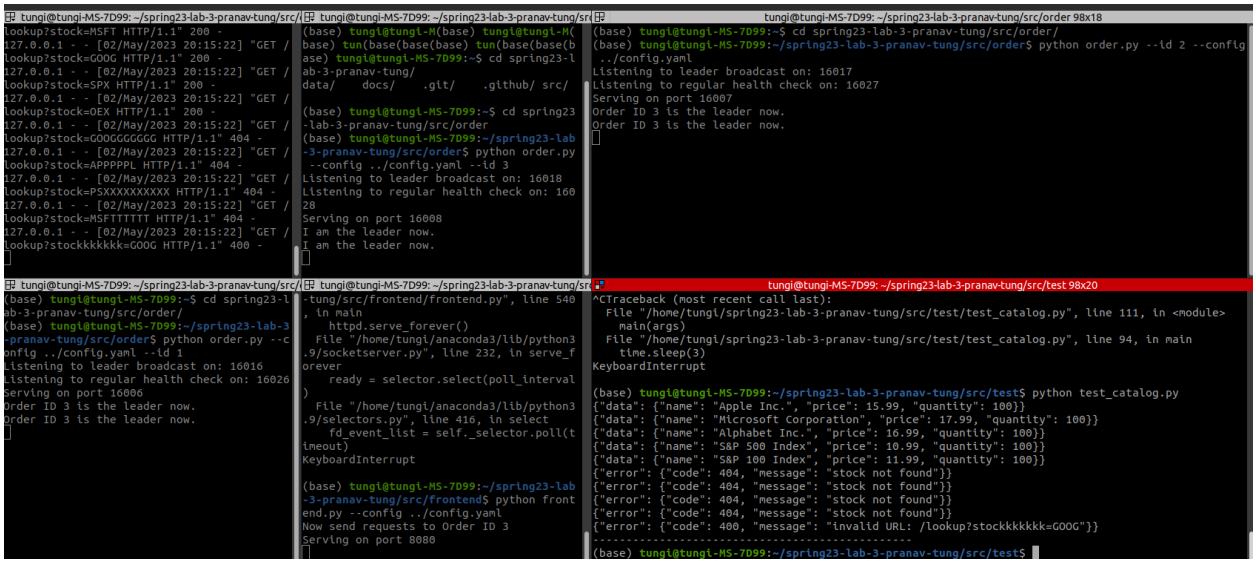
Tung-I Chen 33609886; Pranav Jain 33982032

Lab3
Outputs

Simple Test Cases

First, we tested if the **catalog service** works as expected by sending the following lookup() and order() requests directly to the catalog server. The test cases include the following three parts: fault tolerance, valid orders, and excessive trading.

- **Fault tolerance:** Send 5 lookup requests with valid stock names, and then send 4 lookup requests with invalid names, and then send 1 lookup request with invalid URL format.
  Expected result:
  {"data": {"name": "Apple Inc.", "price": 15.99, "quantity": 100}}
  {"data": {"name": "Microsoft Corporation", "price": 17.99, "quantity": 100}}
  {"data": {"name": "Alphabet Inc.", "price": 16.99, "quantity": 100}}
  {"data": {"name": "S&P 500 Index", "price": 10.99, "quantity": 100}}
  {"data": {"name": "S&P 100 Index", "price": 11.99, "quantity": 100}}
  {"error": {"code": 404, "message": "stock not found"}}
  {"error": {"code": 404, "message": "stock not found"}}
  {"error": {"code": 404, "message": "stock not found"}}
  {"error": {"code": 404, "message": "stock not found"}}
  {"error": {"code": 400, "message": "invalid URL: /lookup?stockkkkkkk=GOOG"}}



