Mahidol University
Faculty of Information
and Communication Technology

PROJECT 2

TESTING FOR AGILE SOFTWARE PROJECT

Echo Group

ITCS473 SW Quality Assurance and Testing

MOTIVATION

Currently, customers of Bash Coffee Shop order **manually** via LINE chat or in person at the store, which slows the ordering process.

Therefore, this project aims to develop a dynamic, **web-based** menu for Bash Coffee Shop, enhancing the customer experience by offering an intuitive platform to explore the menu, search for specific items, apply filters, and customize their selections to personal preferences.

TEACHNOLOGY USED

FOR DEVELOPMENT OF THE APPLICATION

TOOLS USED

FOR TEST

JEST

ROBOT FRAMEWORK

LESSONS LEARNED

THINK LIKE A USER

Learned about how a real user might use or misuse the system.
This helps to find problems that might accidentally be missed.

IMPORTANCE OF TEST PLANNING

Testing needs careful planning. It is essential to identify all functions and scenarios that need testing before starting the process.
This helps us save time and ensures nothing important is missed.

FIXING BUGS IMPROVES TESTING

Each bug found is a lesson for improving test cases.
It shows which areas of the system need more attention.

FOCUS ON EDGE CASES

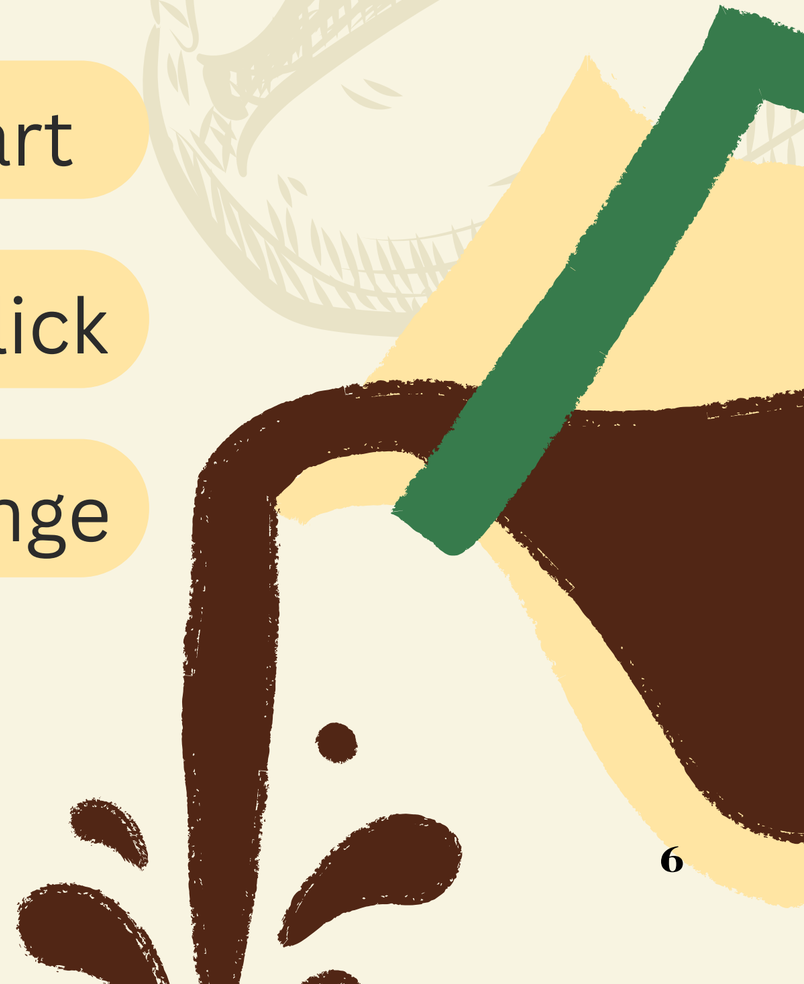
Testing edge cases like invalid inputs or unexpected user actions is as important as testing normal scenarios.
These tests help uncover hidden bugs that might cause issues.

