



21 JULY 2024

# INTERN TECHNICAL CHALLENGE

TEST CASE CREATION AND AUTOMATED TESTING

HASAN TALAL  
POWWR



## Contents

Task 1 .....	0
Functional Test Cases .....	0
Non-Functional Test Cases.....	4
Task2 .....	7
Assumptions: .....	0
Additional Improvements:.....	0
Additional test cases:.....	0
Further scenarios to cover: .....	0
Tools or methodologies: .....	0
Document/references: .....	0
Bugs: .....	0

# Task 1

## Functional Test Cases

Note: the steps for execution are to be done in a Playwright script to allow for easy automation and large scale testing for the test case.

ID	Description	Preconditions	Steps for execution	Expected result	Actual Result	Status
TC001	Testing XS Size button to ensure that products are filtered as expected	> No sizes are selected > Product(s) in the XS size exist	1) Navigate to the size selection buttons 2) Select the XS size	All products that do not have XS sizes are filtered out and the “black batman T-Shirt” is shown		
TC002	Testing XS and S size buttons to ensure that when both are active the expected products are displayed	> No sizes are selected > Product(s) in the XS and S size exist	1) Navigate to the size selection buttons 2) Select the XS size 3) Select the S size	All products that do not have XS or S sizes are filtered out. The “black batman T-Shirt” and “blue sweatshirt” is shown		
TC003	Testing size buttons to ensure that when all sizes are unselected, all products are displayed	>any number of sizes are selected	1) Navigate to the size selection buttons 2) unselect all sizes	All products should be shown		
TC004	Testing Products found label to ensure it the number of products being displayed with no size filters applied	> No sizes are selected >all products are being displayed	1) navigate to the product(s) found label 2) check number displayed is the same as the number of products on screen	Products found label is updated to 16 Product(s) found		
TC005	Testing Products found label to ensure it the number of products being displayed when L size filter is applied	> L sizes are selected >all products of L size are being displayed	1) Navigate to the size selection buttons 2) Select the L size 3) check number displayed is the same as the number of products on screen	Products found label is updated to 10 Product(s) found		
TC006	Testing Products found label to ensure it the number of products being displayed and correctly added together when L and XL size filter is applied	> L and XL sizes are selected >all products of L and XL sizes are being displayed	1) Navigate to the size selection buttons 2) Select the L and XL size 3) check number displayed is the same as the number of products on screen	Products found label is updated to 13 Product(s) found		

<b>TC007</b>	Testing Products found label to ensure it the number of products being displayed and correctly subtracted together when S and M size filter is applied, and M is unchecked	>No other sizes are selected >S and M sizes are selected >Products found is displaying “3 Product(s) found”	1) Navigate to the size selection buttons 2) Deselect the M size 3) check number displayed is the same as the number of products on screen	Products found label is updated to 2 Product(s) found		
<b>TC008</b>	Testing view cart button to ensure that when it is clicked the user can view cart	> no products are added to cart	1) Navigate to cart and click to open it	The user should see a panel on the left that opens to reveal the cart in this case the cart is empty so the user will not see any products		
<b>TC009</b>	Testing view cart button when no products are added the total is 0, no products are in cart and user cannot checkout with 0 products in the cart	> no products are added to cart	1) Navigate to cart and click to open it 2) Check to ensure there are no products in the cart 3) click check out button	Website should display a warning message that tells the user that they need to add items to their shopping cart		
<b>TC010</b>	Testing view cart button when a product is added to cart the user is able to check out.	>User adds any 1 product to their cart >no sizes are selected	1)navigate to “Skater Black Sweatshirt” and click add to cart 2)navigate to checkout button and click	Website should allow the user to check out and display a message to telling the user to that they have successfully done so. It should also display the total/subtotal of \$25.90.		
<b>TC011</b>	Testing add to cart button to ensure that it appears in the cart showing the correct total price	>no sizes are selected >products are visible to the user	1) navigate to the “Cropped Stay Groovy off white” 2)click add to cart	Shopping cart should open and show the product added with a quantity of 1 and a total price of \$10.90		
<b>TC012</b>	Testing add to cart button to ensure that the correct size is added when filtering the product by XXL	>no other sizes are selected	1)navigate to size selection and click XXL button 2)navigate to the “Loose Black T-shirt” and click add to cart	Shopping cart should open and show the added product with the size XXL		

