Lab-6 **Software Engineering**

Name:- **Ramkumar Patel** Student-ID:- **202201509**

(Q1) Develop Use Case Textual Description for "Process Sale" and "Handle Return" use cases.

➢ Use Case Textual Descriptions

Process Sale

Actors: Cashier, Inventory System, Catalog System, Payment Processor
Preconditions: The cashier is logged into the system. The POS is connected to the inventory and catalog systems.
Postconditions: The sale is completed, payment is processed, stock is updated, and a receipt is printed.
Main Flow:

1. The cashier starts a new sale.
2. The cashier scans the product barcode.
3. The system retrieves the product's name and price from the catalog.
4. The system updates the inventory stock for the product.
5. The cashier finalizes the sale and selects the payment method.
6. The payment is processed via cash, card, or check.
7. The system prints a receipt for the customer.
8. If the customer uses a coupon, the system applies a discount before payment.

Extensions:

- If payment fails, prompt the cashier to retry or cancel the transaction.
- If an invalid barcode is scanned, prompt the cashier to rescan.

Handle Return

Actors: Cashier, Inventory System
Preconditions: The cashier is logged into the system. The item to be returned must have a **valid** receipt or sale record.
Postconditions: The return is processed, stock is updated, and a refund is issued.
Main Flow:

1. The cashier initiates the return process.
2. The cashier scans or enters the product details from the receipt.
3. The system verifies the product and price from the sale record.
4. The system updates the inventory stock to add the returned product.
5. The system processes the refund based on the original payment method.
6. The system prints a return receipt for the customer.

Extensions:

- If the product is not found in the sale record, prompt the cashier to check the receipt.
- If the refund amount exceeds a limit, escalate to the administrator for approval.

(Q2) Identify Entity/Boundary Control Objects

➔ **Entity Objects:**

- *Product*: Contains details about the item, such as name, price, and stock.
- *Sale*: Represents a completed transaction, including products sold and total price.
- *Return*: Represents a product return, including refunded amount and restocked products.

➔ **Boundary Objects:**

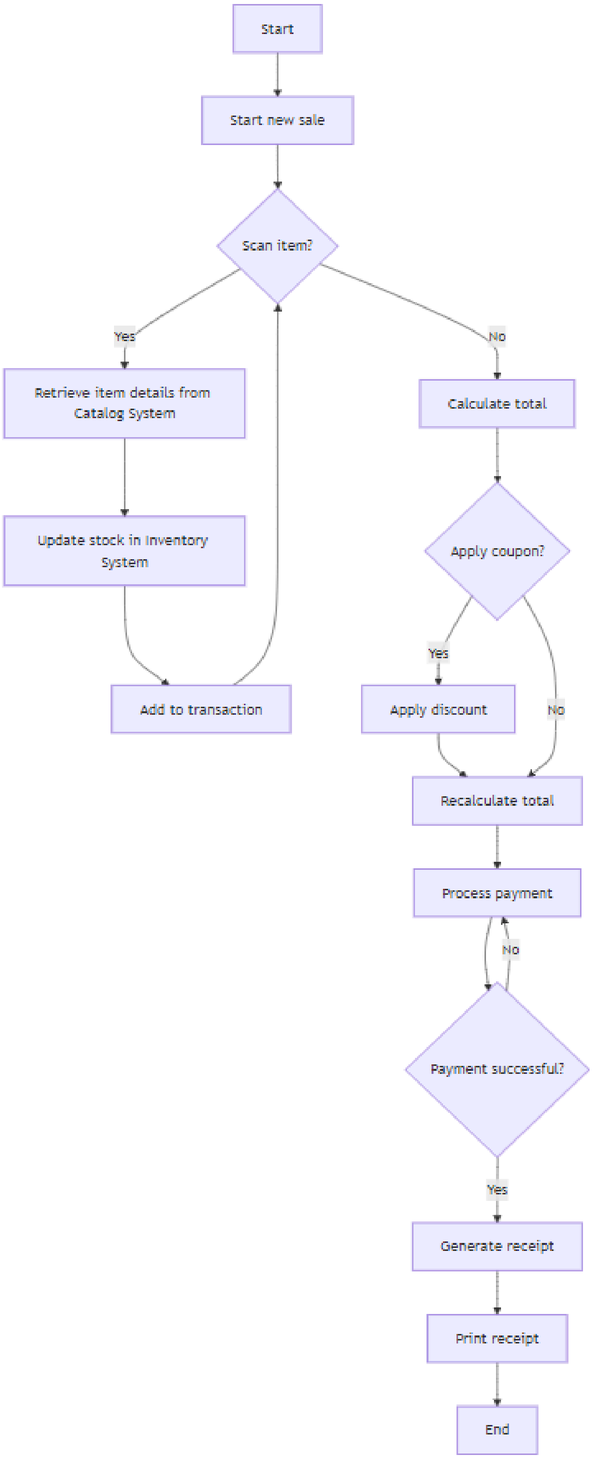
- *POS Terminal*: Interface used by the cashier to process sales and returns.
- *Receipt*: Printed document after a sale or return.
- *Payment Interface*: Manages customer payments (cash, credit card, or check).