UNIT TESTING

Test Suite 1: testHandleAddToCart

Test Suite 2: testHandleAddOnClick

Test Suite 3: testHandleSortChange



Test Suite 1: testHandleAddToCart

Goal: Test the functionality of adding an item to the cart

Identify testable functions

- Testable function: handleAddToCart()

Identify parameters, return types, return values, and exceptional behavior

- **Parameters:**
  - product: Product | null,
  - selectedAddOns: AddOn[],
  - quantity: number,
  - displayTotalPrice: number,
  - addToCart: (item: CartItemType) => number, // returns cart length or updated cart
  - category = “”
  - sweetness = ""
  - type = ""
- **Return type:** object
- **Return value:**
  - { success: false, message: "Product not found" } if the product is null
  - { success: true, cartLength: number } if adding a product to the cart succeeds, where cartLength represents the updated cart size.
- **Exceptional behavior:**
  - If the product to be added to the cart is null, it will return { success: false, message: "Product not found" }
  - If the quantity of the product to be added to the cart is invalid (0 or negative), it will return { success: false, message: "Product quantity is invalid" }
  - If the total price of the product to be added to the cart is invalid (0 or negative), it will return { success: false, message: "Product total price is invalid" }

Test Suite 1: testHandleAddToCart

Model the input domain

- **Develop characteristics**
  - C1: The product is null
  - C2: The selectedAddOns are empty
  - C3: The value of quantity
  - C4: The value of displayTotalPrice
  - C5: The category is empty
  - C6: The sweetness is empty
  - C7: The type is empty

Characteristic	b1	b2	b3
C1: The product is null	True	False	
C2: The selectedAddOns are empty	True	False	
C3: The value of quantity	Quantity <= 0	Quantity > 0	Quantity is not a number
C4: The value of displayTotalPrice	displayTotalPrice <= 0	displayTotalPrice > 0	displayTotalPrice is not a number
C5: The category is empty	True	False	
C6: The sweetness is empty	True	False	
C7: The type is empty	True	False	

Test Suite 1: testHandleAddToCart

Combine partitions to define test requirements

- **BCC Approach**
  - Test requirements: Number of tests = 10 - (1 Infeasible)

Test Case	Test Value	Expected result
T1 (C1b2, C2b2, C3b2, C4b2, C5b2, C6b2, C7b2)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "Beverage", AddOns: [{ name: "Oat Milk", price: 10 }], };selectedAddOns = [{ name: "Oat Milk", price: 10 }];quantity = 1;displayTotalPrice = 60;sweetness = "50%";type = “Hot”;	{ success: true, cartLength: 1 }
T2 (C1b2, C2b1, C3b2, C4b2, C5b2, C6b2, C7b2)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "Beverage", AddOns: [ ], };selectedAddOns = [ ];quantity = 1;displayTotalPrice = 60;sweetness = "50%";type = “Hot”;	{ success: true, cartLength: 1 }
T3 (C1b2, C2b2, C3b1, C4b2, C5b2, C6b2, C7b2)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "Beverage", selectedAddOns = [{ name: "Oat Milk", price: 10 }]; };selectedAddOns = [{ name: "Oat Milk", price: 10 }];quantity = 0;displayTotalPrice = 60;sweetness = "50%";type = “Hot”;	{ success: false, message: "Product quantity is invalid" }

Test Suite 1: testHandleAddToCart

T4 (C1b2, C2b2, C3b3, C4b2, C5b2, C6b2, C7b2)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "Beverage", AddOns: [ ], };selectedAddOns = [{ name: "Oat Milk", price: 10 }];quantity = NaN;displayTotalPrice = 60;sweetness = "50%";type = “Hot”;	{ success: false, message: "Product quantity is invalid" }
T5 (C1b2, C2b2, C3b2, C4b1, C5b2, C6b2, C7b2)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "Beverage", selectedAddOns = [{ name: "Oat Milk", price: 10 }]; };selectedAddOns = [{ name: "Oat Milk", price: 10 }];quantity = 1;displayTotalPrice = -1;sweetness = "50%";type = “Hot”;	{ success: false, message: "Product total price is invalid"}
T6 (C1b2, C2b2, C3b2, C4b3, C5b2, C6b2, C7b2)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "Beverage", selectedAddOns = [{ name: "Oat Milk", price: 10 }]; };selectedAddOns = [{ name: "Oat Milk", price: 10 }];quantity = 1;displayTotalPrice = NaN;sweetness = "50%";type = “Hot”;	{ success: false, message: "Product total price is invalid"}