<b>TC013</b>	Testing shopping cart to ensure that when 2 different products are added to the cart, the total price is added as expected	>no other products are added to cart >no size filters are applied	1)navigate to “Blue T-Shirt” and click add to cart 2)close shopping cart 3)navigate to “Ringer Hall Pass” and click add to cart	Shopping cart should open and display “Blue T-Shirt” and “Ringer Hall Pass” in cart with their respective prices of \$9.00 and \$10.90 with a total price of \$19.90		
<b>TC014</b>	Testing shopping cart to ensure that when 2 of the same products are added to the cart, the total price is added as expected	>no other products are added to cart >no size filters are applied	1)navigate to “Slim black T-shirt” and click add to cart 2)close cart and click add to cart again on “Slim black T-shirt”	Shopping cart should open and display “Slim black T-shirt” in the cart with a price of \$49.90 per item with a total price of \$99.80 as well as showing a quantity of 2		
<b>TC015</b>	Testing to ensure that user can added products of the same type in the cart using the “+” button with the price being adjusted as expected	>One of the products has already been added in the cart in this case “Tropical Wine T-shirt” >the user has the cart view open >no other products are added to the cart >no size filters are added	1) navigate to the “Tropical Wine T-shirt” in the cart 2) navigate to the increase quantity button and click it once	Shopping cart should update the total to display the cost of 2 “Tropical Wine T-shirt” which is \$269.80, and quantity should be updated to 2		
<b>TC016</b>	Testing to ensure that user can remove products of the same type in the cart using the “-” button with the price being adjusted as expected	>Two of the products has already been added in the cart in this case “Tropical Wine T-shirt” >the user has the cart view open >no other products are added to the cart >no size filters are added	1) navigate to the “Tropical Wine T-shirt” in the cart 2) navigate to the decrease quantity button and click it once	Shopping cart should updated the total to display the cost of 1 “Tropical Wine T-shirt” which is \$134.90, and it should show the quantity of the product as 1		
<b>TC017</b>	Testing to ensure that user can added products of the same type in the cart using the “+” button multiple times with the price being adjusted as expected	>One of the products has already been added in the cart in this case “Tropical Wine T-shirt” >the user has the cart view open >no other products are added to the cart	1) navigate to the “Tropical Wine T-shirt” in the cart 2) navigate to the increase quantity button and click it five times	Shopping cart should update the total to display the cost of 6 “Tropical Wine T-shirt” which is \$809.40, and quantity should be updated to 6		

		>no size filters are added				
<b>TC017</b>	Testing to ensure that user can remove products of the same type in the cart using the “-” button multiple time with the price being adjusted as expected	>6 of the products has already been added in the cart in this case “Tropical Wine T-shirt” >the user has the cart view open >no other products are added to the cart >no size filters are added	1) navigate to the “Tropical Wine T-shirt” in the cart 2) navigate to the decrease quantity button and click it five times	Shopping cart should updated the total to display the cost of 1 “Tropical Wine T-shirt” which is \$134.90, and it should show the quantity of the product as 1		
<b>TC018</b>	Testing to ensure that users can update the quantity of the items in the cart while there are multiple products in the cart with the total cost being adjusted accordingly	> 1 “Black Tule Oversized” in cart > 2 “Blue T-Shirt” in cart >1 “Grey T-shirt” in cart >user has the cart open >no other products are added to the cart >no size filters are added	1) Navigate to “Blue T-Shirt” in the cart and click “-” button once 2) navigate to “Grey T-shirt” in the cart and click “+” and click it twice	Shopping cart should be updated to display the quantity of the “Grey T-shirt” to 3, the quantity of the “Blue T-Shirt” to 1 with “Black Tule Oversized” remaining at 1. Total price should be adjusted to \$83.15		

