SHOPPING CART SYSTEM

VISUAL PROGRAMMING

By: **Musa bin Abdullah**

Mohammed Danial

Abdullah Altaf

Contents

Title

Shopping Cart System

Abstract

The online shopping cart system is a comprehensive solution designed to facilitate the e-commerce experience for users and retailers alike. This system enables customers to browse products, manage their selections, and complete purchases seamlessly through a user-friendly interface. Key features include product categorization, real-time inventory management, and the ability to add or remove items from the cart. The system supports various payment gateways, ensuring secure transactions while also incorporating a discount mechanism based on purchase thresholds to enhance customer satisfaction.

Additionally, it provides a recommendation engine that suggests products based on previous purchases and user preferences, thereby improving sales opportunities. The system is built with scalability in mind, allowing for easy integration of new features and third-party services. By leveraging the power of modern web technologies.

INTRODUCTION

The Shopping Cart System is a console-based application developed in C#. It allows users to add, remove, view items in the cart, and proceed to checkout. The system also handles discounts, taxes, and product recommendations, offering a complete e-commerce experience.

2 Project features

- **Add and Remove Products**: Users can add products to the cart by specifying the product ID and quantity. They can also remove items from the cart.
- **View Cart**: Users can view the items added to the cart, along with subtotal, discount, sales tax, and total cost.
- **Item Quantity Management**: Users can add multiple quantities of a product, and the system will update the total accordingly.
- **Apply Discounts and Sales Tax**: The system applies a various discount over the customers shopping bill and 8% sales tax on the cart.
- **Product Recommendations**: Based on the categories of the items in the cart, the system recommends other similar products.
- **Cart Expiration**: The cart expires after 30 minutes, after which users will need to restart the session.
- **Checkout Process**: Users can finalize their purchases and proceed to checkout where they will see a summary of their order.

3 Code Explanation

Classes

1. Product Class:

Represents a product in the shop.

Properties:

- Id: A unique identifier for the product.
- Name: The name of the product.
- Price: The price of the product.
- Category: The category to which the product belongs.
- Constructor initializes the properties.
- ToString() method returns a string representation of the product.

2. **CartItem Class:**

Represents an item in the shopping cart.

Properties:

- Product: The product associated with the cart item.
- Quantity: The quantity of the product in the cart.
- Constructor initializes the properties.
- ToString() method returns a string representation of the cart item.
- GetTotalPrice() method calculates the total price for that cart item based on its quantity.

3. **ShoppingCart Class:**

Manages the shopping cart's items and operations.

Properties:

- items: A list of CartItem objects representing items in the cart.
- salesTax: A decimal representing the sales tax rate (8%).
- cartExpiration: A DateTime indicating when the cart will expire.
- Methods:
- AddProduct(Product product, int quantity): Adds a product to the cart, updating quantity if it already exists.
- RemoveProduct(int productId, int quantity): Removes a specified quantity of a product from the cart.
- ViewCart(): Displays the contents of the cart.
- GetSubtotal(): Calculates the subtotal of all items in the cart.
- GetDiscountAmount(): Calculates any applicable discounts based on the subtotal.

- GetTaxAmount(): Calculates the sales tax based on the subtotal after discounts.
- GetTotal(): Calculates the total amount due, including subtotal, discounts, and tax.
- IsCartExpired(): Checks if the cart has expired.
- RecommendProducts(List<Product> allProducts): Suggests products based on categories of items already in the cart.
- Checkout(List<Product> allProducts): Processes the checkout, displaying the cart details and clearing the cart afterward.

4. Program Class

- Contains the Main method, which serves as the entry point of the application.
- Initializes a ShoppingCart instance and a list of Product instances.
- Displays a welcome message and presents a menu to the user with options to view products, add/remove items, view the cart, checkout, or exit.
- The menu runs in a loop until the user chooses to exit.

5. Helper Methods

- ViewProducts(List<Product> products): Displays all available products.
- AddToCart(ShoppingCart cart, List<Product> products): Prompts the user to enter a product ID and quantity to add to the cart, with an option to add more products.
- RemoveFromCart(ShoppingCart cart): Prompts the user to enter a product ID and quantity to remove from the cart, with an option to remove more products.
- User Interaction
- The user interacts with the application through the console, entering choices and product details as prompted.

