2.Use-Case for Removing a Member

Actions performed by the actor:	**Responses from the system:**
1. Clerk issues a request to remove a member	
	2.The system asks for the member ID.
3. The clerk enters the given member ID.	
	4. The system checks to see if member is valid and in database. If valid, it removes member and their ID from database. Then, it shows confirmation of removal. The system then asks if more members to be removed.
5. The clerk answers in the affirmative or in the negative.	
	6. For affirmative, return to step 2. For negative, exit process.

Use-Case for Adding a Product

Actions performed by the actor:	**Responses from the system:**
2. Clerk issues a request to add new product.	
	2.The system asks for the product name, id, stock in hand, current price, and a reorder level.
7. The clerk enters the unique product name, id, stock in hand, current price and the reorder level.	
	8. The system checks to see if there is product with the same name already in the system. If successful, the system attempt to enter the information into the catalog and informs the clerk about the results. If that was successful, the system generates an order for twice the reorder level. If that was successful, the system reports the order was successful. Then the system then asks the clerk is there is more product to enter.
9. The clerk answers yes or no.	
	10. For yes, return to step 3. For not yes, exit process.

Use case #4: Check out a member' items.

Actions performs by the actor	**Responses from the system**
1. A member arrives at the check-out counter with a cart of items.	
	2. the system asks for the member id
3. the cashier enters the member id	
2. The cashier inputs product id and quantity for each item in the cart.	
	3. The system computes the price and total price. The system displays the product name, quantity, unit price, the total price for that item and the total price for that member.
4. Member will pay with cash.	
	5. The system will reorder products in the previous purchase if any of it reach the reorder level or below for twice the reorder level. If happens, the system will display a message telling what was reordered, how much, and the order number.

Use-Case for Processing a Shipment

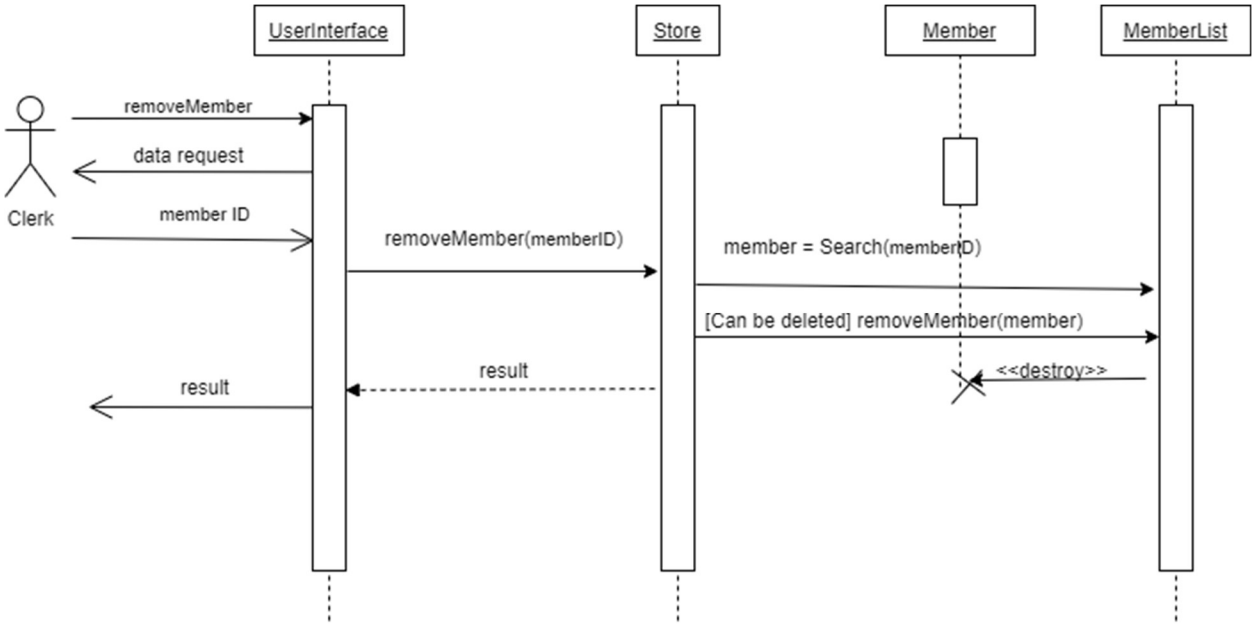
Actions performed by the actor:	**Responses from the system:**
1. The grocery store receives a shipment of products from a supplier.	
3. The clerk issues a request to process a new shipment.	
	3.The system asks for order number of the shipment.
4. The clerk enters the order number of the shipment.	
	11. The system searches through the list of orders for the order with the same number. If the order is not found; the system prints an appropriate message and exits. If the order was found in the list of orders, then the system prints out the product id, name and new stock amount for each product. The system then asks if there is another order number to process.
6. The clerk replies (y or n) yes or no.	
	7. If the answer is yes; the system goes back to step 3; otherwise it exits.

