**1. Test Plan**

Before running the tests, define the key areas to test and the expected outcomes.

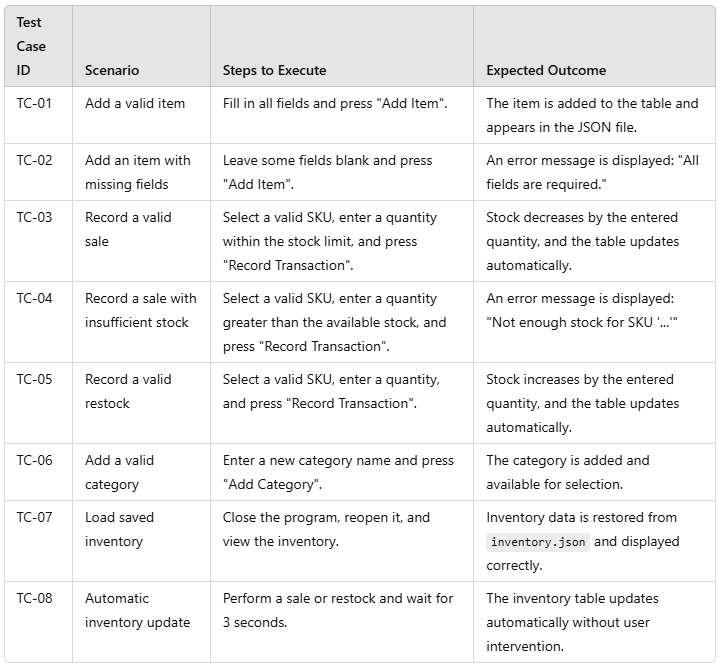
**Key Areas to Test:**

1. **GUI Interaction:**
   * Ensure all buttons, input fields, and tabs function correctly.
   * Validate the responsiveness of the inventory table (automatic updates).
2. **Items Management:**
   * Adding items with valid and invalid inputs.
   * Checking if the inventory table updates immediately.
3. **Category Management:**
   * Adding categories and associating items with categories.
4. **Transactions:**
   * Recording sales with sufficient stock.
   * Handling sales with insufficient stock.
   * Recording restocks and updating the inventory.
5. **Data Persistence:**
   * Verify that data is saved to inventory.json.
   * Reload the program and confirm the data is correctly restored.
6. **Error Handling:**
   * Validate the system gracefully handles invalid inputs.

**2. Test Cases**

Define test cases to systematically verify the system's behavior.

**Test Case Table**



**3. Test Execution**

Perform the test cases step by step and record the results.

**Example Execution Steps:**

1. **Start the Program:**
   * Run main.py and ensure all tabs load without errors.
2. **Add Items:**
   * Add a valid item and verify it appears in the table.
   * Attempt to add an item with incomplete details and confirm the error message.
3. **Perform Transactions:**
   * Test a sale and verify the stock decreases.
   * Attempt a sale with insufficient stock and confirm the error message.
   * Test a restock and verify the stock increases.
4. **Category Management:**
   * Add a category and confirm it is available for selection when adding new items.
5. **Check Data Persistence:**
   * Restart the program and confirm all data is loaded from inventory.json.
6. **Automatic Updates:**
   * Wait for 3 seconds after performing a transaction and confirm the table updates automatically.

**4. Test Results**

Create a table to record the outcomes of each test case.

| **Test Case ID** | **Result** | **Notes** |
| --- | --- | --- |
| TC-01 | Pass | Item added successfully and appeared in the table. |
| TC-02 | Pass | Error message displayed correctly for missing fields. |
| TC-03 | Pass | Stock decreased correctly, and the table updated automatically. |
| TC-04 | Pass | Error message displayed for insufficient stock, and no changes to inventory. |
| TC-05 | Pass | Stock increased correctly, and the table updated automatically. |
| TC-06 | Pass | New category added and available for item association. |
| TC-07 | Pass | Inventory loaded successfully from inventory.json. |
| TC-08 | Pass | Table updated automatically after 3 seconds. |

**5. Report and Observations**

Write a brief summary of your observations:

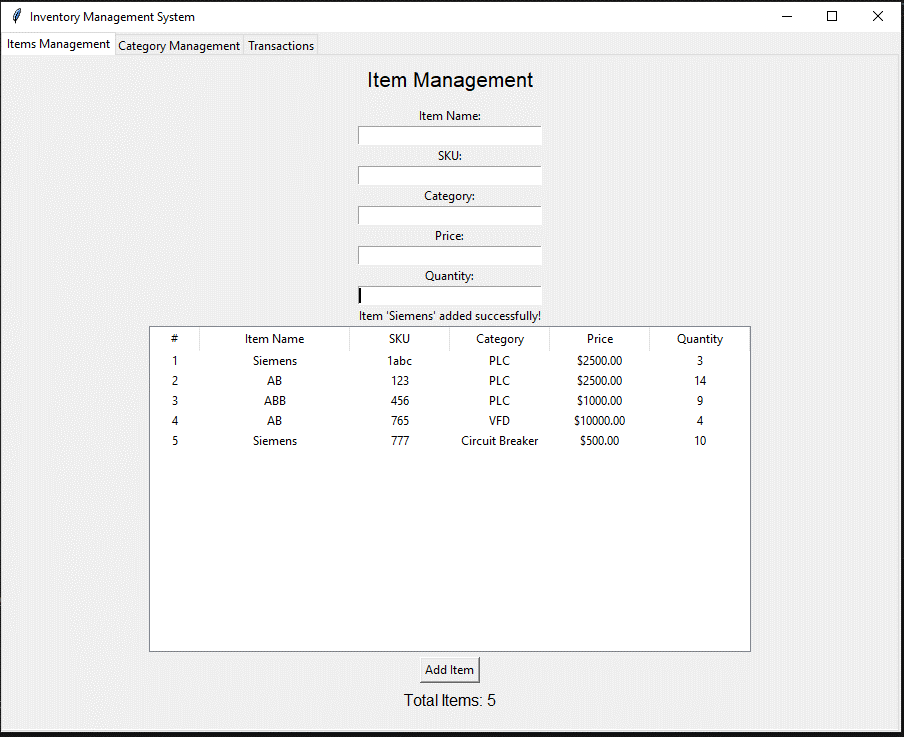
* **Overall Behavior:** The system worked as expected, and all test cases passed successfully.
* **Key Highlights:** Real-time updates and error handling worked seamlessly.
* **Improvements:** Could include additional features like exporting inventory data to a file or adding user authentication for enhanced security.

**6. Documentation**

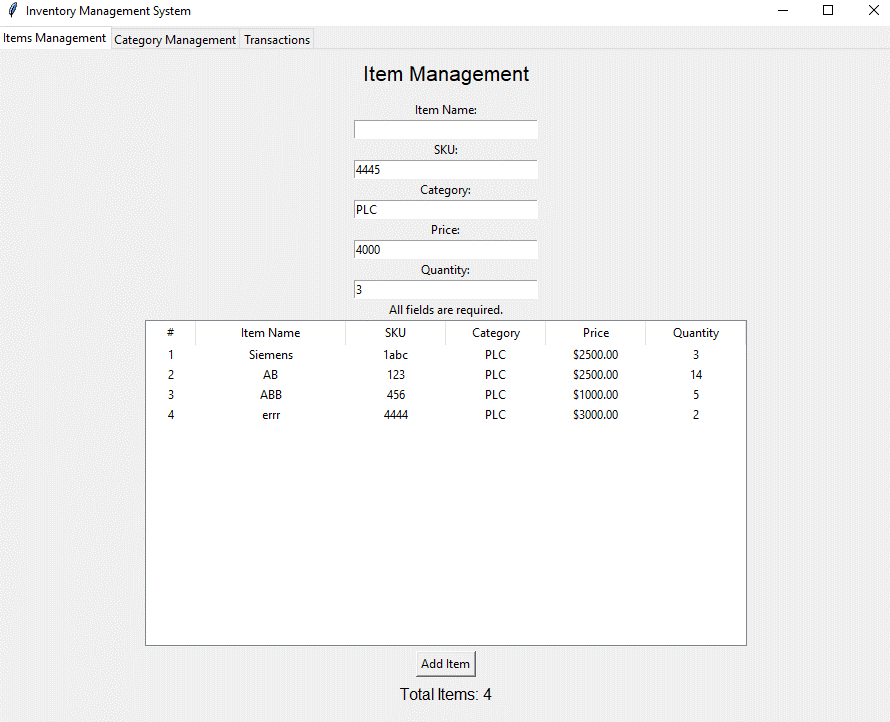
In the next point I add picture of my testing examples.

