Christopher Shi
33061961
cshi@umass.edu

Lab 3 Output Document
CS677 Distributed and Operating Systems

2023-05-01

1 Output of Test Cases:

```
C:\Users\thewi\Desktop\CS677\lab3\src\test-cases>py testcases.py 10.0.0.246 56893
#########################################
##Test Case 1 - Lookup Functionality: ##
#########################################
Input Request: GET /stocks/nvidia
JSON Reply: {'data': {'name': 'nvidia', 'price': 240.63, 'quantity': 1000}}

#############################################################
##Test Case 2 - Lookup Error Handling (Stock not found): ##
#############################################################
Input Request: GET /stocks/imaginarycompany
JSON Reply: {'error': {'code': 404, 'message': 'stock not found'}}

######################################
##Test Case 3 - Sell Functionality: ##
######################################
Input Request: POST /orders
Input JSON: {"name": "meta", "type": "sell", "quantity": 10}
Stock Information before Request: {'data': {'name': 'meta', 'price': 194.02, 'quantity': 1000}}
JSON Reply: {'data': {'transaction_number': 0}}
Stock Information after Request: {'data': {'name': 'meta', 'price': 194.02, 'quantity': 1010}}

#####################################
##Test Case 4 - Buy Functionality: ##
#####################################
Input Request: POST /orders
Input JSON: {"name": "gamestop", "type": "buy", "quantity": 10}
Stock Information before Request: {'data': {'name': 'gamestop', 'price': 20.0, 'quantity': 1000}}
JSON Reply: {'data': {'transaction_number': 1}}
Stock Information after Request: {'data': {'name': 'gamestop', 'price': 20.0, 'quantity': 990}}

##############################################################################
##Test Case 5 - Buy Error Handling (Amount to buy > num of stocks avail.): ##
##############################################################################
Input Request: POST /orders
Input JSON: {"name": "nvidia", "type": "buy", "quantity": 10000000}
JSON Reply: {'error': {'code': 404, 'message': 'not enough stocks available to buy'}}

#######################################################################
##Test Case 6 - Buy/Sell Error Handling (Invalid quantity of stocks): ##
#######################################################################
Note: PROTO already defines quantity needs to be an int. As such, the only invalid value will be negat
Input Request: POST /orders
Input JSON: {"name": "amazon", "type": "buy", "quantity": -1000}
JSON Reply: {'error': {'code': 404, 'message': 'invalid number of stocks'}}

Input Request: POST /orders
Input JSON: {"name": "amazon", "type": "sell", "quantity": -1000}
JSON Reply: {'error': {'code': 404, 'message': 'invalid number of stocks'}}

#################################################################
##Test Case 7 - Buy/Sell Error Handling (Invalid request type): ##
```

```
#################################################################
Input Request: POST /orders
Input JSON: {"name": "meta", "type": "trade", "quantity": 100}
JSON Reply: {'error': {'code': 400, 'message': 'invalid request type'}}

#############################################################
##Test Case 8 - Buy/Sell Error Handling (Stock not found): ##
#############################################################
Input Request: POST /orders
Input JSON: {"name": "imaginarycompany", "type": "buy", "quantity": 100}
JSON Reply: {'error': {'code': 404, 'message': 'stock not found'}}

##########################################
##Test Case 9 - Caching Functionality:  ##
##########################################
Request 1a (s.t. it is stored in local cache): GET /stocks/amazon
Request 1b (s.t. we can retrieve from cache): GET /stocks/amazon
Latency with caching: 0.00027210000553168356
Request 2a (trade stock so cache is invalidated): POST /orders
Request 2b (lookup stock without cache): GET /stocks/amazon
Latency without caching: 0.0015700000076321885
Difference in time: 0.001297900002100505
Percent speed up: 82.6687895408323

##############################################
##Test Case 10 - Order Query Functionality: ##
##############################################
Input Request: POST /orders
