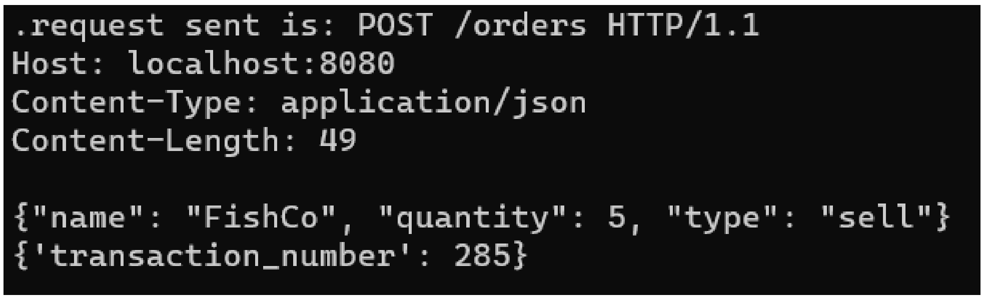
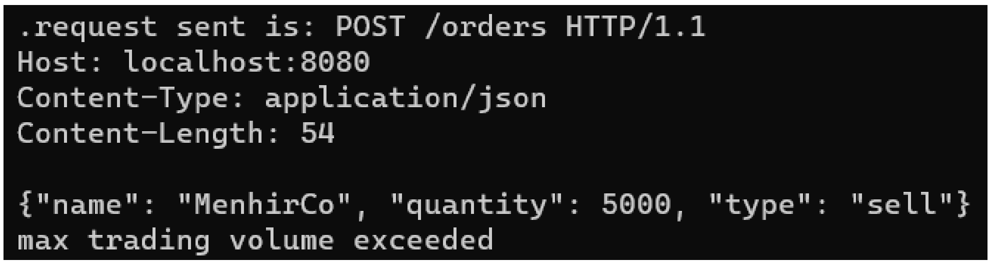
ORDER MICROSERVICE STANDALONE TEST OUTPUT

TC1 sell valid stocks



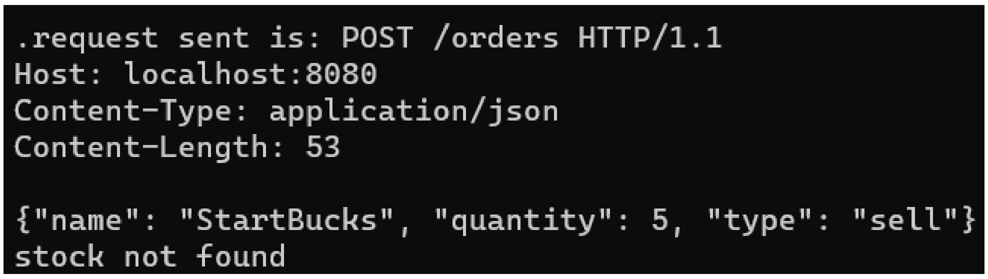
```
.request sent is: POST /orders HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 49

{"name": "FishCo", "quantity": 5, "type": "sell"}
{'transaction_number': 285}
```

TC2 sell stocks quantity exceeding trading limit



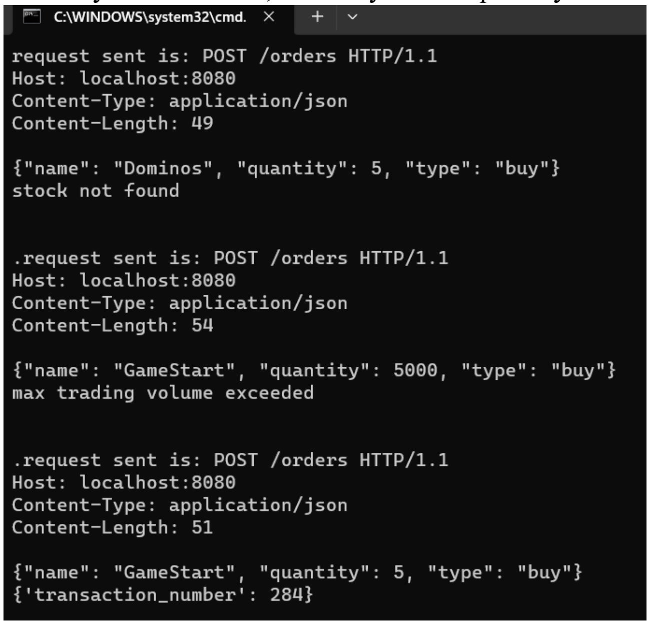
```
.request sent is: POST /orders HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 54

{"name": "MenhirCo", "quantity": 5000, "type": "sell"}
max trading volume exceeded
```

TC3 sell invalid stock name



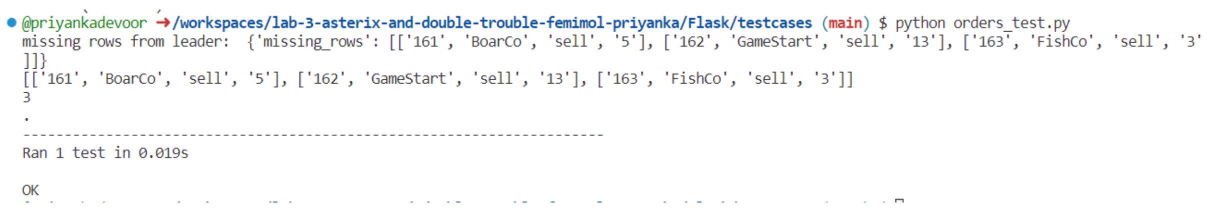
```
.request sent is: POST /orders HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 53

{"name": "StartBucks", "quantity": 5, "type": "sell"}
stock not found
```