**7. Testing Images**

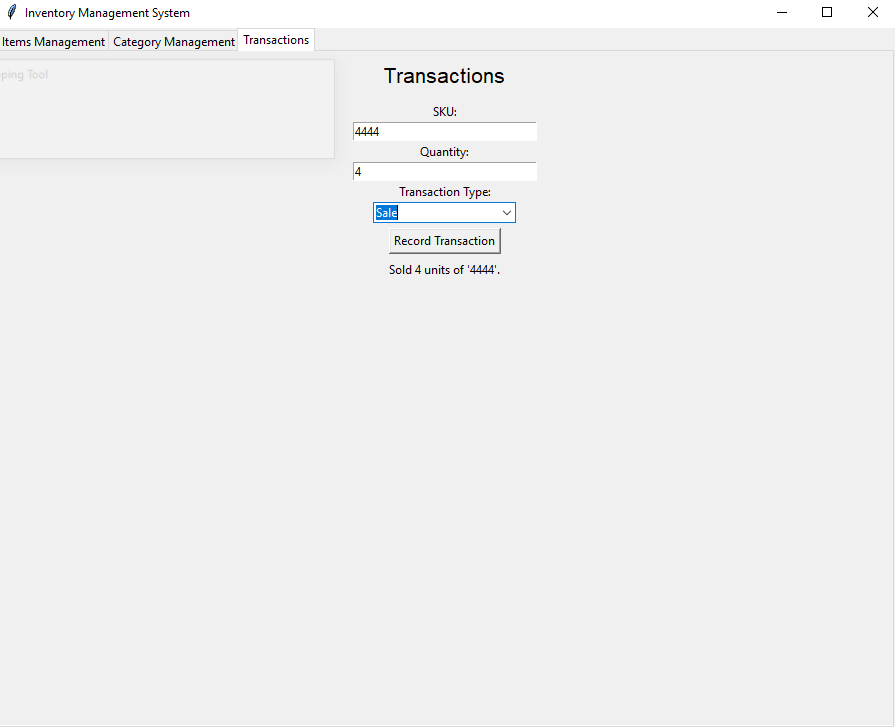
|  |  |
| --- | --- |
| TC-01 | Add a valid Item |



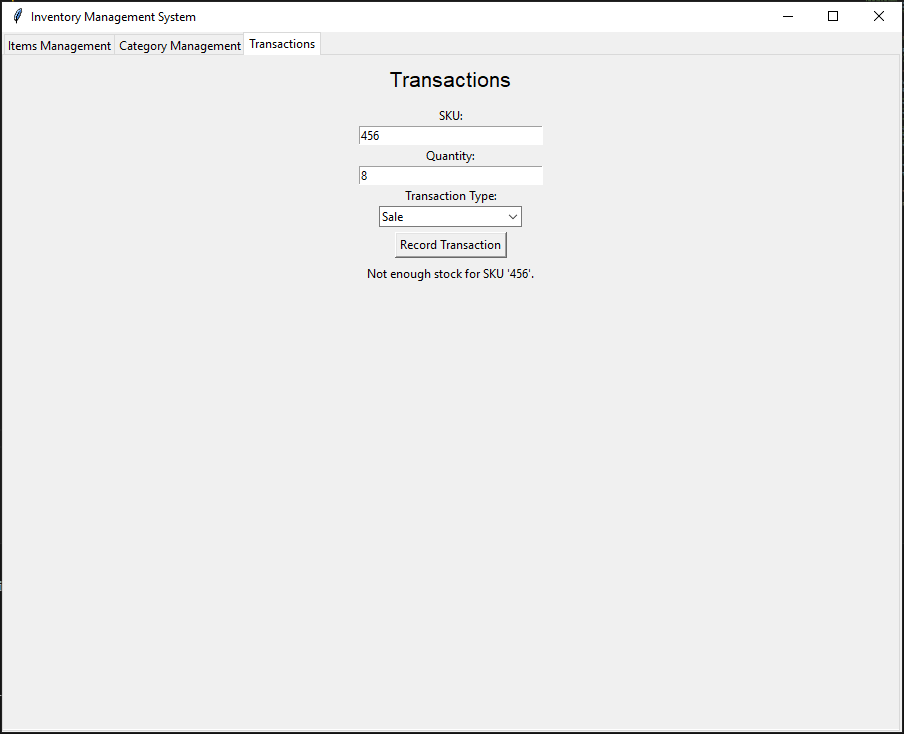
|  |  |
| --- | --- |
| TC-02 | Add an item with missing fields |