Input JSON: {"name": "tesla", "type": "sell", "quantity": 10}
Client Locally Stored Information: {3: {'number': 3, 'name': 'tesla', 'type': 'sell', 'quantity': 10}}
Input Request: GET /orders/3
Output of Order Query: {'data': {'number': 3, 'name': 'tesla', 'type': 'sell', 'quantity': 10}}

###############################################
##Test Case 11 - Order Query Error Handling: ##
###############################################
Input Request: POST /orders
Input JSON: {"name": "amc", "type": "sell", "quantity": 10}
Client Locally Stored Information: {4: {'number': 4, 'name': 'amc', 'type': 'sell', 'quantity': 10}}
Input Request: GET /orders/5
Output of Order Query: {'error': {'code': 404, 'message': 'transaction not found'}}

###############################################
##Test Case 12 - Propagation Functionality: ##
###############################################
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "gamestop", "type": "sell", "quantity": 31}
JSON Reply: {'data': {'transaction_number': 0}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "gamestop", "type": "sell", "quantity": 14}
JSON Reply: {'data': {'transaction_number': 1}}
--- Content of Replica 1 at C:\Users\thewi\Desktop\CS677\lab3\src\order1.txt: ---
['{"0": {"name": "gamestop", "type": "sell", "quantity": 31}, "1": {"name": "gamestop", "type": "sell"
--- Content of Replica 2 at C:\Users\thewi\Desktop\CS677\lab3\src\order2.txt: ---
['{"0": {"name": "gamestop", "type": "sell", "quantity": 31}, "1": {"name": "gamestop", "type": "sell"
--- Content of Replica 3 at C:\Users\thewi\Desktop\CS677\lab3\src\order3.txt: ---
['{"0": {"name": "gamestop", "type": "sell", "quantity": 31}, "1": {"name": "gamestop", "type": "sell"
ALL THREE FILES ARE THE SAME: True
```

```
#######################################################
##Test Case 13 - Largest Replica ID Dead on Arrival: ##
#######################################################
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "tesla", "type": "sell", "quantity": 15}
JSON Reply: {'data': {'transaction_number': 0}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "tesla", "type": "sell", "quantity": 19}
JSON Reply: {'data': {'transaction_number': 1}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "tesla", "type": "buy", "quantity": 67}
JSON Reply: {'data': {'transaction_number': 2}}
--- Content of Replica 1 at C:\Users\thewi\Desktop\CS677\lab3\src\order1.txt: ---
['{"0": {"name": "tesla", "type": "sell", "quantity": 15}, "1": {"name": "tesla", "type": "sell", "qua
--- Content of Replica 2 at C:\Users\thewi\Desktop\CS677\lab3\src\order2.txt: ---
['{"0": {"name": "tesla", "type": "sell", "quantity": 15}, "1": {"name": "tesla", "type": "sell", "qua
--- Content of Replica 3 at C:\Users\thewi\Desktop\CS677\lab3\src\order3.txt: ---
['{}']
ALL THREE FILES ARE THE SAME: False
REPLICAS 2 AND 1 ARE THE SAME: True

#######################################################
##Test Case 14 - Leader Crashes During Execution: ##
#######################################################
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "meta", "type": "sell", "quantity": 69}
JSON Reply: {'data': {'transaction_number': 0}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "meta", "type": "sell", "quantity": 71}
JSON Reply: {'data': {'transaction_number': 1}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "meta", "type": "buy", "quantity": 30}
JSON Reply: {'data': {'transaction_number': 2}}
--- Content of Replica 1 at C:\Users\thewi\Desktop\CS677\lab3\src\order1.txt: ---
['{"0": {"name": "meta", "type": "sell", "quantity": 69}, "1": {"name": "meta", "type": "sell", "quant
--- Content of Replica 2 at C:\Users\thewi\Desktop\CS677\lab3\src\order2.txt: ---
['{"0": {"name": "meta", "type": "sell", "quantity": 69}, "1": {"name": "meta", "type": "sell", "quant
--- Content of Replica 3 at C:\Users\thewi\Desktop\CS677\lab3\src\order3.txt: ---
['{"0": {"name": "meta", "type": "sell", "quantity": 69}}']
ALL THREE FILES ARE THE SAME: False
REPLICAS 2 AND 1 ARE THE SAME: True

#######################################################
##Test Case 15 - Follower Crashes During Execution: ##
#######################################################
