IT314 - Lab6

Modeling Class Diagram and Activity Diagram

Popatiya Harsh Nitinkumar - 202201463

23rd September, 2024

Q.1 Develop Use Case Textual Description for "Process Sale" and "Handle Return" use cases.

Use Case: Process Sale

Actor: Cashier
Precondition: The cashier is logged into the POS system.
Trigger: The customer arrives at the checkout with goods to purchase.

Description:

1. The cashier starts a new sale transaction in the POS system.
2. The barcode of each item is scanned or manually entered, which retrieves the price and name of the product from the catalog system.
3. The inventory system updates the stock amount for the scanned items.
4. After scanning all items, the cashier totals the transaction.
5. The customer selects a payment method (cash, credit card, or check).
6. If there are any gift coupons, they are processed by the POS system to provide the customer with a discount.
7. The payment is processed:
   - For cash, the cashier handles the transaction and provides change if necessary.
   - For credit cards or checks, the POS system verifies and authorizes the payment.
8. After successful payment, a receipt is printed for the customer.
9. The transaction is completed, and the system is updated with the sale details.

Postcondition:

- The sale transaction is saved, and the inventory is updated.
- The customer receives a printed receipt.

Alternate Flows:

- If the payment fails, the transaction is voided or restarted.
- If the barcode of the product is not found in the catalog system, the cashier can manually enter the price or cancel the item.

Use Case: Handle Return

Actor(s): Cashier
Precondition: The cashier is logged into the POS system. The customer requests a return of previously purchased goods.

Trigger: The customer brings an item to the cashier for return.

Description:

1. The cashier initiates a return process in the POS system.
2. The cashier scans or manually enters the details of the returned item from the original purchase receipt.
3. The system checks the inventory system to ensure that the item is returnable (e.g., within the return policy period).
4. The POS system calculates the amount to be refunded to the customer.
5. The cashier processes the refund:
   ○ If the original purchase was made by cash, the cashier refunds the money directly.
   ○ If it was made by credit card or check, the refund is processed through the payment method.
6. The inventory system updates to reflect the returned item being added back to stock.
7. A return receipt is printed for the customer.

Postcondition:

● The returned item is added back to the inventory.
● The customer receives the refund and a printed return receipt.

Alternate Flows:

● If the item is not eligible for return (e.g., due to exceeding the return policy), the cashier informs the customer, and no refund is processed.
● If the payment method used for the original purchase is no longer valid, an alternate method may be used for the refund.

Q.2 Identify Entity and Boundary Control Objects

Entity Objects:

- **Product: Represents the goods being purchased or returned. Includes details like barcode, name, price, and stock.**
- **Sale Transaction: Represents the transaction, including details of items, payment method, total price, and receipt.**
- **Return Transaction: Represents the process of returning an item, the original purchase details, and the refund.**
- **Inventory: Represents the stock information of products that is updated after each sale or return.**
- **Customer: Represents the customer purchasing the product or requesting a return (an external entity interacting indirectly with the system).**

Boundary Objects:

- **POS Interface: The system interface used by the cashier to initiate and complete the sale or return transaction (e.g., screens to scan barcodes, input payment, etc.).**
- **Barcode Scanner: Used to scan products and retrieve details from the catalog.**
- **Receipt Printer: Used to print the sale or return receipt for the customer.**
- **Payment Terminal: Interface for handling payments via credit card, cash, or check.**

Control Objects:

- **Transaction Controller: Manages the overall flow of the sale or return transaction, ensuring that each step (scanning, payment, etc.) is processed correctly.**
- **Payment Processor: Handles the payment validation and authorization for different payment methods (cash, credit card, check).**
- **Inventory Manager: Updates stock levels in the inventory system based on the sale or return transaction.**
- **Coupon Processor: Handles the processing of gift coupons during the sale for discounts.**

