WORKSHOP #8 - SELF CHECKOUT

## APS145NFF - Group 2 Members:

* Natsita - Logic 1 - Pseudocode
* Joshua Civello - Logic 1 - Flowchart
* Seulgi Lee - Logic 2 - Pseudocode
* Md Ridwan Hossain - Logic 2 - Flowchart
* Luca Novello - Logic 3 - Pseudocode
* Luca Novello - Logic 3 - Flowchart

## Workshop Overview

**Black boxes** are widely used and referred to in the world of programming. They represent specific processes that are hidden from the consumer (end-user, programmer, or conceptually by management etc.). In the programming sense, this can be a large-scale process (a system or application) or be as minute as a function (such as a shared Application Program Interface [API], or a web-service). There are two other types; **Grey Boxes** where some knowledge of how the process works is known and **White Boxes** where all the logic is visible/known by the consumer.

In the case of “**functions**”, each function should be limited to a very specific task. Although not necessary, most of the time functions will require some input information and use that input to produce some meaningful output (or result). This workshop is focused on function-based **Black Boxes** and **White Boxes** that do a very specific task and are called from other processes as required which will simplify the overall logic. The **Black Boxes** will be briefly described for you so you will know what functions exist and can implement into your solution and where you will not have to provide an algorithm for, while the **White Boxes** will be the sub-processes you must fully define.

**Recap**

**Black Boxes** are those that you do **NOT** have knowledge of **how it works** – you only know that it should accomplish some meaningful output based on the input you provide it.

**White Boxes** are those that you have **full knowledge** of **how it works** because you will be defining those subprocesses and exposing all the logic (the algorithm) accordingly (ex: Logic-1, Logic-2, and Logic-3 components and any additional supportive subprocesses you may want to add).

**Black** and **White Boxes** can be called from anywhere in your defined processes – these are merely functions that will perform specific tasks to streamline/simplify the overall algorithm.

## Workshop Details

You need to create a **software kiosk application** that will be used in a grocery store **self-checkout machine**. You need to define a solution that will provide a customer the means to **process the items** from their shopping cart (**perishable and non-perishable products**), through to the **payment for the items** that will include an itemized receipt (**total breakdown of all the items**).

# 

# DATA STRUCTURES

| Product | |
| --- | --- |
| sku | *Unique product identifier (****Perishable****) or Barcode (****Non-perishable****)* |
| name | *Product name* |
| category | *Product category (ex. fruit, meat, bakery…)* |
| price | *Price Per Unit (****Quantity****) or Price Per Kilogram (****Weight****)* |
| taxRate | *Tax rate for product (****0% for*** *N****on-Taxed****)* |
| isPerishable | *Returns* ***True if Perishable*** *or* ***False if Non-Perishable*** |

| Transaction | |
| --- | --- |
| perishableProducts | *An array of all* ***Perishable*** *products* |
| nonPerishableProducts | *An array of all* ***Non-Perishable*** *products* |
| subtotal | ***Subtotal*** *cost of all items* |
| taxTotal | ***Product price \* taxRate*** *for all items* |
| total | ***Subtotal + taxTotal*** *cost of all items* |

# PSEUDOCODE:

## Main Pseudocode:

1. Start
2. Create **Transaction.**
3. Is the product perishable?
   1. **Yes**: **(go to Logic 2)**
   2. **No**: **(Step 4)**
4. Is the product non-perishable?
   1. **Yes**: **(go to Logic 1)**
   2. **No**: **(Step 5)**
5. Is there another product?
   1. **Yes**: **(Step 3)**
   2. **No**: **(Step 6)**
6. Does the customer want to cancel the order?
   1. **Yes**: **(End/Restart)**
   2. **No**: **(Step 7)**
7. Is the customer ready to pay?
   1. **Yes**: **(go to Logic 3)**
   2. **No**: **(Step 6)**
8. **Print** and **dispense** Itemized **receipt**.
9. End

