Use Case

## **Instructions**

📌 **What You Need to Do:**

* Write **test cases** for each of the following modules:
  + **Sales**
  + **Sales Order**
  + **Purchase**
  + **Purchase Return**
  + **Product**
* In the Google Drive folder, you will find a file named **"Sample\_Test\_Cases.xlsx"**—please use this as a reference for formatting your test cases.

📌 **Important Notes:**

* **Submission Deadline:** **28th** February 2025. **(10 days)**
* During testing, do not focus on **UI**. Instead, try to analyze the **relationships** between **Sales**, **Purchases**, **and Inventory** to identify core system behaviors.
* **Try to define the core functionality** of each module before writing the test cases.
* **Share your thought process** – Explain your approach while writing the test cases.
* Ensure your submission follows the provided instructions to be considered for the next round.

**Additional Notes:**

* **Be Thorough:** Test all possible scenarios, including invalid inputs and edge cases.
* **Be Detailed:** Provide as much detail as possible in your reports to help us understand and resolve issues quickly.

### **Ask Questions:** If you are unsure about any step or encounter something unexpected, feel free to reach out to me for clarification.

### **Product Creation Use Case:**

**Goal:** Create and manage products in the system, including calculating and updating the **Average Unit Price** based on purchases and sales.

### **Basic Flow (Step-by-Step):**

#### **1. Start Product Creation**

* **Action:**
  + Go to **Products → New Product**
  + The system opens a blank product creation form.

#### **2. Enter Product Details**

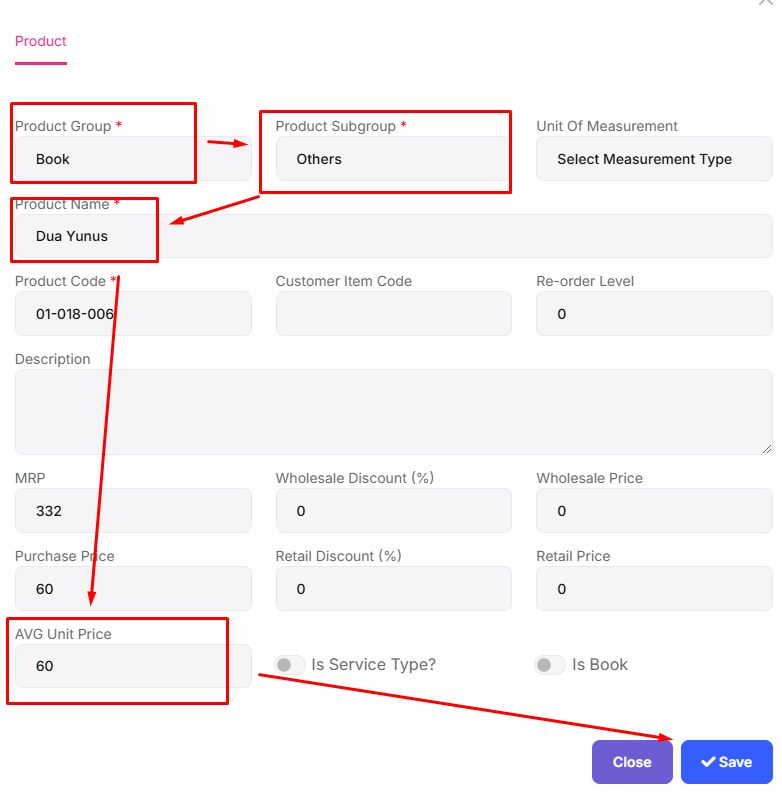
* **Action:**
  + Enter basic product details:
  + Save the product.
* **Outcome:**
  + The product is created with an initial **Average Unit Price** of **0** (since no purchases or sales have been recorded yet).

#### **3. Update Average Unit Price via Purchases**

* **Action:**
  + When a **purchase** is made for the product:
    - The system calculates the new average unit price using the formula:
* Example:
  + Existing stock: 10 units at $500/unit.
  + New purchase: 5 units at $550/unit.
  + New average unit price = ((10 \* 500)+(5∗500)+(5∗550)) / (10 + 5) = $516.67.
* **Outcome:**
  + The product's **Average Unit Price** is updated in the system.

