Engineering Case

For this exercise, we'll be designing a mini Stock Exchange system, similar to how B3 or NYSE would work to be able to receive and process orders. The idea is to allow brokers to submit orders on behalf of their customers and resolve the trades.

Feel free to implement this exercise using whatever language or tech stack you're most comfortable, there are no restrictions on how to implement this exercise. However, we expect you to provide an OCI-compliant container file with a reproducible environment to run the application.

Specifications

The way our mini stock exchange should work is by receiving orders from brokers, keeping track of them and executing as soon as matching orders can be found. To match orders we need to find a buyer and a seller for the same stock, with valid orders and with prices that match.

For price matching you should consider that every price defined by the buyer is a maximum price that they're willing to pay and the one defined by the seller is the minimum price that they're willing to sell for. Whenever they match with a price gap, you should consider the seller price for the execution price.

Examples:

1. Same price
    - A wants to sell 1000 AAPL stock at $10
    - B wants to buy 1000 AAPL stock at $10
    - A and B's orders both get executed at $10
2. No match
    - A wants to sell 1000 AAPL stock at $20
    - B wants to buy 1000 AAPL stock at $10
    - None of the orders are executed

3. Price gap
    - A wants to sell 1000 AAPL stock at $10
    - B wants to buy 1000 AAPL stock at $20
    - A and B's orders both get executed at $10

You should also consider that orders can be partially executed:

- If A wants to sell 1000 and B wants to buy 500 with matching prices, A will sell 500 to B
- If A wants to sell 500, B wants to sell 500 and C wants to buy 1500 with matching prices, A and B will sell 500 each and C will have 500 remaining to be bought

Whenever multiple orders match at the same time, we should resolve them in chronological order:

- If both A and B want to sell 1000 at $10 and C wants to buy 1000 at $10, whoever submitted their order first between A and B would get executed

API

We want you to define the API for the system however you feel is best given the following minimum requirements:

- Brokers need a API in which they can submit bid (buy) and ask (sell) orders for their users
- An order should at least contain:
    - A broker identifier that is submitting the order
    - Document number of the person who owns the order
    - The type of order (bid/ask)
    - Until when the order is valid
    - The symbol of the stock that they want to trade
    - The value of the submitted order - how much is the seller asking or the buyer bidding

    - The quantity of stocks they are bidding/asking for
- Submitting an order should give an identifier back to the broker so they can later get the status of an order by this identifier

Extensions

- Endpoint with current stock price (moving average of ask/bids)
- Endpoint to list the book (eg. first 10 asks/bids) for a given stock
- Endpoint to return the balance of a broker
- Webhook with the operation execution result
- Market price orders