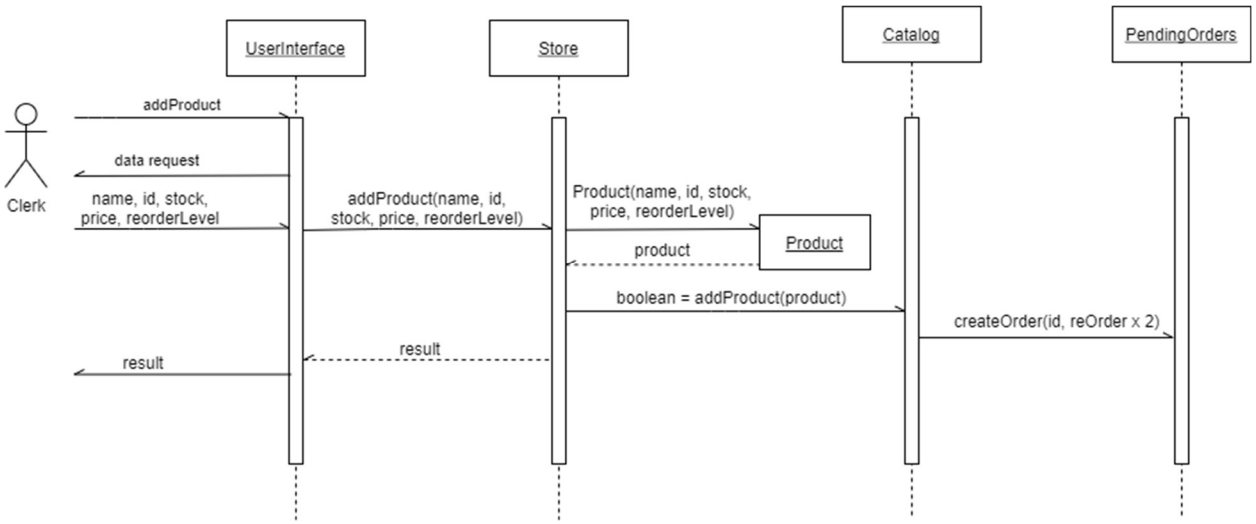
Use case #8: Retrieve member info:

