IT-314
Rakshit Pandhi - 202201426

# Lab Session: Modeling Class Diagram and Activity Diagram (Point of Sale System):

**Question - Develop Use Case Textual Description for "Process Sale" and "Handle Return" use cases.**

**Answer-**

## Use Case Development - Process Sales

Name - Process Sales

Actors - Cashier, catalog system, inventory system.

Goals - The cashier should be able to handle sales.

Pre-conditions -

- The cashier is already logged in to the system.
- The customer has arrived at the counter with the goods.

Trigger -

- The cashier starts a new sale transaction by clicking a button or so.

Main Flow -

1. The cashier scans the goods one by one by using the barcode scanner.
2. The catalog system would display the name, price, and other details on the screen.
3. The inventory system deduces the stock amount of the goods in the backend.
4. Once the scanning is done the total amount is calculated.

5. The cashier asks the customer about the mode of payment (cash, credit card, check).
6. The customer makes the payment.
7. The cashier confirms the payment and generates the receipt.

Alternate Flow -

1a.

- The barcode scanner may be malfunctioning.
- The cashier may need to manually enter the goods.

2a.

- The catalog system may not show the details.
- The cashier must contact the admin regarding the flaw.
- Admin must handle the issue.

3a.

- The inventory system may show the wrong count of the good(issue when count 0 but still the customer is there with the good),
- The cashier must contact the admin regarding the flaw.
- Admin must be able to handle the issue.

5a.

- Before making the payment the customer may ask for a discount by using the gift coupon.
- Hence the cashier should scan the coupon code and update the price of the total bill.

6a.

- Payment may fail, so the cashier should check thoroughly if the amount is debited/credited or not before allowing the customer to leave.

Post-Conditions -

- The cashier has processed the sale of the goods purchased by the customer.

Technologies needed -

- Computer
- Printer
- Barcode scanner
- Software Interface

## Use Case Development - Handle Returns

Name - Handle returns

Actor - Cashier.

Goal - Customers should be able to return certain goods and the cashier should handle it.

Pre-Conditions -

- The customer has already purchased the goods he/she wants to return, along with the receipt (original).
- The cashier has logged in to the system.

Trigger -

- The cashier handles the return by clicking the “return good” button or so.

Main Flow -

1. The cashier asks the customer for the original receipt and the items being returned.
2. The system verifies the receipt and initiates the return process.
3. The cashier scans the barcode of each item being returned.

4. The system retrieves the original sale price of each item and informs the cashier of the total refund amount.
5. The cashier confirms the return with the customer.
6. The customer selects a refund method (cash, credit card).
7. The cashier provides the refund amount to the customer.

Alternate Flow -

1a.

- Customers may not have the original receipt.
- The cashier shouldn't proceed with the return in such a case.

2a.

- The receipt may need to be verified.

3a.

- The cashier isn't able to scan the goods.

4a.

- The price at which the customer bought the goods and the refund price may differ.
- The cashier must address the issue by contacting the admin or so.

7a.

- Payment may fail.
- The cashier may ask the customer for any other alternate payment method.