Test Suite 1: testHandleAddToCart

T7(C1b2, C2b2, C3b2, C4b2, C5b1, C6b2, C7b2)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "", selectedAddOns = [{ name: "Oat Milk", price: 10 }]; };selectedAddOns = [{ name: "Oat Milk", price: 10 }];quantity = 1;displayTotalPrice = 60;sweetness = "50%";type = “Hot”;	{ success: false, message: "Product category is invalid"}
T8 (C1b2, C2b2, C3b2, C4b2, C5b2, C6b1, C7b2)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "Beverage", selectedAddOns = [{ name: "Oat Milk", price: 10 }]; };selectedAddOns = [{ name: "Oat Milk", price: 10 }];quantity = 1;displayTotalPrice = 60;sweetness = "";type = “Hot”;	{ success: false, message: "Product sweetness or type is invalid"}
T9 (C1b2, C2b2, C3b2, C4b2, C5b2, C6b2, C7b1)	product = { id: 101, Drink_Name: "Americano", Description: "A rich espresso with added water for a smooth coffee.", Price: { hotPrice: 50, coldPrice: 55 }, DrinkType: "Hot/Cold", Tag: ["Coffee"], isRecommended: true, img_src: "/images/americano.png", category: "Beverage", AddOns: [{ name: "Oat Milk", price: 10 }], };selectedAddOns = [{ name: "Oat Milk", price: 10 }];quantity = 1;displayTotalPrice = 60;sweetness = "50%";type = “”;	{ success: false, message: "Product sweetness or type is invalid"}

Test Suite 2: testHandleAddOnClick

Goal: Test the functionality of toggling the selection state of an add-on in the selected AddOns

Identify testable functions

- Testable function: handleAddOnClick()

Identify parameters, return types, return values, and exceptional behavior

- Parameters:
  - selectedAddOn: AddOn, // new add-on
  - selectedAddOns: AddOn[] // array of current selected add-ons
- Return type: array
- Return value:
  - If the add-on is already selected (exists in the list), it removes it from the selectedAddOns and returns the new list of selectedAddOns.
  - If the add-on is not already selected, it adds it to the selectedAddOns and returns the new list of selectedAddOns.
- Exceptional behavior: -

Model the input domain

- **Develop characteristics**
  - C1: The initial state of the selectedAddOns list is empty.
  - C2: The selectedAddOn already exists in the selectedAddOns list

Characteristic	b1	b2
C1 = The initial state of selectedAddOns list is empty	True	False
C2 = The selectedAddOn already exists in the selectedAddOns list	True	False

Test Suite 2: testHandleAddOnClick

Combine partitions to define test requirements

- **ACoC Approach**
  - Test requirements: Number of tests = 4 -> (1 Infeasible)

Test Case	Test Value	Expected result
T2 (C1b1, C2b2)	selectedAddOn = { name: "Oat Milk", price: 15 } selectedAddOns = [];	[{ name: "Oat Milk", price: 15 }]
T3 (C1b2, C2b1)	selectedAddOn = { name: "Oat Milk", price: 15 } selectedAddOns = [{ name: "Oat Milk", price: 15 }]	[ ] (Remove the add-on from list)
T4 (C1b2, C2b2)	selectedAddOn = { name: "Brown Sugar Jelly", price: 15 } selectedAddOns = [{ name: "Oat Milk", price: 15 }]	[{ name: "Oat Milk", price: 15 }, { name: "Brown Sugar Jelly", price: 15 }]

Test Suite 3: testHandleSortChange

Goal: Goal: Test the functionality of searching for an item

Identify testable functions

- Testable function: handleSortChange ()

Identify parameters, return types, return values, and exceptional behavior

- Parameters: Product[] updatedProducts, String sortOrder
- Return type: Product[]
- Return value: the unsorted or sorted products by price in ascending or descending.
- Exceptional behavior:

Model the input domain

- **Develop characteristics**
  - C1: length of updatedProducts array
  - C2: sortOder value
  - C3 = hotPrice value
  - C4 = coldPrice value

