**Documentation on a Supermarket Billing Management System**

**An Overview**

This C++ program is about the implementation of Supermarket Management System which includes inventory management, shopping cart functionality, billing, discount application and product recommendation system. The system is designed using object-oriented programming principles and has some simple unit tests.

**List of Classes**

**1. Product**

This is a base class that represents an item in stock generically.

**Properties:**

* name: string
* price: double
* stock: int
* description: string

**Behaviors:**

* getPrice(): This method returns the price of a product.
* getName(): This function gives out the name of the product.
* getStock(): It returns the current stock level of this item in store.
* updateStock(int quantity): Reduce the amount available when a customer makes purchases.
* display(): Shows just basic details about this product to screen.
* displayDetails(): Presents all information related to this particular item in question like its price, description et cetera.

**2. FreshProduct**

A derived class from Product showing products with expiry dates only called shelf life hereafter if you please.

**Additional Attributes:**

shelfLife : int (in days)

**Additional Methods:**

-overriden methods are present for display and displaydetails so as to include shelf life information

**3. ShoppingCart**

Manages the items added to the cart by the user.

**Attributes:**

* items: vector of CartItem structs

**Methods:**

* addToCart(shared\_ptr<Product> product, int quantity): Adds a product to the cart
* getTotalCost(): Calculates the total cost of items in the cart
* display(): Displays the contents of the cart

**4. DiscountStrategy**

Abstract base class for implementing different discount strategies.

**Methods:**

* applyDiscount(double amount): Pure virtual method to apply a discount

**5. PercentageDiscount**

Derived class from DiscountStrategy, applies a percentage-based discount.

**Methods:**

* applyDiscount(double amount): Applies a percentage discount

**6. FixedAmountDiscount**

Derived class from DiscountStrategy, applies a fixed amount discount.

**Methods:**

* applyDiscount(double amount): Applies a fixed amount discount

**7. Billing**

Handles billing operations including discount application.

**Static Methods:**

* generateBill(const ShoppingCart& cart): Generates a bill for the given cart
* applyDiscount(double amount, DiscountStrategy\* strategy): Applies a discount using the given strategy

**8. RecommendationSystem**

Provides product recommendations based on purchase history.

**Attributes:**

* purchaseHistory: unordered\_map storing user purchase
* Methods:
* -recordPurchase(const string& user, const string& product)`: It saves a user's purchase.
* - getRecommendations(const string& user)`: It gives the user product recommendations.

Main Functionality

* 1. Inventory Management: The system entails product information for each product in the inventory.
* 2. Shopping Cart: The users are provided with a shopping cart to add a jn to buy the product.
* 3. Billing: Generates bills for the items in the cart.
* 4. Discount Application:Discounts are applied by various strategies.
* 5. Payment Processing: The payment processing of the bank and M-PESA transactions is imitated.
* 6.Recommendation System: It gives product recommendations based on the purchase history.
* User Interface
* A command-line interface is the main function of the program and there are options you can use to execute the statement you want by typing in the command-line.
* 1. View the Product List
* 2. View the Product Details
* 3. Go to Cart
* 4. View the Cart
* 5. Go to Checkout
* 6. Get the Results
* 7. Exit
* Testing
* `runTests()` test covers only the essential basic unit tests for the different system elements
* Error Handling
* Furthermore, this process, using exceptions, handles errors by either notifying the user if the stock is less than required or the user has entered an invalid input. Discount Application:Discounts are applied by various strategies.