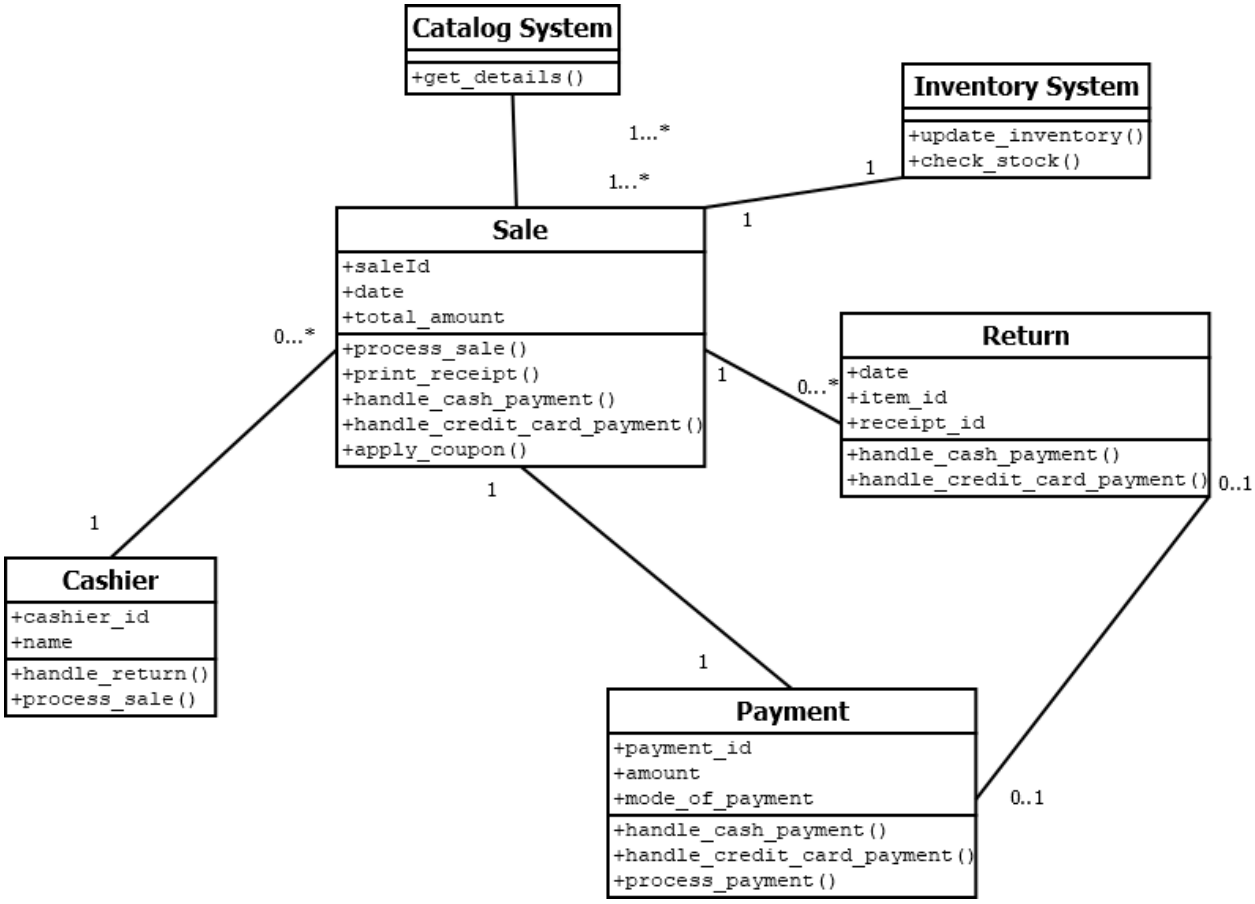
Post-Conditions -

- The customer must be able to return the goods and the cashier handles it and returns the refund amount.

Technologies needed -

- Computer
- Printer
- Barcode scanner
- Software Interface

## Question  - Develop Analysis Domain Models




Assumptions regarding the multiplicities -

### Cashier ↔ Sale

- **Multiplicity**:
    - One `Cashier` can process **multiple** `Sale` instances, but each `Sale` is processed by exactly **one** `Cashier`.
    - `Cashier` (1) ↔ `Sale` (0..*)

**Sale ↔ Payment**

- **Multiplicity**:
    - One `Sale` can be linked to exactly **one** `Payment`.
    - A `Payment` can be used for **one** `Sale`.
    - `Sale` (1) ↔ `Payment` (1)

**Sale ↔ Return**

- **Multiplicity**:
    - One `Sale` can have **zero or more** `Return` instances.
    - Each `Return` is linked to exactly **one** `Sale`.
    - `Sale` (1) ↔ `Return` (0..*)

**Sale ↔ Catalog System**

- **Multiplicity**:
    - A `Sale` can retrieve details from **one or more** product catalogs, while the `Catalog System` services **many** `Sales`.
    - `Sale` (1..*)* ↔ *`Catalog System`* (1..*)*

