

COMSATS UNIVERSITY ISLAMABAD

(ATTOCK CAMPUS)



Assignment No: 01

Mobile App Development

Submitted to

Sir Muhammad Kamran

Submitted by

Hashir Fayyaz

SP22-Bse-037

INTRODUCTION:

The primary objective of this code is to implement a simple shopping cart system that allows users to manage their selected products efficiently. It provides functionality for adding, removing, and updating product quantities, calculating total costs, applying discounts, and displaying a summary of the items in the cart.

Operations Implemented:

- **Add Items to the Cart**
 - **Function:** `addItemToCart(productId, productName, quantity, price)``
 - **Operation:** Adds a new product to the cart or updates the quantity of an existing product if it is already in the cart.
- **Remove Items from the Cart**
 - **Function:** `removeItemFromCart(productId)``
 - **Operation:** Removes a product from the cart based on its unique ``productId``.
- **Update Item Quantity**
 - **Function:** `updateItemQuantity(productId, newQuantity)``
 - **Operation:** Updates the quantity of a specified product in the cart.
- **Calculate Total Cost**
 - **Function:** `calculateTotalCost()``
 - **Operation:** Computes the total cost of all items in the cart by multiplying each item's price by its quantity and summing them up.
- **Display Cart Summary**
 - **Function:** `displayCartSummary()``
 - **Operation:** Generates and logs a summary of items in the cart, including product names, quantities, and total prices for each item.
- **Filter Out Items with Zero Quantity**
 - **Function:** `filterZeroQuantityItems()``
 - **Operation:** Removes any items from the cart that have a quantity of zero.
- **Apply Discount Code**
 - **Function:** `applyDiscount(discountCode)``
 - **Operation:** Checks for valid discount codes and applies them to the total cost if applicable.

Code Expalanation:

Logics:

1. Add Items to the Cart:

```
javascript
const addItemToCart = (productId, productName, quantity, price) => { ... }
```

- **Logic:** This function checks if an item with the given `productId` already exists in the `cart`. If it does, it increments the existing item's quantity by the specified amount. If not, it creates a new product object and adds it to the cart.

2. Remove Items from the Cart:

```
javascript
const removeItemFromCart = (productId) => { ... }
...
```

- **Logic:** This function searches for the index of the product in the cart using its `productId`. If found, it removes that product from the cart using the `splice` method.

3. Update Item Quantity:

```
javascript
const updateItemQuantity = (productId, newQuantity) => { ... }
...
```

- **Logic:** This function iterates through the cart and updates the quantity of the item that matches the provided `productId`. It uses `map` to create a new array with updated quantities while keeping other items unchanged.

4. Calculate Total Cost:

```
`javascript
const calculateTotalCost = () => { ... }
...
```

- **Logic:** This function calculates the total cost of all items in the cart by using `reduce`. It multiplies each item's price by its quantity and accumulates these values to return the total.

5. Display Cart Summary:

```
`javascript
const displayCartSummary = () => { ... }
...
```

- **Logic:** This function generates a summary of items in the cart by mapping over each item and creating an object that includes the product name, quantity, and total price for each item. It then logs this summary to the console.

6. Filter Out Items with Zero Quantity

```
javascript
const filterZeroQuantityItems = () => { ... }
...
```

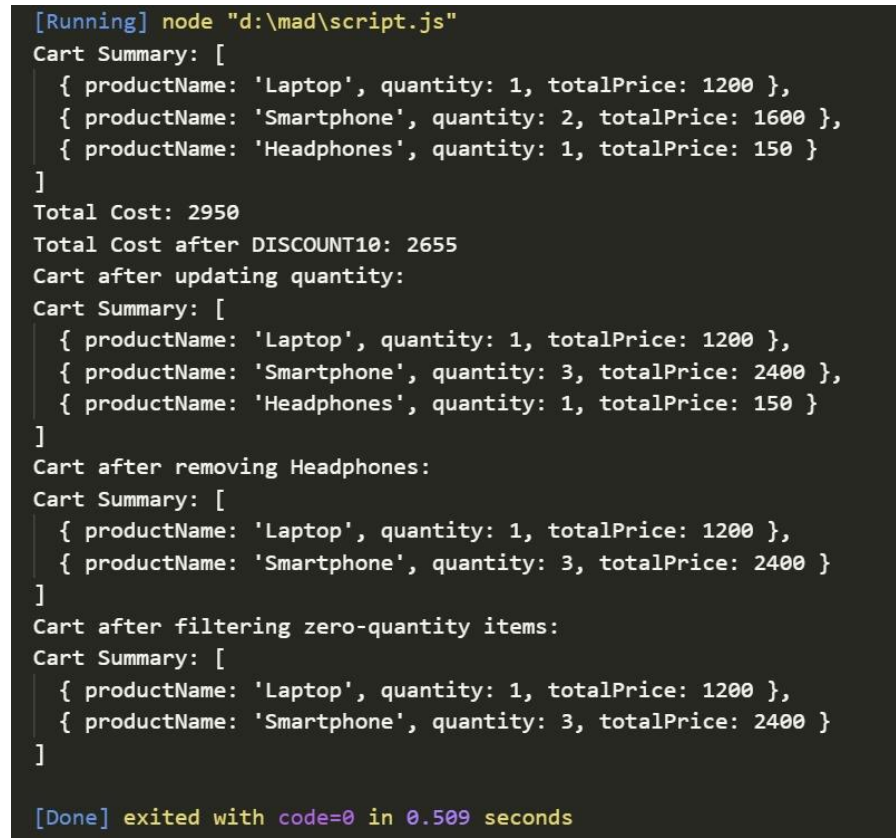
- **Logic:** This function filters out any items from the cart that have a quantity of zero using the `filter` method. It updates the `cart` array to only include items with a positive quantity.