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "amc", "type": "sell", "quantity": 47}
JSON Reply: {'data': {'transaction_number': 0}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "amc", "type": "buy", "quantity": 83}
JSON Reply: {'data': {'transaction_number': 1}}
-------- Input --------
Input Request: POST /orders
```

```
Input JSON: {"name": "amc", "type": "buy", "quantity": 45}
JSON Reply: {'data': {'transaction_number': 2}}
--- Content of Replica 1 at C:\Users\thewi\Desktop\CS677\lab3\src\order1.txt: ---
['{"0": {"name": "amc", "type": "sell", "quantity": 47}, "1": {"name": "amc", "type": "buy", "quantity
--- Content of Replica 2 at C:\Users\thewi\Desktop\CS677\lab3\src\order2.txt: ---
['{"0": {"name": "amc", "type": "sell", "quantity": 47}, "1": {"name": "amc", "type": "buy", "quantity
--- Content of Replica 3 at C:\Users\thewi\Desktop\CS677\lab3\src\order3.txt: ---
['{"0": {"name": "amc", "type": "sell", "quantity": 47}, "1": {"name": "amc", "type": "buy", "quantity
ALL THREE FILES ARE THE SAME: False
REPLICAS 3 AND 1 ARE THE SAME: True

################################################################
##Test Case 16 - Leader Crashes and Rejoins During Execution: ##
################################################################
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "amc", "type": "sell", "quantity": 7}
JSON Reply: {'data': {'transaction_number': 0}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "amc", "type": "buy", "quantity": 40}
--- Content of Replica 3 at Time of Crash: ---
['{"0": {"name": "amc", "type": "sell", "quantity": 7}}']
JSON Reply: {'data': {'transaction_number': 1}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "amc", "type": "sell", "quantity": 75}
JSON Reply: {'data': {'transaction_number': 2}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "amc", "type": "sell", "quantity": 70}
JSON Reply: {'data': {'transaction_number': 3}}
--- Content of Replica 1 at C:\Users\thewi\Desktop\CS677\lab3\src\order1.txt: ---
['{"0": {"name": "amc", "type": "sell", "quantity": 7}, "1": {"name": "amc", "type": "buy", "quantity"
--- Content of Replica 2 at C:\Users\thewi\Desktop\CS677\lab3\src\order2.txt: ---
['{"0": {"name": "amc", "type": "sell", "quantity": 7}, "1": {"name": "amc", "type": "buy", "quantity"
--- Content of Replica 3 at C:\Users\thewi\Desktop\CS677\lab3\src\order3.txt: ---
['{"0": {"name": "amc", "type": "sell", "quantity": 7}, "1": {"name": "amc", "type": "buy", "quantity"
ALL THREE FILES ARE THE SAME: True
REPLICAS 3 AND 1 ARE THE SAME: True

################################################################
##Test Case 17 - Follower Crashes and Rejoins During Execution: ##
################################################################
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "nvidia", "type": "buy", "quantity": 44}
JSON Reply: {'data': {'transaction_number': 0}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "nvidia", "type": "sell", "quantity": 6}
--- Content of Replica 2 at Time of Crash: ---
['{"0": {"name": "nvidia", "type": "buy", "quantity": 44}, "1": {"name": "nvidia", "type": "sell", "qu
JSON Reply: {'data': {'transaction_number': 1}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "nvidia", "type": "buy", "quantity": 94}
JSON Reply: {'data': {'transaction_number': 2}}
-------- Input --------
Input Request: POST /orders
Input JSON: {"name": "nvidia", "type": "buy", "quantity": 51}
```

```
JSON Reply: {'data': {'transaction_number': 3}}
--- Content of Replica 1 at C:\Users\thewi\Desktop\CS677\lab3\src\order1.txt: ---
['{"0": {"name": "nvidia", "type": "buy", "quantity": 44}, "1": {"name": "nvidia", "type": "sell", "qu
--- Content of Replica 2 at C:\Users\thewi\Desktop\CS677\lab3\src\order2.txt: ---
['{"0": {"name": "nvidia", "type": "buy", "quantity": 44}, "1": {"name": "nvidia", "type": "sell", "qu
--- Content of Replica 3 at C:\Users\thewi\Desktop\CS677\lab3\src\order3.txt: ---
['{"0": {"name": "nvidia", "type": "buy", "quantity": 44}, "1": {"name": "nvidia", "type": "sell", "qu
ALL THREE FILES ARE THE SAME: True
REPLICAS 3 AND 1 ARE THE SAME: True


C:\Users\thewi\Desktop\CS677\lab3\src\test-cases>
```

