Flipkart Daily

Problem

Description: Flipkart wants to build a product to deliver groceries and daily essentials by next morning. Users can search for items from inventory with some filters and will be allowed to place an order only if (S)he has enough money in the wallet. To keep it simple, the use case of adding Wallet money later is out of scope.

Features

1. Add user -> AddUser(name, email, wallet_amount)
2. Define price for the items -> AddItem(category, brand, mrp, price)
3. Add items to inventory -> AddInventory(category, brand, quantity)
4. Add items to the user's cart (this places the order directly using the wallet amount)
5. Update price of a particular item for a {category, brand}
6. Update price of all items in category and brand basis below 2 options:
    a. Update price by applying flat x% off on price
    b. Update price by applying x% increase on price
7. Get current user cart information
8. Search items in inventory by applying multiple filters as mentioned below and can perform search with zero or many of these filters
    a. brand
    b. category
    c. priceRange (priceFrom , priceTo)
    d. quantity (greaterThanEqualTo, lessThanEqualTo)

Bonus

1. A User should be able to choose the Order of Search results
    a. Items with lowest price first
    b. Items with highest price first
    c. Items with least no of quantity
    d. We might add more such options in the future
2. Admin should be able to add more pricing options in future to the existing ones seamlessly like below:
    a. Update price by applying flat x% off on mrp
    b. Update price by increasing x% on price for all brands within a category

Guidelines

1. Do not use any database (Store all interim/output data **in-memory)**
2. Do not create any UI for the application.
3. Write a driver class for demo purposes which will execute all the commands at one place in the code and test cases.
4. Start with min 5 items with more quantities varying from 0-30
5. Assume that only one active session will be there for one user and there can be many users online at the same time. Your code should be able to support multiple users using the system at the same time but this need not be demonstrated(using the driver).
6. Problem Constraints:
    a. Wallet money will be at max 10000
    b. No of items in the inventory < 1000
    c. Maximum inventory quantity < 1000

Expectations

- **The code should be demo-able (<u>very important</u>)**. The code should be functionally correct and complete.
    - At the end of this interview round, an interviewer will provide multiple inputs to your program for which it is expected to work
- The code should handle edge cases properly and fail gracefully. Add suitable exception handling, wherever applicable.
- The code should have a good object-oriented design.
- The code should be readable, modular, testable, and extensible. Use intuitive names for your variables, methods, and classes.
    - It should be easy to add/remove functionality without rewriting a lot of code.
    - Do not write a monolithic code

Test cases

(You need not follow the same method signatures and output)

1. Add Item (category, brand, mrp, price)
2. Update Item (category, brand, price)
3. Add inventory (category,brand, quantity)
4. Add user (“Name”, ”Email”,”Wallet Amount”)
5. Search (“brand”,”category”) , Search(“category”), Search(“category”,Order_By=Price), Search(“category”, Order_By=ItemQty)
6. Add to cart(user, category, brand, quantity)
7. Get cart info(user)
8. Update Items (category, brand, pricing_option)

Sample input/output

AddItem(Amul, Milk, 120, 100)
AddItem(Amul, Curd, 60, 50)
AddItem(Nestle, Milk, 70, 60)
AddItem(Nestle, Curd, 100, 90)

AddInventory(Amul, Milk, 10)
AddInventory(Nestle, Milk, 5)
AddInventory(Nestle,Curd, 10)
AddInventory(Nestle, Milk, 30)
AddInventory(Amul, Curd, 5)

Inventory :
Amul - > Milk -> 20
Amul -> Curd -> 5
Nestle -> Milk -> 35
Nestle -> Curd-> 10

AddUser(Jhonny, jhonny@gmail.com , 600)
AddUser(Naruto, naruto@gmail.com, Anime, 500)
AddUser(Goku, goku@gmail.com, 3000)

AddToCart(Jhonny, Milk, Nestle, 5)
AddToCart(Naruto, Milk, Nestle,12)
UpdateItem(Nestle, Milk, 5)
AddToCart(Goku, Milk, Nestle, 10)

GetCart(Goku)
    Nestle -> Milk - > 10 → Total Price : 50
GetCart(Jhonny)
    Nestle -> Milk -> 5 -> Total Price : 300
GetCart(Naruto)
    Empty Cart

UpdateItems(Nestle, Curd, “FLAT_10%_OFF”)

SearchItems([Nestle])

Nestle -> Curd -> 10 -> 81

SearchItems([Nestle, Amul], [Curd])

Nestle -> Curd -> 10 -> 81

Amul -> Curd -> 5 -> 50

SearchItems([Nestle, Amul], [Curd],OrderBy InventoryQty)

Amul -> Curd -> 5 -> 50

Nestle -> Curd -> 10 -> 81

SearchItems([Nestle], [Curd, Milk, Paneer])

Nestle -> Curd -> 10 -> 81