**-DOCUMENTATION-**

**Additional test cases**

**1. Apply Discount Code**

**Test Case:** Verify that applying a discount code affects the total price correctly.

**Description:** Ensure that when a valid discount code is applied, the cart’s total price is updated to reflect the discount. Additionally, check that invalid discount codes show appropriate error messages and do not alter the total price.

**Steps:**

1. **Navigate to Cart Page:**
   * Go to the cart page where you can see the list of items added and the total price.
2. **Verify Initial Total Price:**
   * Record the initial total price of the cart before applying any discount.
3. **Apply Valid Discount Code:**
   * Enter a valid discount code in the discount code field and click the "Apply" button.
   * The discount code should apply, and the total price should be updated accordingly.
   * Verify that the new total price reflects the discount correctly.
4. **Verify Discount Code Application:**
   * Check that the applied discount is shown in the cart summary or breakdown.
5. **Apply Invalid Discount Code:**
   * Enter an invalid or expired discount code and click the "Apply" button.
   * Verify that the system displays an appropriate error message indicating the code is invalid or expired.
   * Ensure that the total price remains unchanged.
6. **Remove Discount Code:**
   * Remove the discount code and verify that the total price reverts to its original amount before the discount was applied.
7. **Edge Case: Expired Discount Code:**
   * Test with a discount code that has expired to ensure the system correctly identifies it as expired and does not apply any discount.

**Expected Results:**

* Valid discount codes should correctly adjust the total price.
* Invalid or expired codes should show error messages and leave the total price unchanged.
* The system should handle discount code removals properly and revert to the original total price.

**2. Cart Instance**

**Test Case:** Verify that the cart contents persist after refreshing the page or reloading.

**Description:** Ensure that items added to the cart are retained across page reloads and sessions. This test case is crucial for confirming that cart data is stored and managed correctly across user interactions.

**Steps:**

1. **Add Items to Cart:**
   * Navigate to the product page and add several items to the cart.
   * Confirm that the items are correctly added by checking the cart’s item count and details.
2. **Refresh the Page:**
   * Refresh the cart page (using F5 or a browser refresh button).
   * Verify that the items added to the cart remain intact and that the item details (quantity, price) are accurate.
3. **Navigate Away and Return:**
   * Navigate to another page (e.g., home page) and then return to the cart page.
   * Check that the cart contents persist and are consistent with what was added before.
4. **Log Out and Log In:**
   * If applicable, log out of the website and then log back in.
   * Verify that the cart contents are retained after logging back in.
5. **Close and Reopen Browser:**
   * Close the browser completely and reopen it.
   * Navigate to the website and go to the cart page.
   * Check that the cart contents are still present.

**Expected Results:**

* Cart contents should persist across page reloads, navigation, and user sessions.
* Items added to the cart should be retained and correctly displayed, even after refreshing, logging out, or reopening the browser.

**3. Cart Limit**

**Test Case:** Verify that the cart enforces any quantity limits (e.g., maximum quantity allowed for an item).

**Description:** Ensure that the cart enforces rules regarding the maximum quantity of an item that can be added. This is important to prevent users from exceeding allowed limits.

**Steps:**

1. **Navigate to Product Page:**
   * Go to a product page with quantity limits defined (e.g., a product with a max limit of 5 items).
2. **Add Items to Cart:**
   * Attempt to add items to the cart, starting with a quantity below the limit (e.g., 1 or 2).
3. **Verify Adding Up to Limit:**
   * Gradually increase the quantity of the item in the cart up to the maximum limit (e.g., 5).
   * Verify that the item can be added to the cart up to the allowed limit.
4. **Attempt to Exceed Limit:**
   * Try to add more items beyond the allowed limit (e.g., 6 items if the limit is 5).
   * Verify that the system prevents adding more items and displays an appropriate error message.
5. **Check Error Message:**
   * Ensure that the error message displayed when exceeding the limit is clear and instructive (e.g., "Cannot add more than 5 items of this product").
6. **Remove Items:**
   * Remove some items from the cart and verify that you can now add up to the limit again.