2 Proof of Functionality:

2.1 Basic Functionality:

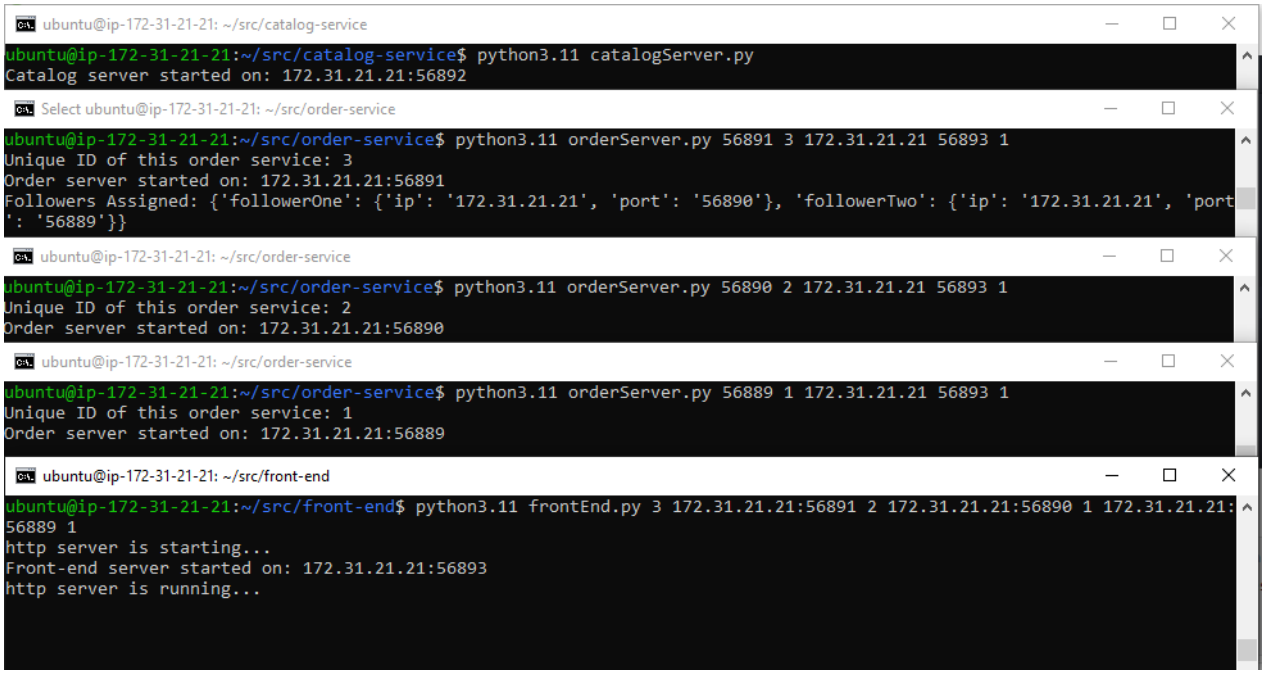

Figure 1: Output of the initialized microservices.

```
ubuntu@ip-172-31-16-170: ~/src/catalog-service
ubuntu@ip-172-31-16-170:~/src/catalog-service$ python3.11 catalogServer.py
Catalog server started on: 172.31.16.170:56892
```

```
ubuntu@ip-172-31-16-170: ~/src/order-service
ubuntu@ip-172-31-16-170:~/src/order-service$ python3.11 orderServer.py 56891 3 172.31.16.170 56893 1
Unique ID of this order service: 3
Order server started on: 172.31.16.170:56891
Followers Assigned: {'followerOne': {'ip': '172.31.16.170', 'port': '56890'}, 'followerTwo': {'ip': '172.31.16.170', 'po
rt': '56889'}}
Written on Replica 3: {41: {'name': 'nike', 'type': 'buy', 'quantity': 41}}
Written on Replica 3: {42: {'name': 'nvidia', 'type': 'sell', 'quantity': 33}}
Written on Replica 3: {43: {'name': 'nvidia', 'type': 'sell', 'quantity': 99}}
Written on Replica 3: {44: {'name': 'nvidia', 'type': 'buy', 'quantity': 97}}
```

```
ubuntu@ip-172-31-16-170: ~/src/order-service
ubuntu@ip-172-31-16-170:~/src/order-service$ python3.11 orderServer.py 56890 2 172.31.16.170 56893 1
Unique ID of this order service: 2
Order server started on: 172.31.16.170:56890
Propagated on Replica 2: {41: {'name': 'nike', 'type': 'buy', 'quantity': 41}}
Propagated on Replica 2: {42: {'name': 'nvidia', 'type': 'sell', 'quantity': 33}}
Propagated on Replica 2: {43: {'name': 'nvidia', 'type': 'sell', 'quantity': 99}}
Propagated on Replica 2: {44: {'name': 'nvidia', 'type': 'buy', 'quantity': 97}}
```

```
ubuntu@ip-172-31-16-170: ~/src/order-service
ubuntu@ip-172-31-16-170:~/src/order-service$ python3.11 orderServer.py 56889 1 172.31.16.170 56893 1
Unique ID of this order service: 1
Order server started on: 172.31.16.170:56889
Propagated on Replica 1: {41: {'name': 'nike', 'type': 'buy', 'quantity': 41}}
Propagated on Replica 1: {42: {'name': 'nvidia', 'type': 'sell', 'quantity': 33}}
Propagated on Replica 1: {43: {'name': 'nvidia', 'type': 'sell', 'quantity': 99}}
Propagated on Replica 1: {44: {'name': 'nvidia', 'type': 'buy', 'quantity': 97}}
```

```
ubuntu@ip-172-31-16-170: ~/src/front-end
ubuntu@ip-172-31-16-170:~/src/front-end$ python3.11 frontEnd.py 3 172.31.16.170:56891 2 172.31.16.170:56890 1 172.31.16.
170:56889 1
http server is starting...
Front-end server started on: 172.31.16.170:56893
http server is running...
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/tesla HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/intel HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/intel HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/nike HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/nvidia HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/nvidia HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/amazon HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/gamestop HTTP/1.1" 200 -
172.31.16.170 - - [27/Apr/2023 21:33:06] "GET /cache/nike HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "POST /orders HTTP/1.1" 200 -
172.31.16.170 - - [27/Apr/2023 21:33:06] "GET /cache/nvidia HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "POST /orders HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/nvidia HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "GET /stocks/nvidia HTTP/1.1" 200 -
172.31.16.170 - - [27/Apr/2023 21:33:06] "GET /cache/nvidia HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "POST /orders HTTP/1.1" 200 -
172.31.16.170 - - [27/Apr/2023 21:33:06] "GET /cache/nvidia HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:06] "POST /orders HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:07] "GET /orders/41 HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:07] "GET /orders/42 HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:07] "GET /orders/43 HTTP/1.1" 200 -
73.186.87.78 - - [27/Apr/2023 21:33:07] "GET /orders/44 HTTP/1.1" 200 -
```