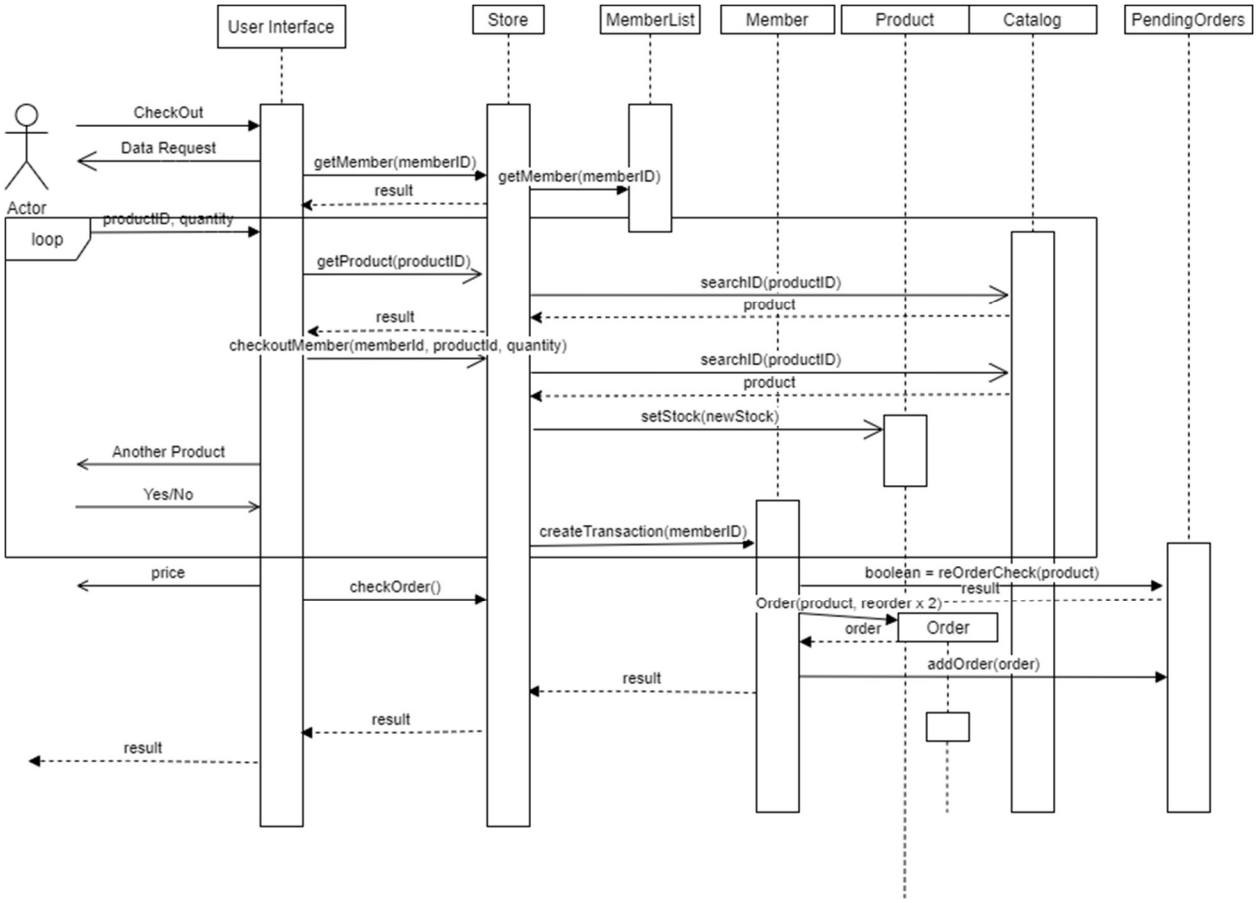
Actions performs by the actor	Responses from the system
1. A string (a name or part of a name) is given to the system.	
	2. The system displays members' address, fee paid, and id of all members whose name has the given string.