## Logic 1 - Pseudocode:

1. Start

2. Import **product, Transaction**

3. Initialize **product.price**

4. Does **product.isPerishable** return true?

a. Yes, the product is perishable (go to Logic 2)

b. No, the product is non-perishable (continue #4)

5. Scan the item

6. Is the barcode scanned successfully?

a. No, the button "Need Help/Assistance" needed to be pressed

i. If the button is pressed, send a notification to the store staff to assist the customer

b. Yes, continue #5

7. Lookup product info and price from the product database (Black box)

8. Is the item found in the product database?

a. No, display "Item not found. Please ask for assistance from a staff member."

b. Yes, continue #7

9. Store the product price in **product.price**

10. Add **product** to **Transaction.nonPerishableProducts**

11. Return/Output **Transaction**

12. End

## Logic 2 - Pseudocode:

1. Start/Previous Logic.
2. Import & store **Product, Transaction.**
3. If the price of the item is determined by **weight**?
   1. **Yes: (Step 4)**
   2. **No: (Step 7)**
4. Weight item on scale
5. Calculate cost = weight \* product.price(Weight)
6. Add cost, product into Trasaction.perishabeProducts.
7. If the item is priced by **quantity**?
   1. **Yes: (Step 8)**
   2. **No : (Step 12**)
8. Ask the user to enter the quantity of the item they want to purchase.
9. Calculate cost = quantity \* product.price(Quantity)
10. Add cost, product into Trasaction.perishabeProducts.
11. Return/Output **Transaction**
12. End/Next Logic.