TC4 buy invalid stock, TC5 buy stocks quantity exceeding trading limit, TC6 buy a valid stock



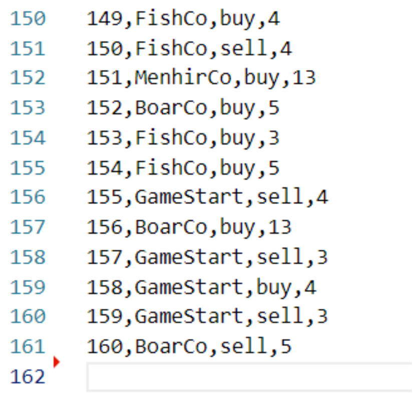
```
request sent is: POST /orders HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 49

{"name": "Dominos", "quantity": 5, "type": "buy"}
stock not found


.request sent is: POST /orders HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 54

{"name": "GameStart", "quantity": 5000, "type": "buy"}
max trading volume exceeded


.request sent is: POST /orders HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 51

{"name": "GameStart", "quantity": 5, "type": "buy"}
{'transaction_number': 284}
```

TC5 Sync Database



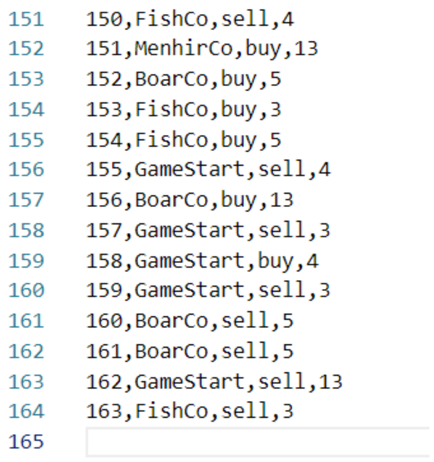
```
● @priyankadevoor →/workspaces/lab-3-asterix-and-double-trouble-femimol-priyanka/Flask/testcases (main) $ python orders_test.py
  missing rows from leader:  {'missing_rows': [['161', 'BoarCo', 'sell', '5'], ['162', 'GameStart', 'sell', '13'], ['163', 'FishCo', 'sell', '3'
  ]]}
  [['161', 'BoarCo', 'sell', '5'], ['162', 'GameStart', 'sell', '13'], ['163', 'FishCo', 'sell', '3']]
  3
  .
  ----------------------------------------------------------------------
  Ran 1 test in 0.019s

  OK
```

Before sync the csv file



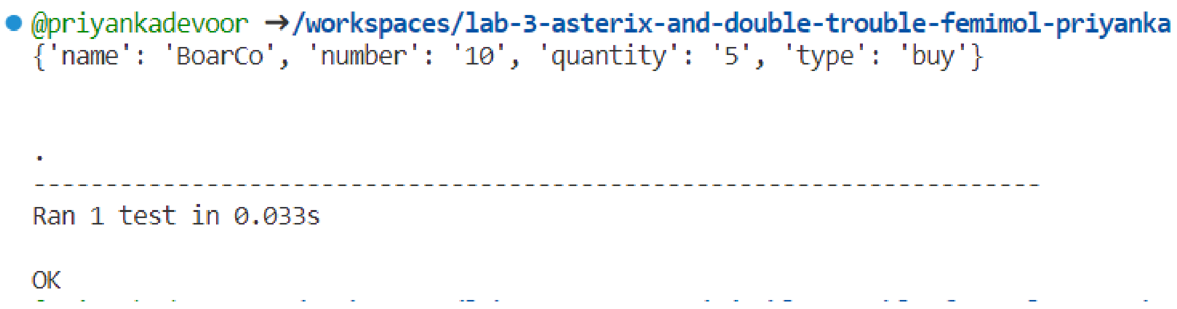
```
149,FishCo,buy,4
150,FishCo,sell,4
151,MenhirCo,buy,13
152,BoarCo,buy,5
153,FishCo,buy,3
154,FishCo,buy,5
155,GameStart,sell,4
156,BoarCo,buy,13
157,GameStart,sell,3
158,GameStart,buy,4
159,GameStart,sell,3
160,BoarCo,sell,5
```

After sync the csv file



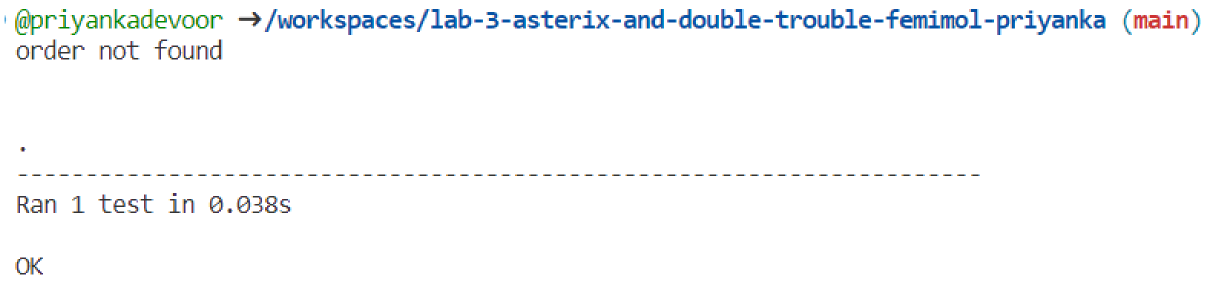
```
150,FishCo,sell,4
151,MenhirCo,buy,13
152,BoarCo,buy,5
153,FishCo,buy,3
154,FishCo,buy,5
155,GameStart,sell,4
156,BoarCo,buy,13
157,GameStart,sell,3
158,GameStart,buy,4
159,GameStart,sell,3
160,BoarCo,sell,5
161,BoarCo,sell,5
162,GameStart,sell,13
163,FishCo,sell,3
```

