ASSIGNMENT 3

Group 12 Members

Vaibhav Kumar Singh	180101086
Nishchay Manwani	180101051

Application ID 2: Client Server Trading System using socket programming

CREATING EXECUTABLE FILES

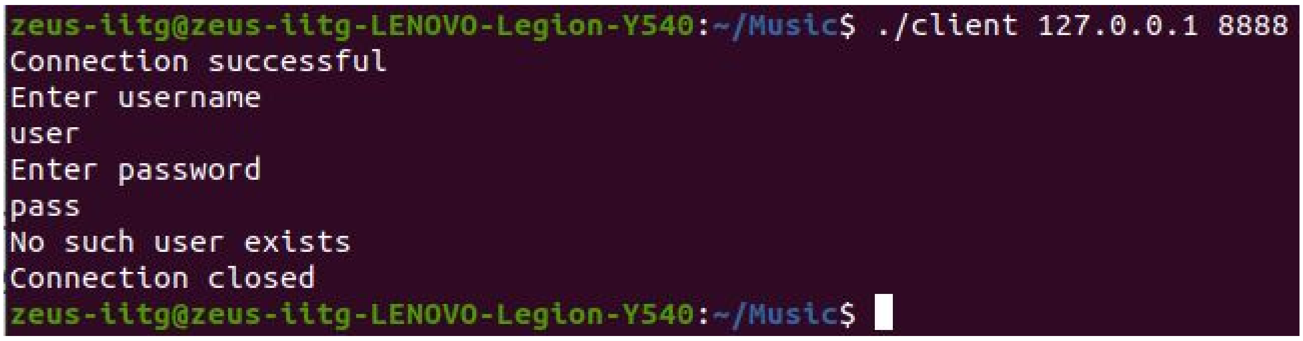
- In order to create executable files for server and client, run the command **"`make all`"** from the terminal. The Makefile provided will create the executable files. The name of the executable files for server and client are **"`server`"** and **"`client`"** respectively. To remove any previously made executable files, simply run **"`make clean`"** from the terminal.
- Alternatively, the executable files can be made directly using the terminal using the following two commands:
    - gcc -Wall server.c -o server
    - gcc -Wall client.c -o client

RUNNING EXECUTABLE FILES

- Open a new terminal window. Firstly we will run our server. For this, we need to specify the port number on which the server will run. Port numbers can run from 0 to 65535. Port numbers from 0 to 1023 are reserved for common TCP/IP applications. Hence, we generally provide port numbers greater than 1023 for our server. The command for running the server is
    - ./server <port_number>   (e.g.   ./server 8888)
- To run our client, we open a new terminal window. We need to provide 2 arguments to our client, namely
    1. The ip address of the server
    2. The port number at which the server is listening
- Since we are running the server on our local machine, we give the ip address of the loopback interface (127.0.0.1). The command for running the client is
    - ./client <ip_address> <port_number>   (e.g.   ./client 127.0.0.1 8888)

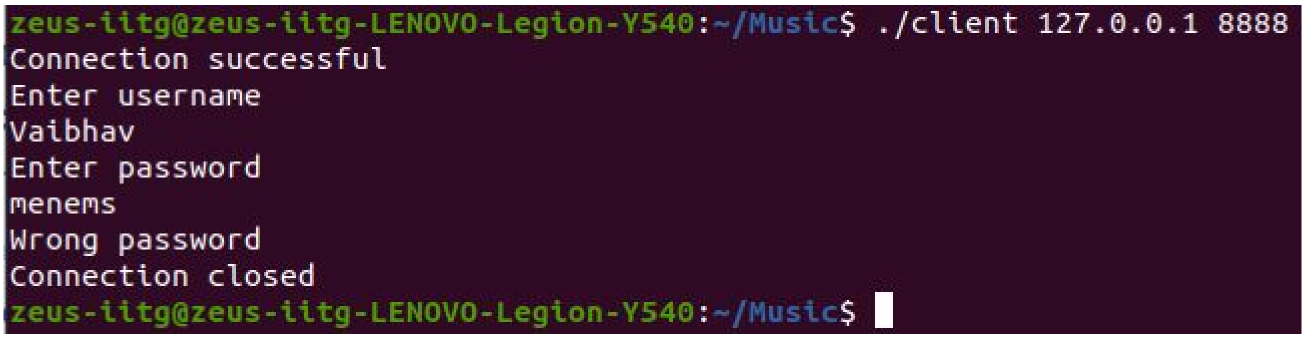
FUNCTIONALITIES AND USAGE

- On running the client, the user needs to enter the username and password. The server side searches for the entered credentials in its database and returns a value which indicates one of the following possibilities:
    - wrong username