Characteristic	b1	b2	b3
C1 = length of updatedProducts	Empty	Single Product	Multiple Products
C2 = sortOder value	Ascending	Descending	Invalid value
C3 = hotPrice value	Valid	Invalid	
C4 = coldPrice value	Valid	Invalid	

Test Suite 3: testHandleSortChange

Combine partitions to define test requirements

- **BCC Approach**
  - Base choice: C1b3, C2b1, C3b1, C4b1
  - Test requirements: Number of tests = 7

Test Case	Test Value	Expected result
T1 (C1b3, C2b1, C3b1, C4b1)	handleSortChange ([{ id: 1, name: "Dirty", description: "A rich espresso shot served over cold milk, resulting in a bold flavor contrast.", **hotPrice: "85", coldPrice: "90"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: true, image: "latte.png", Tag: ["Coffee","Recommend"]}, { id: 2, name: "Americano", description: "An Americano is made by diluting an espresso shot with hot water for a smooth, robust coffee.", **hotPrice: "55", coldPrice: "60"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "americano.png", Tag: ["Coffee"]}, { id: 3, name: "Latte", description: "A smooth blend of espresso and steamed milk with a creamy finish.", **hotPrice: "60", coldPrice: "65"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "latte.png", Tag: ["Coffee", "Milk"]}], "**Price Low to High**")	[{ id: 2, ... }, { id: 3, ... }, { id: 1, ... }]
T2 (C1b3, C2b1, C3b1, C4b2)	handleSortChange ([{ id: 1, name: "Dirty", description: "A rich espresso shot served over cold milk, resulting in a bold flavor contrast.", **hotPrice: "85", coldPrice: "-"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: true, image: "latte.png", Tag: ["Coffee","Recommend"]}, { id: 2, name: "Americano", description: "An Americano is made by diluting an espresso shot with hot water for a smooth, robust coffee.", **hotPrice: "55", coldPrice: "-"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "americano.png", Tag: ["Coffee"]}, { id: 3, name: "Latte", description: "A smooth blend of espresso and steamed milk with a creamy finish.", **hotPrice: "60", coldPrice: "-"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "latte.png", Tag: ["Coffee", "Milk"]}], "**Price Low to High**")	[{ id: 2, ... }, { id: 3, ... }, { id: 1, ... }]
T3 (C1b3, C2b1, C3b2, C4b1)	handleSortChange ([{ id: 1, name: "Dirty", description: "A rich espresso shot served over cold milk, resulting in a bold flavor contrast.", **hotPrice: "-", coldPrice: "90",** category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: true, image: "latte.png", Tag: ["Coffee","Recommend"]}, { id: 2, name: "Americano", description: "An Americano is made by diluting an espresso shot with hot water for a smooth, robust coffee.", **hotPrice: "-", coldPrice: "60"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "americano.png", Tag: ["Coffee"]}, { id: 3, name: "Latte", description: "A smooth blend of espresso and steamed milk with a creamy finish.", **hotPrice: "-", coldPrice: "65"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "latte.png", Tag: ["Coffee", "Milk"]}], "**Price Low to High**")	[{ id: 2, ... }, { id: 3, ... }, { id: 1, ... }]

