Unit Tests

Unit Test 1: Adding a New Product to Inventory

This test ensures that adding a new product to the inventory works correctly. We create an Inventory object and add an item (name = "Jacket", price = 50, quantity = 20). The test passes if the item is correctly added with the expected details, otherwise it fails.

```
item = new Item(name="Jacket", price=50, quantity=20)
inventory.add(item)
if inventory.search("Jacket") is not empty and inventory.search("Jacket").price == 50 and
inventory.search("Jacket").quantity == 20:
  PASS
else:
  FAIL
```

Unit Test 2: Deleting a Product from Inventory

This test verifies that deleting a product from the inventory works as intended. We create an Inventory object, add an item (name = "Shoes", price = 30, quantity = 10), and then delete it. The test passes if the item is successfully removed from the inventory.

```
inventory = new Inventory()
item = new Item(name="Shoes", price=30, quantity=10)
inventory.add(item)
inventory.delete("Shoes")
if inventory.search("Shoes") is empty:
  PASS
else:
  FAIL
```

System Tests

System Test 1: Purchase Transaction with Inventory Update and Receipt Generation

A customer picks up a pair of jeans which sells at $50. The employee scans the barcode and operates the POS system. The employee selects those 2 jeans from the POS. The POS then returns the total, which in this case would be $100 without tax. The customer taps a credit card and waits for the transaction handler to process and confirm the purchase. Once it is confirmed, inventory will automatically update itself, and the database will record the transaction history as well. Finally, the POS will generate a receipt with all the relevant information printed on it.

System Test 2: Refund Transaction with Inventory Reinstatement and Refund Confirmation

A customer requests a refund for a previously purchased shirt which was priced at $20. The employee requests the customer’s receipt and scans the barcode or types in the transaction ID on the receipt. Then, the employee confirms the purchase details. If it is verified as eligible for a refund, then the POS will connect to the cashier and refund the customer in cash. Meanwhile, the inventory will be updated since the refunded shirt is now added back. The POS will also generate a refund receipt for the customer and record refund information such as date, refund amount, etc., into the transaction history, saving it to the database in further steps.

System Test 3: Inventory Search by Multiple Criteria

An employee searches for items in inventory by inputting information into the POS. The employee can enter names, item IDs, prices, or even colors to search for items in the inventory. For instance, if the employee enters "Sweater" as the name, "Brown" as the color, and "150" as the price, the POS will search for items that match all inputted criteria and return those matched items for the employee to view.

System Test 4: Adding New Items to Inventory and Verifying Details

An employee wants to add new items to the inventory through the POS. The POS will then require the employee to provide and input information such as names, prices, item ID, colors, quantities, sizes, etc. Once the POS confirms that the item was successfully added to the inventory, the employee can double-check by using the search function with the item ID or any other condition to verify if it was added correctly.

Functional Tests

Function Test on Inventory Real-time Update Function

```
TransactionHandler txhandler = new TransactionHandler();
txhandler.transactionID = 1234566778976433;
Inventory inventory = new Inventory();
Item x1 = new Item(name = "x1");
Item x2 = new Item(name = "x2");
Item x3 = new Item(name = "x3");

inventory.addItem(x1);
inventory.addItem(x2);
inventory.addItem(x3);

txhandler.addItem(x1);
txhandler.addItem(x3);

txhandler.updateInventory(1234566778976433);

if (inventory.search("x1").isEmpty() && inventory.search("x3").isEmpty() &&
!inventory.search("x2").isEmpty()) {
    PASS;
} else {
    FAIL;
}
```

Function Test on Database Update Function

```
TransactionHandler txhandler = new TransactionHandler();
txhandler.transactionID = 42456754323;
Database db = new Database();
Item x1 = new Item();

txhandler.addItem(x1);
TransactionHistory txhistory = new TransactionHistory();

db.syncData();

if (txhistory.retrieveTransaction().contains(42456754323)) {
    if (db.execute("SELECT * FROM history WHERE txID = 42456754323")) {
        PASS;
    } else {
        FAIL;
    }
```

```
} else {
    FAIL;
}
```

Features Tested

Transaction Handling

We designed function tests for the inventory update and the database update. From our understanding, transaction handling is not only about payment but also about recording information after payment. Our function tests can tell if the inventory update module works or not, preventing the sale of sold-out items. It also checks the database update module to ensure that the database has a record, which might be helpful if a refund is issued.

Inventory Update and Management

We designed function tests for the Inventory add and delete functions, as those are the most basic functions in a well-organized inventory. Checking these two functions ensures that other functions that rely on adding or deleting items work properly. We also designed two system tests for the inventory search and the process of adding items to the POS. These system tests ensure that overloaded functions work as intended from the employee's perspective, without requiring knowledge of how the functions are implemented.