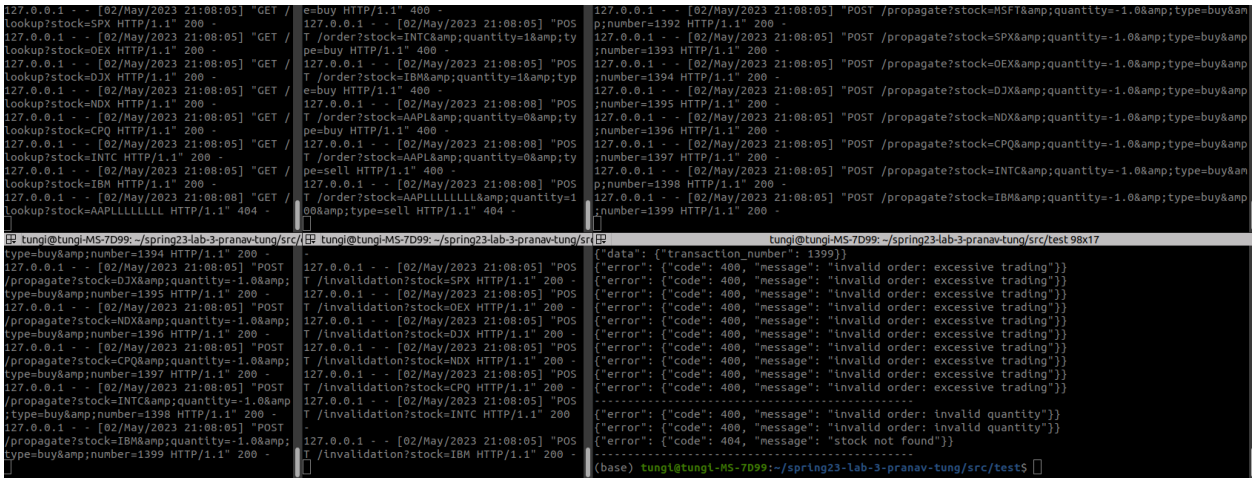
- **Valid orders**: Perform the following operation 100 times: Sell 1 stock of AAPL, and then buy 1 stock of AAPL, and then sell 1 stock of MSFT, and then buy 1 stock of MSFT.
  Expected result:

All the orders should be successful. Also check /src/catalog/data/catalog.json. The remaining quantity should remain the same yet the accumulated volume should have increased by 100.

- **Excessive trading:** Perform the following operation 101 times: Buy 1 quantity of each stock.
  Expected result:
  10 “Excessive trading” messages show up on the terminal. Also, the remaining_quantity of each stock in /src/catalog/data/catalog.json should be 0.

Secondly, we tested **both the catalog and order services**. The test cases in test_order.py include:

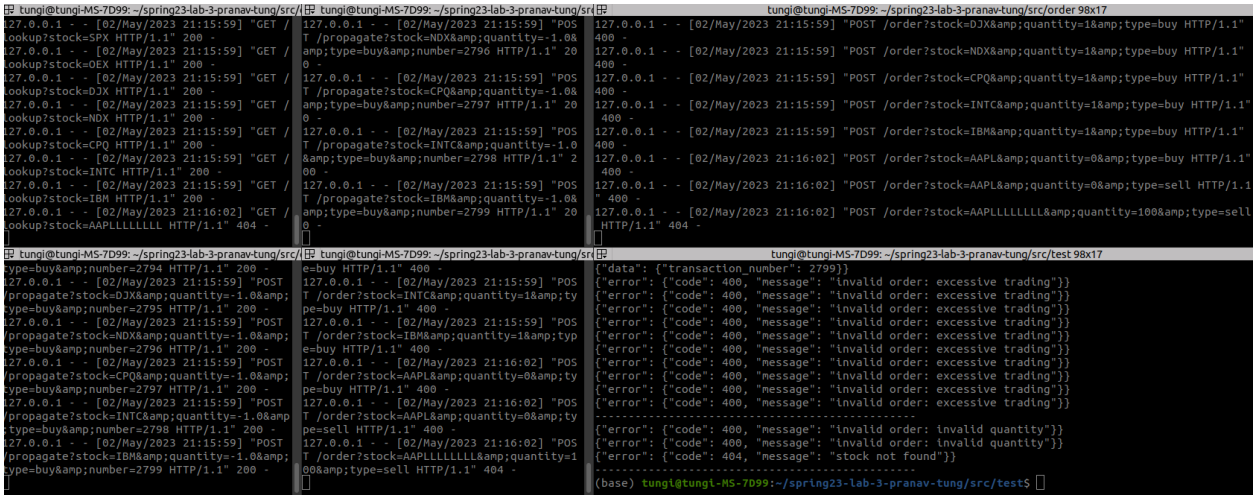
- 4 legal order requests
- 4 legal order lookup requests and 2 illegal lookup requests (order number not found)
- Buy 1 quantity of each stock. Do this 101 times:
- 3 invalid URL request
- Thus, 10 excessive trading messages and 3 invalid request messages are expected.