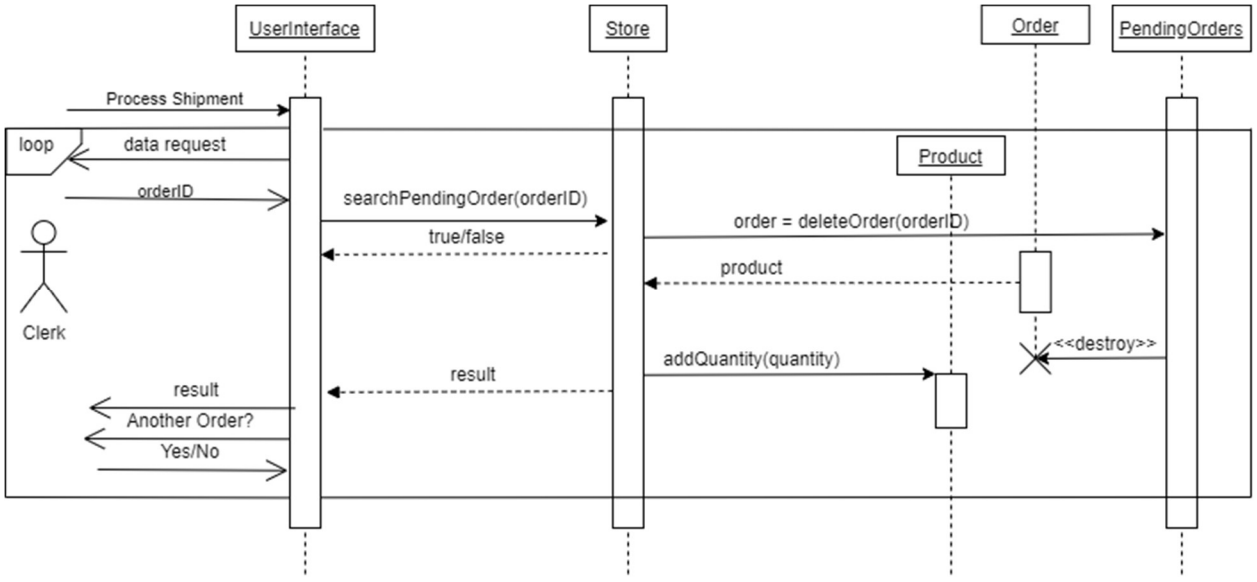
Use-Case for Printing Transactions

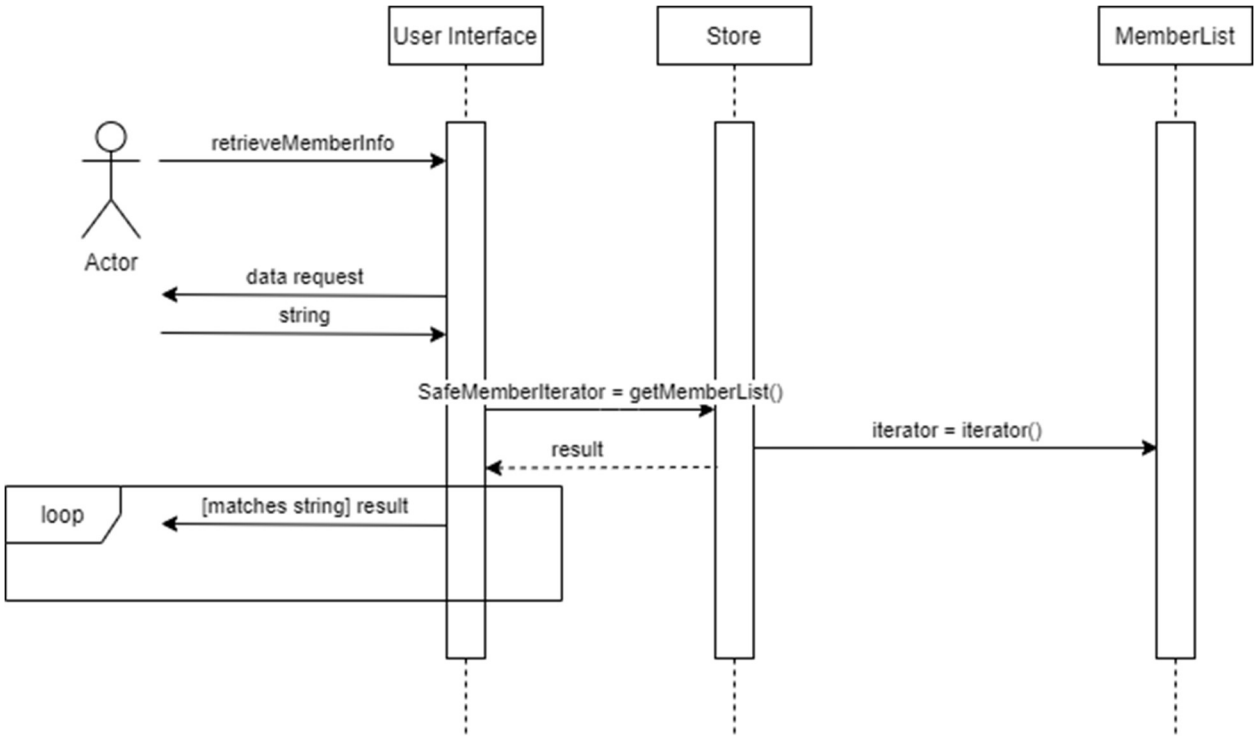
Actions performed by the actor:	**Responses from the system:**
4. The clerk issues a request get member transactions.	
	2.The system asks for the id of the member and asks for two dates that span the period of which the desired transactions are to be printed.
12. The clerk enters the member's id and the two dates.	
	13. The system searches for the id of that member in the registry, and determines if the span of the dates is valid by ensuring that the first date comes before the second, or that they are equal to each other. If the id was not found or the dates are not valid; the system exits. If the system finds the member of the id, and the dates were valid, the system displays a message confirming that the member was found. For each transaction made by the member on and between the two dates the system displays the product name, quantity, price and total amount for each transaction.
8. The clerk prints out the transactions and gives them to the customer.	