Figure 2: Example output of the stock service. ”GET /cache/stock name” requests shown proving server-push invalidation functionality. Replication shown between the 3 order replicas.

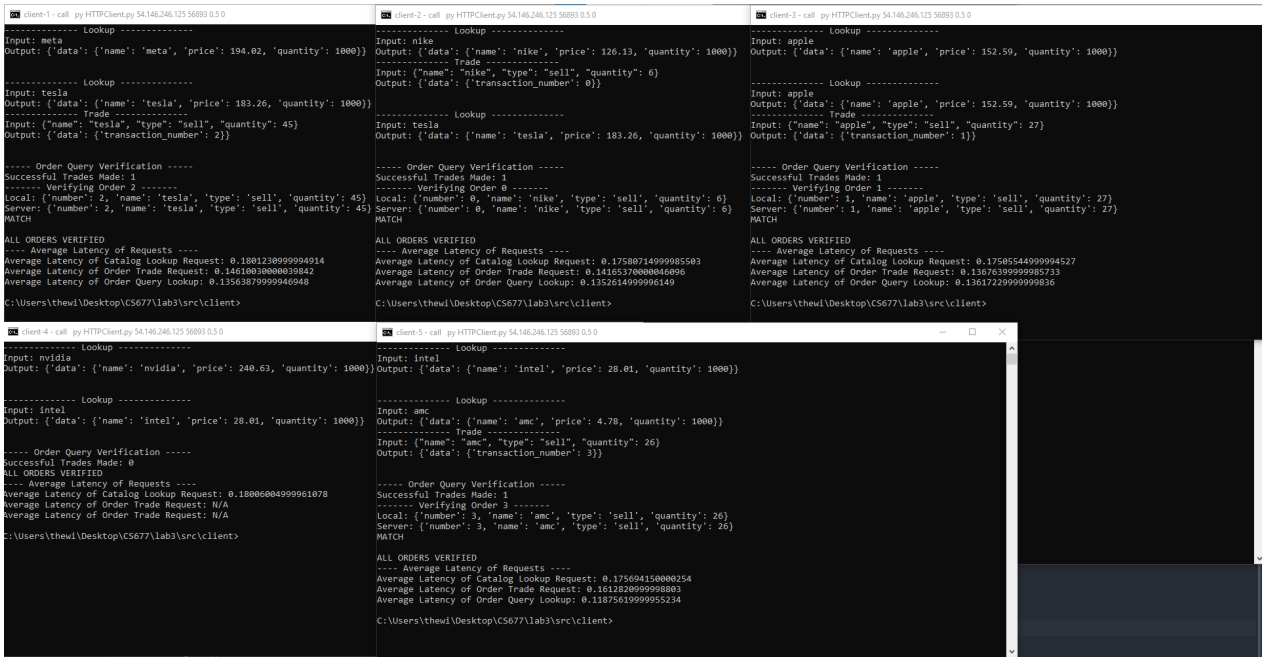

Figure 3: Example output of the 5 clients connecting to the stock service. Clients display order query verification functionality.

2.2 Fault Tolerance:

```
ubuntu@ip-172-31-16-170: ~/src/catalog-service
ubuntu@ip-172-31-16-170:~/src/catalog-service$ python3.11 catalogServer.py
Catalog server started on: 172.31.16.170:56892

ubuntu@ip-172-31-16-170: ~/src/order-service
ubuntu@ip-172-31-16-170:~/src/order-service$ python3.11 orderServer.py 56890 2 172.31.16.170 56893 1
Unique ID of this order service: 2
Order server started on: 172.31.16.170:56890
Followers Assigned: {'followerOne': {'ip': '172.31.16.170', 'port': '56889'}}

ubuntu@ip-172-31-16-170: ~/src/order-service
ubuntu@ip-172-31-16-170:~/src/order-service$ python3.11 orderServer.py 56889 1 172.31.16.170 56893 1
Unique ID of this order service: 1
Order server started on: 172.31.16.170:56889

ubuntu@ip-172-31-16-170: ~/src/front-end
ubuntu@ip-172-31-16-170:~/src/front-end$ python3.11 frontEnd.py 3 172.31.16.170:56891 2 172.31.16.170:56890 1 172.31.16.
170:56889 1
 Connection to 172.31.16.170:56891 timed out.
http server is starting...
Front-end server started on: 172.31.16.170:56893
http server is running...
```