- The application provides feedback after each action (e.g., adding or removing products, viewing the cart).

4 Work Flow

- **Step 1**: The user views the available products.
- **Step 2**: The user adds products to the cart.
- **Step 3**: The user can view the cart, which shows all items along with subtotal, discount, tax, and total.
- **Step 4**: The user proceeds to checkout or removes items from the cart.
- **Step 5**: The system recommends additional products based on items already in the cart.

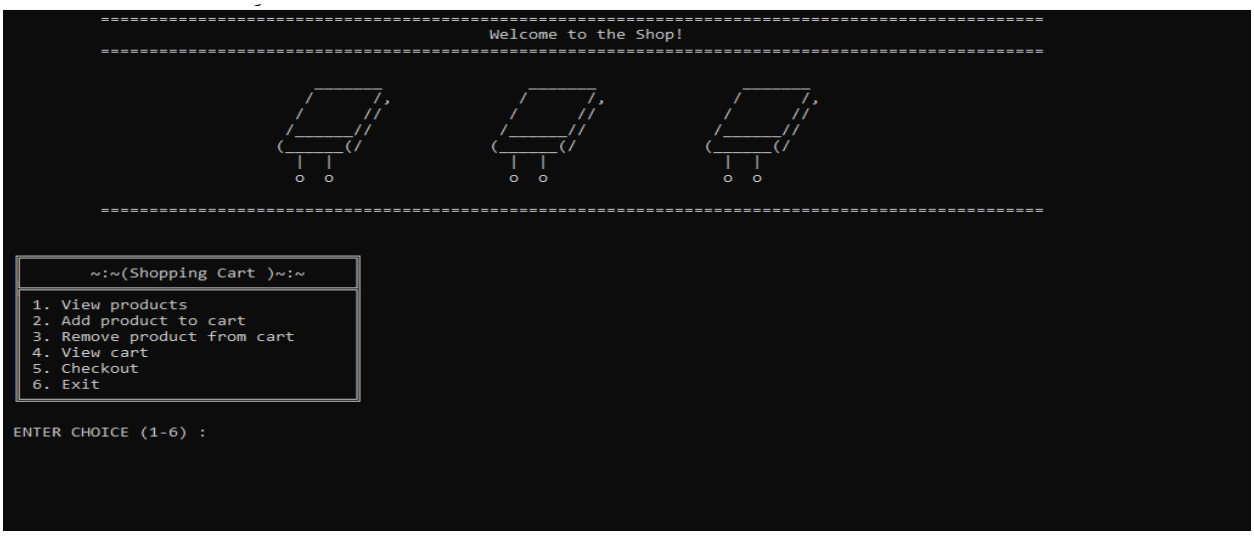
5 Collection Used

The following collections are utilized in the project:

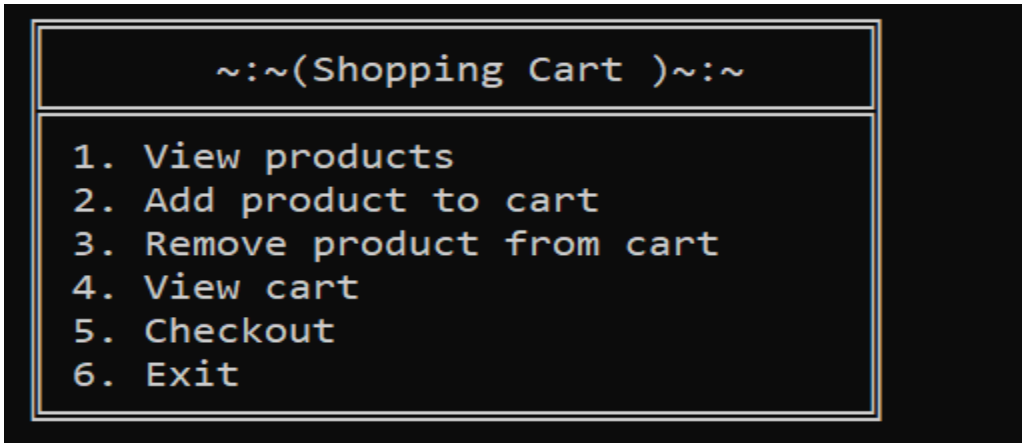
- **List<Product>**: Stores the available products.
- **List<CartItem>**: Stores the items added to the shopping cart.