TC6 – Order query for a valid transaction



```
● @priyankadevoor →/workspaces/lab-3-asterix-and-double-trouble-femimol-priyanka
  {'name': 'BoarCo', 'number': '10', 'quantity': '5', 'type': 'buy'}


  .
  ----------------------------------------------------------------------
  Ran 1 test in 0.033s

  OK
```

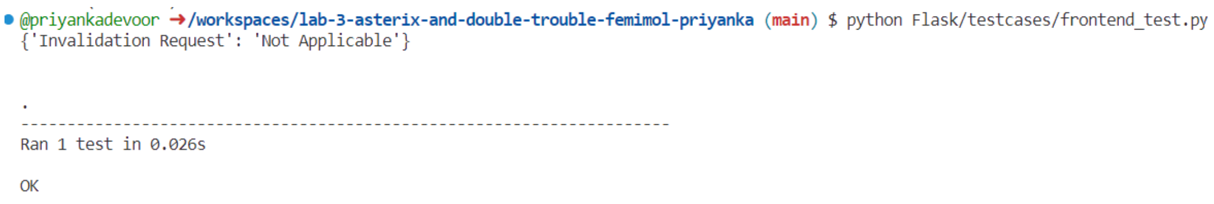
TC7 Order query for a transaction number that deos not exist.



```
@priyankadevoor →/workspaces/lab-3-asterix-and-double-trouble-femimol-priyanka (main)
order not found


.
----------------------------------------------------------------------
Ran 1 test in 0.038s

OK
```

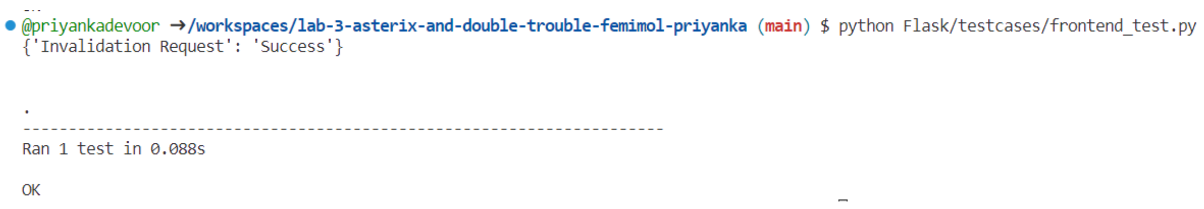
FRONT-END STANDALONE TEST CASE OUTPUT

1. Caching testcase when the stock name has not changed in the database.



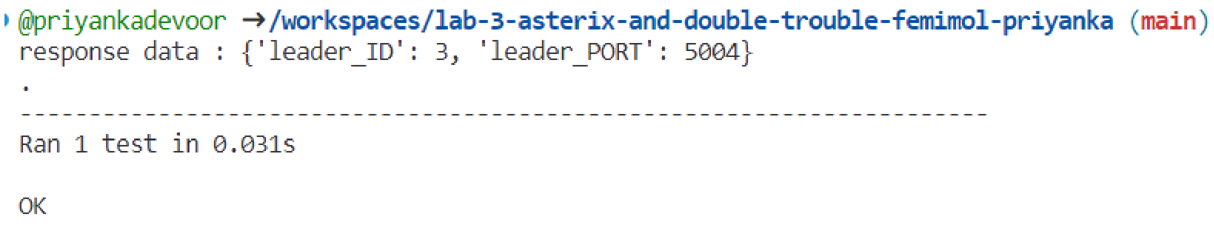
```
● @priyankadevoor →/workspaces/lab-3-asterix-and-double-trouble-femimol-priyanka (main) $ python Flask/testcases/frontend_test.py
  {'Invalidation Request': 'Not Applicable'}


  .
  ----------------------------------------------------------------------
  Ran 1 test in 0.026s

  OK
```

2. Caching testcase output when the stock database has updated hence, invalidate memcache is called which will delete that particular stock name from the inmemory dictionary.



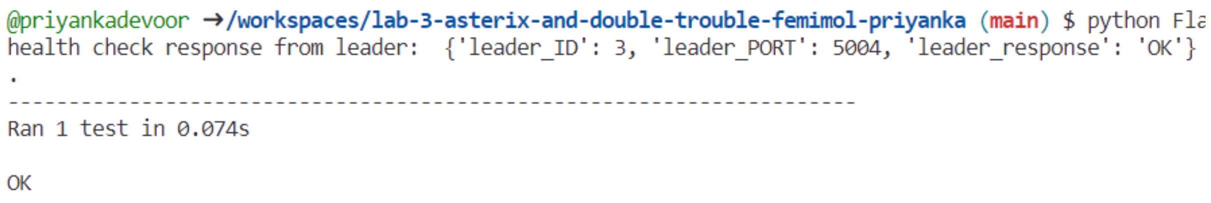
```
● @priyankadevoor →/workspaces/lab-3-asterix-and-double-trouble-femimol-priyanka (main) $ python Flask/testcases/frontend_test.py
  {'Invalidation Request': 'Success'}


  .
  ----------------------------------------------------------------------
  Ran 1 test in 0.088s

  OK
```

