Phase 3 - VATTO Tech

Introduction (Brief Overview)

Approach to Testing

The overall approach we will be utilizing for testing, which will ensure that all the functionalities of the VATTO tech inventory tracking system are functional, is by using unit, integration, and system tests. To categorize our tests we will also apply the testing views Clear Box (CB), Translucent Box (TB), and Opaque Box (OB) wherever appropriate. Finally, Automation Testing will be implemented wherever possible to ensure more efficient verification of the software.

System Components to be Tested:

CB Cases

- **Add Product Functionality:** Inspect to see if the add function adds a new product to the database, double check to see if the product is saved correctly
- **Delete Product Functionality**: Inspect to see if the delete/remove function removes a new product from the database, double check to see if the product is gone from the list
- **Update/Edit Product Functionality:** Ensure updating existing item details works correctly.
- **Search Functionality**:
    - Validate that the search function can successfully find an item when given the correct search parameters.
    - Confirm that the search function works regardless of the letter casing (case-insensitive search).
    - An empty search query returns no results or handles the edge case properly.

- **Sort/Filter Functionality:**
    - Verify that the sort/filter function works correctly
    - Determine that filtering products by relevance is working as expected

- **Login Functionality:**
    - Confirm that the login function works as anticipated, correctly authenticates valid users
    - Test that the username and password provided is valid
    - Make sure that the edge cases are handled correctly whether it be a wrong username or password
- **Overview of Inventory:**
    - Check that the inventory overview correctly displays the list of products in stock.
    - Ensure that the inventory overview includes key product details like name, price, and quantity.
    - Test that the inventory overview updates when items are added or removed from the database

TB Cases

- **Login System & Inventory System - Valid user credentials**
    - Enter valid credentials (username and password) in the login form.
    - Submit login request.
    - Verify that the login system fetches user data and grants access to the inventory system.

- **Login System & Database - Invalid user credentials exist**
    - Enter incorrect credentials (wrong username/password combination).
    - Submit the login form three times with the wrong credentials.
    - Verify that the login system interacts with the database to check credentials and deny access.
    - Check if the system displays an error message and exits after three failed login attempts.

- **Inventory System & File I/O - Inventory file exists**
    - Ensure that the inventory file is available.
    - Trigger the system to load the inventory data from the file.
    - Verify that the inventory system receives and correctly displays the data from the file.
    - Ensure that the inventory system interacts with the file system to read the data and display it in the UI.
- **Inventory System & File I/O - The inventory File is missing**
    - Start the system with the inventory file missing.
    - Check that the system handles this error by displaying an appropriate message.
    - Verify that the system continues running despite the missing file, without crashing.
- **Inventory System & File-based Add Function - Valid inventory file exists**
    - Ensure a valid inventory file with products is available.
    - Trigger the system to load products from the file.
    - Verify that the inventory system correctly adds the products to the inventory list.
    - Check that the products are integrated properly into the system without data loss.
- **Inventory System & File-based Add Function - Invalid inventory file exists**
    - Ensure an invalid inventory file with incorrect data is available.
    - Trigger the system to load products from the incorrect file.
    - Verify that the system detects the issue and displays an appropriate error message.
    - Ensure the system continues to operate normally after the error.

OB Cases

- **Inventory stock alert - Product stock is low**
    - Check the inventory status report after some products have low stock levels.
    - Review the stock alert in the inventory overview.
    - Verify that the system generates and displays a clear stock alert for low-stock products.
    - Ensure the stock alert is shown in a user-friendly format on the UI.

- **Bulk product import - A valid inventory file exists**
    - Upload a valid inventory file that contains product data.
    - Trigger the bulk import process.
    - Observe the product addition process from the user interface (UI).
    - Check the inventory to ensure that all products are added successfully and that existing products have their stock levels increased accordingly.
- **Data persistence after restart - The system has active data**
    - Start the system with an active inventory.
    - Restart the system (close and reopen the application).
    - After the restart, check the inventory overview to confirm the data (inventory) is still intact.
    - Verify that all the product information is still present after the restart.

Testing Frameworks and Automation Tools:

- Unit Testing: JUnit (JavaScript framework)
- Integration Testing: Postman (API testing)
- System Testing: Manual user testing (UI automation)

Application of Testing Views (CB, TB, OB)

For this project, VATTO Tech will be applying the three key testing views, Clear Box (CB), Translucent Box (TB), and also Opaque Box (OB) to ensure every part of the system is fully tested. First, we will be using the Clear Box testing for our unit tests. This will do a thorough check of all individual functions and methods, so we can see exactly how they work and make sure they do what they're supposed to. Next, we will be using the Translucent Box for the integration tests, we'll be testing how different parts of the system will work together. We'll focus on making sure the different parts of the system work well together and everything runs smoothly. Lastly, we have the Opaque Box, which will be used for system tests. We will test how the system works as a whole from the user's perspective, without focusing on the code behind it. This includes making sure features like inventory overview, login, search, etc, work properly and look right to the user.