**Expected Results:**

* The cart should enforce quantity limits correctly, allowing users to add only up to the maximum allowed quantity.
* Attempting to exceed the limit should show an appropriate error message and prevent further additions.

**4. Responsive Design**

**Test Case:** Verify that the cart is functional and visually correct on different screen sizes (mobile, tablet, desktop).

**Description:** Ensure that the cart interface is properly displayed and remains usable across various device sizes and orientations. This is important for a good user experience on all devices.

**Steps:**

1. **Test on Desktop:**
   * Open the cart page on a desktop browser.
   * Verify that the cart layout, item details, and functionality are correct and usable.
2. **Test on Tablet:**
   * Resize the browser window or use a tablet emulator to view the cart page on a tablet-sized screen.
   * Ensure that the cart layout adjusts appropriately and remains functional.
3. **Test on Mobile:**
   * Use a mobile device or mobile emulator to access the cart page.
   * Verify that the cart is displayed correctly in portrait and landscape orientations.
   * Check that touch interactions (e.g., adding/removing items) work as expected.
4. **Check for Responsiveness:**
   * Ensure that elements such as buttons, input fields, and item details are properly sized and spaced for readability and usability on different screen sizes.
5. **Test Navigation:**
   * Verify that navigation and interactions (e.g., adding/removing items, applying discount codes) work smoothly across all device sizes.
6. **Test Accessibility:**
   * Ensure that responsive design also adheres to accessibility standards, such as readable font sizes and accessible navigation.

**Expected Results:**

* The cart should be fully functional and visually appropriate across all tested device sizes and orientations.
* Interactions and layout adjustments should be seamless and user-friendly on mobile, tablet, and desktop screens.

.

**Implementations and improvements**

**1. Account Creation and Management**

**Sign-Up**

* **Functionality:** Allow users to create an account with email and password.
* **Fields:** Name, Email, Password, Confirm Password.
* **Process:** Validate input, confirm email (optional), and create an account.

**Login**

* **Functionality:** Enable users to log in with email and password.
* **Fields:** Email, Password.
* **Process:** Authenticate user credentials and grant access to the account.

**Forgot Password**

* **Functionality:** Allow users to reset their password if forgotten.
* **Process:** Send a password reset link to the user's registered email.

**Guest Checkout**

* **Functionality:** Allow users to make purchases without creating an account.
* **Fields:** Basic details such as email and shipping address.
* **Process:** Provide a streamlined checkout experience without account creation.

**2. Product Filtering**

**Add Filtering Options**

* **Functionality:** Allow users to filter products by additional parameters.
* **Options:** Colour, Design, Size, Brand, Price Range.
* **UI:** Include filters in a sidebar or dropdown menu for easy access.

**3. Detailed Pricing**

**Subtotal and VAT**

* **Functionality:** Clearly display subtotal and VAT (if applicable).
* **Details:** Include a breakdown of the subtotal, VAT percentage, any other charges and total price.

**Price Transparency**

* **UI:** Show VAT inclusively or exclusively and clarify in the cart/checkout page.
* **Backend:** Ensure accurate calculation and display of all charges and total cost.

**4. Footer with Contact Form**

**Design**

* **Functionality:** Add a footer with a contact form for user inquiries.
* **Fields:** Name, Email, Subject, Message, Submit Button.
* **UI:** Ensure consistent design with the website theme.

**Backend Handling**

* **Functionality:** Process form submissions.
* **Process:** Post data to a backend endpoint, handle validation, and send responses.
* **Security:** Implement spam protection and validate inputs to prevent abuse.

**Buggs or issues**

During my recent attempt to test the cart update functionality, I faced a challenge related to the button classes used for incrementing and decrementing the item quantity. The core of the issue was that both the increment and decrement buttons shared the same CSS class. As a result, the testing script, which relies on these classes to identify and interact with specific elements, was unable to distinguish between the two buttons.