3. Leader election when all 3 replicas are running



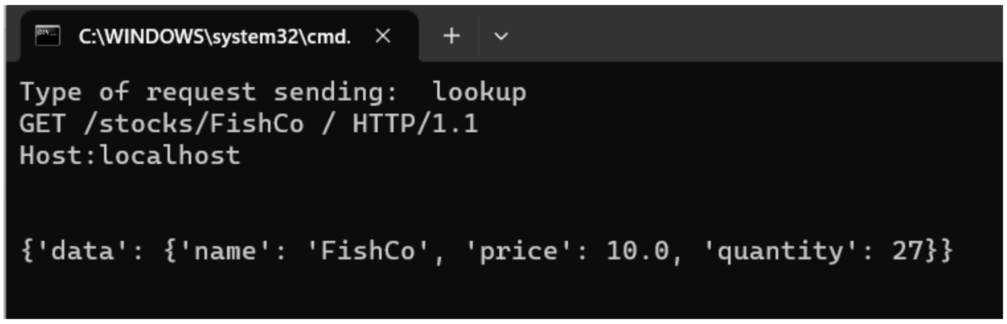
```
@priyankadevoor →/workspaces/lab-3-asterix-and-double-trouble-femimol-priyanka (main)
response data : {'leader_ID': 3, 'leader_PORT': 5004}
.
----------------------------------------------------------------------
Ran 1 test in 0.031s

OK
```

4. health check to the leader



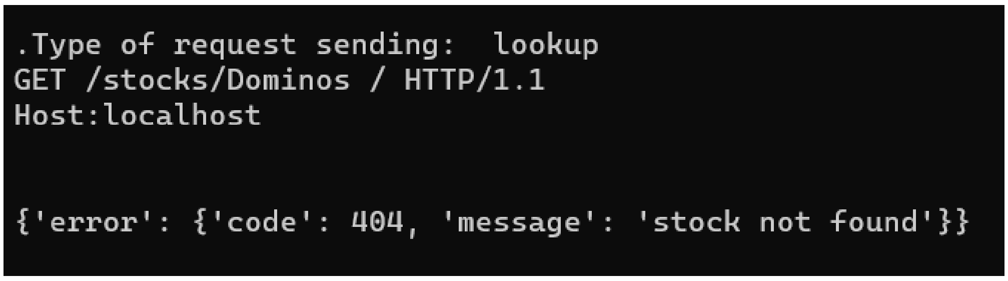
```
@priyankadevoor →/workspaces/lab-3-asterix-and-double-trouble-femimol-priyanka (main) $ python Fla
health check response from leader:  {'leader_ID': 3, 'leader_PORT': 5004, 'leader_response': 'OK'}
.
----------------------------------------------------------------------
Ran 1 test in 0.074s

OK
```

5. Lookup of valid stock name



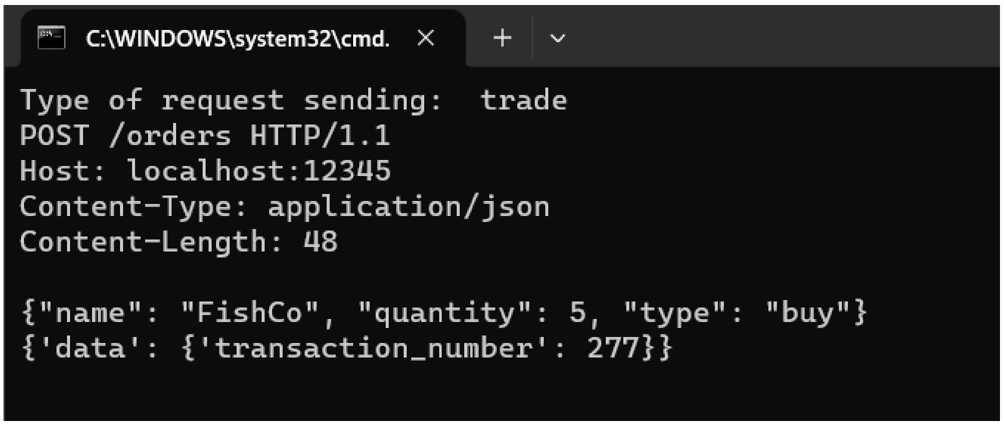
```
Type of request sending:  lookup
GET /stocks/FishCo / HTTP/1.1
Host:localhost


{'data': {'name': 'FishCo', 'price': 10.0, 'quantity': 27}}
```

6. Lookup for invalid stock name



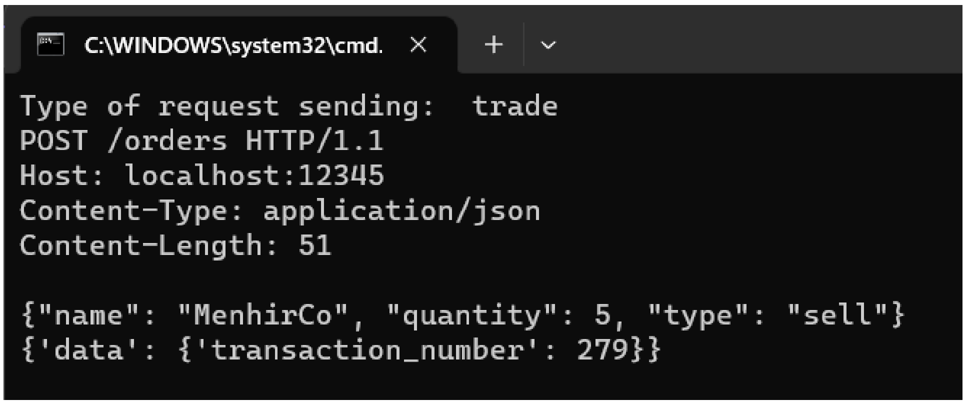
```
.Type of request sending:  lookup
GET /stocks/Dominos / HTTP/1.1
Host:localhost


{'error': {'code': 404, 'message': 'stock not found'}}
```