Figure 4: Example output of when the largest replica ID is dead on arrival.

```
ubuntu@ip-172-31-16-170: ~/src/catalog-service
ubuntu@ip-172-31-16-170:~/src/catalog-service$ python3.11 catalogServer.py
Catalog server started on: 172.31.16.170:56892

ubuntu@ip-172-31-16-170: ~/src/order-service
ubuntu@ip-172-31-16-170:~/src/order-service$ python3.11 orderServer.py 56889 1 172.31.16.170 56893 1
Unique ID of this order service: 1
Order server started on: 172.31.16.170:56889
Followers Assigned: {}

ubuntu@ip-172-31-16-170: ~/src/front-end

ubuntu@ip-172-31-16-170:~/src/front-end$ python3.11 frontEnd.py 3 172.31.16.170:56891 2 172.31.16.170:56890 1 172.31.16.
170:56889 1
ubuntu@ip-172-31-16-170:~/src/front-end$ python3.11 frontEnd.py 3 172.31.16.170:56891 2 172.31.16.170:56890 1 172.31.16.
170:56889 1

 Connection to 172.31.16.170:56891 timed out.
 Connection to 172.31.16.170:56890 timed out.
http server is starting...
Front-end server started on: 172.31.16.170:56893
http server is running...
```

Figure 5: Example output of when the two largest replica IDs are dead on arrival.

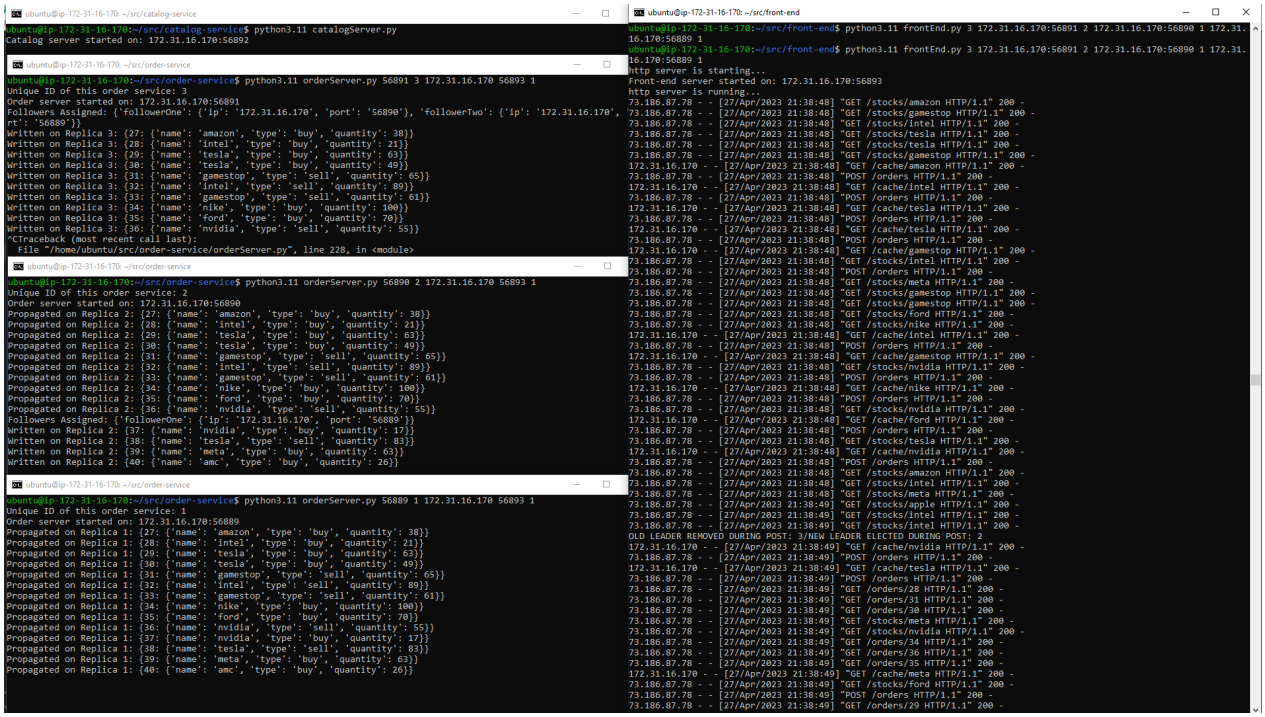

Figure 6: Example output when the leader replica crashes during execution.

