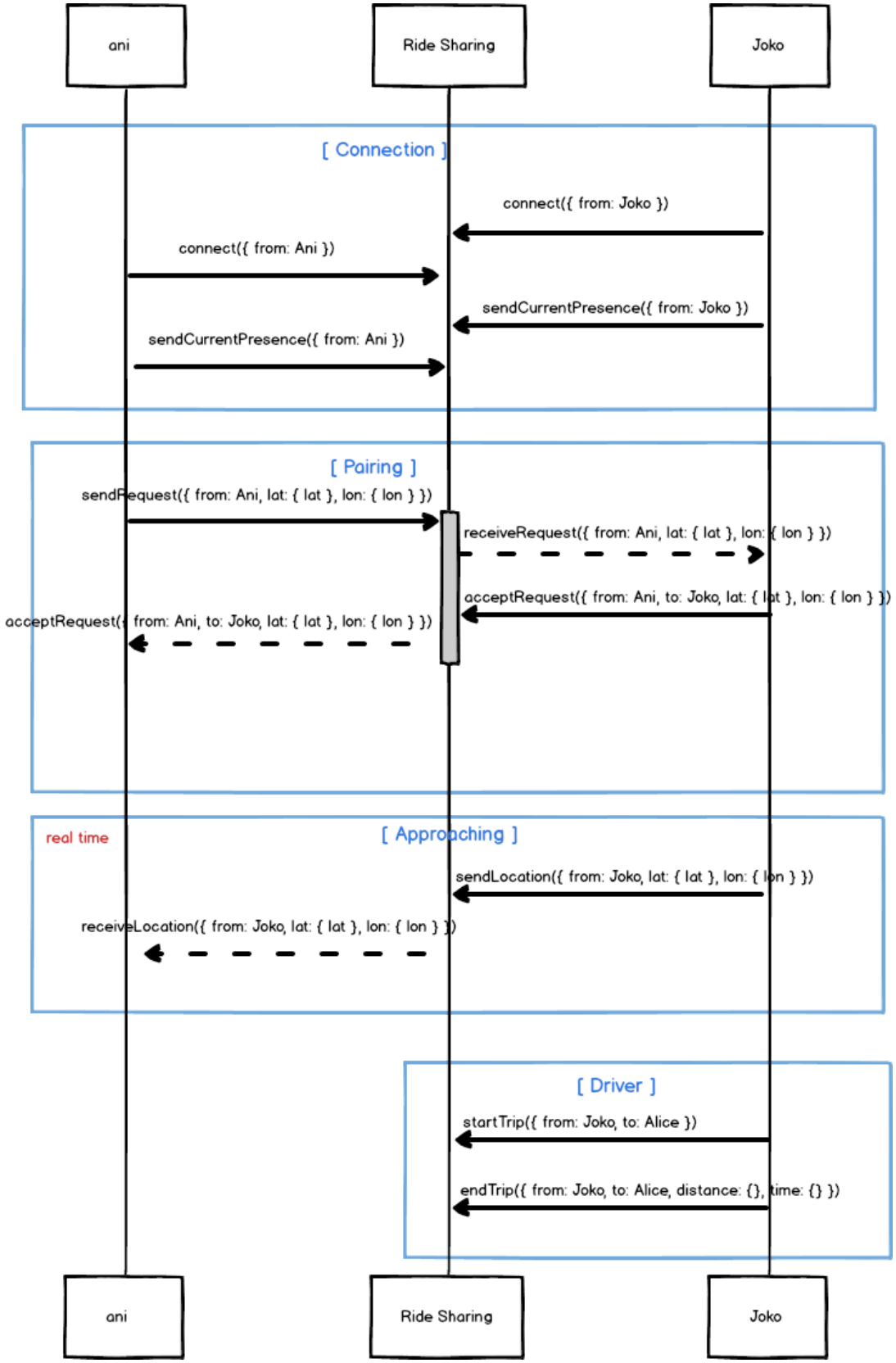

susan
inventory
manda
Apel = 5
Pepaya = 1
Mangga = 4
[Selecting]
selectItems({ items: [Apel : { 2 } , Mangga: { 4 }], customerId: manda })
selectItems({ items: [Apel : { 5 } , Pepaya: { 1 }], customerId: susan })
[Ordering]
order({ items: [Apel : { 5 } , Pepaya: { 1 }], customerId: susan })
order({ items: [Apel : { 2 } , Mangga: { 4 }], customerId: manda })
Apel = ?
Pepaya = ?
Mangga = ?
susan
inventory
manda

problem 2

Below are the list of user requirements that are need to be implemented as the technical test for backend engineer. It should be implemented for a week effectively from [DATETIME]. Please send the github link or bitbucket link that tried to solve those problem to us, **don't forget the readme**, so we can run and test your code.

Please implement it using one of the following languages, Java or NodeJs, or **if you want to implement with another languages, give us the explanation/benchmark why you choose that language**. If you need to ask something, feel free to contact [CONTACTPERSON]

1. (Connection) As a Driver, I can connect to Ride Sharing system
2. (Connection) As a Passenger, I can connect to Ride Sharing system
3. (Connection) As a Driver, I can send my presence to Ride Sharing system
4. (Connection) As a Passenger, I can send my presence to Ride Sharing system
5. (Pairing) As a Passenger, I can send requests to Driver
6. (Pairing) As a Driver, I can receive requests from Passenger
7. (Pairing) As a Driver, I can accept requests from Passenger
8. (Pairing) As a Passenger, I can receive accepted requests from Drivers
9. (Approaching) As a Driver, I can send my location
10. (Approaching) As a Passenger, I can receive Driver location
11. (Driving) As a Driver, I can start the trip
12. (Driving) As a Driver, I can end the trip


ani
Ride Sharing
Joko
[Connection]
connect({ from: Joko })
connect({ from: Ani })
sendCurrentPresence({ from: Joko })
sendCurrentPresence({ from: Ani })
[Pairing]
sendRequest({ from: Ani, lat: { lat }, lon: { lon } })
receiveRequest({ from: Ani, lat: { lat }, lon: { lon } })
acceptRequest({ from: Ani, to: Joko, lat: { lat }, lon: { lon } })
acceptRequest({ from: Ani, to: Joko, lat: { lat }, lon: { lon } })
real time
[Approaching]
sendLocation({ from: Joko, lat: { lat }, lon: { lon } })
receiveLocation({ from: Joko, lat: { lat }, lon: { lon } })
[Driver]
startTrip({ from: Joko, to: Alice })
endTrip({ from: Joko, to: Alice, distance: {}, time: {} })
ani
Ride Sharing
Joko