7. Buy valid stock



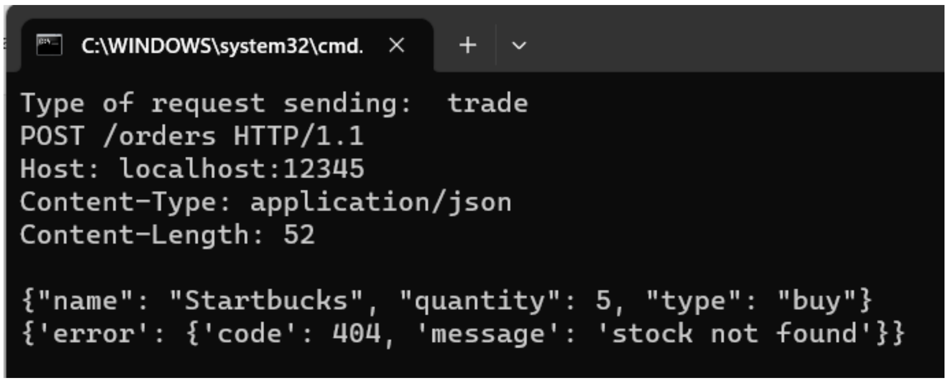
```
Type of request sending:  trade
POST /orders HTTP/1.1
Host: localhost:12345
Content-Type: application/json
Content-Length: 48

{"name": "FishCo", "quantity": 5, "type": "buy"}
{'data': {'transaction_number': 277}}
```

8. Sell valid stock



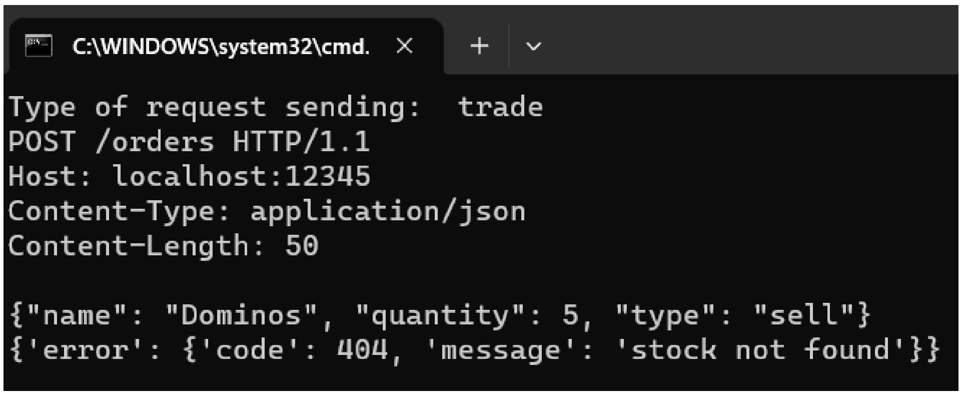
```
Type of request sending:  trade
POST /orders HTTP/1.1
Host: localhost:12345
Content-Type: application/json
Content-Length: 51

{"name": "MenhirCo", "quantity": 5, "type": "sell"}
{'data': {'transaction_number': 279}}
```

9. Buy invalid stock



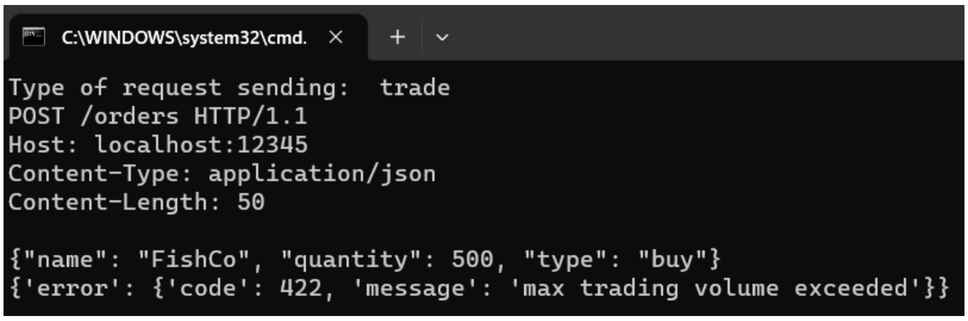
```
Type of request sending:  trade
POST /orders HTTP/1.1
Host: localhost:12345
Content-Type: application/json
Content-Length: 52

{"name": "Startbucks", "quantity": 5, "type": "buy"}
{'error': {'code': 404, 'message': 'stock not found'}}
```

10. Sell invalid stock



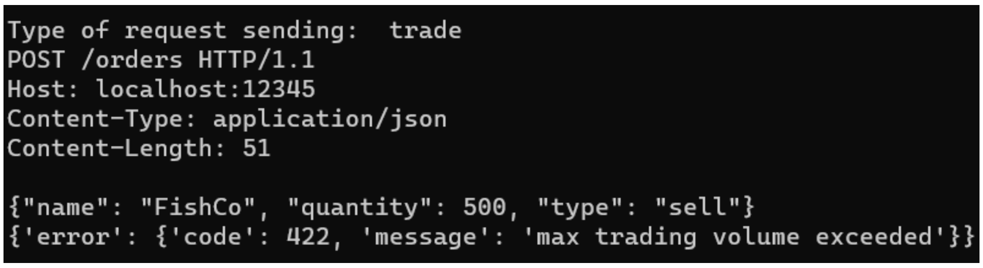
```
Type of request sending:  trade
POST /orders HTTP/1.1
Host: localhost:12345
Content-Type: application/json
Content-Length: 50

{"name": "Dominos", "quantity": 5, "type": "sell"}
{'error': {'code': 404, 'message': 'stock not found'}}
```