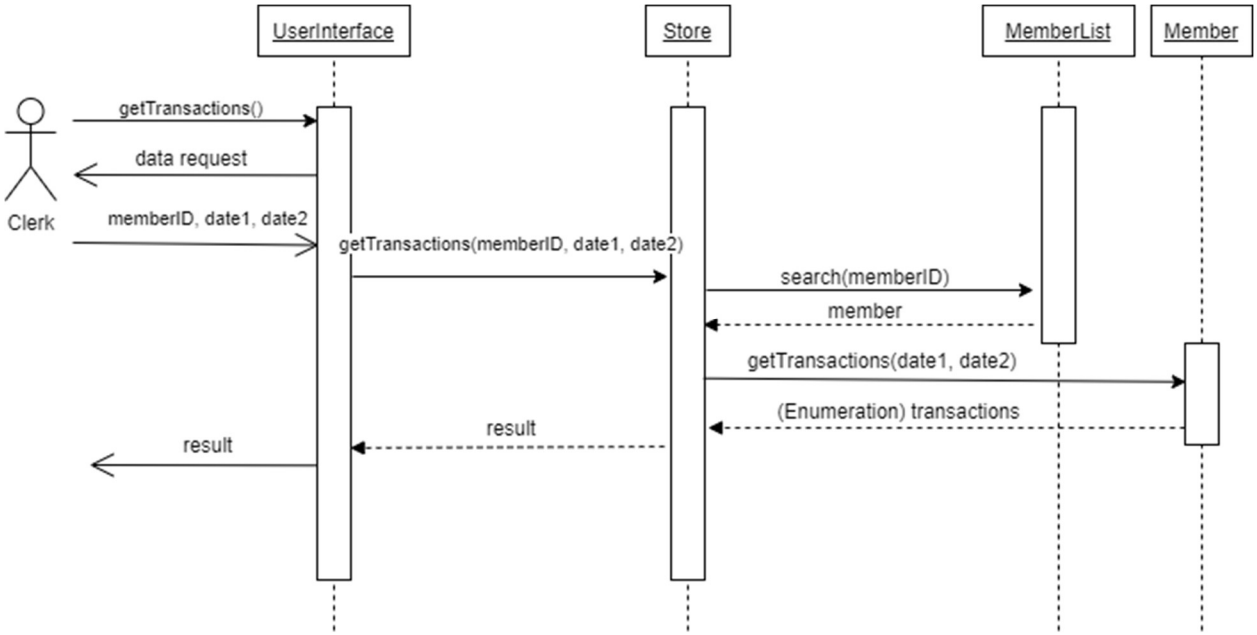

UserInterface
Store
Member
MemberList
Clerk
removeMember
data request
member ID
removeMember(memberID)
member = Search(memberID)
[Can be deleted] removeMember(member)
result
result
<<destroy>>