## Non-Functional Test Cases

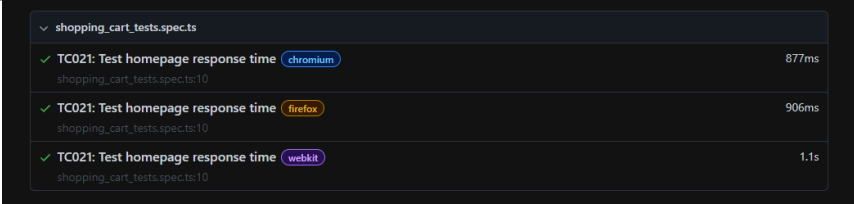
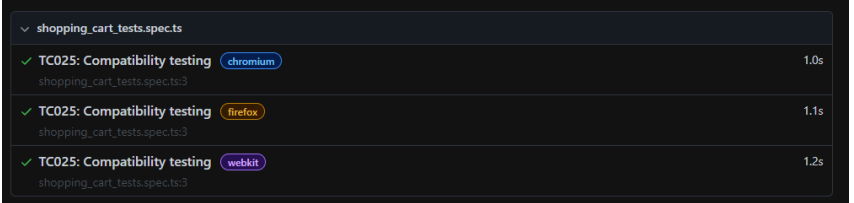
ID	Description	Preconditions	Steps for execution	Expected result	Actual Result	Status
TC019	Testing to ensure that the system can handle a moderate amount of using simultaneously accessing the homepage	>no users are currently accessing the page	1)using a tool such as K6 set the number of users to 500 2)run the load test over 1 minute 3)obtain report	Report should display the total requests with only a few failed requests in comparison to the total sent requests This can be checked by viewing the number of status code being 200		
TC020	Testing to ensure that the system can handle a large amount of using simultaneously accessing the homepage	>no users are currently accessing the page	1)using a tool such as K6 set the number of users to 5000 2)run the load test over 1 minute 3)obtain report	Report should result in more failed attempts due the server not being capable of handle such load. This can be checked by viewing the number of error codes generated such as 500,502 and 204		
TC021	Testing to see if the response time for the initial homepage is fast enough	>no users are currently accessing the page	1)using a tool such as playwright set the number of users to 1 2)run the load test over 1 minute 3)obtain report	Report should display the median response time as ~1000ms		
TC022	Testing to see if the response time for the initial homepage is fast enough with a large number of users on the website	>no users are currently accessing the page	1)using a tool such as playwright set the number of users to 5000 2)run the load test over 1 minute 3)obtain report	Report should display the median response time as ~4500ms		
TC023	Testing to see if system can handle a moderate number of transactions being processed at the same time	> no other users are currently on the website > no other transactions are being processed	1)using a tool such as K6 set the number of users checking out to 500 2)run the load test over 1 minute 3)obtain report	Report should show most if not all users being able to complete the transactions in under 5 seconds		

<b>TC024</b>	Testing to see if system can handle a large number of transactions being processed at the same time	> no other users are currently on the website > no other transactions are being processed	1)using a tool such as K6 set the number of users checking out to 5000 2)run the load test over 1 minute 3)obtain report	Report should show most if not all users being able to compete the transactions in under 5 seconds		
<b>TC025</b>	Compatibility testing on chromium-based browsers, Firefox browsers as well as Webkit browsers to ensure that functionality and ease of use	>test to be done in a headed mode >there isn't an overwhelming load on the server	1)using a tool such as playwright configure the script to run on chromium-based browsers, Firefox browsers as well as Webkit  2)in the tests include tests that check all elements of the webapp are being loaded, the shopping cart is functional etc  3)run tests in a headed mode 4)obtain report	Report should show all tests as passed		
<b>TC026</b>	Compatibility testing on Android and ISO devices to ensure that functionality and ease of use	>there isn't an overwhelming load on the server >user is on combatable browser and has internet access	1)using a tool such as playwright configure the script to run a mobile browser such as google chrome and safari  2)in the tests include tests that check all elements of the webapp are being loaded, the shopping cart is functional etc  3)run tests in a headed mode 4)obtain report	Report should show all tests as passed		



<b>TC027</b>	Compatibility testing to ensure accessibility and easy navigation on mobile devices. This mainly assess reactivity of UI elements, visual clutter and how nice the website looks	>there isn't an overwhelming load on the server >user is on combatable browser and has internet access	1)navigate the website as described in TC002,TC008 and TC012 2)While completing the above step, look for irregular UI elements 3)confirm with design team to ensure that this is how they want the application to look on mobile	All UI elements are as intended. Easy to access, placed in appropriate and recognisable places with icons that users will understand at a glance		
<b>TC028</b>	Testing accessibility for people who may be visually impaired, blind, colour blind. Overall testing perceivability	>Webapp is fully functional >WCAG 2.2 guidelines are taken into consideration >testing team is trained in testing accessibility	1)Check all UI elements are accessible using keyboard, this includes tabbing order 2)Use the webapp with a screen reader 3)Ensure that all images have alt text 4)Ensure that there is enough colour contrast 5)use screen magnifiers to ensure everything is still readable 6)Ensure that webapp is still usable without fine motor control	Web App is fully usable with keyboard with clear focus indicators and is navigable with screen reader. Images have descriptive alt text. Colour contrast meets WCAG requirements. Content is readable with magnifiers and is operable without fine motor control		

