Thndr Programming Assignment

Description

You are requested to implement a simple API that would simulate parts of the trading process we have at thndr. Given a stream representing stock data, you should implement an API that supports the following actions:

- For a given stock, what's the current price
- For a given stock, what's the lowest/highest price so far in the day/hour
- Add funds to a given user's account
- Withdraw funds from a given user's account
- Buy stock (if available and if the funds allow it)
- Sell stock

Your Task

- Attached, there's a docker-compose file that runs 2 containers
    - A streamer that pushes data to the queue
    - vernemq, the message queue used for this task
- You will need to consume data from the topic `thndr-trading`
- This will give you data for 3 given stocks, structure is explained below
- Feel free to edit the docker-compose file with more images of yours (if you want to)
- To run the docker-compose.yml file:

```
docker-compose up -d
docker-compose logs # to view the logs
```

Input

Example of the expected input fromn the queue:

```
{
    "stock_id": "6ffb8e62-92c1-40c7-9d38-5b976a346b62", // stock id
    "name": "CIB", // stock name
    "price": 28, // current stock price
    "availability": 46, // current stock availability
    "timestamp": "2019-12-15 14:36:33.462393" // current timestamp
}
```

Endpoints

The are the endpoints you need to support:

#	Endpoint	Content-Type	Json Structure
1	/deposit	application/json	`{ "user_id": <user_id>, "amount": <amount> }`
2	/withdraw	application/json	`{ "user_id": <user_id>, "amount": <amount> }`
3	/buy	application/json	`{ "user_id": <user_id>, stock_id: <stock_id>, "total": <total>, "upper_bound": <upper_bound>, "lower_bound": <lower_bound> }`
4	/sell	application/json	`{ "user_id": <user_id>, stock_id: <stock_id>, "total": <total>, "upper_bound": <upper_bound>, "lower_bound": <lower_bound> }`
5	/stock	application/json	`{ "stock_id": <stock_id> }`
6	/user	application/json	`{ "user_id": <user_id>}`

- For endpoints 1-4, it's sufficient just to return some ack/status code to tell whether the action was successful
- For endpoints 5&6, please come up with a json structure that you see fit to be returned. This structure should give an overview of the current stock/user state
- Since the price can change between the order time and the actual execution time, `upper_bound` and `lower_bound` set an acceptable window for executing the order, otherwise we will have to retry after a given point in time. Feel free to decide what will be the exact behaviour in that case and explain your assumptions in your solution.

Implementation Notes

- You don't have to create new users, you can come with a few user ids and assume they already exist
- An API with json input/output is fine, no need to implement any front-end
- Please use python for your solution (whether to use a framework or not, that's up to you)
- You can use any 3rd party tools/packages you prefer