### **Summary Workflow:**

**Create Product → Enter Details → Save → Update Average Unit Price via Purchases**

****

### 

### 

### 

### 

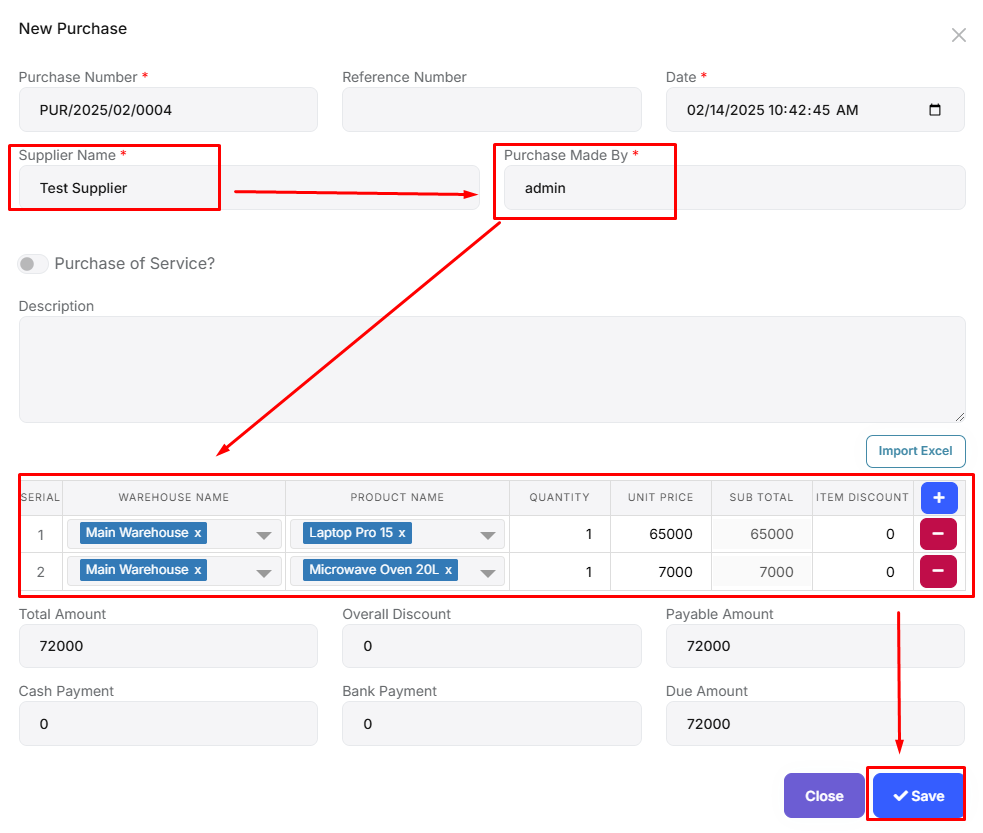
### **Purchase Use Case: Step-by-Step Use Case for Purchase:**

#### **1. Create a Purchase Order**

* **Steps:**
  + Go to **Purchase** → **New Purchase**.
  + **Select Supplier:**
    - If the supplier doesn’t exist:
      * Create a new supplier (Name, Contact, Address).
      * Link to the pre-created supplier account (e.g., "Supplier XYZ").
  + **Add Products:**
    - Search and select products from the list.
    - Enter quantity and price (e.g., 10 units @ $500 each).
  + **Payment:**
    - Select payment mode:
      * **Cash Payment**
      * **Bank Payment**
  + Save Purchase Order.
* **Outcome:**
  + Inventory stock increases (e.g., Samsung Galaxy S23: 0 → 10 units).
  + A **Purchase Voucher** is auto-generated with details.

#### **2. Edit/Delete Purchase Order**

* **Editing a Purchase:**
  + Update product quantity (e.g., change from 10 → 15 units).
  + **Outcome:** Stock increases by +5 units.
* **Deleting a Purchase:**
  + **Outcome:** Stock decreases by the original purchased quantity (e.g., 10 → 0 units).



### 

### 

### 

### 

### 

### 

### 

### 

### **Sales Use Case:**

**Goal:** Sell products to customers, update inventory, track payments, and generate sales vouchers.

### **Basic Flow (Step-by-Step):**

#### **1. Start a New Sale**

* **Action:**
  + Go to **Sales → New Sale**.
  + The system opens a blank sales form.

#### **2. Select or Create a Customer**

* **Action:**
  + Search for an existing customer (e.g., "Customer ABC").
  + **If a customer does NOT exist:**
    - Click **"Create New Customer"**.
    - Enter customer details: Name, Phone, Address.
    - Link to the **Customer Account Group** (e.g., "Customers" under "Accounts Receivable").
  + **If a customer exists:**
    - Select the customer from the list.
* **Outcome:** Customer is linked to the sale.

#### **3. Add Products to the Sale**

* **Action:**
  + Search for products (e.g., "Samsung Galaxy S23").
  + Select the product from the inventory list.
  + Enter **quantity** to sell (e.g., 2 units).
* **Outcome:** Products added to the sale.

