Satwika Nino Wandhana 23/516202/PA/22066
Daniel Winston Mandela Tulung 23/516260/PA/22080
Mohammad Azka Khairur Rahman 23/511608/PA/21830
Adam Maulana Haq 23/511623/PA/21832 (AFK)
Rindra Adriansyah 23/511559/PA//21820 (AFK)

🛒 Cart Module — Unit Test Documentation

Overview

Our e-commerce website allows users to browse and purchase products online. The shopping cart is a key part of the website, where users can add products they want to buy, update the quantity, see the total price, and remove items before checking out.

To ensure the shopping cart works correctly, we write automated tests using Jest, a popular JavaScript testing framework. Jest helps us quickly verify that all cart functions behave as expected, so we can fix problems early and keep our website reliable.

This document explains the unit tests written for the `cart` module, which manages adding, removing, and updating products in a shopping cart.

Each test checks a core functionality to ensure the cart behaves as expected.

Setup

- **Mock Data**: 3 sample products used for testing.
- **Reset**: Before each test, the cart is reset to an empty state to avoid side effects.

```
beforeEach(() => {
  cart._reset();
});
```

Test Cases & Explanations

1. Add Item to Cart

```
test('add item to cart', () => {
```

```
  const product = mockProducts[0];
  cart.addItem(product.name, '67a1f4e43f34a77b6dde9144', product.price);
  expect(cart.getItems()).toEqual([{ name: product.name, quantity: 1,
price: product.price }]);
});
```

Explanation:
Checks that adding an item to the cart creates an entry with the correct product name, price, and sets the quantity to 1 by default.

2. Remove Item from Cart

```
test('remove item from cart', () => {
  const product = mockProducts[1];
  cart.addItem(product.name, '67a1f52e3f34a77b6dde914a', product.price);
  cart.removeItem(product.name);
  expect(cart.getItems()).toEqual([]);
});
```

Explanation:
Confirms that removing a product by name deletes it from the cart, resulting in an empty cart if it was the only item.

3. Update Total Price Based on Quantity

```
test('update total price based on quantity', () => {
  const product = mockProducts[2];
  cart.addItem(product.name, '67a1f5663f34a77b6dde914c', product.price);
  expect(cart.getTotalPrice()).toBe(product.price); // 899.99
  cart.updateQuantity(product.name, 5);
  expect(cart.getTotalPrice()).toBe(product.price * 5); // 4499.95
});
```

Explanation:
Validates that the total price reflects changes when item quantity is updated. Initially, the price matches one unit; after updating quantity, total price is multiplied accordingly.

4. Add Item Increases Cart Size

```
test('add item to cart increases cart size', () => {
  cart.addItem('Apple AirPods Pro 2nd gen', 1, 10);
  expect(cart.getItems().length).toBe(1);
```

```
  cart.addItem('Bose QuietComfort 45', 1, 20);
  expect(cart.getItems().length).toBe(2);
});
```

Explanation:
Tests that adding different products increases the number of distinct items in the cart.

5. Remove Item Decreases Cart Size

```
test('remove item from cart decreases cart size', () => {
  cart.addItem('Apple AirPods Pro 2nd gen', 1, 10);
  cart.addItem('Bose QuietComfort 45', 1, 20);
  cart.removeItem('Apple AirPods Pro 2nd gen');
  expect(cart.getItems().length).toBe(1);
  cart.removeItem('Bose QuietComfort 45');
  expect(cart.getItems().length).toBe(0);
});
```

Explanation:
Verifies that removing items decreases the number of distinct items in the cart until it is empty.

6. Update Quantity Changes Quantity but Not Cart Size

```
test('update quantity does not change cart size but affects total
quantity', () => {
    cart.addItem('Apple AirPods Pro 2nd gen', 1, 10); // Add a product
    expect(cart.getItems().length).toBe(1); // Cart size should be 1
    cart.updateQuantity('Apple AirPods Pro 2nd gen', 5); // Update
quantity
    expect(cart.getItems()[0].quantity).toBe(5); // Quantity should be
updated to 5
});
```

Explanation:
Checks that changing the quantity updates the item's quantity property without adding or removing items from the cart.

Summary

Functionality	Tested Behavior
Add Item	Adds product with quantity 1 and correct price
Remove Item	Removes product completely, updates cart size
Update Quantity	Changes quantity without altering cart size
Total Price	Correctly reflects quantity × price
Cart Size	Reflects number of distinct products

```
PS C:\Users\Daniel\Documents\UNIVERSITY\Semester 4 2025\WRPL\QuickCart> npx jest
 PASS  __tests__/cart.test.js
  √ add item to cart (3 ms)
  √ remove item from cart (1 ms)
  √ update total price based on quantity (1 ms)
  √ add item to cart increases cart size (1 ms)
  √ remove item from cart decreases cart size (1 ms)
  √ update quantity does not change cart size but affects total quantity (1 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   94.11 |       50 |     100 |   91.66 |
 cart.js  |   94.11 |       50 |     100 |   91.66 | 7
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        0.543 s, estimated 1 s
Ran all test suites.
```

Github link: https://github.com/danielwinstonmandela/WorkshopRPLDIUP.git