Analyze the relationships between Sales, Purchases, and Inventory to identify core system behaviors:

Before writing test cases for the Sales, Sales Order, Purchase, Purchase Return, and Product modules, I need to understand how these modules interact with each other. These relationships form the core logic of any inventory management system.

Relationship Between Sales and Inventory:

- ❖ When a product is sold (Sales Order is created), its quantity decreases from the inventory.
- ❖ If a product is out of stock, a Sales Order should not be processed unless there is a back-order option.
- ❖ Discounts, taxes, promotions, and payment conditions are essential in Sales but may not directly affect inventory.
- ❖ If a Sales Order is canceled, the product should be restored into inventory.

- ◆ Example:

1. Initial Inventory → 50 units of a product
2. A Sales Order is created for 5 units
3. Updated Inventory → 50 - 5 = 45 units

Relationship Between Purchases and Inventory:

- ❖ When a product is purchased (Purchase Order is created) and received, its quantity increases in the inventory.
- ❖ If a product is out of stock and a Sales Order is placed, the system should create a Purchase Order to restock the product.
- ❖ A Purchase Order updates inventory once the supplier delivers the items.
- ❖ If a Purchase Return occurs, the inventory should decrease accordingly.

- ◆ Example:

1. Initial Inventory → 45 units
2. A Purchase Order is placed for 20 units
3. Updated Inventory → 45 + 20 = 65 units

Combined Relationship Between Sales, Purchases, and Inventory:

- ❖ Purchase Orders add new stock to the inventory.
- ❖ Sales Orders reduce stock from the inventory.
- ❖ Sales Returns bring stock back into inventory.
- ❖ Purchase Returns decrease stock from inventory.

- Example:

1. Initial Inventory: 50 units
2. Sales Order: 5 units sold → New Inventory: 45
3. Purchase Order: 20 units purchased → New Inventory: 65
4. Purchase Return: 5 units returned to the supplier → New Inventory: 60
5. Sales Return: 2 units returned by a customer → New Inventory: 62

Core Functionality of Modules and Test Cases:

1. Sales Module Core Functionality:

The Sales module handles product purchases, edit order, discount applications, order cancelation, and payment processing.

2. Sales Orders Module Core Functionality:

The Sales Orders module manages placed orders, including add product with Sales order, order details, search, filtering, and exporting order information.

3. Purchase Module Core Functionality:

The Purchase module facilitates the procurement of products from suppliers, including purchase order creation, order approval, and inventory updates.

4. Purchase Return Module Core Functionality:

The Purchase Return module manages the return of purchased products due to defects or other reasons, verify confirm the purchase return and refund processing.

5. Product Module Core Functionality:

The Product module manages product catalog information, including adding, updating, deleting, and categorizing products.

Thought Process & Approach for Writing Test Cases:

When writing test cases for different modules, I followed a structured approach to ensure comprehensive coverage of the system's functionality. Below is my thought process:

1. Understanding the Core Functionality:

Before writing test cases, I first defined the core functionality of each module:

Sales Module: Focuses on product purchasing, order placement, and payment.

Sales Orders Module: Handles order tracking, status updates, and order history.

Purchase Module: Manages procurement, supplier orders, and inventory updates.

Purchase Return Module: Deals with product returns and refund processing.

Product Module: Manages adding, updating, and deleting products in the system.

Defining the core functionality helps in identifying critical areas that need testing.

2. Categorizing Test Scenarios:

For each module, I created test cases based on:

*Positive Scenarios (expected workflows)

*Negative Scenarios (handling invalid inputs, failures, and edge cases)

In the Purchase Module, I included test cases for creating and approving purchase orders.

3. Writing Clear, Step-by-Step Test Cases:

Each test case is structured with:

Test Case ID (unique identifier)

Module (which feature it belongs to)

Test Scenario (brief description)

Test Steps (actions to execute)

Expected Result (what should happen if the system works correctly)

Actual Result (to be filled during execution)

Status (Passed/Failed)

Priority (High, Medium, Low)

Remarks (additional notes)

This format ensures that testers can easily follow and execute each test case.