Needed APIs:

Store

1. Method GET

@desc check availability and price of the item

@req orderItemId

@res itemQuantity, itemPrice

1. Method PUT

@desc update quantity of an item

@req orderItemId, itemQuantity

@res A feedback showing successful/ unsuccessful update

1. Method POST

@desc get total weight of an order

@req orderItemId, itemQuantity

@res totalWeight

Accounting

1. Method POST

@desc receive accounting Id and total price

@req salesOrderId, orderItemId, itemQuantity, itemPrice, orderDelivery

@res accountingId/ invalid id, totalPrice

1. Method PUT

@desc put paymentId

@req accountingId, paymantId

@res A feedback showing successful/ unsuccessful update

Delivery

1. Method GET

@desc declares whether a kind of delivery is possible

@req totalWeight, orderDelivery

@res A feedback showing possibility of delivery

Payment

1. Method POST

@desc send salesOrderId and total price for payment

@req salesOrderId, totalPrice

@res paymentId/ invalid id

Customer

1. Method PUT

@desc save salesOrderId for a customer

@req customerId, salesOrderId

@res A feedback showing successful/ unsuccessful update

APIs needed to be deployed:

Customer

1. Method GET

@desc show the list of all orders of a customer

@req -

@res List<SalesOrder>

1. Method PUT

@desc delete an order (set/ reset a flag showing activity of an order)

@req customerId, salesOrderId

@res A feedback showing successful/ unsuccessful deletion

1. Method POST

@desc create a new order

@req an object of SalesOrder class

@res A feedback showing successful/ unsuccessful order registration