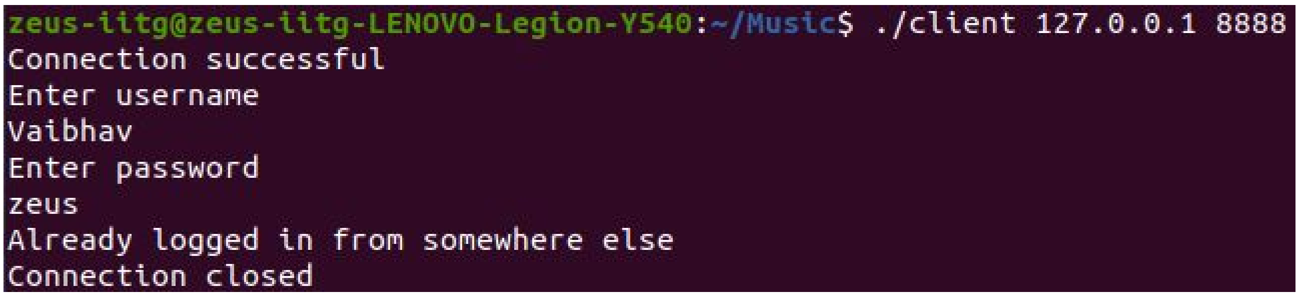
```
zeus-iitg@zeus-iitg-LENOVO-Legion-Y540:~/Music$ ./client 127.0.0.1 8888
Connection successful
Enter username
user
Enter password
pass
No such user exists
Connection closed
zeus-iitg@zeus-iitg-LENOVO-Legion-Y540:~/Music$
```

- ➢ wrong password



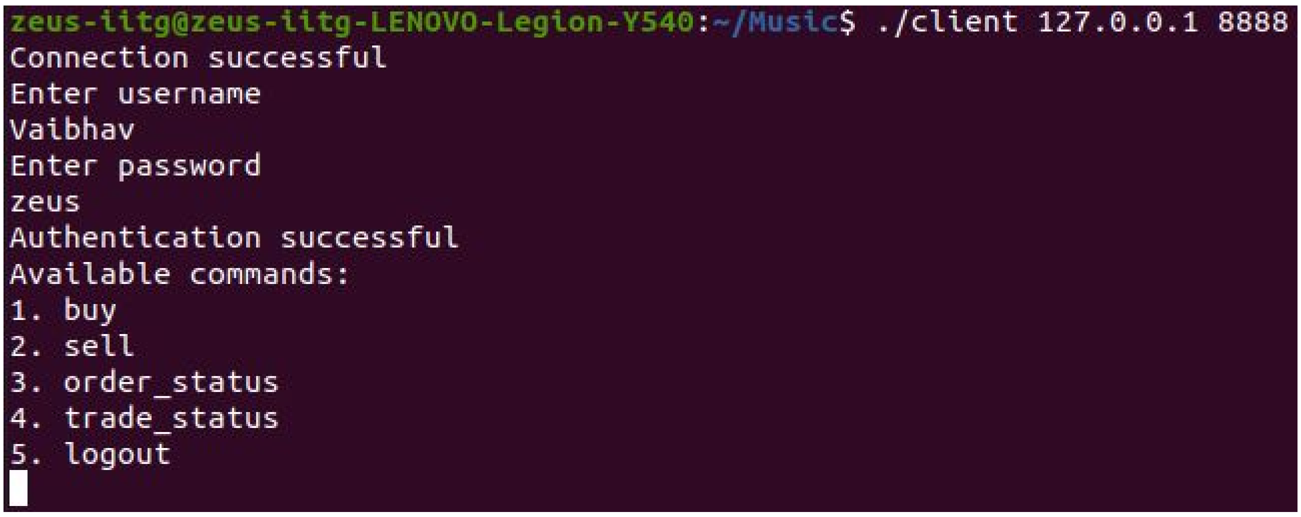
```
zeus-iitg@zeus-iitg-LENOVO-Legion-Y540:~/Music$ ./client 127.0.0.1 8888
Connection successful
Enter username
Vaibhav
Enter password
menems
Wrong password
Connection closed
zeus-iitg@zeus-iitg-LENOVO-Legion-Y540:~/Music$
```

- ➢ already logged in from somewhere else



```
zeus-iitg@zeus-iitg-LENOVO-Legion-Y540:~/Music$ ./client 127.0.0.1 8888
Connection successful
Enter username
Vaibhav
Enter password
zeus
Already logged in from somewhere else
Connection closed
```