Unit Tests

Test ID	Method/Class	Input(s)	Expected Output(s)	Testing Approach (Manual/Automated)	Assigned Team Member
UT-01-CB	AddProduct()	"1" "Chess" "5" "12.99" "9999" "Board Game" "Unknown" "5"	"Product added successfully!"	JUnit	Tony
UT-02-CB	RemoveProduct()	"1125"	"Product removed successfully!"	JUnit	Tony
UT-03-CB	ViewInventory()	"5"	"Chess | SKU:9999 | Quantity: 5 | Price: $12.99 | Tags: Board Game,Unknown, 5"	JUnit	Abdullah
UT-04-CB	searchInventory()	"4" "1" "Chess"	"Chess | SKU:9999 | Quantity: 5 | Price: $12.99 | Tags: Board Game,Unknown, 5"	JUnit	Abdullah
UT-05-CB	inventoryOverview()	"6"	"==== Inventory Overview ==== Stock Status Report:	JUnit	Abdullah

			Summary: Total items below threshold (5): 0 Items at zero stock: 0”		
UT-06-CB	sortInventory()	“7” “1”	“Chess | SKU:9999 | Quantity: 5 | Price: $12.99 | Tags: Board Game,Unknown, 5”	JUnit	Tony
UT-07-CB	editInventory()	“8” “9999” “3” “7”	“Editing product: Chess | SKU:9999 | Quantity: 5 | Price: $12.99 | Tags: Board Game,Unknown, 5” “1. Edit Name 2. Edit SKU 3. Edit Quantity 4. Edit Price 5. Edit Tags 6. Exit Editing Enter your choice” “Enter your new quantity:”	JUnit	Tony

Integration Tests

Test ID	Components Involved	Preconditions	Steps	Expected Result	Assigned Team Member
IT-01-TB	Login System & Database	User exists in DB	Enter valid credentials, submit, view inventory	Login is successful,the session starts, can see the full inventory	Omar
IT-02-TB	Login System & Database	Invalid user credentials exist	Enter incorrect credentials three times	Access denied, program exits	Omar
IT-03-TB	Inventory File I/O	Inventory file exists	Load inventory from the file	Inventory is populated correctly	Abdullah
IT-04-TB	Inventory File I/O	The inventory file is missing	Start the system	Error message displayed, system continues	Abdullah
IT-05-TB	File-Based Add	A valid inventory file exists	Load products from the file	Products are added successfully	Tony

IT-06-TB	File-Based Add	Incorrect inventory file	Load products from the incorrect file	Error message displayed, system continues	Tony

System Tests

Test ID	Scenario	Preconditions	Steps	Expected Outcome	Assigned Team Member
ST-01-OB	Inventory stock alert	Product stock is low	View inventory report	The overview report indicates the stock is low	Abdullah
ST-02-OB	Bulk product import	Valid inventory file	Import products with a file	Products are added successfully and existing products have their stock increased	Tony
ST-03-OB	Data persistence after the restart	The system has active data	Restart the system and check the inventory again	The inventory persists after restart	Tony

Coverage

Justification of Test Coverage

To ensure that the VATTO Tech project is reliable and working, we've structured our testing plan to fully cover and verify our system. We choose to identify the high-risk components, summarize test cases across unit, integration, and system levels, and lastly, we will be ensuring the appropriate application of all three testing views (CB, TB, and OB).

Identifying High-Risk Components:

We've gone over our components and identified the high-risk components of the system that are critical for the system and user experience. These include the add, edit, and remove functions, our overview function, search function, and also our sort/filter function. We believed

that these were our high-risk components for the system core operations and user interaction, which is why we have thorough test cases for all.

Summarizing Test Cases Across Unit, Integration, and System Levels:

Overall, our testing plan includes a good mix of unit, integration, and system tests to make sure everything works as expected. Our unit tests will focus on individual methods and functions such as our login function, search function, etc. The integration tests will examine how the components will interact with each other, an example of this testing is ensuring that after the login system is completed, it will then lead you to the main commands. The system tests will validate end-to-end processes, like user account management and product search functionality. We've ensured a comprehensive number of test cases to cover all major functionality across these three levels.

Ensuring Appropriate Application of all Three Testing Views (CB. TB, OB):

To make sure to achieve a thorough validation of the system, thus we've applied all three of the testing views, Clear Box(CB), Translucent Box(TB), and Opaque Box(OB). Unit tests will verify individual methods and functions, integration tests will check if component iterations are successful, and system tests will validate real-world scenarios function correctly. CB testing checks internal logic, TB testing examines component interactions, and OB testing verifies expected system behavior. These approaches will confirm core functionalities are working, components are integrated correctly, and the systems perform perfectly, providing confidence in the software's functionality.