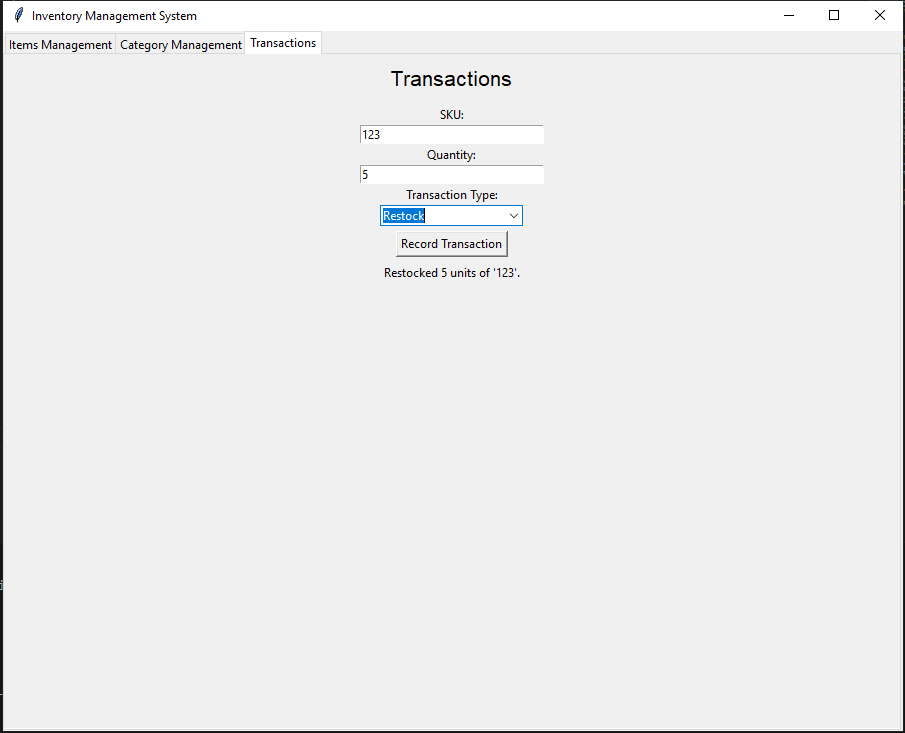
|  |  |
| --- | --- |
| TC-03 | Record a valid sale |



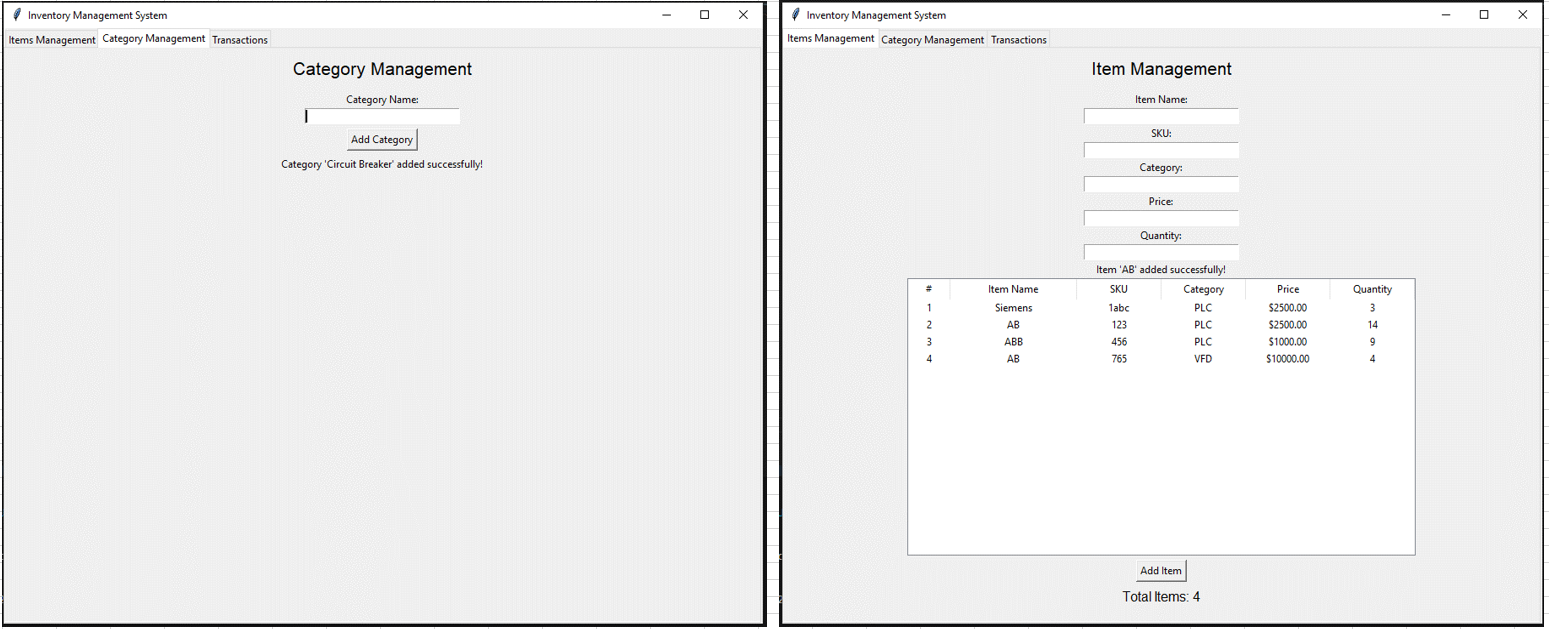
|  |  |
| --- | --- |
| TC-04 | Record a valid sale |