Test Suite 3: testHandleSortChange

Test Case	Test Value	Expected result
T4 (C1b3, C2b2, C3b1, C4b1)	handleSortChange ([{ id: 1, name: "Dirty", description: "A rich espresso shot served over cold milk, resulting in a bold flavor contrast.", **hotPrice: "85", coldPrice: "90"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: true, image: "latte.png", Tag: ["Coffee","Recommend"]}, { id: 2, name: "Americano", description: "An Americano is made by diluting an espresso shot with hot water for a smooth, robust coffee.", **hotPrice: "55", coldPrice: "60"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "americano.png", Tag: ["Coffee"]}, { id: 3, name: "Latte", description: "A smooth blend of espresso and steamed milk with a creamy finish.", **hotPrice: "60", coldPrice: "65**", category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "latte.png", Tag: ["Coffee", "Milk"]}], "**Price High to Low**")	[{ id: 1, ... }, { id: 3, ... }, { id: 2, ... }]
T5 (C1b3, C2b3, C3b1, C4b1)	handleSortChange ([{ id: 1, name: "Dirty", description: "A rich espresso shot served over cold milk, resulting in a bold flavor contrast.", **hotPrice: "85", coldPrice: "90"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: true, image: "latte.png", Tag: ["Coffee","Recommend"]}, { id: 2, name: "Americano", description: "An Americano is made by diluting an espresso shot with hot water for a smooth, robust coffee.", **hotPrice: "55", coldPrice: "60"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "americano.png", Tag: ["Coffee"]}, { id: 3, name: "Latte", description: "A smooth blend of espresso and steamed milk with a creamy finish.", **hotPrice: "60", coldPrice: "65**", category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: false, image: "latte.png", Tag: ["Coffee", "Milk"]}], "**Invalid value**")	[{ id: 1, ... }, { id: 2, ... }, { id: 3, ... }]
T6 (C1b2, C2b3, C3b1, C4b1)	handleSortChange ([{ id: 1, name: "Dirty", description: "A rich espresso shot served over cold milk, resulting in a bold flavor contrast.", **hotPrice: "50", coldPrice: "60"**, category: "Drink", TypeOfDrinks: "Hot/Cold", isRecommended: true, image: "latte.png", Tag: ["Coffee","Recommend"]}], "**Price Low to High**")	[{ id: 1, ... }]

SYSTEM TESTING

Search Functionality Module

Sort Functionality Module

Add to cart Functionality Module

SEARCH MUDULE

Test Case 1.1: Search	
Test Case ID: S-01	**Test Designed by:** Echo
Test priority: Medium	**Test Designed Date:** 14/11/2024
Module Name: Search Functionality	**Test Executed By:** Echo
Test Title: Verify search menu item with valid inputs	**Test Execution Date:** 14/11/2024
Description: Ensure the search functionality works for valid inputs.	
Pre-conditions: The frontend is running on http://localhost:3000/ and the backend web service of http://localhost:3030/ is running.	
Dependencies: A database with searchable menu items.	

Step	Test Steps	Test Data	Expected Result	Actual result	Status	Notes
1.	Open Browser	N/A	The browser is opened.	The browser is opened.	Pass	
2.	Launch the website	N/A	The website homepage is displayed	Bash Coffee homepage(Navigate to 'http://localhost:3000/' successfully)	Pass	
3.	Enter the valid menu item name	“Espresso”	Match Espresso menu	The product card of Espresso is displayed in the explore menu section and the number of matched items is one.	Pass	
4.	Close Browser	N/A	Browser is closed	Browser is closed	Pass	

- **Post-condition: The browser is closed, and no further interactions are possible.**

SEARCH MUDULE

Test Case 1.2: Search	
Test Case ID: S-02	**Test Designed by:** Echo
Test priority: Medium	**Test Designed Date:** 14/11/2024
Module Name: Search Functionality	**Test Executed By:** Echo
Test Title: Verify search menu item with invalid inputs	**Test Execution Date:** 14/11/2024
Description: Ensure the search functionality works for invalid inputs.	
Pre-conditions: The frontend is running on http://localhost:3000/ and the backend web service of http://localhost:3030/ is running.	
Dependencies: A database with searchable menu items.	

Step	Test Steps	Test Data	Expected Result	Actual result	Status	Notes
1.	Open Browser	N/A	The browser is opened.	The browser is opened.	Pass	
2.	Launch the website	N/A	The website homepage is displayed	Bash Coffee homepage(Navigate to 'http://localhost:3000/' successfully)	Pass	
3.	Enter invalid menu item name	“zzz”	Not match the menu item	The Explore Our Menu section shows a ‘No result’ image.	Pass	
4.	Close Browser	N/A	Browser is closed	Browser is closed	Pass	

- **Post-condition: The browser is closed, and no further interactions are possible.**

SEARCH MUDULE

Test Case 1.3: Search	
Test Case ID: S-03	**Test Designed by**: Echo
Test priority: Medium	**Test Designed Date:** 14/11/2024
Module Name: Search Functionality	**Test Executed By**: Echo
Test Title: Verify search menu item that returns multiple matched items	**Test Execution Date:** 14/11/2024
Description: Ensure the search functionality works for multiple matched items.	
Pre-conditions: The frontend is running on http://localhost:3000/ and the backend web service of http://localhost:3030/ is running.	
Dependencies: A database with searchable menu items.	