Q.3 Develop Sequence Diagrams

● Process Sales
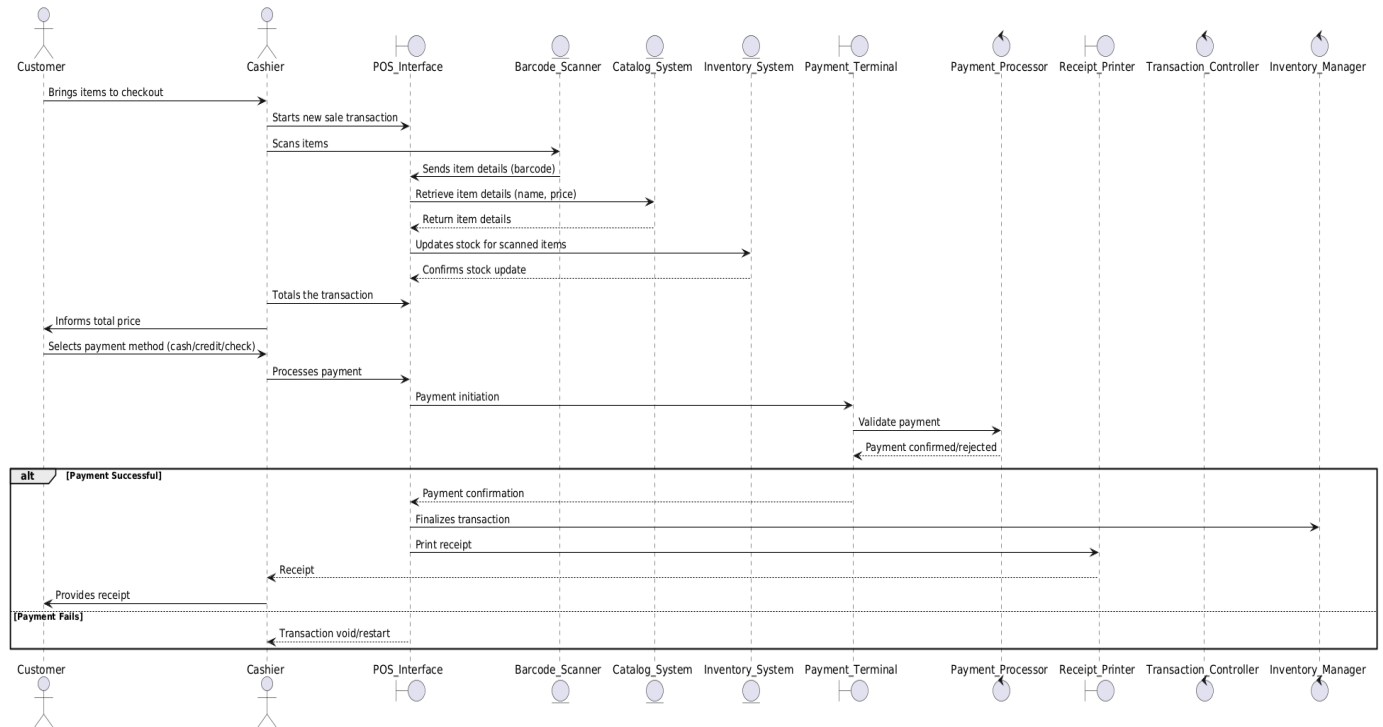
● Handle Return
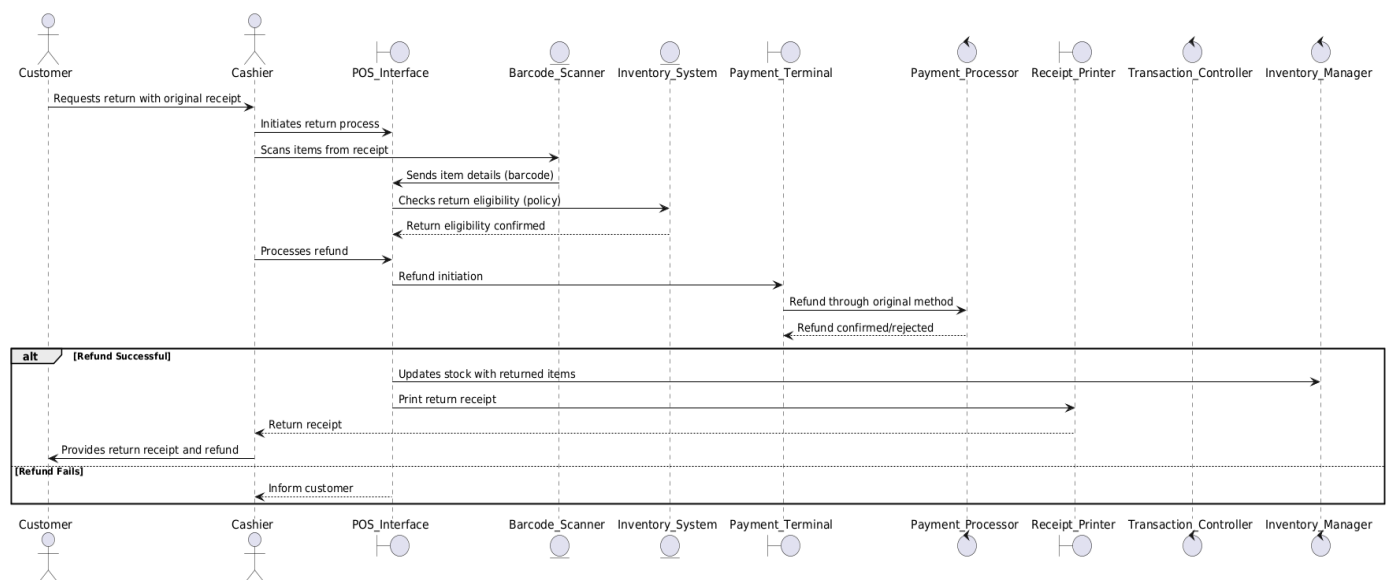
Q.4 Develop activity diagram for "Process Sale" and "Handle Return"