6 Code Output

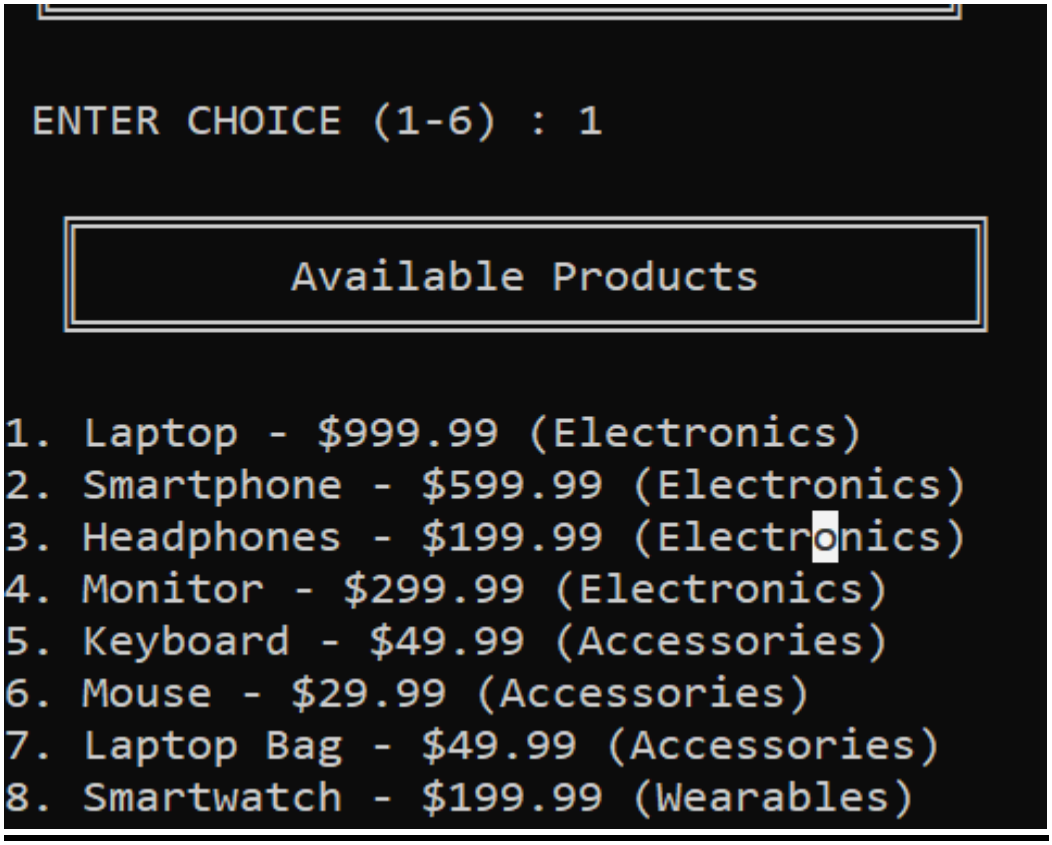
starting



Menu



View Product



Add To Cart

```
 ENTER CHOICE (1-6) : 2
Enter the product ID to add to the cart:2
Enter quantity:2
2x Smartphone added to the cart.
Do you want to add another product? (yes/no)  : yes
Enter the product ID to add to the cart:2
Enter quantity:1
1x Smartphone added to the cart.
Do you want to add another product? (yes/no)  : no
```

Remove From Cart

```
 ENTER CHOICE (1-6) : 3
Enter the product ID to remove from the cart:2
Enter the quantity to remove:1
Removed 1x Smartphone from the cart.

Do you want to remove another product? (yes/no) : no
```

View Cart

```
 ENTER CHOICE (1-6) : 4

        Items in your cart

2x Smartphone - $599.99
Smartphone x 2 - $1199.98
```