Finally, we tested the whole application. The test cases include:

- 5 legal lookup requests
- 4 lookup requests with non-existent stock names
- 1 invalid URL lookup requests
- 100 sell-and-then-buy operations, the same as that in the catalog testing
- buy 1 quantity of each stock 101 times, the same as that in the catalog testing
- 3 invalid URL order requests

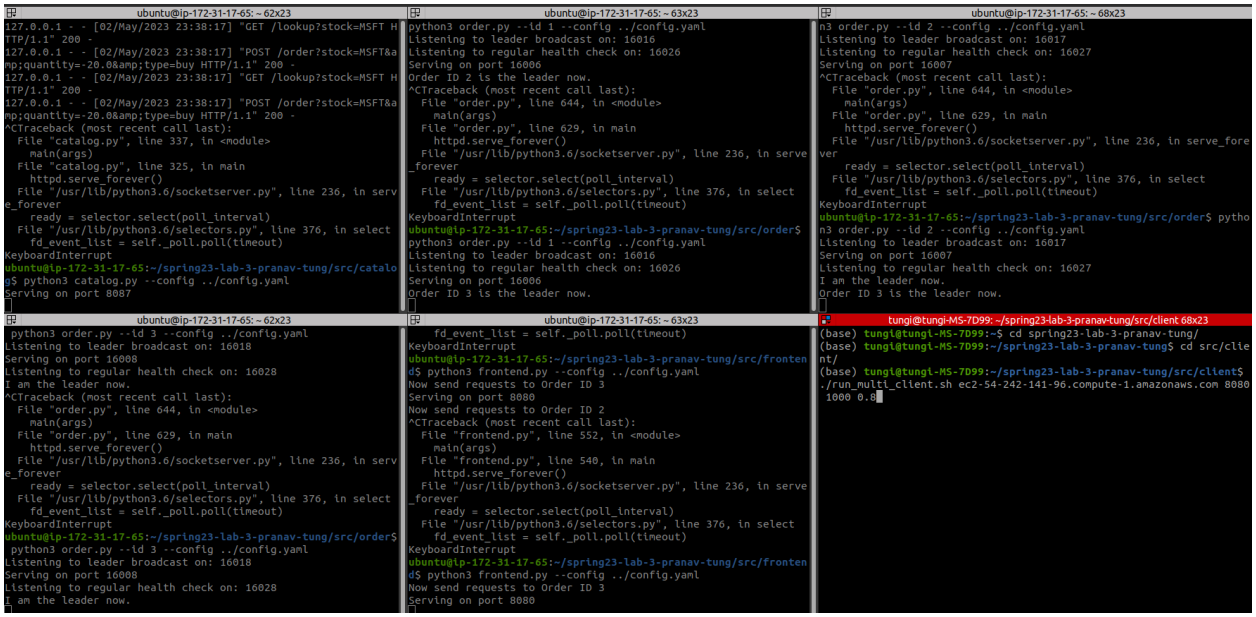
- The expected output:



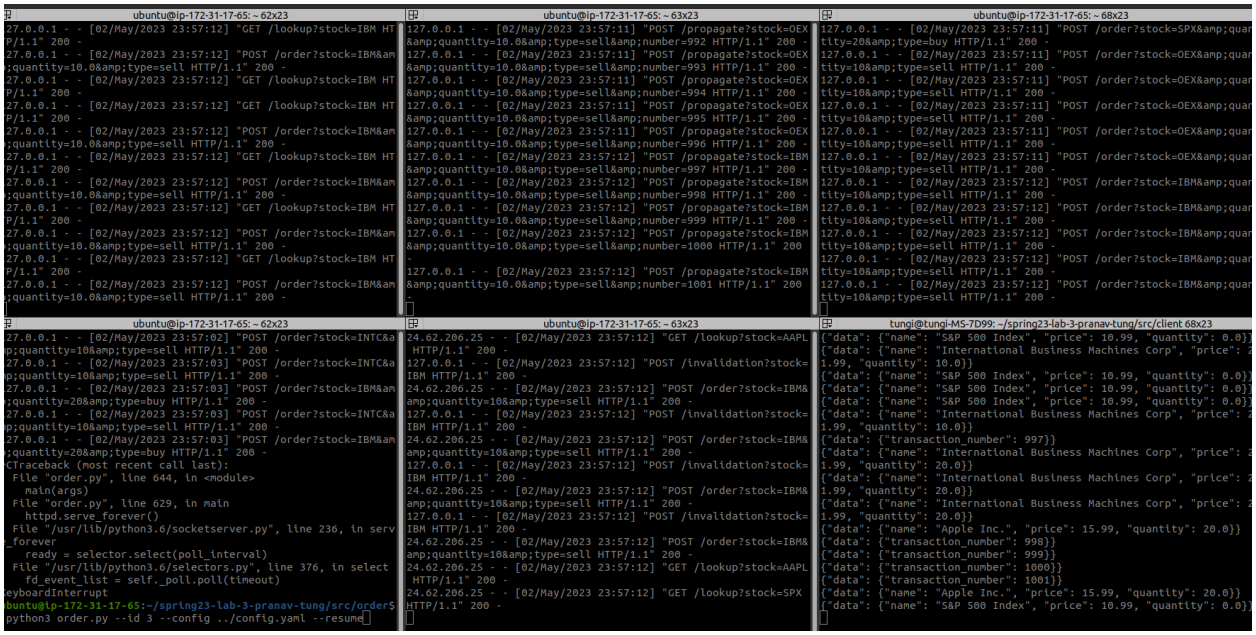
Testing by deploying the application on AWS

The following screenshots demonstrate the process of testing our application on the AWS m5a.large instance.

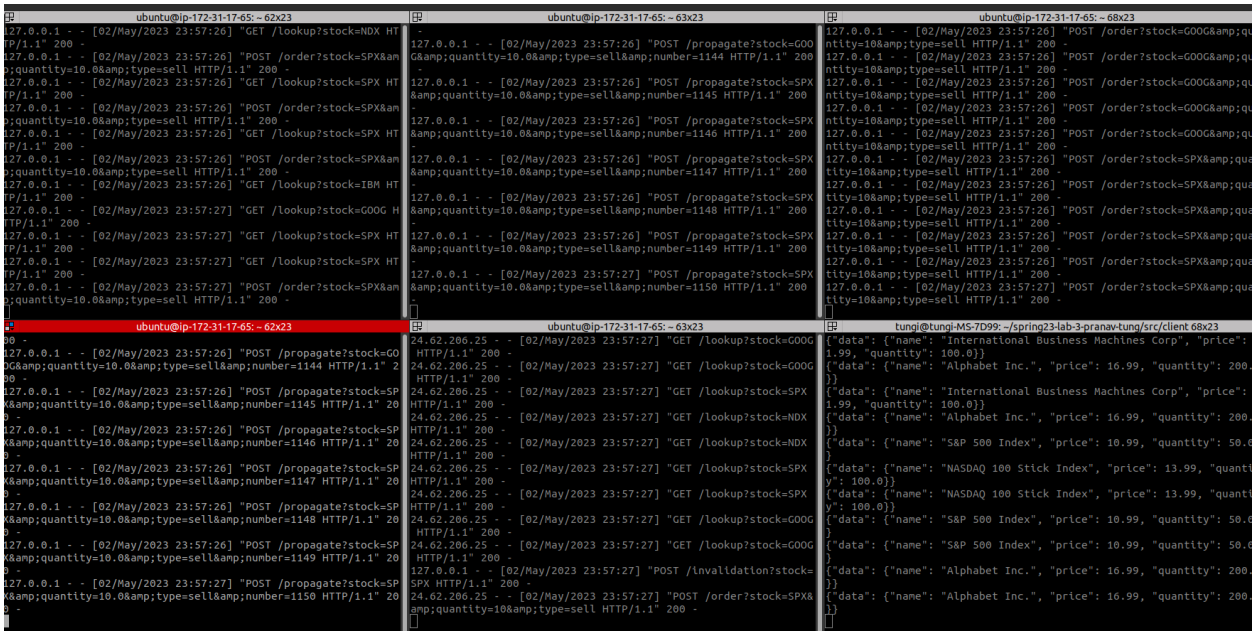
To start out testing, we open five terminals on the instance, one for the catalog, one for the frontend, and three for the order. The client is running on my local machine.



