**Testing and System Validation**

**Question:** Write a test plan designed to ensure that the coded solution works as expected. The test plan should include specific instructions about the data and conditions the program will be tested with.

**Shopping Cart Application Test Plan**

Test Environment

* Operating System: Windows 11 / macOS / Linux
* Java Version: JDK 8 or later
* IDE: Eclipse / IntelliJ IDEA / NetBeans

**Launch Application**

Instructions:

* Execute the application.

Expected Result:

* The IDE should launch the application without encountering any errors.
* If there is no existing file to store the product details, generate a new file within the src directory.
* If a file for saving product details already exists, ensure that the product details from the corresponding file are successfully loaded into the application.
* Confirm that the Westminster Shopping Manager Menu is visible in the IDE console and that the user is prompted to input their choice.

**WestminsterShoppingManager Console Menu Implementation Test Plan**

1. Add a New Product to the System

**Test Case 1: Successful Inclusion of an Electronic Product.**

Instructions:

* Opt for the "Add a new product to the system" choice from the menu.
* Select the option to include an electronics product.
* Input valid details for all attributes.
* Verify that the system has available space for additional products.

Expected Result:

* Confirm the successful addition of the product.
* Validate that the product count increments by 1.
* Verify the update in the total number of products in the system.
* Ensure the details of the added product match the entered information.

**Test Case 2: Successful Inclusion of a Clothing Product.**

Instructions:

* Opt for the "Add a new product to the system" choice from the menu.
* Select the option to include a clothing product.
* Input valid details for all attributes.
* Verify that the system has available space for additional products.

Expected Result:

* Confirm the successful addition of the product.
* Validate that the product count increments by 1.
* Verify the update in the total number of products in the system.
* Ensure the details of the added product match the entered information.

**Test Case 3: Maximum Products Reached**

Instructions:

* Try to include a new product when the system has reached its maximum limit.

Expected Result:

* Validate that the system gracefully manages the limit by presenting a suitable error message.
* Confirm that the user receives information about reaching the maximum limit.
* Ensure that the addition of the product is not executed.

1. Delete a Product from the System

**Test Case 1: Successful Deletion with a Valid Product ID**

Instructions:

* Opt for the "Delete a product from the system" choice from the menu.
* Enter the product ID of an existing product for deletion.

Expected Result:

* Validate the successful deletion of the specified product.
* Confirm the display of the accurate deletion message along with the correct product information.
* Verify that the product count decreases by 1.
* Check for the update in the total number of products.

**Test Case 2: Invalid Product ID**

Instructions:

* Opt for the "Delete a product from the system" choice from the menu.
* Try to delete a product using an invalid or non-existing product ID.

Expected Result:

* Confirm that the system handles invalid input gracefully by presenting a suitable error message.
* Verify that no products are deleted.
* Check that the total number of products remains unchanged.

1. Print the List of Products in the System

**Test Case 1: Successful Printing in Alphabetic Order**

Instructions:

* Select the "Print the list of products in the system" option from the menu.

Expected Result:

* Validate the display of the product list on the console.
* Confirm that the product list is arranged alphabetically by product ID.
* Ensure accurate representation of all relevant attributes and product types.

**Test Case 2: Empty Product List**

Instructions:

* Remove all products from the system.
* Print the list of products.

Expected Result:

* Confirm the display of a suitable message for an empty product list.
* Verify that no products are printed in such a scenario.

1. Save and Load Products from a File

**Test Case 1: Successful Save to File**

Instructions:

* Add a few products to the system.
* Save the list of products to a file.

Expected Result:

* Validate the creation of the file and successful saving of products.
* Verify that the saved products match the ones entered earlier.

**Test Case 2: Successful Load from File**

Instructions:

* Restart the application.

Expected Result:

* Validate the error-free restart of the application.
* Confirm the loading of the product list from the file.
* Verify that the loaded products match those previously saved, are correctly displayed, and can be utilized in subsequent operations.

**Test Case 3: Empty File**

Instructions:

* Erase all products from the system.
* Save the empty product list to a file.
* Restart the application and load products.

Expected Result:

* Confirm the display of an appropriate message for an empty file, and ensure no products are loaded.

**WestminsterShoppingManager GUI Implementation Test Plan**

1. Open GUI from the Menu Console

**Test Case 1: GUI Opening**

Instructions:

* Open the GUI using the additional option in the menu console.

Expected Result:

* Verify that the GUI window opens successfully without errors.
* Check that it matches the specified mock-up.
* Verify that the product categories are displayed in the dropdown menu.

1. Product Visualization

**Test Case 1: Dropdown Menu**

Instructions:

* Select each option from the product type dropdown menu (All, Electronics, Clothes).

Expected Result:

* Verify that the displayed products match the selected category.
* The table should display the selected category of products from the "products.txt" file.
* Confirm that the dropdown menu functions correctly.

**Test Case 2: Product List Display**

Instructions:

* Check if the product list is displayed in a table.

Expected Result:

* Verify that the product information is correctly displayed from the products.txt file.

**Test Case 3: Sort Products**

Instructions:

* Click on each column header in the product table.

Expected Result:

* Products should be sorted in ascending order based on the clicked column.
* Clicking the same column header again should toggle the sorting to descending order.

**Test Case 4: Check Availability Highlighting**

Instructions:

* Ensure that the "products.txt" file contains products with low availability (e.g., less than 3 items).

Expected Result:

* Verify that the corresponding rows of products with reduced availability in the products table are highlighted in red (background color indicating low availability).

1. Adding Products to Shopping Cart

**Test Case 1: Product Details Display**

Instructions:

* Click on a product in the table.

Expected Result:

* Confirm that product details appear in a panel below the table.
* The details panel should update with information about the selected product.
* Verify that all relevant information is displayed.
* Verify that the "Add to Shopping Cart" button is enabled.

**Test Case 2: Adding Product to Shopping Cart**

Instructions:

* Click on a product in the table.
* Add a product to the shopping cart by clicking the "Add to Shopping Cart" button.
* Repeat for multiple products.

Expected Result:

* Verify that the selected products are successfully added to the shopping cart.
* Check that the shopping cart updates accordingly.
* Verify that the shopping cart window is updated with the added product.

**Test Case 3: View Shopping Cart**

Instructions:

* Click the "Shopping Cart" button.

Expected Result:

* Confirm that the shopping cart window opens.
* Check if the list of products and final price are displayed correctly.

1. Test Shopping Cart Discounts

**Test Case 1: Category Discount**

Instructions:

* Add three or more products of the same category to the shopping cart.

Expected Result:

* Verify that a 20% category discount is applied.
* Confirm that the final price reflects the discount.

**Test Case 2: First Purchase Discount**

Instructions:

* Add products to the shopping cart and complete the purchase.
* Repeat for a second purchase.

Expected Result:

* Verify that a 10% first purchase discount is applied on the second purchase.
* Confirm that the final price reflects the discount.

**Test Case 3: Discount Message**

Instructions:

* Apply discounts and check the message.

Expected Result:

* Verify that the discount message appears on the GUI.
* Confirm that the final cost is displayed correctly.