7. Apply Discount Code

```
javascript
const applyDiscount = (discountCode) => { ... }
...
```

- **Logic:** This function checks if a provided discount code is valid by looking it up in a predefined discounts object. It calculates the total cost and applies the discount percentage to return the final cost after discount.

Screenshot:



```
[Running] node "d:\mad\script.js"
Cart Summary: [
  { productName: 'Laptop', quantity: 1, totalPrice: 1200 },
  { productName: 'Smartphone', quantity: 2, totalPrice: 1600 },
  { productName: 'Headphones', quantity: 1, totalPrice: 150 }
]
Total Cost: 2950
Total Cost after DISCOUNT10: 2655
Cart after updating quantity:
Cart Summary: [
  { productName: 'Laptop', quantity: 1, totalPrice: 1200 },
  { productName: 'Smartphone', quantity: 3, totalPrice: 2400 },
  { productName: 'Headphones', quantity: 1, totalPrice: 150 }
]
Cart after removing Headphones:
Cart Summary: [
  { productName: 'Laptop', quantity: 1, totalPrice: 1200 },
  { productName: 'Smartphone', quantity: 3, totalPrice: 2400 }
]
Cart after filtering zero-quantity items:
Cart Summary: [
  { productName: 'Laptop', quantity: 1, totalPrice: 1200 },
  { productName: 'Smartphone', quantity: 3, totalPrice: 2400 }
]
[Done] exited with code=0 in 0.509 seconds
```

Code:

```
// Shopping cart array to hold product items
let cart = [];
```

```
// 1. Add Items to the Cart
```

```
const addItemToCart = (productId, productName, quantity, price) => {  
  // Check if item already exists in the cart  
  const existingProduct = cart.find(item => item.productId === productId);  
  if (existingProduct) {  
    // Update the quantity if product already exists  
    existingProduct.quantity += quantity;  
  } else {  
    // Push a new product object into the cart  
    cart.push({ productId, productName, quantity, price });  
  }  
};
```

// 2. Remove Items from the Cart

```
const removeItemFromCart = (productId) => {  
  // Find index of the product to remove by productId  
  const productIndex = cart.findIndex(item => item.productId === productId);  
  if (productIndex !== -1) {  
    // Remove product using splice  
    cart.splice(productIndex, 1);  
  }  
};
```

// 2. Update Item Quantity

```
const updateItemQuantity = (productId, newQuantity) => {  
  cart = cart.map(item =>  
    item.productId === productId ? { ...item, quantity: newQuantity } : item  
  );  
};
```

// 3. Calculate Total Cost

```
const calculateTotalCost = () => {  
  // Use reduce to calculate total price based on quantity and price  
  return cart.reduce((total, item) => total + (item.price * item.quantity), 0);  
};
```

// 4. Display Cart Summary

```
const displayCartSummary = () => {  
  // Use map to generate summary for each product  
  const summary = cart.map(item => ({  
    productName: item.productName,
```

```

        quantity: item.quantity,
        totalPrice: item.price * item.quantity
    ));

    console.log('Cart Summary:', summary);
    return summary;
};

// 4. Filter out Items with Zero Quantity
const filterZeroQuantityItems = () => {
    // Use filter to remove items with zero quantity
    cart = cart.filter(item => item.quantity > 0);
};

// 5. Apply Discount Code (Optional)
const applyDiscount = (discountCode) => {
    const discounts = {
        'DISCOUNT10': 0.10, // 10% discount
        'DISCOUNT20': 0.20, // 20% discount
    };

    // Check if discount code is valid
    const discount = discounts[discountCode] || 0;
    const totalCost = calculateTotalCost();

    // Apply discount to total cost
    return totalCost - (totalCost * discount);
};

// Test the shopping cart functionality

// Adding items to the cart
addItemToCart(1, 'Laptop', 1, 1200);
addItemToCart(2, 'Smartphone', 2, 800);
addItemToCart(3, 'Headphones', 1, 150);

// Display cart summary
displayCartSummary();

// Calculate total cost

```

```
console.log('Total Cost:', calculateTotalCost());

// Apply a discount
console.log('Total Cost after DISCOUNT10:', applyDiscount('DISCOUNT10'));

// Update quantity of an item
updateItemQuantity(2, 3);
console.log('Cart after updating quantity:');
displayCartSummary();

// Remove an item from the cart
removeItemFromCart(3);
console.log('Cart after removing Headphones:');
displayCartSummary();

// Filter out items with zero quantity (if any)
filterZeroQuantityItems();
console.log('Cart after filtering zero-quantity items:');
displayCartSummary();
```

Conclusion:

From this code, i learned how to implement basic CRUD operations for managing a shopping cart, including adding, removing, and updating items, as well as calculating totals and applying discounts. It also highlights the use of JavaScript array methods for data manipulation.

Challenges Faced:

In this code, i faced a few challenges. First, keeping track of the shopping cart's items and their quantities can get complicated, especially if we add more features later. Second, the code doesn't handle errors well, so if someone tries to remove or update an item that doesn't exist, it might not work as expected. Third, as the number of items in the cart grows, some functions might slow down because they search through the entire list each time. Lastly, since the output is shown in the console, it might not be very user-friendly, which means we would need to create a better interface for users in future updates.