**Sale ↔ Inventory System**

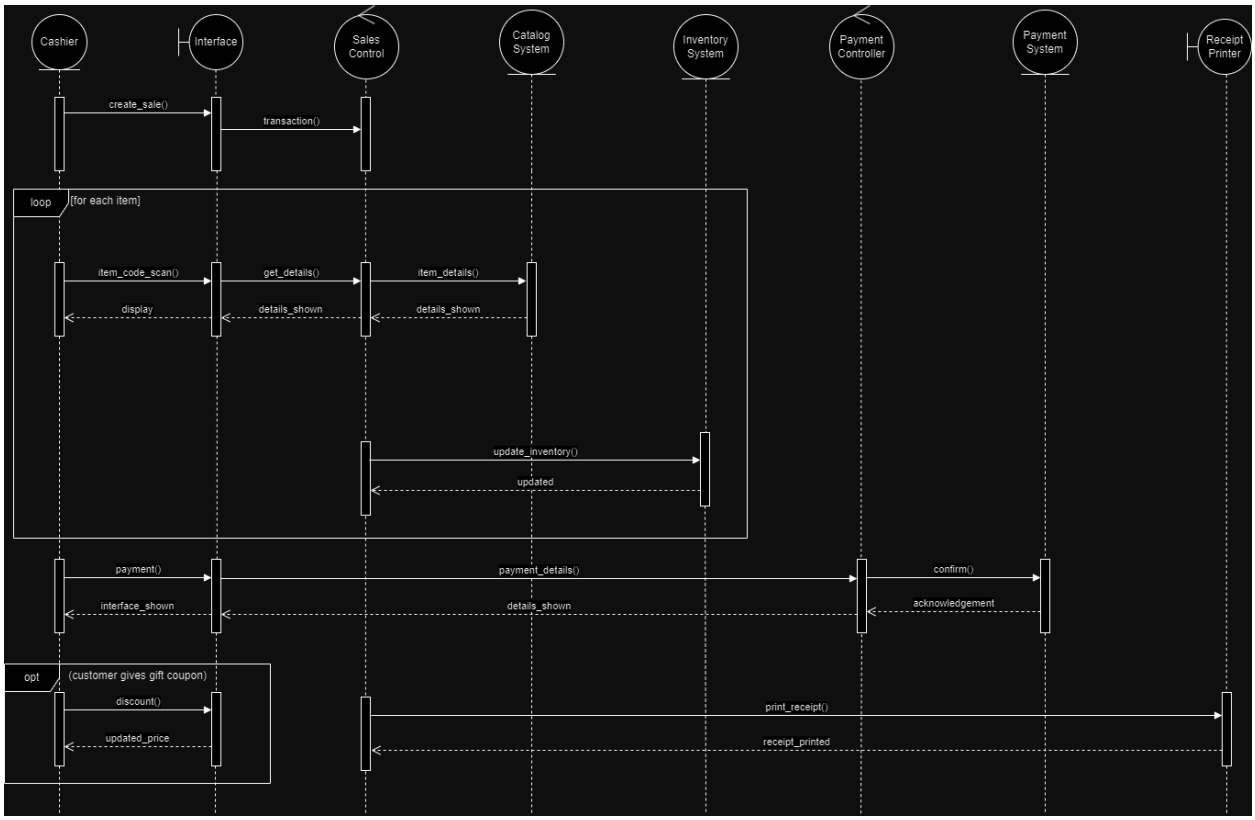
- **Multiplicity**:
    - Each `Sale` interacts with the `Inventory System` once.
    - `Sale` (1) ↔ `Inventory System` (1)