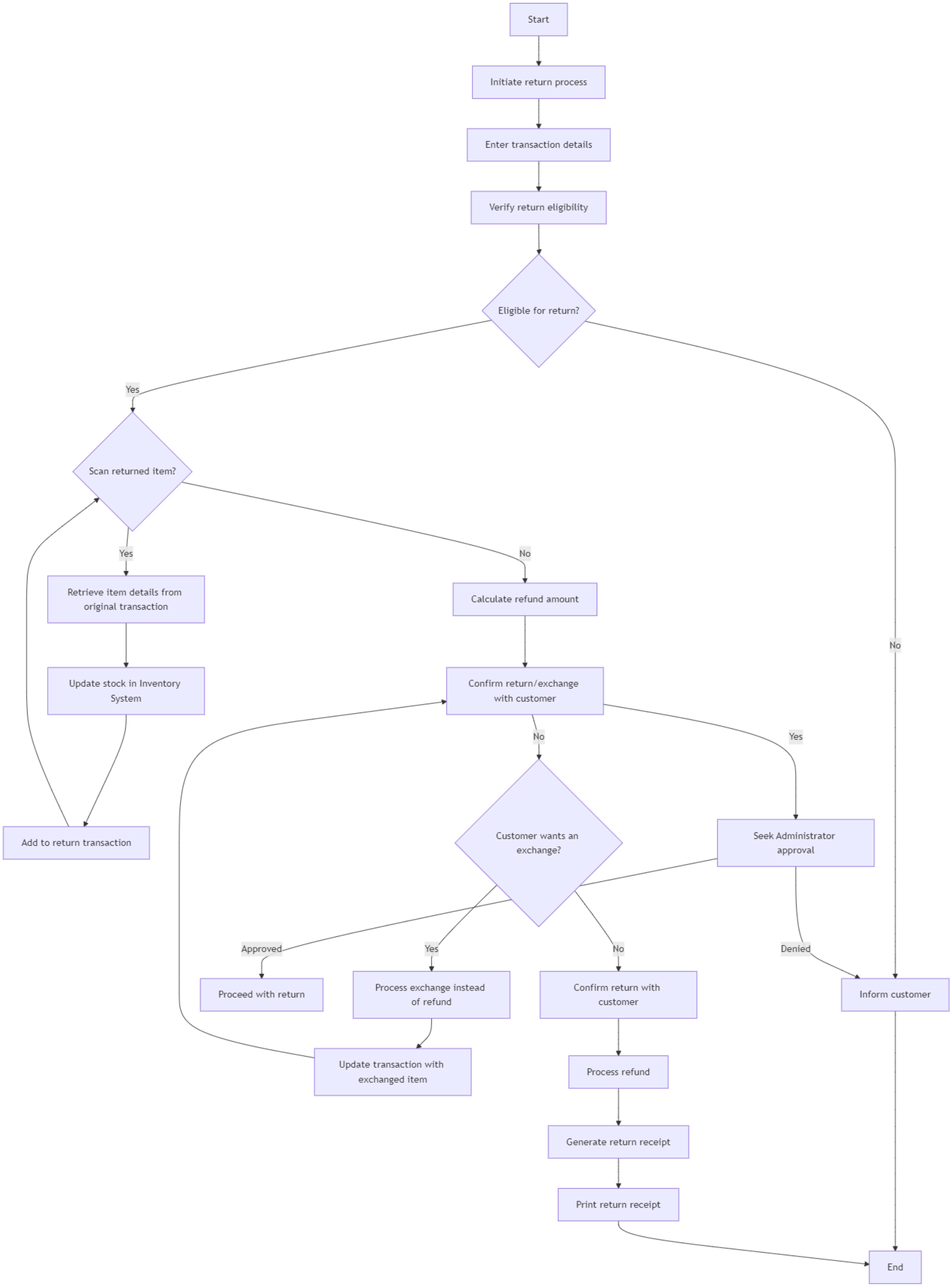
➔ **Control Objects:**

- *Sale Controller*: Manages the sale process, updates inventory, and communicates with payment systems.
- *Return Controller*: Manages product returns, updates stock, and handles refunds.