Checkout

```
ENTER CHOICE (1-6) : 5

        Items in your cart

2x Smartphone - $599.99
Smartphone x 2 - $1199.98

        Proceeding to checkout


---------------------------------
Sub_total: $1199.98
After_Discount: $0.00
Tax: $95.9984
Total: $1295.9784
---------------------------------
Checkout Date and Time: 10/25/2024 6:29:42 PM
Thank you for your purchase!.........
```

Recommendation

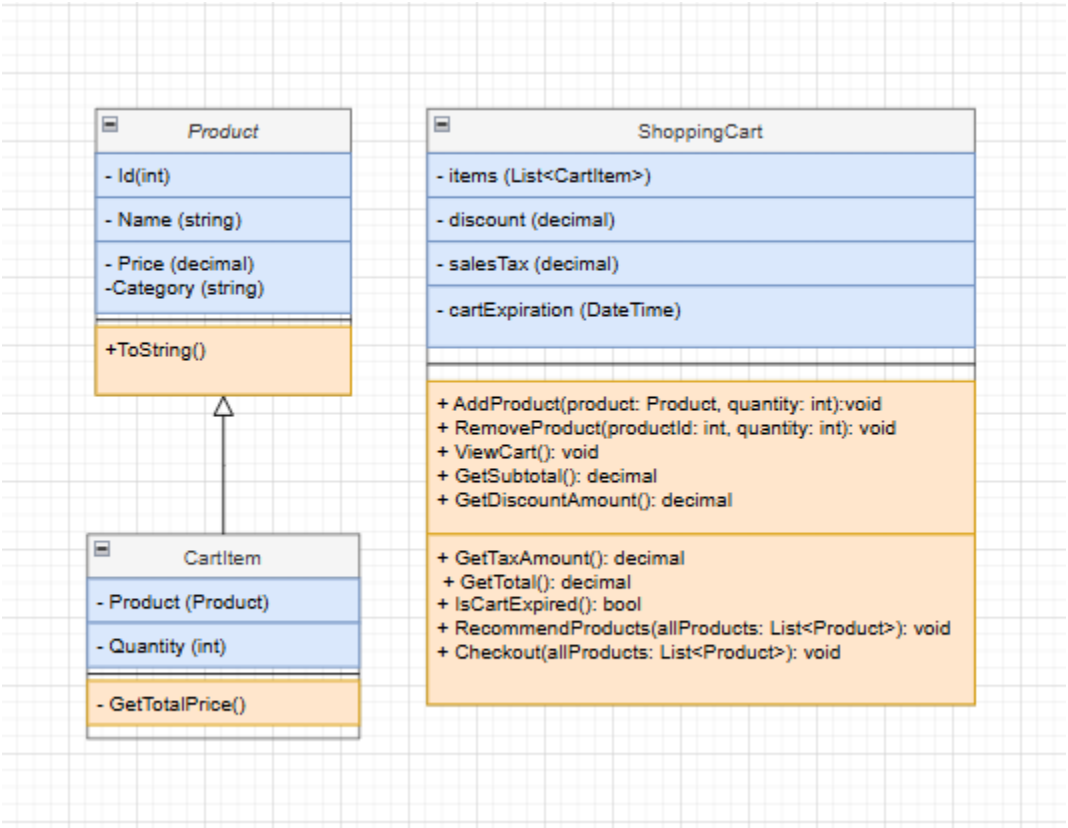
```
Based on your cart, we recommend:
Laptop - $999.99 (Electronics)
Smartphone - $599.99 (Electronics)
Headphones - $199.99 (Electronics)
Monitor - $299.99 (Electronics)
```

Exit

```
 ENTER CHOICE (1-6) : 6

F:\visual studio\test\test\bin\Debug\net8.0\test.exe (process 228) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automaticall
he console when debugging stops.
Press any key to close this window . . .
```

6. UML DIAGHRAME :



7. Conclusion

This project demonstrates the effective use of object-oriented programming principles in C#. The Shopping Cart system covers essential e-commerce operations, including adding/removing items, managing cart expiration, calculating totals with discounts and taxes, and offering product recommendations.

This code provides a straightforward implementation of a shopping cart system in a console application. It includes basic functionalities like adding/removing products, calculating totals with discounts and taxes, and providing product recommendations based on the user's cart. The structure of the code is modular, with separate classes for products, cart items, and the shopping cart itself, making it easy to extend and maintain.