**Return ↔ Payment**

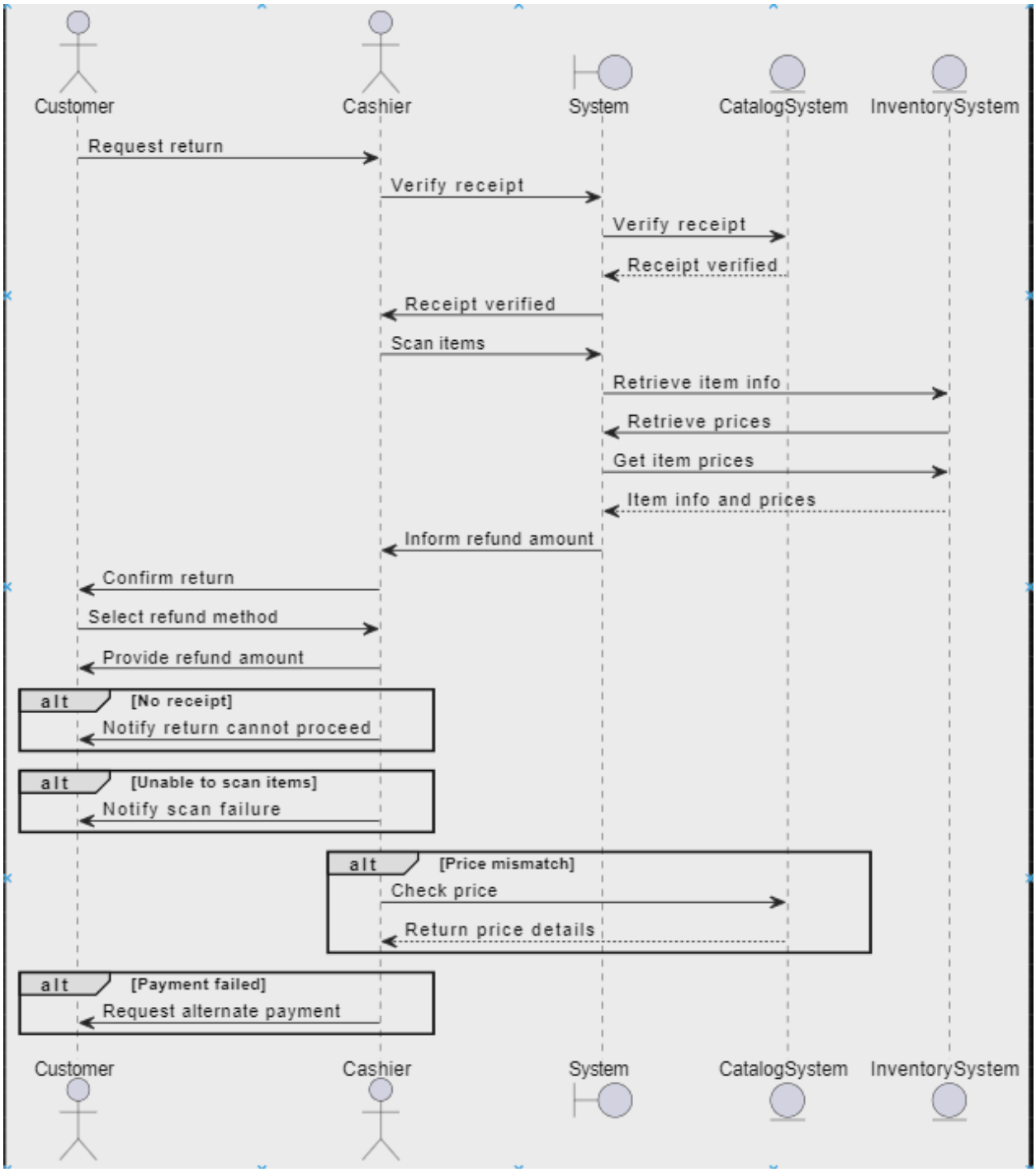
- **Multiplicity**:
    - One `Return` may result in **one** `Payment` for the refund.
    - `Return` (0..1) ↔ `Payment` (0..1)

## Question  - Develop Sequence Diagrams

Process Sale Use Case -

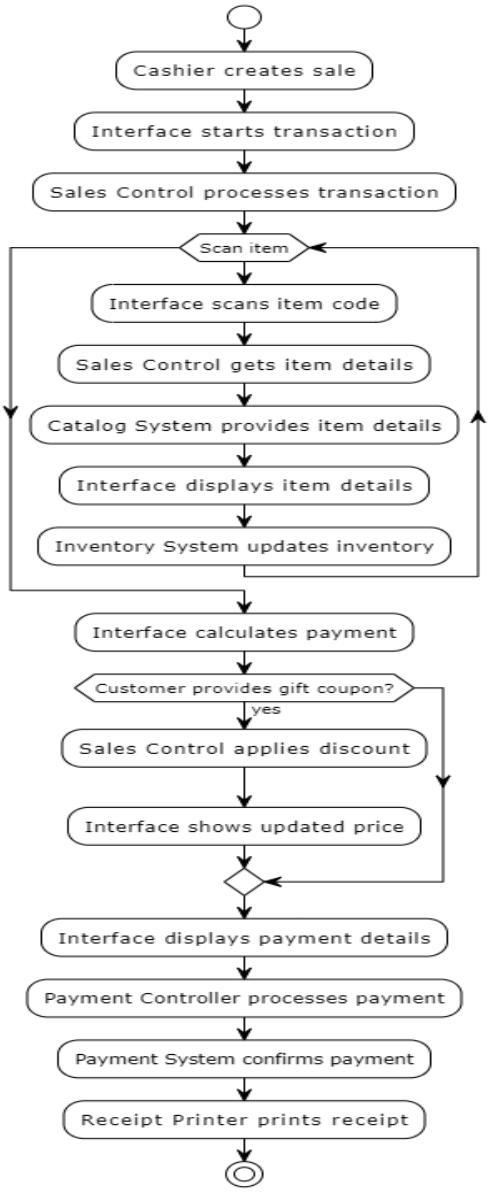


Handle Return Use Case -



## Question  - Develop Activity Diagrams for Process Sale and Handle Return

Process Sale -



Handle return -