(Q5) Develop activity diagram for "Process Sale" and "Handle Return" use cases.

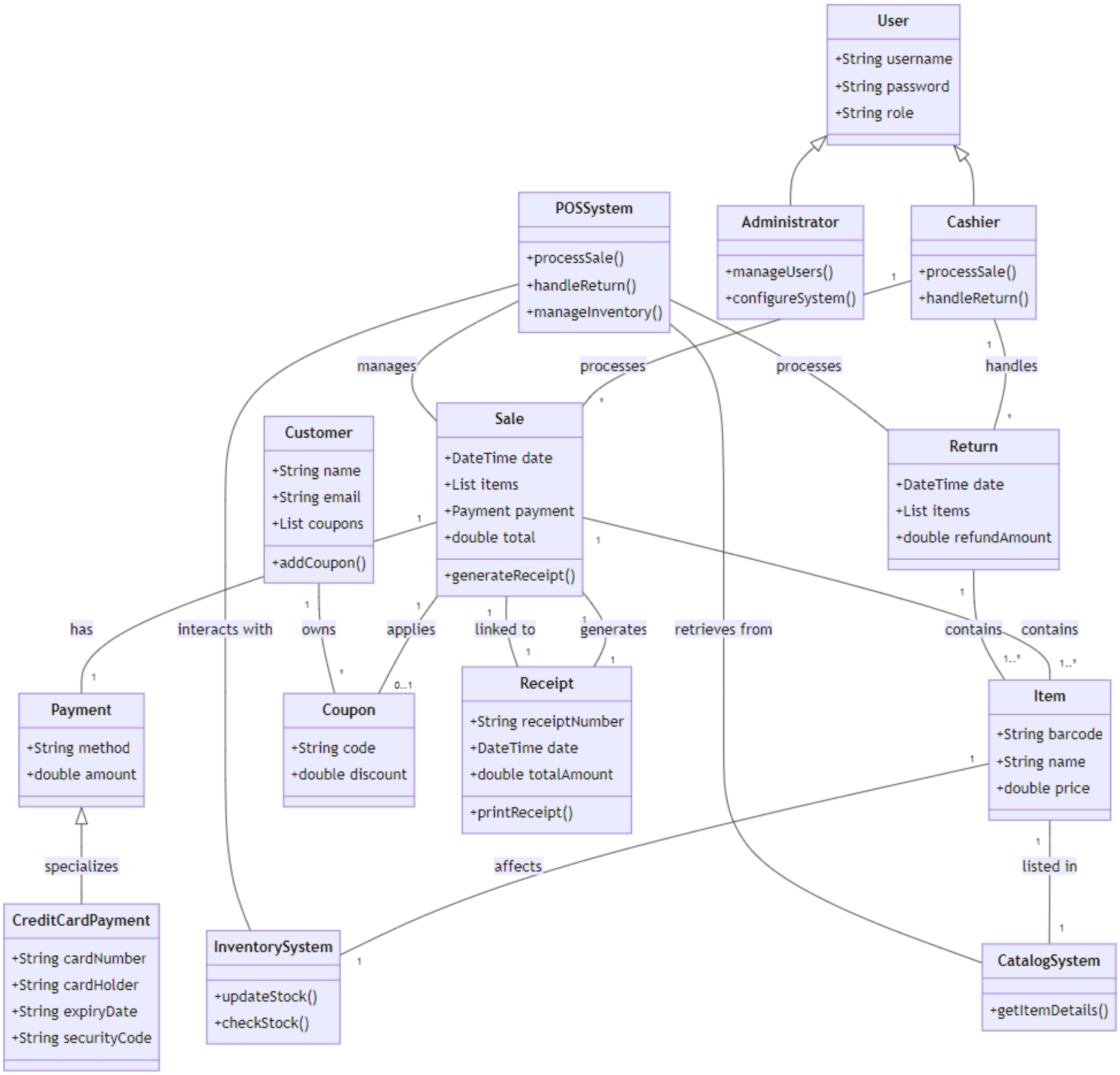
(i & ii) Activity Diagram for “Process-Sale” and “Handle-Return”




Start
Initiate return process
Enter transaction details
Verify return eligibility
Eligible for return?
Yes
No
Scan returned item?
Yes
No
Retrieve item details from original transaction
Calculate refund amount
Update stock in Inventory System
Confirm return/exchange with customer
No
Yes
Add to return transaction
Customer wants an exchange?
Seek Administrator approval
Approved
Yes
No
Denied
Proceed with return
Process exchange instead of refund
Confirm return with customer
Inform customer
Update transaction with exchanged item
Process refund
Generate return receipt
Print return receipt
End