Step	Test Steps	Test Data	Expected Result	Actual result	Status	Notes
1.	Open Browser	N/A	The browser is opened.	The browser is opened.	Pass	
2.	Launch the website	N/A	The website homepage is displayed	Bash Coffee homepage(Navigate to 'http://localhost:3000/' successfully)	Pass	
3.	Enter the name of the menu	“matcha”	The website will display the menu items that include "matcha" in their name and the total number of matched items is 5.	The website displays the 5 menus that include "matcha" in their name.	Pass	
4.	Close Browser	N/A	Browser is closed	Browser is closed	Pass	

Post-condition: The browser is closed, and no further interactions are possible.

SORT MODULE

Test Case 2.1: Sort by Price, high to low						
Test Case ID: P-01			**Test Designed by**: Echo			
Test priority: Medium			**Test Designed Date**: 14/11/2024			
Module Name: Sort Functionality			**Test Executed By**: Echo			
Test Title: Verify sorting by price, high to low			**Test Execution Date**: 14/11/2024			
Description: Ensure that the sort functionality works correctly to display menu items sorted by price, from high to low.						
Pre-conditions: The frontend is running on http://localhost:3000/ and the backend web service of http://localhost:3030/ is running.						
Dependencies: A database with searchable menu items.						
Step	**Test Steps**	**Test Data**	**Expected Result**	**Actual result**	**Status**	**Notes**
1.	Open Browser	N/A	The browser is opened.	The browser is opened.	Pass	
2.	Launch the website	N/A	The website homepage is displayed	Bash Coffee homepage(Navigate to 'http://localhost:3000/' successfully)	Pass	
3.	Select "Sort by Price: High to Low" option	button_SortPriceHighLow	The website displays menu items sorted by price in descending order.	Items are displayed in descending order by price.	Pass	
4.	Close Browser	N/A	Browser is closed	Browser is closed	Pass	
Post-condition: The browser is closed, and no further interactions are possible.						

SORT MODULE

Test Case 2.2: Sort by Price, low to high						
Test Case ID: P-02			**Test Designed by**: Echo			
Test priority: Medium			**Test Designed Date**: 14/11/2024			
Module Name: Sort Functionality			**Test Executed By**: Echo			
Test Title: Verify sorting by price, low to high			**Test Execution Date**: 14/11/2024			
Description: Ensure that the sort functionality works correctly to display menu items sorted by price, from low to high.						
Pre-conditions: The frontend is running on http://localhost:3000/ and the backend web service of http://localhost:3030/ is running.						
Dependencies: A database with searchable menu items.						
Step	**Test Steps**	**Test Data**	**Expected Result**	**Actual result**	**Status**	**Notes**
1.	Open Browser	N/A	The browser is opened.	The browser is opened.	Pass	
2.	Launch the website	N/A	The website homepage is displayed	Bash Coffee homepage(Navigate to 'http://localhost:3000/' successfully)	Pass	
3.	Select "Sort by Price: Low to High" option	button_SortPriceLowHigh	The website displays menu items sorted by price in ascending order.	Items are displayed in ascending order by price.	Pass	
4.	Close Browser	N/A	Browser is closed	Browser is closed	Pass	
Post-condition: The browser is closed, and no further interactions are possible.						

