FoodKart

Total time: 150 mins

Description:

Implement an online food ordering system. Below are the expected features from the system.

Features:

1. This system has tie-ups with restaurants where each restaurant has a menu with all the items with its price.
2. Each restaurant has a maximum processing capacity of items at any given time. Beyond that it won't accept any further item requests until items which are in processing are completed. Every item preparation time is same for simplicity
3. Each restaurant takes some time to prepare and dispatch food. Once the item is fulfilled the system gets the notification of it which adds back the processing capacity of that restaurant.
4. Order is accepted from customers only if all the items can be fulfilled by one restaurants.
5. One out of multiple candidate restaurants can be selected based on restaurant selection strategy (eg: lowest order amount)

Requirements:

1. Onboard new restaurant with its menu and item processing capacity.
2. Restaurant should be able to change its menu.
3. Customers should be able to place an order by giving multiple items and quantity details.
4. Implement one restaurant selection strategy (lowest price offered by the restaurant for that order.) Have the option of selection using a different strategy.
5. System should be able to keep track of all items served by each restaurant.
6. Option to mark the order as delivered, so that restaurant can add the capacity back

Other Details:

1. Do not use any database or NoSQL store, use in-memory store for now.
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 the system with onboarding 5 restaurants, each restaurant serving 3 items and has processing power for a maximum of 4 items at a given time.

Expectations:

1. Make sure that you can execute your code and show that it is working.
2. Make sure that code is functionally correct.
3. Write the unit test-cases
4. Work on the expected output first and then add good-to-have features of your own.
5. Code should be modular and readable.

6. Separation of concern should be addressed.
7. Code should easily accommodate new requirements with minimal changes.
8. Code should be easily testable.
9. Concurrency must be handled wherever applicable with scalability in mind (*good to have*)

Extensions:

1. Estimated delivery time of the order on placing it
2. Delivery cost can also come
3. Custom options to menu like king_burger with cheese, w/o cheese etc
4. Cancellation of order

Test cases: (*You need not follow the same method signatures and output*)

- add_restaurant(“restaurant_1”, {“item1": <item1_price>, “item_2”: <item_2_price>, ….}, <simlultaneous_processing_capacity>)
- add_restaurant(resta1, {‘king_burger’: 10, ‘samosa_pizza’: 20, ‘alu_pasta’: 19}, 15)
- add_restaurant(resta2, {‘bendi_macaroni’: 12, ‘samosa_pizza’: 25, ‘alu_pasta’: 17}, 12)
- update_menu(resta1, {‘bendi_macaroni’: 8, ‘king_burger’: 15})
- Print all restaurant details
  [{‘name’: ‘resta1’, ‘menu’: {‘king_burger’: 15, ‘samosa_pizza’: 20, ‘alu_pasta’: 19, ‘bendi_macaroni’: 8}, ‘total_capacity’: 15, ‘capacity_in_use’: 0},{‘name’: ‘resta2’, ‘menu’: {‘bendi_macaroni’: 12, ‘samosa_pizza’: 25, ‘alu_pasta’: 17}, ‘total_capacity’: 12, ‘capacity_in_use’: 0}]
- book(‘cust1’, {‘bendi_macaroni’: 3, ‘samosa_pizza’: 2’}) -- should return resta1, order1
- Print all restaurant details
  [{‘name’: ‘resta1’, ‘menu’: {‘king_burger’: 15, ‘samosa_pizza’: 20, ‘alu_pasta’: 19, ‘bendi_macaroni’: 8}, ‘total_capacity’: 15, ‘capacity_in_use’: 5},{...}]
- Print all orders placed
  [{ ‘order_id’: ‘order1’, ‘user’: ‘cust1’, ‘order_details’: {‘restaurant’: ‘resta1’, ‘items’: {‘bendi_macaroni’: 3, ‘samosa_pizza’: 2’}, ‘cost’: 64} }]
- MarkAsDelivered
  [{‘order_id’: ‘order1’}
- Print all restaurant details
  [{‘name’: ‘resta1’, ‘menu’: {‘king_burger’: 15, ‘samosa_pizza’: 20, ‘alu_pasta’: 19, ‘bendi_macaroni’: 8}, ‘total_capacity’: 15, ‘capacity_in_use’: 0},{‘name’: ‘resta2’, ‘menu’: {‘bendi_macaroni’: 12, ‘samosa_pizza’: 25, ‘alu_pasta’: 17}, ‘total_capacity’: 12, ‘capacity_in_use’: 0}]