- Process Sales

● Handle Return

Initiate return process

Scan or enter return item
details

Eligible for Return

Yes

No

Calculate refund amount

Inform customer of
ineligibility

Original Payment

Cash

Credit Card or Check

Process cash refund

Process refund through
original payment method

Update inventory for return

Print return receipt

End return process
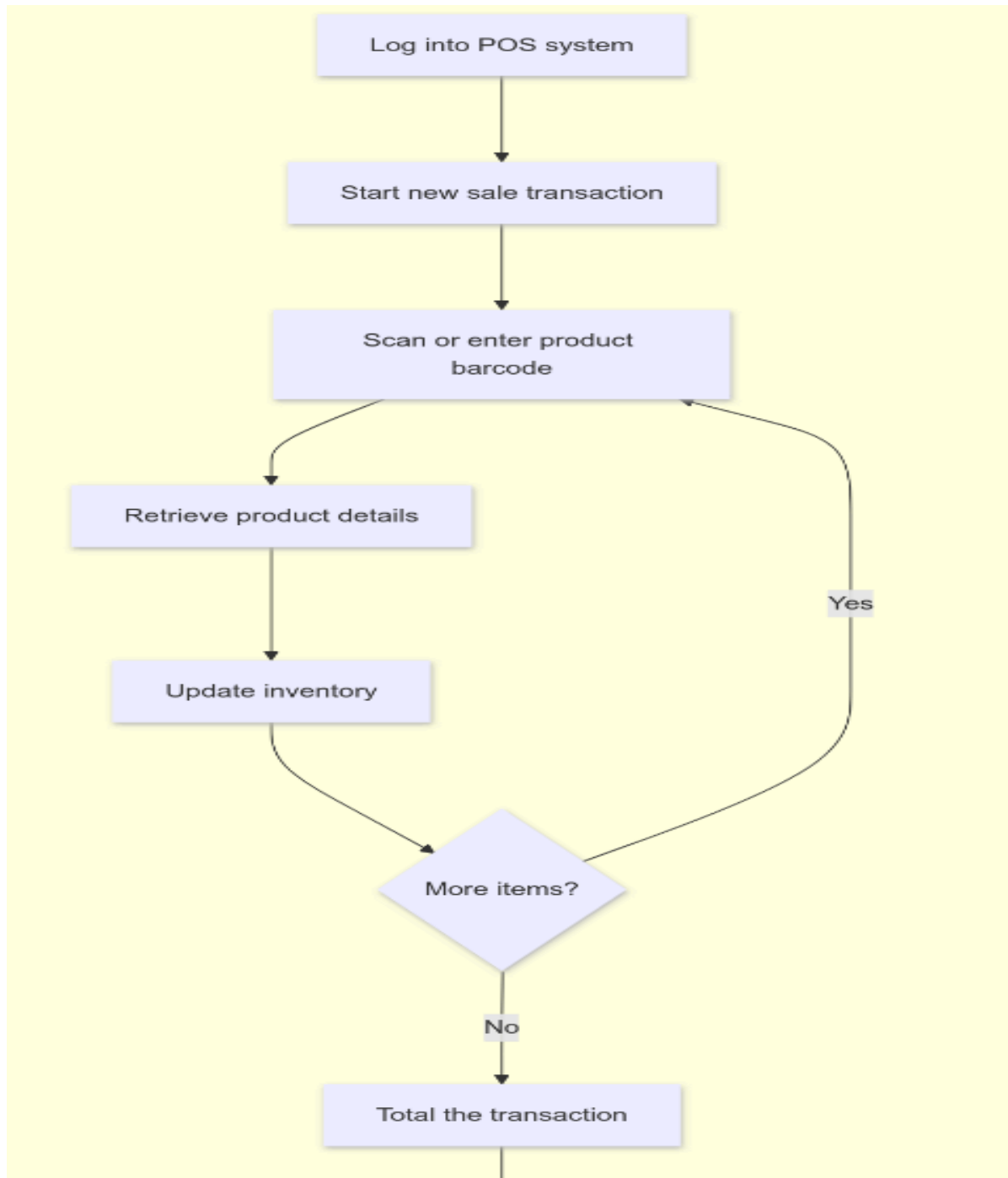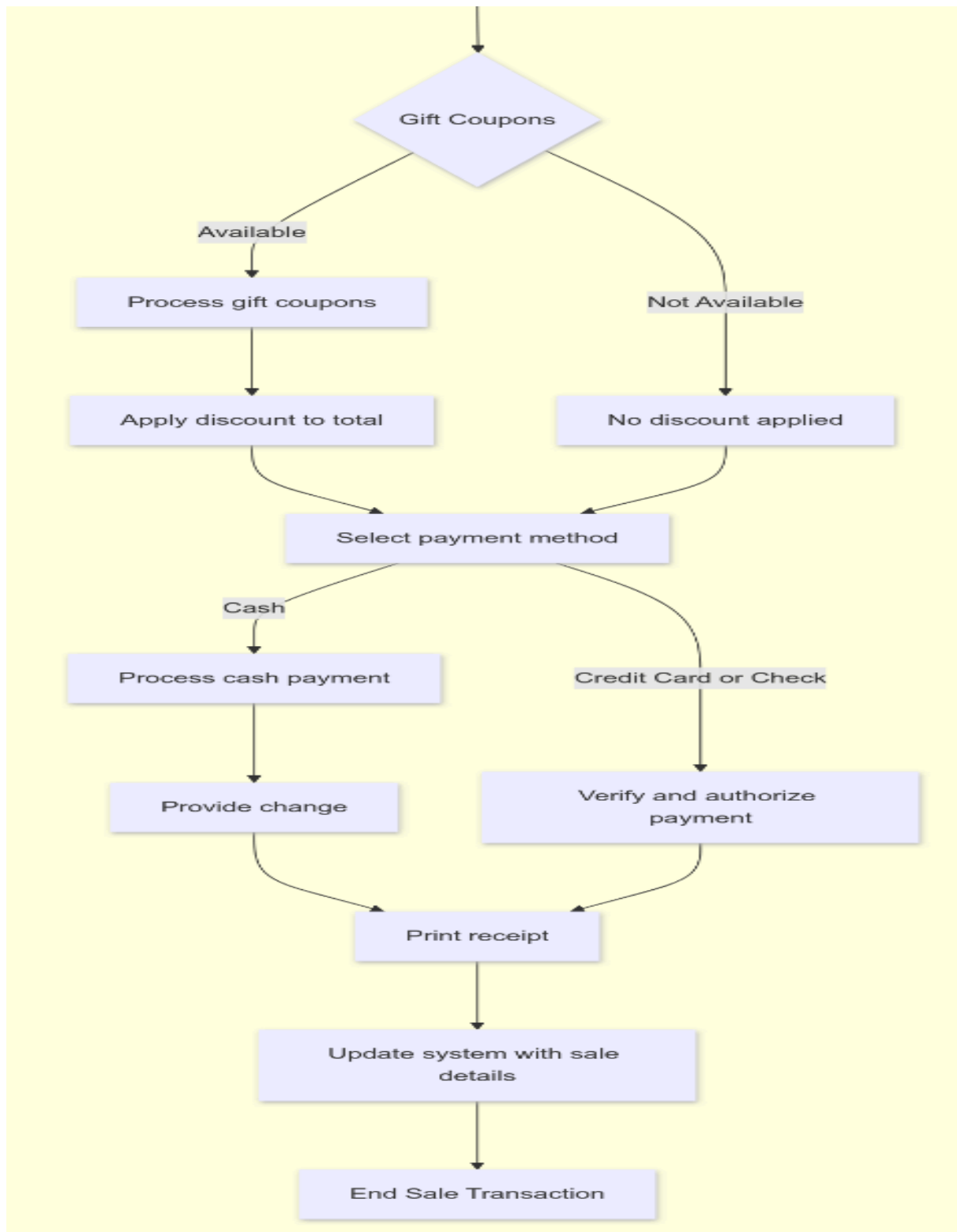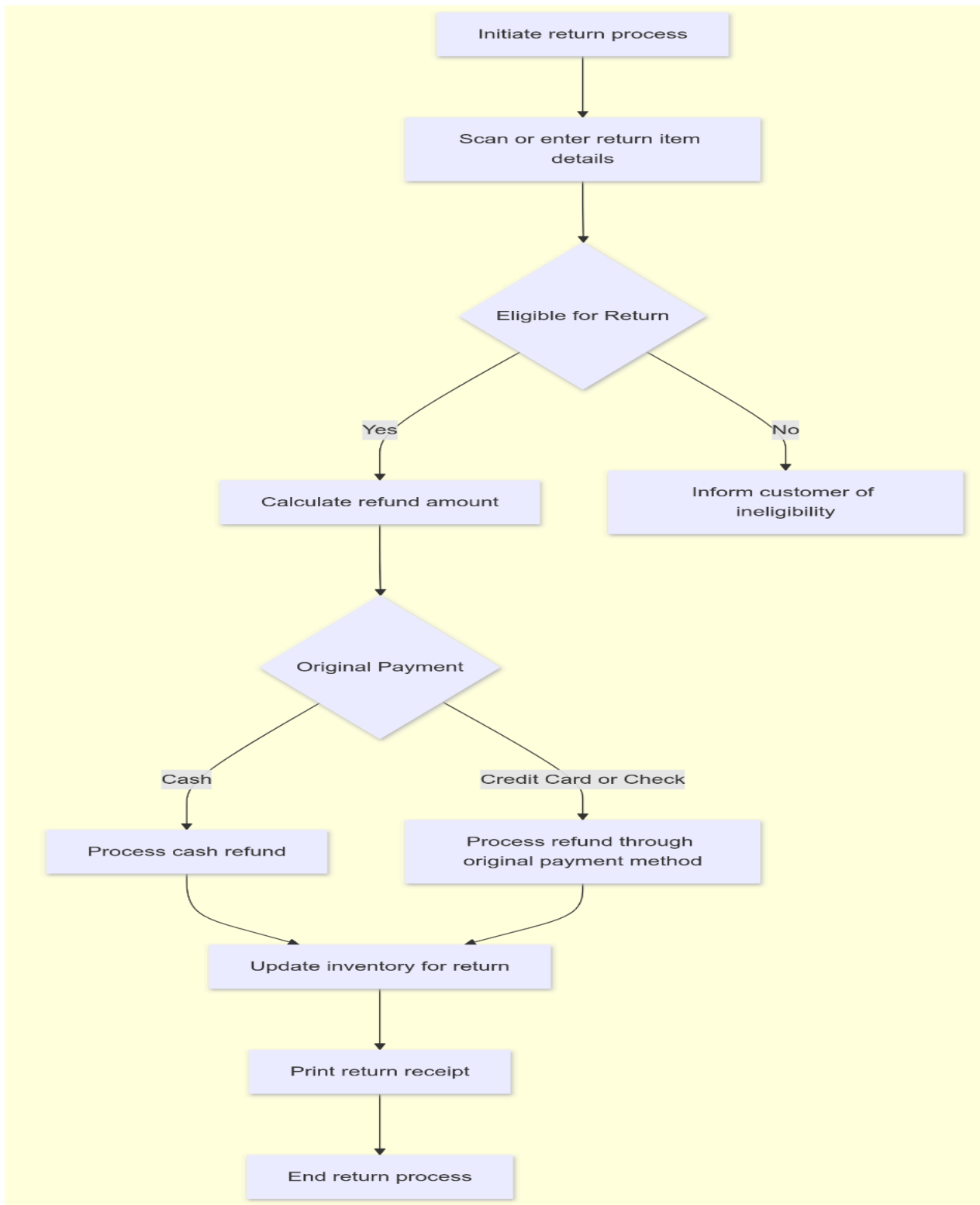
Q.5 Develop Analysis Domain Model