Specifically, the script was consistently locating the decrement button, which was disabled because there was only one item in the cart. This situation prevented the script from interacting with the increment button, which was essential for verifying the functionality of increasing the item quantity. Given that the decrement button could not be used for testing under these conditions, the script could not complete the intended actions.

To address this problem, it's crucial to update the class selectors or identify a more reliable method to differentiate between the increment and decrement buttons. This would ensure that the script can accurately target and interact with the correct button, enabling a comprehensive test of the cart update functionality.

A screen shot of a computer code

Description automatically generated

What I would like to add

In the case that I accidentally refresh the website, my whole cart is gone so I would add the permission to ask for the use of cookies to make sure that I can keep everything I added but if the user doesn’t want to, I would also implement a create account or ‘continue as guest’ buttons to store the information of the user.

If I change my mind and decided that what I have added to basket is not longer what I wanted I have to remove each product one by one, so I would implement a ‘clear whole basket button’.

Add more filters to adjust to the customers likes and interests like color and designs.

VAT? Why does it say subtotal as if there’s any additional charge. It is stated that the delivery is free so I guess that it’s the VAT what still has to be included

During my recent attempt to test the cart update functionality, I faced a challenge related to the button classes used for incrementing and decrementing the item quantity. The core of the issue was that both the increment and decrement buttons shared the same CSS class. As a result, the testing script, which relies on these classes to identify and interact with specific elements, was unable to distinguish between the two buttons.

Specifically, the script was consistently locating the decrement button, which was disabled because there was only one item in the cart. This situation prevented the script from interacting with the increment button, which was essential for verifying the functionality of increasing the item quantity. Given that the decrement button could not be used for testing under these conditions, the script could not complete the intended actions.

To address this problem, it's crucial to update the class selectors or identify a more reliable method to differentiate between the increment and decrement buttons. This would ensure that the script can accurately target and interact with the correct button, enabling a comprehensive test of the cart update functionality.n

|  |  |
| --- | --- |
| **TEST CASE ID** | **TC001** |
| Description | Users can add products to the cart from the product listing page. |
| Preconditions | User must be on the product listing page |
| Steps to execute | 1. User must navigate to the product listing page.  2. User must select a size that will filter if the desired item is available in the user’s size.  3. If such item is available in the desired size, the user must click ‘Add to cart’ |
| Expected result |  |
| Actual result |  |
| Status |  |

|  |  |
| --- | --- |
| **TEST CASE ID** | **TC002** |
| Description | Users can add products to the cart from the product listing page. |
| Preconditions | User must be on the product listing page |
| Steps to execute | 1. User must navigate to the product listing page.  2. User must select a size that will filter if the desired item is available in the user’s size.  3. If such item is available in the desired size, the user must click ‘Add to cart’ |
| Expected result |  |
| Actual result |  |
| Status |  |

|  |  |
| --- | --- |
| **TEST CASE ID** | **TC003** |
| Description | Users can add products to the cart from the product listing page. |
| Preconditions | User must be on the product listing page |
| Steps to execute | 1. User must navigate to the product listing page.  2. User must select a size that will filter if the desired item is available in the user’s size.  3. If such item is available in the desired size, the user must click ‘Add to cart’ |
| Expected result |  |
| Actual result |  |
| Status |  |

|  |  |
| --- | --- |
| **TEST CASE ID** | **TC004** |
| Description | Users can add products to the cart from the product listing page. |
| Preconditions | User must be on the product listing page |
| Steps to execute | 1. User must navigate to the product listing page.  2. User must select a size that will filter if the desired item is available in the user’s size.  3. If such item is available in the desired size, the user must click ‘Add to cart’ |
| Expected result |  |
| Actual result |  |
| Status |  |

|  |  |
| --- | --- |
| **TEST CASE ID** | **TC005** |
| Description | Users can add products to the cart from the product listing page. |
| Preconditions | User must be on the product listing page |
| Steps to execute | 1. User must navigate to the product listing page.  2. User must select a size that will filter if the desired item is available in the user’s size.  3. If such item is available in the desired size, the user must click ‘Add to cart’ |
| Expected result |  |
| Actual result |  |
| Status |  |