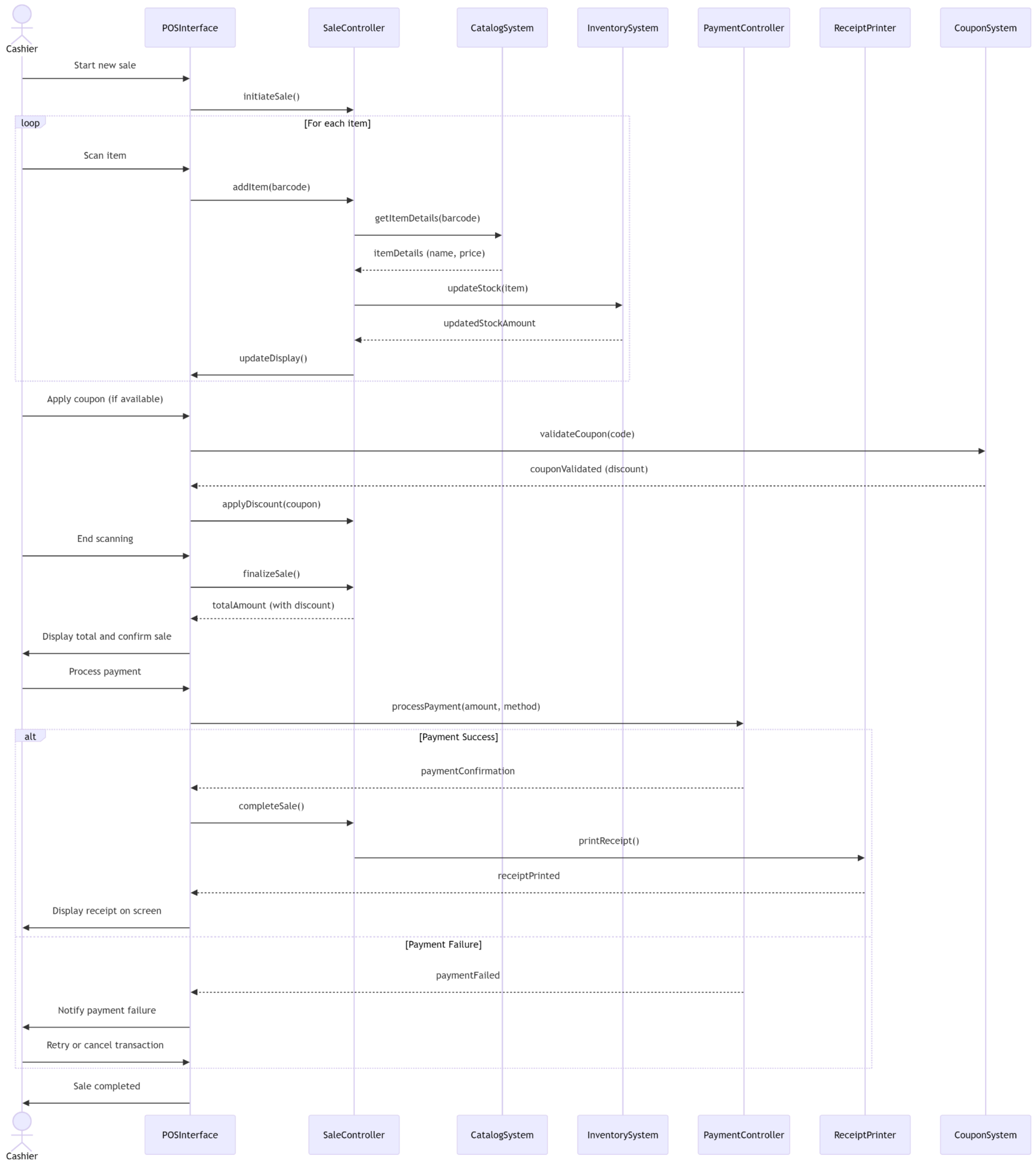
(Q4) Develop Analysis Domain Models

➢

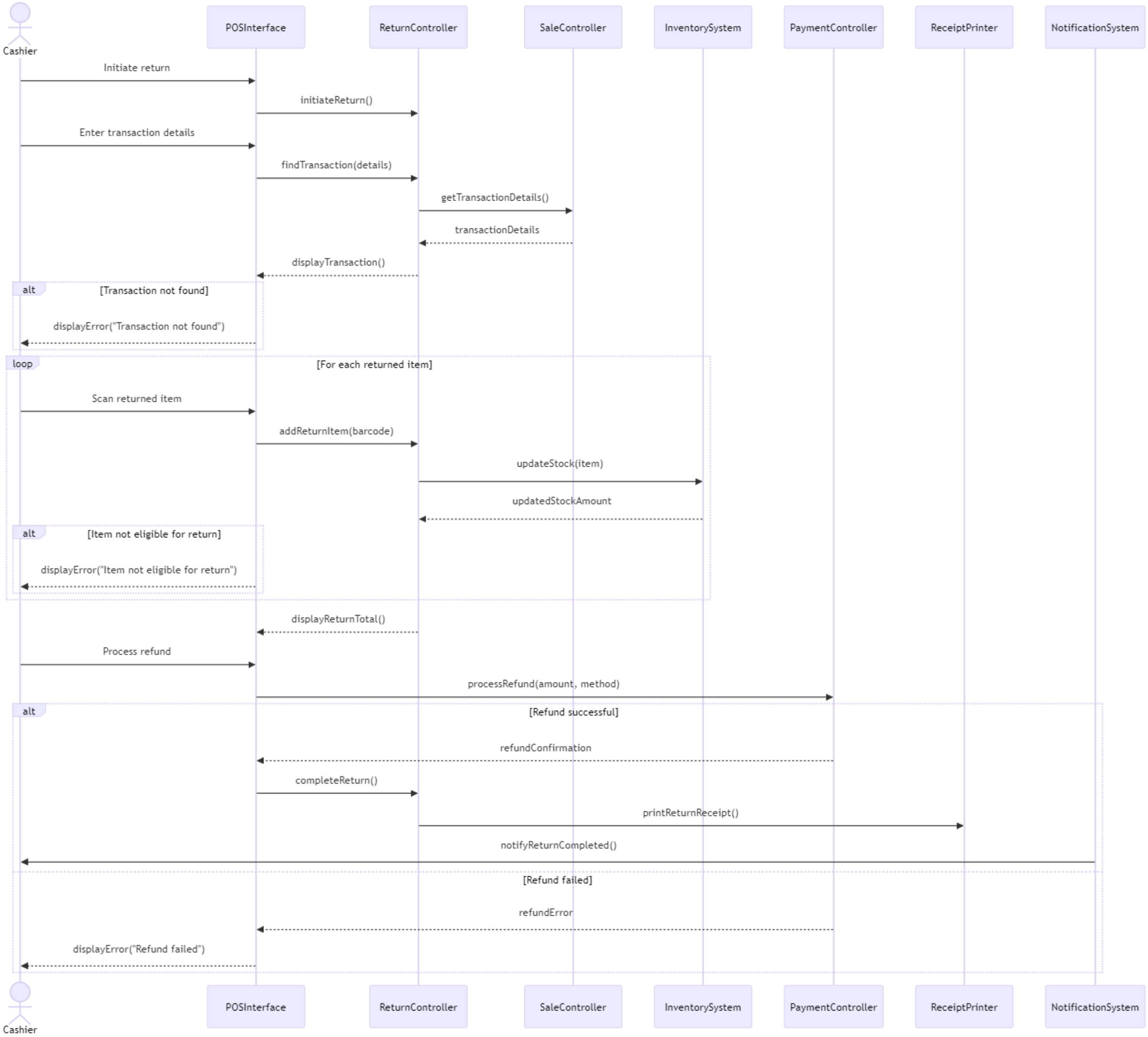


(Q2) Develop Sequence Diagram

- Below is the sequence diagram for **“Process-Sale”**



➢ Below is the sequence diagram for **“Handle-Return”**


Cashier
POSInterface
ReturnController
SaleController
InventorySystem
PaymentController
ReceiptPrinter
NotificationSystem
Initiate return
initiateReturn()
Enter transaction details
findTransaction(details)
getTransactionDetails()
transactionDetails
displayTransaction()
alt
[Transaction not found]
displayError("Transaction not found")
loop
[For each returned item]
Scan returned item
addReturnItem(barcode)
updateStock(item)
updatedStockAmount
alt
[Item not eligible for return]
displayError("Item not eligible for return")
displayReturnTotal()
Process refund
processRefund(amount, method)
alt
[Refund successful]
refundConfirmation
completeReturn()
printReturnReceipt()
notifyReturnCompleted()
[Refund failed]
refundError
displayError("Refund failed")
Cashier
POSInterface
ReturnController
SaleController
InventorySystem
PaymentController
ReceiptPrinter
NotificationSystem