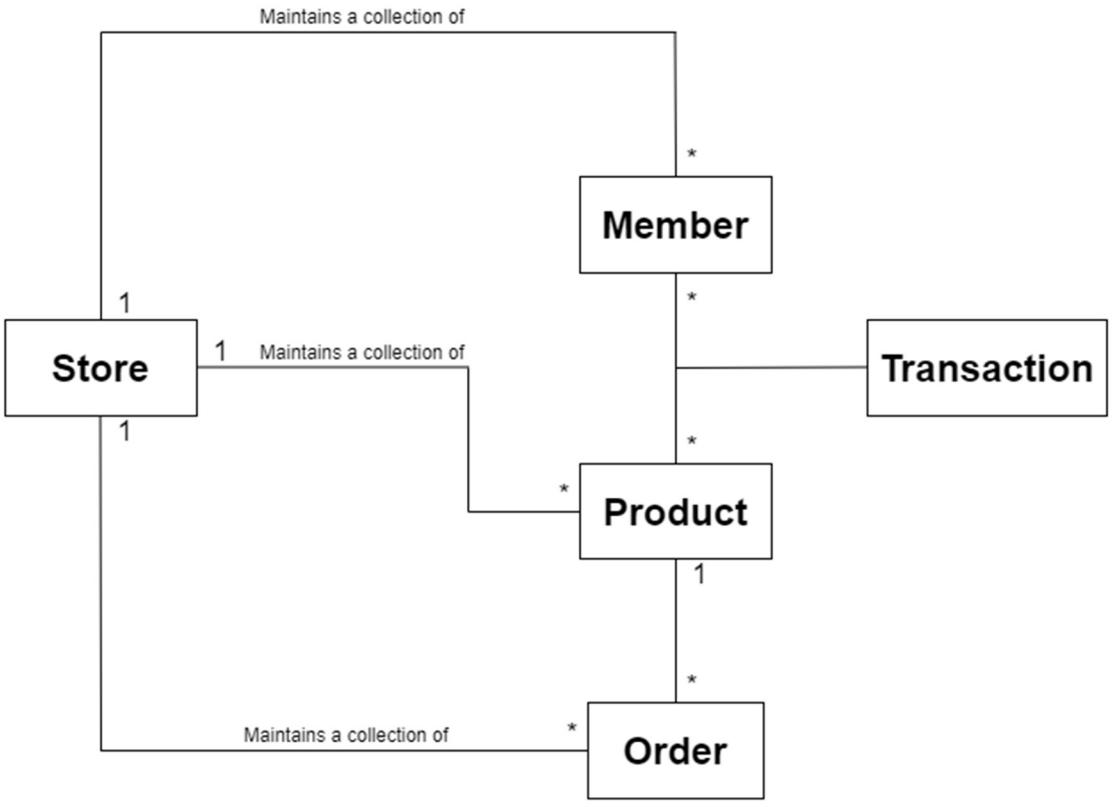

UserInterface
Store
Catalog
PendingOrders
Clerk
addProduct
data request
name, id, stock,
price, reorderLevel
addProduct(name, id,
stock, price, reorderLevel)
Product(name, id, stock,
price, reorderLevel)
Product
product
boolean = addProduct(product)
createOrder(id, reOrder x 2)
result
result


User Interface
Store
MemberList
Member
Product
Catalog
PendingOrders
Actor
CheckOut
Data Request
getMember(memberID)
result
getMember(memberID)
loop
productID, quantity
getProduct(productID)
searchID(productID)
product
result
checkoutMember(memberId, productId, quantity)
searchID(productID)
product
setStock(newStock)
Another Product
Yes/No
createTransaction(memberID)
price
checkOrder()
boolean = reOrderCheck(product)
result
Order(product, reorder x 2)
order
Order
addOrder(order)
result
result
result


UserInterface
Store
Order
PendingOrders
Process Shipment
loop
data request
Product
orderID
searchPendingOrder(orderID)
true/false
order = deleteOrder(orderID)
product
Clerk
<<destroy>>
addQuantity(quantity)
result
result
Another Order?
Yes/No


User Interface
Store
MemberList
retrieveMemberInfo
Actor
data request
string
SafeMemberIterator = getMemberList()
result
iterator = iterator()
loop
[matches string] result


UserInterface
Store
MemberList
Member
getTransactions()
Clerk
data request
memberID, date1, date2
getTransactions(memberID, date1, date2)
search(memberID)
member
getTransactions(date1, date2)
(Enumeration) transactions
result
result



Member
id
name
dateJoined
address
phoneNumber
feePaid

Product
id
name
stock
price
reorderLevel

Order
id
product
dateOrdered
quantity

Transaction
date
product
quantity

Store