#### **4. Set Payment Method**

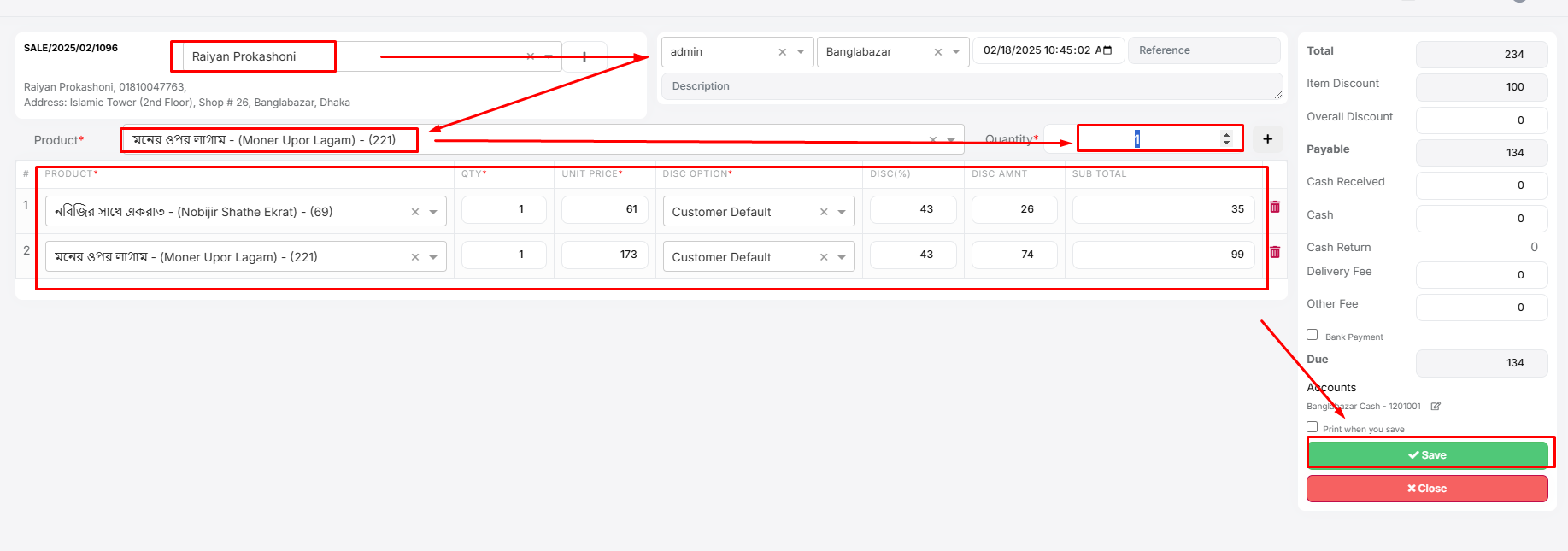
* **Action:**
  + Select payment mode:
    - **Cash Account**
    - **Bank Account**
  + Enter the amount received (if partial/full payment).
* **Outcome:** Payment details saved.

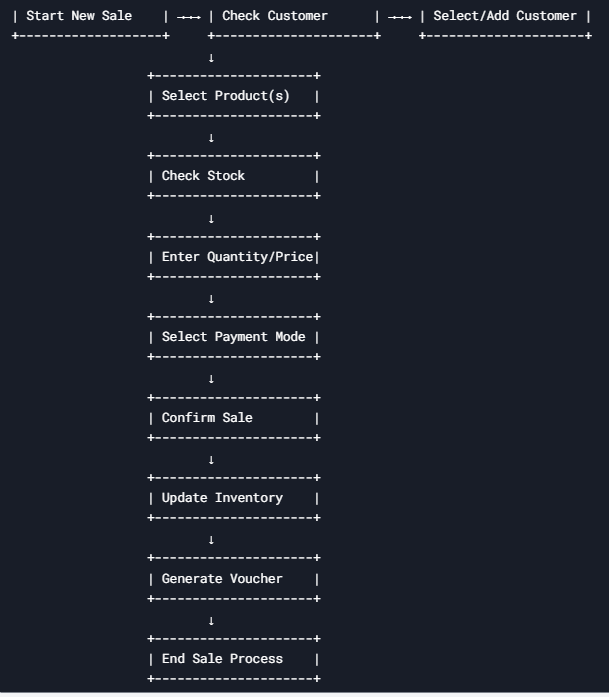
#### **5. Confirm the Sale**

* **Action:**
  + Click **"Save"**.
  + The system validates:
    - Stock availability.
    - Customer and payment details.
* **Outcome:**
  + Inventory stock **decreases** (e.g., Samsung Galaxy S23: 10 → 8 units).