## Task2

TC021	Testing to see if the response time for the initial homepage is fast enough	>no users are currently accessing the page	1)using a tool such as playwright set the number of users to 1 2)run the load test over 1 minute 3)obtain report	Report should display the median response time as ~1000ms	 <p>shopping_cart_tests.spec.ts</p> <ul style="list-style-type: none"> <li>✓ TC021: Test homepage response time chromium 877ms shopping_cart_tests.spec.ts:10</li> <li>✓ TC021: Test homepage response time firefox 906ms shopping_cart_tests.spec.ts:10</li> <li>✓ TC021: Test homepage response time webkit 1.1s shopping_cart_tests.spec.ts:10</li> </ul>	Pass
TC025	Compatibility testing on chromium-based browsers, Firefox browsers as well as Webkit browsers to ensure that functionality and ease of use	>test to be done in a headed mode >there isn't an overwhelming load on the server	1)using a tool such as playwright configure the script to run on chromium-based browsers, Firefox browsers as well as Webkit  2)in the tests include tests that check all elements of the webapp are being loaded, the shopping cart is functional etc  3)run tests in a headed mode 4)obtain report	Report should show all tests as passed	 <p>shopping_cart_tests.spec.ts</p> <ul style="list-style-type: none"> <li>✓ TC025: Compatibility testing chromium 1.0s shopping_cart_tests.spec.ts:3</li> <li>✓ TC025: Compatibility testing firefox 1.1s shopping_cart_tests.spec.ts:3</li> <li>✓ TC025: Compatibility testing webkit 1.2s shopping_cart_tests.spec.ts:3</li> </ul>	pass

TS example.spec.ts U

TS shopping\_cart\_tests.spec.ts U ●

TS useCartProducts.ts

tests > TS shopping\_cart\_tests.spec.ts > ...

```
1 import { test, expect } from '@playwright/test';
2
3 test('TC025: Compatibility testing ', async ({ page }) => {
4
5     await page.goto('http://localhost:3000');
6     await expect(page).toHaveTitle(/Typescript React Shopping cart/);
7
8 });
9
10 test('TC021: Test homepage response time', async ({ page }) => {
11
12     const minResponseTime = 1000;
13
14     //calculating response time
15     const startT = Date.now();
16     await page.goto('http://localhost:3000');
17     const endT = Date.now();
18     const responseTime = endT - startT;
19     console.log(`Response time: ${responseTime} ms`);
20
21     expect(responseTime).toBeLessThanOrEqual(minResponseTime);
22 });
23
24
25
```

## Assumptions:

- Functionalities of the Web application is working as intended such as calculating total price of the cart
- The test user has permission to perform all actions related to the shopping cart and filters
- The cart starts empty when the user first navigates to the product listing page. This ensures that adding an item is accurately reflected when calculating the total cost.
- No filters are applied when the application is first launched
- The application is assumed to be running with its default settings and configuration.
- The items that the user is adding to cart is in stock and can be purchased
- The application is supported for different operating systems and browsers
- The application was build understanding the WCAG 2.2 concepts and regulations
- The servers that are hosting the application are able to support up to 5000 users (for TC024)
- The application is optimised for a fast response time
- The application has DDOS protection

## Additional Improvements:

### Additional test cases:

- test cases to ensure that the user can completely remove an item from the cart using the “x” button
- test cases to see the maximum number of the same product a user can purchase
- test cases to ensure accessibility by displaying current languages and potentially translation into other languages
- compatibility testing for Linux OS

### Further scenarios to cover:

- a scenario that could be addressed is users with slow internet connections
- users with slow devices or small screens
- users who are blind and hard of hearing/deaf
- users who are tech illiterate

### Tools or methodologies:

- using stressing testing software such as k6 to perform actual load testing
- split test case into more specific phases such as unit and integration testing to help the readability of the document and cover more bases
- a methodology that would be useful for test driven software development would be agile, specifically the scrum process as it allows for reviews after each sprint with unit and acceptance testing being done at regular intervals

### Document/references:

- Understanding WCAG 2.2: <https://www.gov.uk/service-manual/helping-people-to-use-your-service/understanding-wcag>

### Bugs:

- When sorting by a size that has 2 characters such as XS or XL and adding the product to the cart. The size of the item will appear as just the first character of the size. So, adding XL or XXL will just show up as X