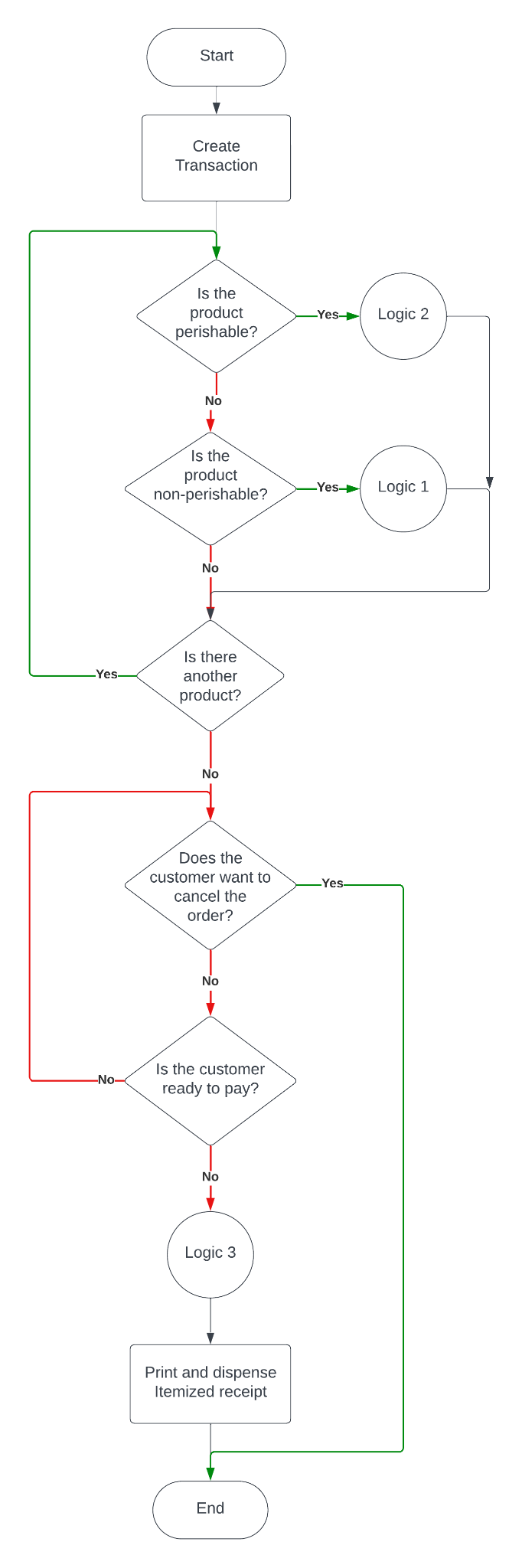
## 

## Logic 3 - Pseudocode:

1. Start/Previous Logic.
2. Import & store **Transaction.**
3. Does the customer want to **cancel** the order?
   1. **Yes**:Clear **Transaction: (End/Restart Logic)**
   2. **No**: loop through **Transaction.nonPerishableProducts**: **(Step 4)**
4. Does **Transaction.perishableProducts** have any items to **loop** through?
   1. **Yes**: **(Step 6)**
   2. **No**: **(Step 5)**
5. Does **Transaction.nonPerishableProducts** have any items to **loop** through?
   1. **Yes**: **(Step 6)**
   2. **No**: **(Step 9)**
6. Add **product.price** to **Transaction.subtotal.**
7. Multiply **product.price** with **product.taxRate** and add to **Transaction.taxTotal.**
8. Are there any more items to **loop** though?
   1. **Yes**: **(Step 4)**
   2. **No**: **(Step 9)**
9. Add **Transaction.subtotal** & **Transaction.taxTotal** to **Transaction.total**
10. Does the customer want to **proceed with payment**?
    1. **Yes**: **Process Payment**: **(Step 11)**
    2. **No**:Clear **Transaction: (End/Restart Logic)**
11. Was payment successful?
12. End/Next Logic.

# FLOWCHARTS:

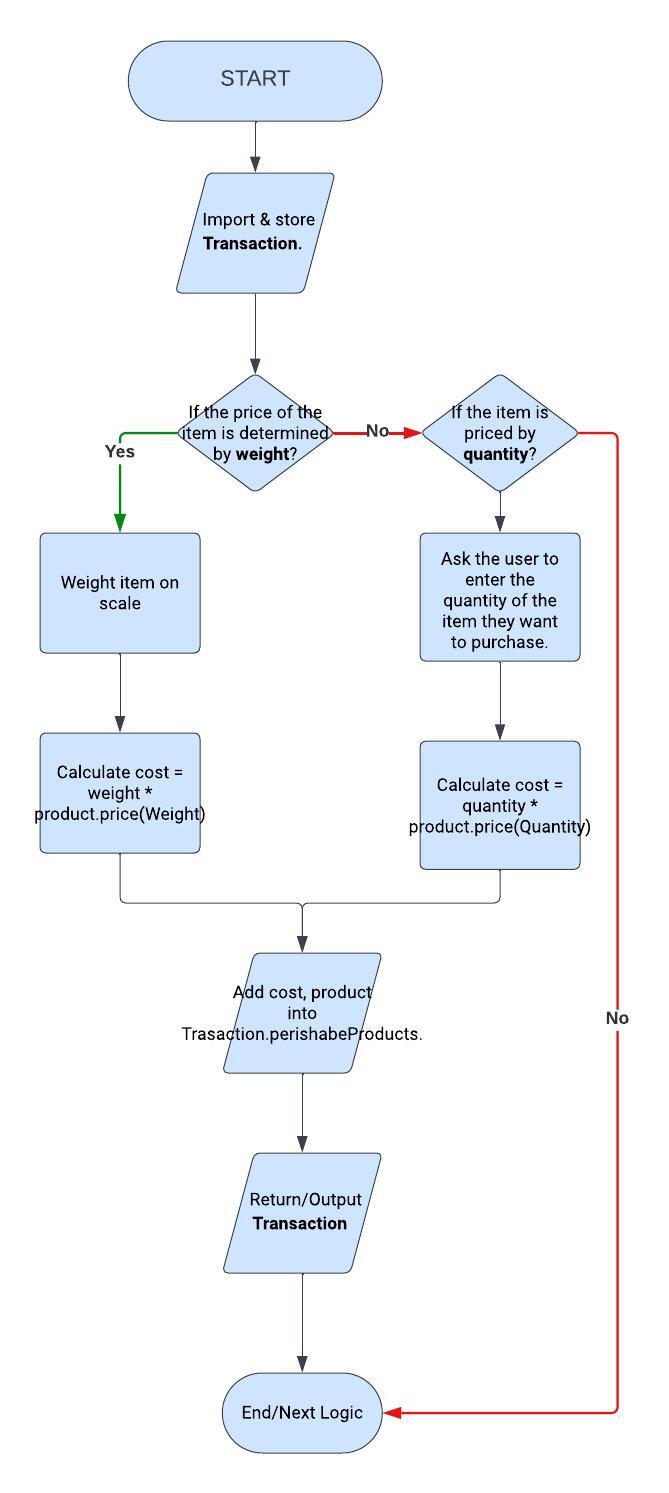
## Main Flowchart:



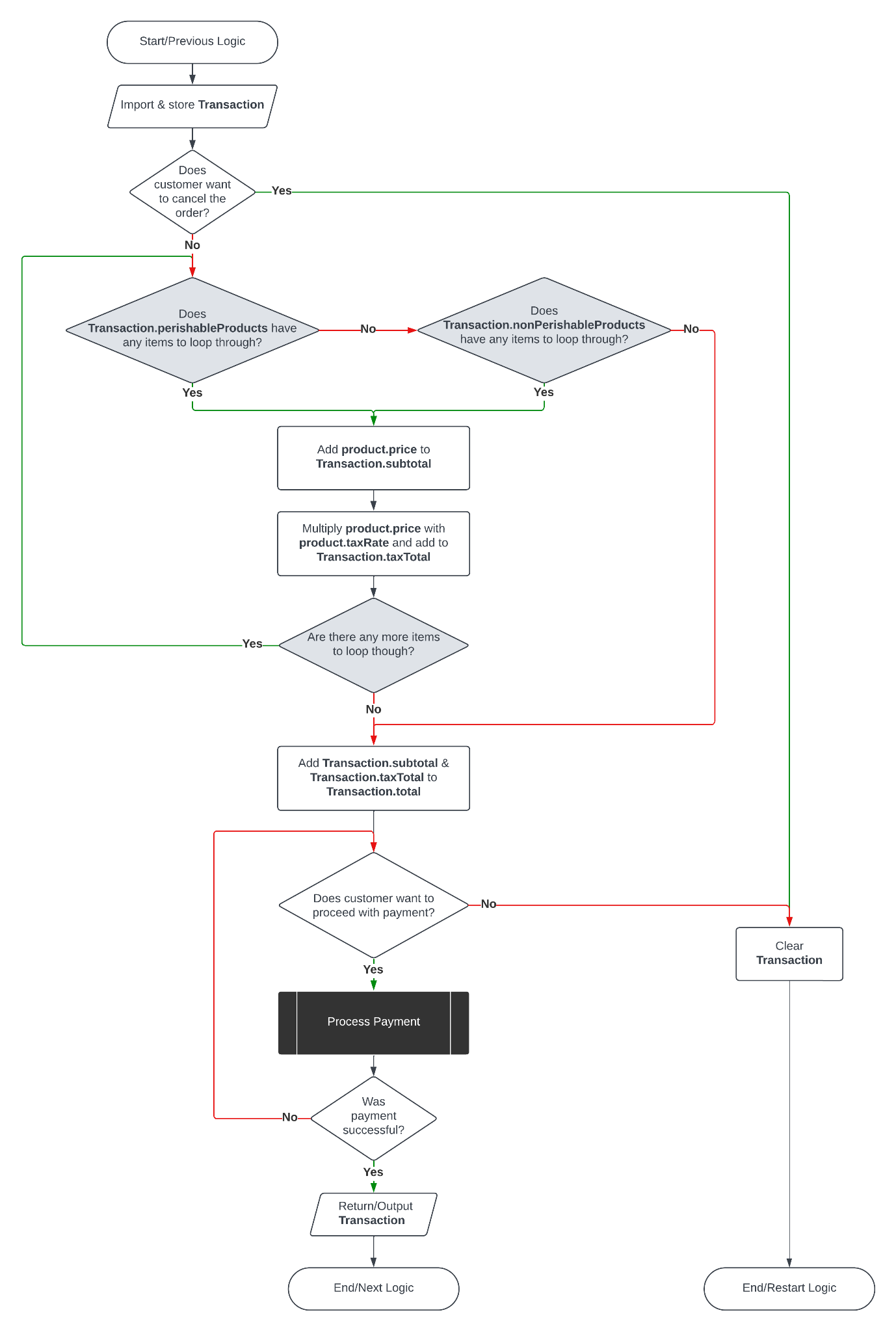
## Logic 1 Flowchart:

## 

## Logic 2 Flowchart:



## Logic 3 Flowchart:



# TEST CASES

**1. Scan/Weigh/Enter Items For Non-Perishable Items:**

**Input:** Scan the barcode for all non-perishable items.

**Expected Output:** Verify that the system scanned the items correctly.

**2. Scan/Weigh/Enter Items For Perishable Items:**

**Input:** Scanning the barcode for all scanned perishable items without barcodes.

**Expected Output:** Make sure the system accepts the input for all perishable items and registers them.

**3. Bagging The Items:**

**Input:** Customer places the items in the bag after scanning or weighing.

**Expected Output:** Ensure that the system allows items to be bagged.

**4. Customer’s Payment Process:**

**Input:** Complete the payment transaction using any of the various payment methods.

**Expected Output:** Verify that the payment process is completed successfully and that a receipt has been generated.

**5. (Black Box) Data Lookup for the Product:**

**Input:** Scan the barcode or enter the product’s code.

**Expected Output:** Check the system to retrieve the correct product information for the database.

**6. (Black Box) Weighing All Perishable Items:**

**Input:** Place all perishable items with no barcode on the weight scale.

**Expected Output:** Ensure the system accurately reads the weight and prices all the scanned items accordingly.

**7. (Black Box) Searching Products by Category:**

**Input:** Search for products within the specific category.

**Expected Output:** Verify that the system receives the correct products based on the specified category.

**8. Weighing All Perishable Items - By Product’s Weight:**

**Input:** Place perishable items on the weight scale.

**Expected Output:** Verify that the system calculates the price based on the items' weight.

**9. Weighing Perishable Items - By Quantity:**

**Input:** Manually input the quantity for perishable items.

**Expected Output:** Ensure the system correctly calculates the price based on the quantity entered.

**10. Integration Testing:**

**Input:** Perform the self-checkout process with all non-perishable and perishable items.

**Expected Output:** Check that all components (scanning, weighing, payment) work seamlessly together.

# FINAL DESCRIPTION

The procedure focuses on an application used in a Self-checkout machine that manages sales of both perishable and non-perishable goods, permits refunds, determines costs according to weight or volume, and makes payments more manageable.

The process starts by creating a transaction and determining whether the product is perishable or non-perishable. Users are presented with a list of unprocessed foods. They select items and provide necessary information (e.g., weight for items priced by weight, quantity for items priced by unit). Prices are calculated accordingly.

Users can continue scanning additional items throughout this process or proceed to the checkout phase. Users can cancel the transaction if necessary when transitioning to order management and payment processing. Comprehensive computations, including prices and taxes, are conducted and incorporated into the transaction's subtotal and tax total. Users can then proceed with the payment or abort the transaction entirely. Payment processing occurs, and if successful, the transaction details are returned.

The system's design embodies a methodical approach to transaction handling, adeptly accommodating diverse product types while ensuring clarity and operational ease for customers and administrators. It effectively manages the intricacies of perishable and non-perishable item management, facilitates accurate price calculations, and provides the seamless execution of payment processing operations.