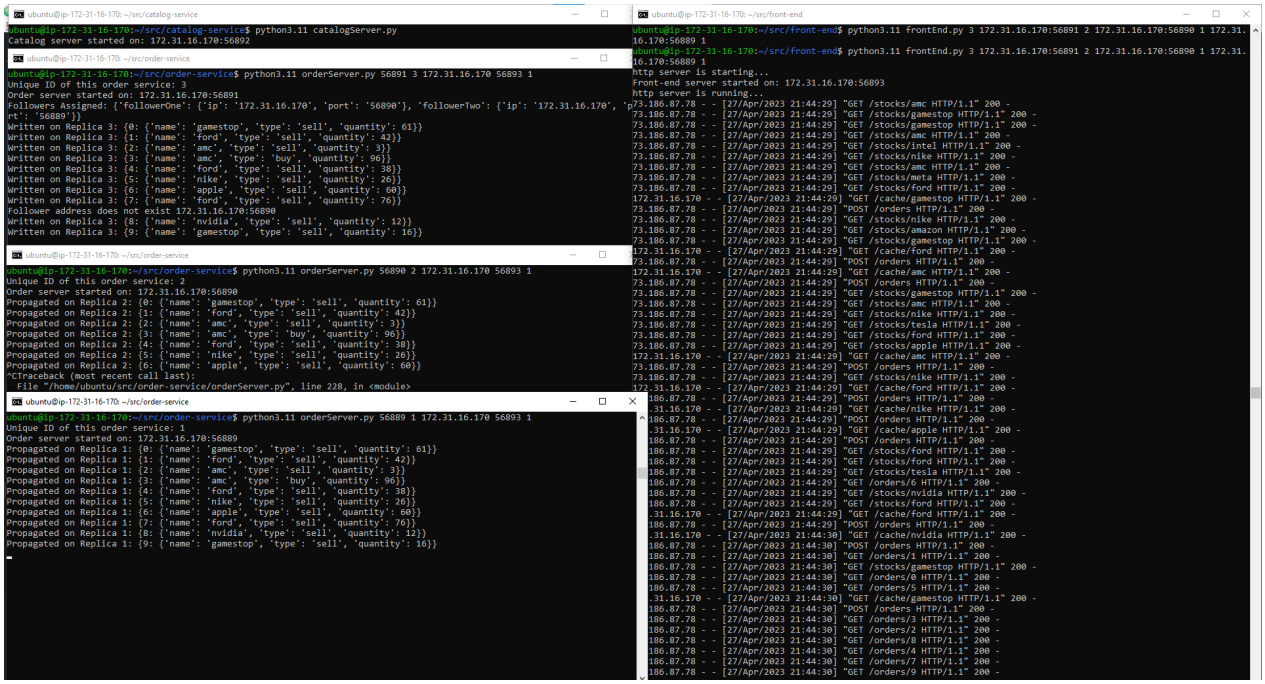

Figure 7: Example output when a follower replica crashes during execution.

