|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | 25/10/2024  Visual Programming  BSCS(B) | |
| **Shopping Cart Application** | | | | |
| C:\Users\Air Univeristy\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\838FF43E.tmp | | | | |
| **Submitted By:**  Amna Sajjad (233582)  Komal Naz (233520)  Tooba Ghaffar (233518)  **Submitted to:**  Mam Aatka | |  |  |  |

|  |
| --- |
| **Table of Content:**  **1.Introduction.**  **2.Classes and Key Components:**  2.1 Category Class.  2.2 Product Class.  2.3 CartItem Class.  2.4 Cart Class.  2.5 Checkout Class,  2.6 Program Class.  **3. Key Features:**  3.1 Cart Expiration.  3.2 Product Recommendations.  3.3 Discount and Tax Calculation.  **4. File Handling.**  **5. Improvements and Considerations.**  **6. Conclusion.**  **1.Introduction:**  The code represents a simple console-based shopping cart system written in C#. It allows users to view products by category, add or remove products from their cart, view the cart, and proceed to checkout. The program supports product recommendations and cart expiration functionality. The main components include `Category`, `Product`, `CartItem`, `Cart`, `Checkout`, and a driver class `Program`. |
|  |

|  |  |
| --- | --- |
|  |  |
| **2.**  **Classes and Key Components:** |  |
|  |  |
| **2.1 Category Class:**  The Category class represents product categories:  **Properties-**   * **Id** : Category ID * **Name** : Category name * **-Constructor** : Initializes a category with `id` and `name`.   **The snippet is:**    **2.2 Product Class:**  The `Product` class represents a product:  **Properties:**   * **Id :** Product ID * **Name :** Product name * **Price :** Product price * **Description :** Brief description of the product * **Category :** Reference to the `Category` the product belongs to * **Constructor :** Initializes a product with an ID, name, price, description, and reference.   **The snippet is:**    **2.3 CartItem Class:**  The `CartItem` class represents an item in the shopping cart:  **Properties:**   * **Product** : Reference to a `Product` * **Quantity** : Quantity of the product.   **Methods** :   * **TotalPrice()** : Calculates the total price by multiplying the product price with the quantity.   **The snippet is:**    **2.4 Cart Class:**  The `Cart` class manages the shopping cart, holding a list of `CartItem` objects:  **Fields :**   * **Items :** A list of `CartItem` representing the products in the cart. * **CartExpirationInMinutes** : Constant representing the cart expiration time (30 minutes). * **ExpirationTime :** Stores the expiration time of the cart.   **Methods :**   * **AddItem(Product product, int quantity) :** Adds a product to the cart or increases its quantity if it already exists. * **RemoveItem(int productId) :** Removes a product from the cart based on its ID. * **ViewCart() :** Displays the contents of the cart. * **CalculateTotal() :** Returns the total price of the items in the cart. * **IsCartExpired() :** Checks if the cart has expired. * **RecommendProducts(List<Product> allProducts) :** Returns a list of up to 3 product recommendations that are not already in the cart.   **The snippet is:**        **2.5 Checkout Class:**  The `Checkout` class handles discount and tax calculations during checkout:  **Properties :**   * **Discount :** Percentage discount applied to the total amount. * **TaxRate :** Tax percentage to be added to the final total.   **Methods:**   * **ApplyDiscount(decimal total) :** Applies a discount to the subtotal. * **ApplyTax(decimal total) :** Adds tax to the discounted subtotal. * **FinalizeTotal(decimal total) :** Combines the discount and tax operations to finalize the total price.   **The snippet is:**    **2.6 Program Class:**  The `Program` class is the main entry point for the application and handles user interaction:  **Fields :**   * Categories : List of all product categories. * products : List of all available products.   **Methods :**   * **Main(string[] args) :** Main execution loop that allows users to interact with the shopping cart system through a menu. * **ViewProductsByCategory() :** Displays products by their categories. * **AddProductToCart(Cart cart) :** Adds a product to the cart after taking the product ID and quantity from the user. * **RemoveProductFromCart(Cart cart) :** Removes a product from the cart based on the product ID. * **CheckoutCart(Cart cart, Checkout checkout) :** Handles the checkout process, applying discounts and tax. * **ViewRecommendations(Cart cart) :** Shows product recommendations. * **LoadCategoriesFromFile(string fileName) :** Loads categories from a file. * **LoadProductsFromFile(string fileName) :** Loads products from a file.   **The snippet is:**    **Main Function :**        **3. Key Features:**  **3.1 Cart Expiration:**  The cart expires 30 minutes after it is created. The method `IsCartExpired()` checks if the cart is still valid during the checkout process.  **3.2 Product Recommendations:**  The `RecommendProducts()` method suggests up to 3 products that are not currently in the cart. This feature encourages further purchases based on available products.    **3.3 Discount and Tax Calculation:**  The `Checkout` class applies a discount and tax to the total cart value. The discount is applied first, followed by tax. The user can customize the discount and tax rates.  **4. File Handling:**  Categories and products are loaded from text files (`category.txt` and `inventory.txt`). Each file is expected to follow a specific format:   * **Categories:** `CategoryId,CategoryName`. * **Products:** `ProductId,ProductName,ProductPrice,ProductDescription,CategoryId`.   This design allows for easy updates to the product inventory by modifying the text files.    **5. Improvements and Considerations :**   * **Error Handling :** The program has basic error handling for file reading but could benefit from more robust validation in user input, such as invalid product IDs or quantities. * **Persistence :** Currently, the cart data is not saved. Adding functionality to persist cart items across sessions (e.g., using a database or local storage) could enhance user experience. * **Scalability:** The program is designed for small product lists. For larger inventories, optimizing file loading (e.g., using a database) and implementing pagination for product listings would improve performance. * **Testing :** Integration of unit tests would improve the reliability of key functionalities, such as adding/removing items, calculating totals, and checking expiration.   **6. Conclusion :**  The `ShoppingCartApp` is a well-structured console application that implements basic e-commerce functionalities. It provides a simple and user-friendly way for customers to manage a shopping cart, calculate totals with discounts and taxes, and receive product recommendations. Expanding the app to support more robust input validation, persistence, and scalability would make it more suitable for larger use cases.  **THANK YOU…** | | |
|  | | |
|  | | |
|  | | |
|  | | |
|  | | |
|  | |

|  |
| --- |
|  |