Then, we ran multiple clients currently using $ ./run_multi_client.sh {public_DNS_name_of_instance} 8080 100 0.8. During sending the requests, we shut down the order server with ID=3 (the highest ID, bottom left terminal):



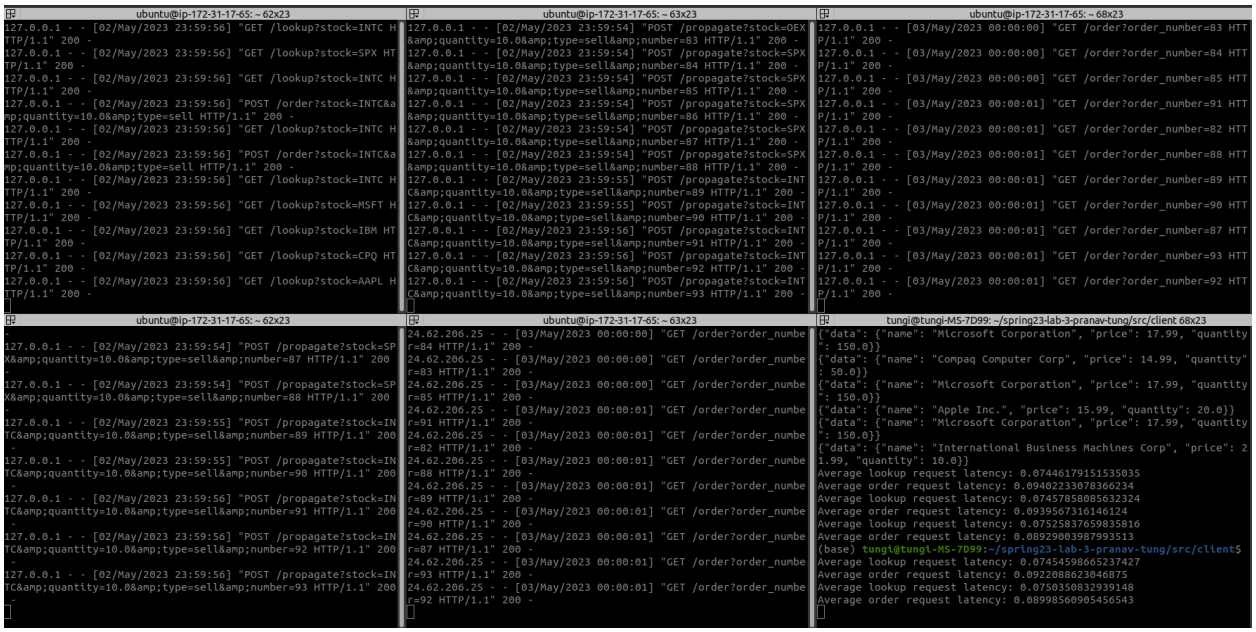
Then, we resumed the order ID=3 by appending the - -resume argument.



-

Notice that now the node with ID=2 becomes the leader (the upper right terminal). The ID=1 and ID=3 become the follower nodes which receive propagation requests from the leader.

After the clients exit, check if there are any error messages showing on the client's terminal. If not, that means the order information retrieved from the order database are the same with the locally stored order information.



We conducted several simulations of crash failures on both local machines and AWS to evaluate the resilience of our application. Specifically, we randomly killed a replica in the order server, including the leader, and observed the application's behavior. Remarkably, we found that the application continued to function normally after the replica was killed. Our findings indicate that the application runs as usual, and the client does not notice the crash failures either during order requests or the final order checking phase. Furthermore, when the order server was restored, it automatically synchronized with the databases of other replicas, ensuring that the order database files (order1/2/3.csv) were identical when the client exited. We also verified that the order information retrieved from the database was consistent with the client's locally stored information. Based on these findings, we can confidently conclude that our application is robust to crash failures, and clients will not experience any disruptions.