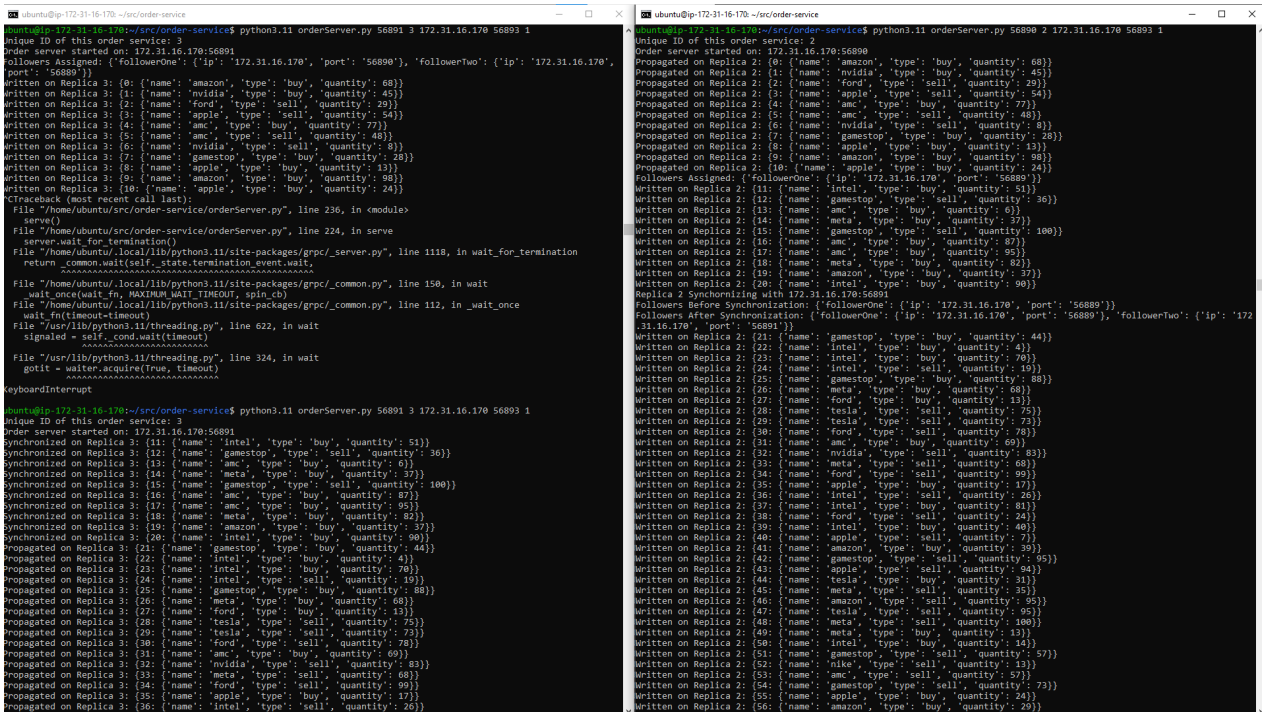

Figure 8: Example output when the leader replica crashes during execution and rejoins. Replica ID 3 shown on left, replica ID 2 shown on right.

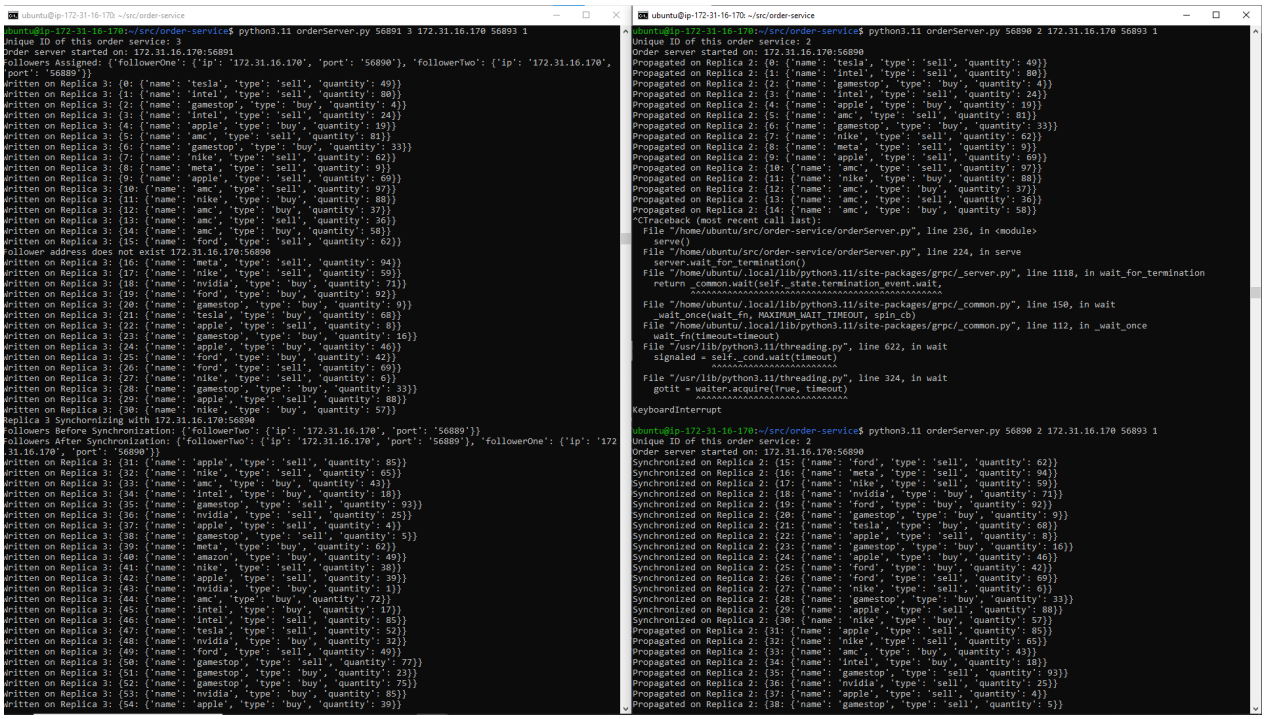

Figure 9: Example output when a follower replica crashes during execution and rejoins. Replica ID 3 shown on left, replica ID 2 shown on right.