\

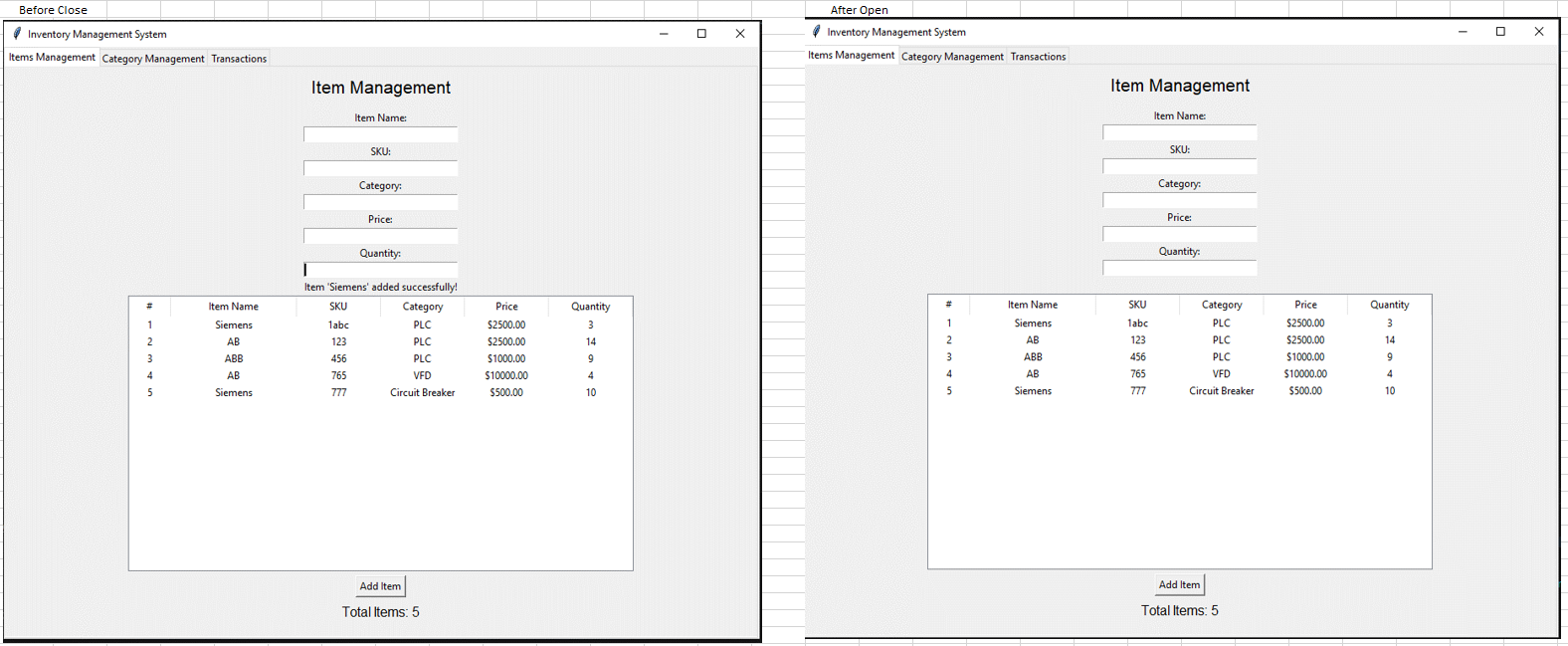
|  |  |
| --- | --- |
| TC-05 | Record a valid restock |



|  |  |
| --- | --- |
| TC-06 | Add a valid category |



|  |  |
| --- | --- |
| TC-07 | Load saved inventory |



|  |  |
| --- | --- |
| TC-07 | Load saved inventory |

This is happening every time we do a transaction or every time we add an Item.