COMSATS UNIVERSITY ISLAMABAD

ATTOCK CAMPUS

SUBMITTED BY:

Syed Usman Ali

Registration Number:

(SP22-BSE-017)

Course:

Mobile Application Development

SUBMITTED TO:

Mr. Muhammad Kamran

DATED:

September 26,2024

Assignment No#01

Introduction:

The ShoppingCart class is a simple JavaScript implementation of a shopping cart functionality. It allows users to manage a collection of products, providing features to add, remove, update quantities, calculate totals, and even apply discounts.

Explanation of the Code

The ShoppingCart class consists of several methods, each designed to perform specific actions related to cart management. Here's a breakdown of the key components:

1. **Constructor**:
    - Initializes an empty array cart to store the items.
2. **Add Items** (addItem):
    - Takes parameters like productId, productName, quantity, and price.
    - Creates a product object and adds it to the cart array.
3. **Remove Items** (removeItem):
    - Accepts a productId and filters out the corresponding item from the cart.
4. **Update Quantity** (updateItemQuantity):
    - Finds an item by its productId and updates its quantity.
5. **Calculate Total Cost** (calculateTotal):
    - Uses the reduce method to calculate the total price of all items in the cart based on their quantities and prices.
6. **Display Cart Summary** (displayCartSummary):
    - Returns an array of objects representing each item with its productName, quantity, and the total price for that item.
7. **Filter Out Zero Quantity** (filterOutZeroQuantity):
    - Removes items from the cart that have a quantity of zero, ensuring the cart only contains valid items.
8. **Apply Discount** (applyDiscount):

- Checks for valid discount codes and applies the corresponding discount to the total cost.

How It Works:

To use the ShoppingCart class, you can create an instance of it and utilize its methods to manage products. Here's a quick example flow:

1. **Adding Items**: Items can be added to the cart by calling the addItem method with the required details.
2. **Viewing the Cart**: The displayCartSummary method can be called to see a summary of the items in the cart.
3. **Calculating Total**: The total cost can be computed with calculateTotal.
4. **Updating and Removing Items**: You can change quantities or remove items as needed.
5. **Applying Discounts**: Finally, you can apply a discount code to see the adjusted total.

Add or remove Items:

```
// 1. Add Items to the Cart
addItem = (productId, productName, quantity, price) => {
    const product = { productId, productName, quantity, price };
    this.cart.push(product); // Add product to the cart
};

// 2. Remove Items from the Cart
removeItem = (productId) => {
    this.cart = this.cart.filter(item => item.productId !== productId); // Filter out the item
};
```

Update or Calculate Total Cost:

```
// Update quantity of an item
updateItemQuantity = (productId, newQuantity) => {
    const item = this.cart.find(item => item.productId === productId);
    if (item) {
        item.quantity = newQuantity; // Update the quantity
    }
};

// 3. Calculate Total Cost
calculateTotal = () => {
    return this.cart.reduce((total, item) => total + (item.price * item.quantity), 0); // Calculate total
};
```

Display Cart Summary:

```
// 4. Display Cart Summary
displayCartSummary = () => {
    return this.cart.map(item => ({
        productName: item.productName,
        quantity: item.quantity,
        totalPrice: item.price * item.quantity // Calculate total price for each product
    }));
};
```

Filter out Items:

```
// Filter out items with zero quantity
filterOutZeroQuantity = () => {
    this.cart = this.cart.filter(item => item.quantity > 0); // Keep only items with quantity > 0
};
```

Discount:

```
// Bonus: Apply Discount Code
applyDiscount = (code) => {
    const discountCodes = { 'SAVE10': 0.10, 'SAVE20': 0.20 }; // Example discount codes
    const discount = discountCodes[code] || 0; // Get discount if valid
    const total = this.calculateTotal();
    return total - (total * discount); // Apply discount
};
```

Output of the Code:

What I Learned from This Code:

- **Encapsulation**: The ShoppingCart class encapsulates related data and behaviors, demonstrating how to organize code effectively.
- **Array Methods**: I learned to use array methods like push, filter, and reduce to manipulate collections of data.
- **Real-world Application**: This code serves as a practical example of how shopping carts work in e-commerce, laying a foundation for further development and more complex features.