### **Summary Workflow:**

**Select/Create Customer → Add Products → Payment → Confirm → Update Stock & Accounts → Generate Voucher**





### 

### 

### **Purchase Return Use Case:**

**Goal:** Handle product returns to suppliers, update inventory, adjust payments, and generate purchase return vouchers.

### **Basic Flow (Step-by-Step):**

#### **1. Start a New Purchase Return**

* **Action:**
  + Go to **Purchases → Purchase Returns → New Return**.
  + The system opens a blank purchase return form.

#### **2. Select the Original Purchase**

* **Action:**
  + Search for the original purchase
  + Select the purchase from the list.
* **Outcome:**
  + The system loads the original purchase details

#### **3. Select Products to Return**

* **Action:**
  + From the loaded purchase, select the products to return (e.g., "Samsung Galaxy S23").
  + Enter the **quantity** to return (e.g., 5 units).
  + The system validates:
    - **If the quantity is valid:** Proceed.
    - **If the quantity exceeds the original purchase:** Show the error.
    - **Outcome:** Products added to the return cart.

#### **4. Set Refund Method**

* **Action:**
  + Select refund mode:
    - **Cash Account**
    - **Bank Account**
  + Enter the amount to refund (if partial/full refund).
* **Outcome:** Refund details saved.

#### **5. Confirm the Purchase Return**

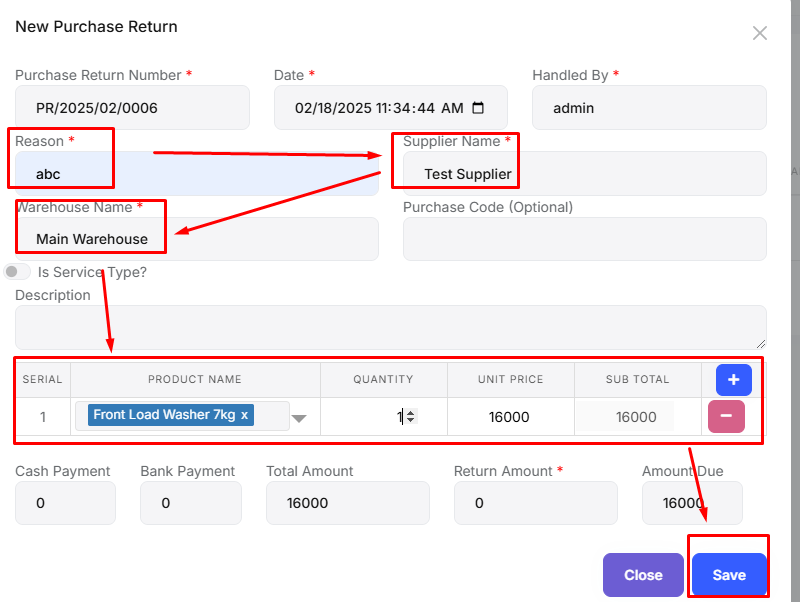
* **Action:**
  + Click **"Confirm Return"**.
* **Outcome:**
  + Inventory stock **decreases** (e.g., Samsung Galaxy S23: 50 → 45 units).

#### **Edit the Return**

* **Action:**
  + Go to **Purchases → Purchase Returns → Edit Return**.
  + Adjust product quantity (e.g., change from 5 → 3 units).
  + **Outcome:**
    - If quantity increases: Stock decreases further (45 → 42 units).
    - If quantity decreases: Stock increases (45 → 48 units).

### **Summary Workflow:**

**Setup → Select Original Purchase → Add Return Products → Refund → Confirm → Update Stock & Accounts → Generate Purchase Return Voucher**





### 

### 

### 

### 

### **Sales Order Use Case:**

**Goal:** Allow customers to create sales orders for products without updating inventory or generating vouchers.

### **Basic Flow (Step-by-Step):**

#### **1. Create a New Sales Order**

* **Action:**
  + Click **"New Sales Order"**.
  + The system opens a blank sales order form.

#### **2. Add Products to the Sales Order**

* **Action:**
  + Search for products (e.g., "Samsung Galaxy S23").
  + Select the product from the list.
  + Enter **quantity** (e.g., 2 units).
* **Outcome:** Products added to the sales order.

#### **3. Submit the Sales Order**

* **Action:**
  + Review the sales order details (products, quantities, total amount).
  + Click **"Submit Order"**.
* **Outcome:**
  + The sales order is saved in the system.
  + No inventory or stock is updated.
  + No voucher is created.

#### **Case: Customer Edits the Sales Order**

* **Action:**
  + Go to **Sales → My Orders → Edit Order**.
  + Adjust product quantity or add/remove products.
  + Click **"Update Order"**.
* **Outcome:**
  + The sales order is updated with the new details.

### **Summary Workflow:**

**Login → Create Sales Order → Add Products → Submit → Save Order->Sales Order List**