ADD TO CART MODULE

Test Case 3.1: Add to cart						
Test Case ID: A-01			**Test Designed by:** Echo			
Test priority: High			**Test Designed Date**: 14/11/2024			
Module Name: Add to cart Functionality			**Test Executed By**: Echo			
Test Title: Verify both select adding add-ons for drinks			**Test Execution Date:** 14/11/2024			
Description: Ensure users can add optional add-ons (e.g., milk, sugar) to their drink orders.						
Pre-conditions: The frontend is running on http://localhost:3000/ and the backend web service of http://localhost:3030/ is running.						
Step	**Test Steps**	**Test Data**	**Expected Result**	**Actual result**	**Status**	**Notes**
1.	Open Browser	N/A	The browser is opened.	The browser is opened.	Pass	
2.	Launch the website	N/A	The website homepage is displayed.	Bash Coffee homepage(Navigate to 'http://localhost:3000/' successfully)	Pass	
3.	Scroll down the page to find a Latte product card and click on the '+' button located on the Latte product card.	button_	The website will show the Latte menu.	The website will show the Latte menu.	Pass	
4.	Select the 'Cold' button to change the drink to 'Cold'.	button_Cold	The drink option changes to Cold.	The drink option changes to Cold.	Pass	
5.	Choose an add-on of Oat milk (15.-).	button_Oat Milk (15.-)	The Oat milk (15.-) is selected, and the price is updated.	The Oat milk (15.-) is selected, and the price is updated to 80.	Pass	
6.	Choose another add-on of Brown Sugar Jelly (15.-).	button_Brown Sugar Jelly (15.-).	The second add-on is selected, and the price is updated.	The Brown Sugar Jelly (15.-) is selected, and the price is updated to 95.	Pass	
7.	Change the sweetness to 100%.	button_100	The sweetness changes to 100.	The sweetness changes to 100.	Pass	
8.	Click add to cart button	button_Add to cart	The message displays "Item added to cart successfully!"	The message displays "Item added to cart successfully!"	Pass	
9.	Close Browser	N/A	Browser is closed	Browser is closed	Pass	
Post-condition: The browser is closed, and no further interactions are possible.						

ADD TO CART MODULE

Test Case 3.2: Add to cart						
Test Case ID: A-02			**Test Designed by:** Echo			
Test priority: High			**Test Designed Date**: 14/11/2024			
Module Name: Add to cart Functionality			**Test Executed By:** Echo			
Test Title: Verify non-select adding add-ons for drinks			**Test Execution Date:** 14/11/2024			
Description: Ensure users can add optional add-ons (e.g., milk, sugar) to their drink orders.						
1.	**Open Browser**	**N/A**	**The browser is opened.**	**The browser is opened.**	**Pass**	**Notes**
2.	Launch the website	N/A	The website homepage is displayed.	Coffee homepage(Navigate to 'http://localhost:3000/' successfully)	Pass	
3.	Scroll down the page to find a Pinky Milk product card	N/A	The website does not show the Pinky Milk menu.	The website does not show the Pinky Milk menu.	Pass	
4.	Navigate the pagination to the next page	Next button	The website redirects to the next page.	The website redirects to the next page.	Pass	
5.	Scroll down the page to find a Pinky Milk product card and click on the '+' button on the Latte product card.	button_Plus	The website shows the Pinky Milk menu.	The website shows the Pinky Milk menu.	Pass	
4.	Select the 'Cold' button to change the drink to 'Cold'.	button_Cold	The drink option changes to Cold.	The drink option changes to Cold.	Pass	
5.	Change the sweetness to 100%.	button_100	The sweetness changes to 100.	The sweetness changes to 100.	Pass	
6.	Click add to cart button	button_Add to cart	The message displays "Item added to cart successfully!"	The message displays "Item added to cart successfully!"	Pass	
7.	Close the success popup	button_close	The success popup is closed.	The browser is closed and the system navigates the user to the homepage.	Pass	
8.	Close Browser	N/A	Browser is closed	Browser is closed	Pass	
Post-condition: The browser is closed, and no further interactions are possible.						

COVERAGE REPORT

All files

98.3% Statements `116/118` **82.35%** Branches `28/34` **100%** Functions `3/3` **98.3%** Lines `116/118`

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
utilsCart.ts	97.61%	82/84	81.81%	18/22	100%	2/2	97.61%	82/84
utilsSort.ts	100%	34/34	83.33%	10/12	100%	1/1	100%	34/34

testHandleAddToCart & testHandleAddOnClick
testHandleSortChange

COVERAGE REPORT

Search Test:

All | Tags | Suites | Search

Status: 3 tests total, 3 passed, 0 failed, 0 skipped

Total Time: 00:00:29.775

Name	Documentation	Tags	Status	Message	Elapsed	Start / End
SearchTest . Test No Results Search	Verify that the website displays a "No Result" image when no items match the search term.		PASS		00:00:09.127	20241115 01:25:44.183 20241115 01:25:53.310
SearchTest . Test Multiple Matched Search	Verify that the website works for multiple matched items.		PASS		00:00:10.557	20241115 01:25:53.312 20241115 01:26:03.869
SearchTest . Test Valid Menu Item Search	Verify that the website accepts a valid menu item name and displays the correct matching count.		PASS		00:00:10.091	20241115 01:25:34.091 20241115 01:25:44.182