- Process Sales

Customer
- customerID: String
+ purchaseItems(): void

Inventory
- stockLevels: Map<Product, int>
+ checkStock(Product): bool
+ updateStock(Product): void

1..*
Initiates
0..*

0..*
Monitors
0..*

SaleTransaction
- transactionID: String
- totalAmount: Double
+ addProduct(product: Product): void
+ finalizeTransaction(payment: Payment): void

1..*———Constains———0..*

Product
- id: String
- name: String
- price: Double
- stock: int
+ getDetails(): void
+ updateStock(): void

1..1
Associated With
1..1

1..1

Generates

1..*
Provides Info for
0..*

Payment
- paymentID: String
- amount: Double
- method: PaymentMethod
+ processPayment(): bool

Catalog
- productList: List<Product>
+ getProductDetails(Product): void

1..1

1..1
Generates———1..1→

Receipt
- receiptID: String
+ printReceipt(): void

● Handle Return

SaleTransaction

- transactionID: String
- totalAmount: Double

+ addProduct(product: Product): void
+ finalizeTransaction(payment: Payment): void

Inventory

- stockLevels: Map<Product, int>

+ checkStock(Product): bool
+ updateStock(Product): void

0..*
Moniters
0..*

1..1
Linked to
1..1

ReturnTransaction

- returnID: String
- returnReason: String
- isApproved: bool

+ verifyReturn(SaleTransaction): bool
+ processReturn(): void

1..* Contains 1..*

Product

- id: String
- name: String
- price: Double
- stock: int

+ getDetails(): void
+ updateStock(): void

1..1
Has
1..1

Refund

- refundID: String
- refundAmount: Double

+ issueRefund(): void

1..*
Has
1..1

StorePolicy

- policyID: String
- returnTimeLimit: int

+ checkEligibility(Product): bool

0..*
Initiates
1..*

1.1
Issued to
1..1

Customer

- customerID: String

+ purchaseItems(): void