11. Buy stocks greater than trading limit



```
Type of request sending:  trade
POST /orders HTTP/1.1
Host: localhost:12345
Content-Type: application/json
Content-Length: 50

{"name": "FishCo", "quantity": 500, "type": "buy"}
{'error': {'code': 422, 'message': 'max trading volume exceeded'}}
```

12. Sell stocks greater than trading limit

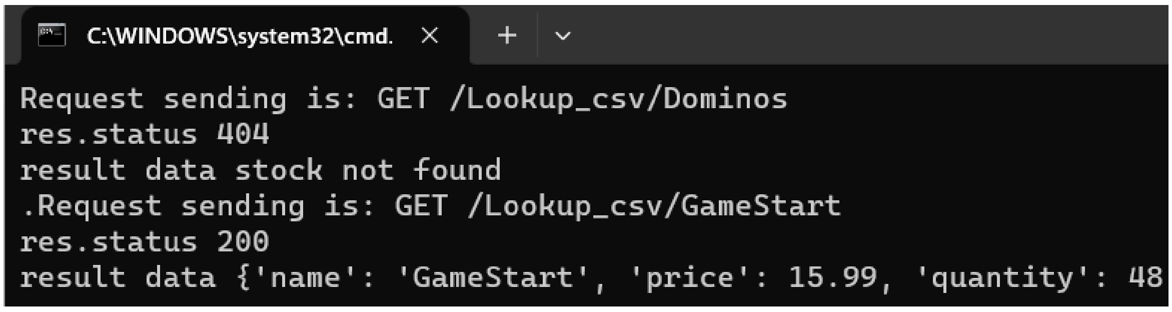


```
Type of request sending:  trade
POST /orders HTTP/1.1
Host: localhost:12345
Content-Type: application/json
Content-Length: 51

{"name": "FishCo", "quantity": 500, "type": "sell"}
{'error': {'code': 422, 'message': 'max trading volume exceeded'}}
```

CATALOG MICROSERVICE STANDALONE TEST OUTPUT

TC1 send valid stock name lookup

TC2 send request for invalid stock name lookup

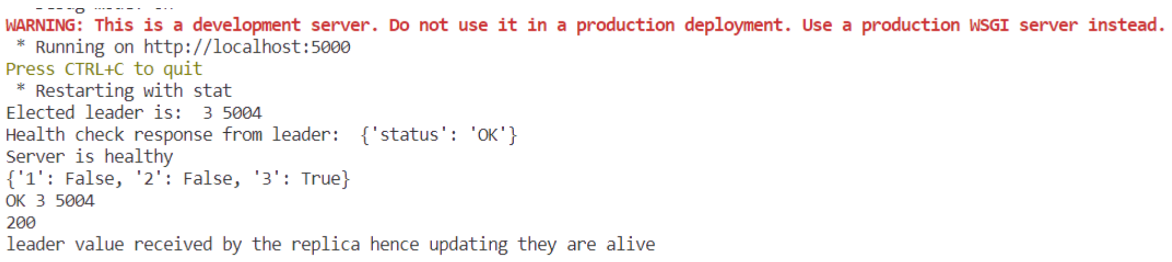


```
Request sending is: GET /Lookup_csv/Dominos
res.status 404
result data stock not found
.Request sending is: GET /Lookup_csv/GameStart
res.status 200
result data {'name': 'GameStart', 'price': 15.99, 'quantity': 48
```

MANUAL TESTING OF REPLICATION AND LEADER ELECTION AFTER CRASHING

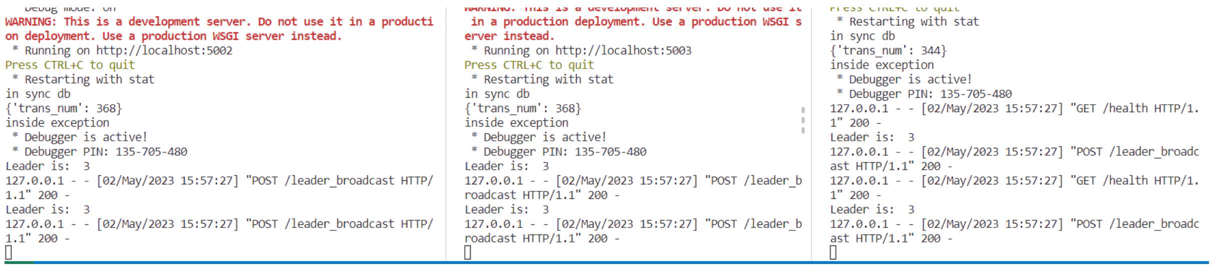
1. Leader elected is 3 when all 3 replicas are running and leader respond OK for health check message.