[Automated Test]

COVERAGE REPORT

Sort Test:

LOG

SortTest Report

Generated
20241115 01:28:56 UTC+07:00
13 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20241115 01:28:37.900
End Time:	20241115 01:28:56.291
Elapsed Time:	00:00:18.391
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	2	2	0	0	00:00:17	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
SortTest	2	2	0	0	00:00:18	

Name	Documentation	Tags	Status	Message	Elapsed	Start / End
SortTest . Test Sort by Price from High to Low			PASS		00:00:09.151	20241115 01:28:47.138 20241115 01:28:56.289
SortTest . Test Sort by Price from Low to High			PASS		00:00:08.207	20241115 01:28:38.930 20241115 01:28:47.137

[Automated Test]

COVERAGE REPORT

Add to CartTest:

LOG

AddToCartTest Report

Generated
20241115 01:23:54 UTC+07:00
15 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20241115 01:23:17.807
End Time:	20241115 01:23:54.237
Elapsed Time:	00:00:36.430
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	2	2	0	0	00:00:36	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
AddToCartTest	2	2	0	0	00:00:36	

Test Details

All | Tags | Suites | Search

Status:	2 tests total, 2 passed, 0 failed, 0 skipped
Total Time:	00:00:36.135

Name	Documentation	Tags	Status	Message	Elapsed	Start / End
AddToCartTest.Test Non Select Adding Add Ons For Drinks IN NEXT PAGE			PASS		00:00:22.044	20241115 01:23:32.190 20241115 01:23:54.234
AddToCartTest.Test Valid Menu Item Search And Add To Cart			PASS		00:00:14.091	20241115 01:23:18.099 20241115 01:23:32.190

[Automated Test]

CI INTEGRATION

```yaml
name: NodeJS with Webpack

on:
 pull_request:
 branches: [ "main" ]

jobs:
 build:
 runs-on: ubuntu-latest

 strategy:
 matrix:
 node-version: [18.x, 20.x, 22.x]

 steps:
 - uses: actions/checkout@v4

 - name: Use Node.js ${{ matrix.node-version }}
 uses: actions/setup-node@v4
 with:
 node-version: ${{ matrix.node-version }}

 - name: Install Dependencies
 run: |
 npm install
 npm install -D webpack-cli

 - name: Build
 run: npx webpack

 - name: Run Unit Tests
  run: npm test

 - name: Jest Coverage Comment
  id: coverageComment
  uses: MishaKav/jest-coverage-comment@main
  with:
   hide-comment: true
   coverage-summary-path: ./coverage/coverage-summary.json

 - name: Check the output coverage
  run: |
   echo "Coverage Percentage - ${{ steps.coverageComment.outputs.coverage }}"
   echo "Coverage Color - ${{ steps.coverageComment.outputs.color }}"
   echo "Summary HTML - ${{ steps.coverageComment.outputs.summaryHtml }}"

 - name: Create the badge
  if: github.ref == 'refs/heads/main'
  uses: schneegans/dynamic-badges-action@v1.6.0
  with:
   auth: ${{ secrets.JEST_COVERAGE_COMMENT }}
   gistID: 5e90d640f8c212ab7bbac38f72323f80
   filename: jest-coverage-comment__main.json
   label: Coverage
   message: ${{ steps.coverageComment.outputs.coverage }}%
   color: ${{ steps.coverageComment.outputs.color }}
   namedLogo: javascript
```

CI INTEGRATION

README

Bash Coffee Shop Frontend ☕

NodeJS with Webpack passing | Coverage 97%

This project is the frontend application for the **Bash Coffee Shop**, designed to provide a dynamic, user-friendly web-based interface for customers to browse the menu, search for items, sort by preferences, and customize orders before adding them to the cart.

This project was developed by using Next.js, performed unit tests using Jest, and implemented automated UI tests using Robot Framework.

THANK YOU

ECHO GROUP