FRONT_END OUTPUT



```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://localhost:5000
Press CTRL+C to quit
 * Restarting with stat
Elected leader is:  3 5004
Health check response from leader:  {'status': 'OK'}
Server is healthy
{'1': False, '2': False, '3': True}
OK 3 5004
200
leader value received by the replica hence updating they are alive
```

REPLICAS OUTPUT

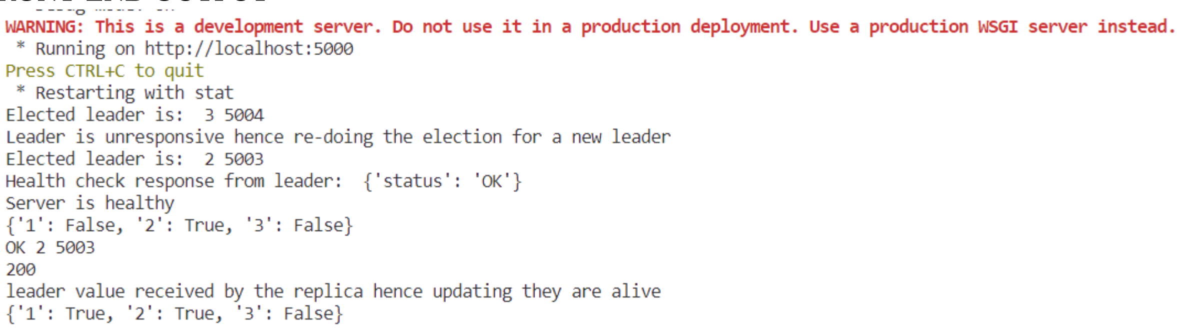


```
WARNING: This is a development server. Do not use it in a producti
on deployment. Use a production WSGI server instead.
 * Running on http://localhost:5002
Press CTRL+C to quit
 * Restarting with stat
in sync db
{'trans_num': 368}
inside exception
 * Debugger is active!
 * Debugger PIN: 135-705-480
Leader is:  3
127.0.0.1 - - [02/May/2023 15:57:27] "POST /leader_broadcast HTTP/
1.1" 200 -
Leader is:  3
127.0.0.1 - - [02/May/2023 15:57:27] "POST /leader_broadcast HTTP/
1.1" 200 -
```

```
 in a production deployment. Use a production WSGI s
erver instead.
 * Running on http://localhost:5003
Press CTRL+C to quit
 * Restarting with stat
in sync db
{'trans_num': 368}
inside exception
 * Debugger is active!
 * Debugger PIN: 135-705-480
Leader is:  3
127.0.0.1 - - [02/May/2023 15:57:27] "POST /leader_b
roadcast HTTP/1.1" 200 -
Leader is:  3
127.0.0.1 - - [02/May/2023 15:57:27] "POST /leader_b
roadcast HTTP/1.1" 200 -
```

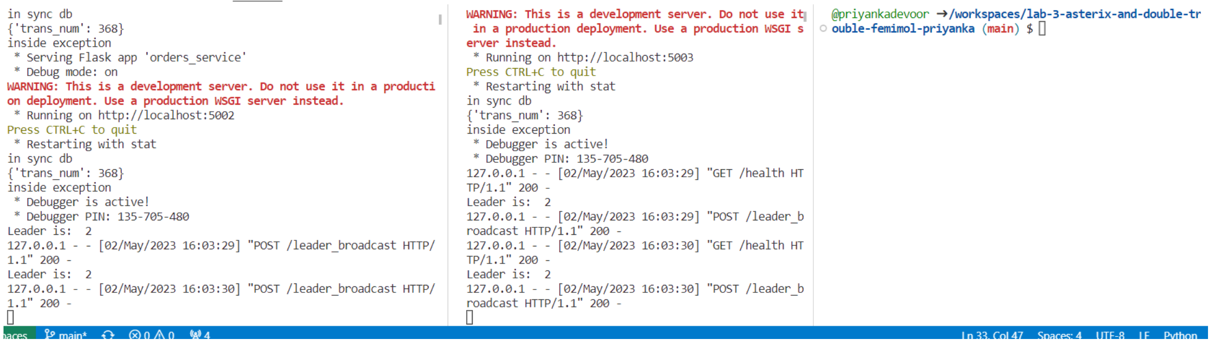
```
 * Restarting with stat
in sync db
{'trans_num': 344}
inside exception
 * Debugger is active!
 * Debugger PIN: 135-705-480
127.0.0.1 - - [02/May/2023 15:57:27] "GET /health HTTP/1.
1" 200 -
Leader is:  3
127.0.0.1 - - [02/May/2023 15:57:27] "POST /leader_broadc
ast HTTP/1.1" 200 -
127.0.0.1 - - [02/May/2023 15:57:27] "GET /health HTTP/1.
1" 200 -
Leader is:  3
127.0.0.1 - - [02/May/2023 15:57:27] "POST /leader_broadc
ast HTTP/1.1" 200 -
```

2. When the leader with ID – 3 has crashed, redoing the leader election.
   FRONT-END OUTPUT



```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://localhost:5000
Press CTRL+C to quit
 * Restarting with stat
Elected leader is:  3 5004
Leader is unresponsive hence re-doing the election for a new leader
Elected leader is:  2 5003
Health check response from leader:  {'status': 'OK'}
Server is healthy
{'1': False, '2': True, '3': False}
OK 2 5003
200
leader value received by the replica hence updating they are alive
{'1': True, '2': True, '3': False}
```

REPLICAS OUTPUT



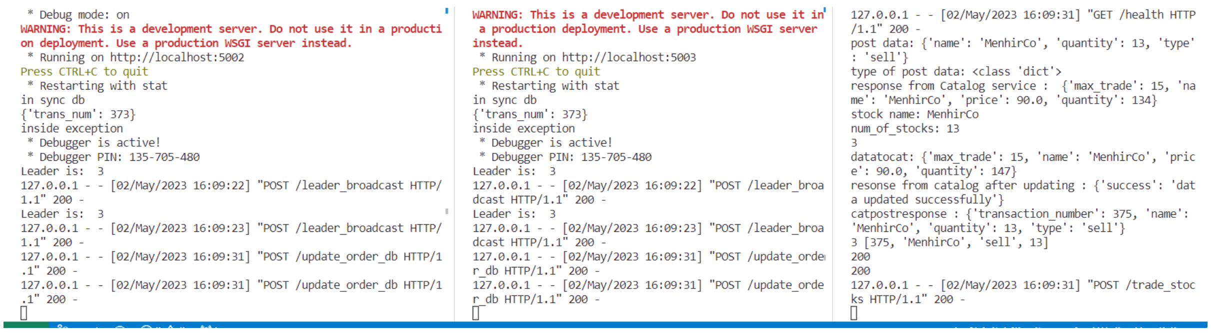
```
in sync db
{'trans_num': 368}
inside exception
 * Serving Flask app 'orders_service'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a producti
on deployment. Use a production WSGI server instead.
 * Running on http://localhost:5002
Press CTRL+C to quit
 * Restarting with stat
in sync db
{'trans_num': 368}
inside exception
 * Debugger is active!
 * Debugger PIN: 135-705-480
Leader is:  2
127.0.0.1 - - [02/May/2023 16:03:29] "POST /leader_broadcast HTTP/
1.1" 200 -
Leader is:  2
127.0.0.1 - - [02/May/2023 16:03:30] "POST /leader_broadcast HTTP/
1.1" 200 -
```

```
WARNING: This is a development server. Do not use it
 in a production deployment. Use a production WSGI s
erver instead.
 * Running on http://localhost:5003
Press CTRL+C to quit
 * Restarting with stat
in sync db
{'trans_num': 368}
inside exception
 * Debugger is active!
 * Debugger PIN: 135-705-480
127.0.0.1 - - [02/May/2023 16:03:29] "GET /health HT
TP/1.1" 200 -
Leader is:  2
127.0.0.1 - - [02/May/2023 16:03:29] "POST /leader_b
roadcast HTTP/1.1" 200 -
127.0.0.1 - - [02/May/2023 16:03:30] "GET /health HT
TP/1.1" 200 -
Leader is:  2
127.0.0.1 - - [02/May/2023 16:03:30] "POST /leader_b
roadcast HTTP/1.1" 200 -
```

```
@priyankadevoor →/workspaces/lab-3-asterix-and-double-tr
ouble-femimol-priyanka (main) $
```

```
paces   main*   ⊗ 0 ⚠ 0   4                                   Ln 33, Col 47   Spaces: 4   UTF-8   LF   Python
```

OUTPUTS of all trade request being forwarded only to the leader and all the replicas are getting update_db request after a successful trade transaction in order to keep the DB in sync.



```
 * Debug mode: on
WARNING: This is a development server. Do not use it in a producti
on deployment. Use a production WSGI server instead.
 * Running on http://localhost:5002
Press CTRL+C to quit
 * Restarting with stat
in sync db
{'trans_num': 373}
inside exception
 * Debugger is active!
 * Debugger PIN: 135-705-480
Leader is:  3
127.0.0.1 - - [02/May/2023 16:09:22] "POST /leader_broadcast HTTP/
1.1" 200 -
Leader is:  3
127.0.0.1 - - [02/May/2023 16:09:23] "POST /leader_broadcast HTTP/
1.1" 200 -
127.0.0.1 - - [02/May/2023 16:09:31] "POST /update_order_db HTTP/1
.1" 200 -
127.0.0.1 - - [02/May/2023 16:09:31] "POST /update_order_db HTTP/1
.1" 200 -
```

```
WARNING: This is a development server. Do not use it in
 a production deployment. Use a production WSGI server
instead.
 * Running on http://localhost:5003
Press CTRL+C to quit
 * Restarting with stat
in sync db
{'trans_num': 373}
inside exception
 * Debugger is active!
 * Debugger PIN: 135-705-480
Leader is:  3
127.0.0.1 - - [02/May/2023 16:09:22] "POST /leader_broa
dcast HTTP/1.1" 200 -
Leader is:  3
127.0.0.1 - - [02/May/2023 16:09:23] "POST /leader_broa
dcast HTTP/1.1" 200 -
127.0.0.1 - - [02/May/2023 16:09:31] "POST /update_orde
r_db HTTP/1.1" 200 -
127.0.0.1 - - [02/May/2023 16:09:31] "POST /update_orde
r_db HTTP/1.1" 200 -
```

```
127.0.0.1 - - [02/May/2023 16:09:31] "GET /health HTTP
/1.1" 200 -
post data: {'name': 'MenhirCo', 'quantity': 13, 'type'
: 'sell'}
type of post data: <class 'dict'>
response from Catalog service :  {'max_trade': 15, 'na
me': 'MenhirCo', 'price': 90.0, 'quantity': 134}
stock name: MenhirCo
num_of_stocks: 13
3
datatocat: {'max_trade': 15, 'name': 'MenhirCo', 'pric
e': 90.0, 'quantity': 147}
resonse from catalog after updating : {'success': 'dat
a updated successfully'}
catpostresponse : {'transaction_number': 375, 'name':
'MenhirCo', 'quantity': 13, 'type': 'sell'}
3 [375, 'MenhirCo', 'sell', 13]
200
200
127.0.0.1 - - [02/May/2023 16:09:31] "POST /trade_stoc
ks HTTP/1.1" 200 -
```