- ➢ authentication successful

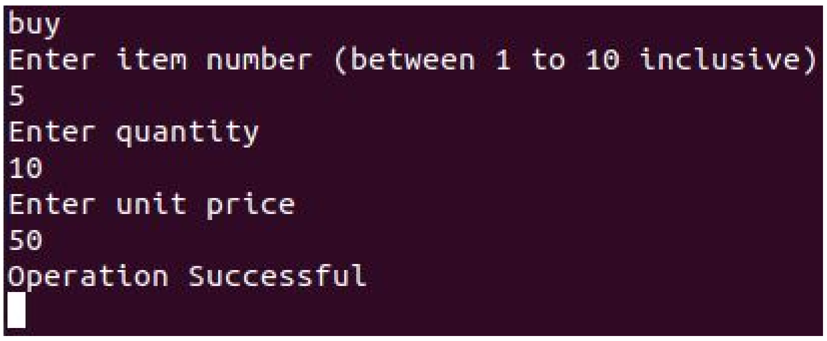


```
zeus-iitg@zeus-iitg-LENOVO-Legion-Y540:~/Music$ ./client 127.0.0.1 8888
Connection successful
Enter username
Vaibhav
Enter password
zeus
Authentication successful
Available commands:
1. buy
2. sell
3. order_status
4. trade_status
5. logout
```

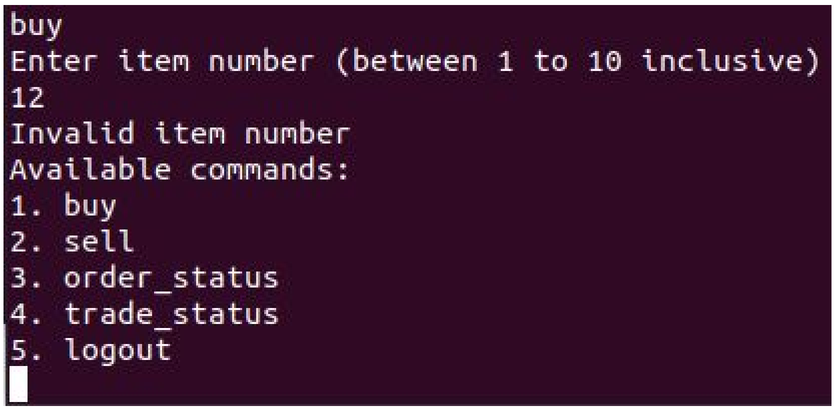
❑ As shown in the above image, there are 5 options the user can choose from. They are described in detail below.

- ➢ buy

  This option enables the user to make a buy request. The user is asked for the item number, quantity and unit price. If the user gives incorrect input for any of the above fields, the server shows an error and returns back to the main menu, else it accepts the request and shows Operation Successful before returning back to the main menu.



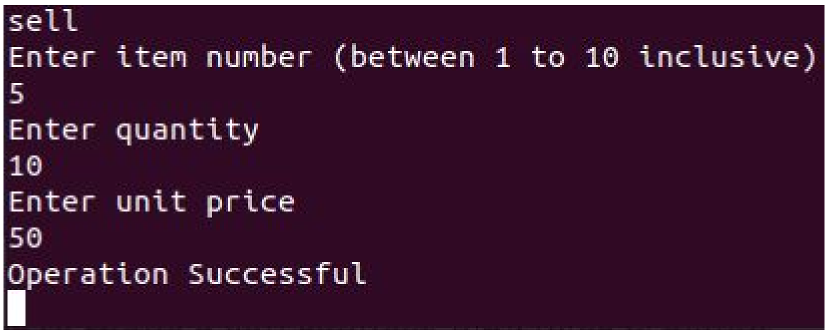
```
buy
Enter item number (between 1 to 10 inclusive)
5
Enter quantity
10
Enter unit price
50
Operation Successful
```



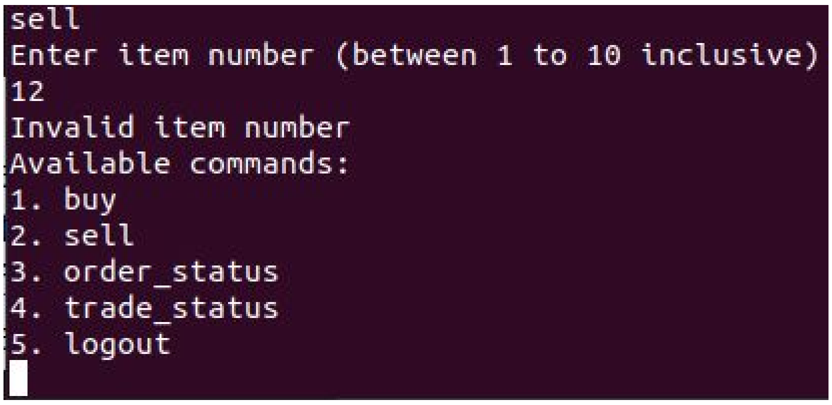
```
buy
Enter item number (between 1 to 10 inclusive)
12
Invalid item number
Available commands:
1. buy
2. sell
3. order_status
4. trade_status
5. logout
```

- sell

  This option enables the user to make a sell request. The user is asked for the item number, quantity and unit price. If the user gives incorrect input for any of the above fields, the server shows an error and returns back to the main menu, else it accepts the request and shows Operation Successful before returning back to the main menu.



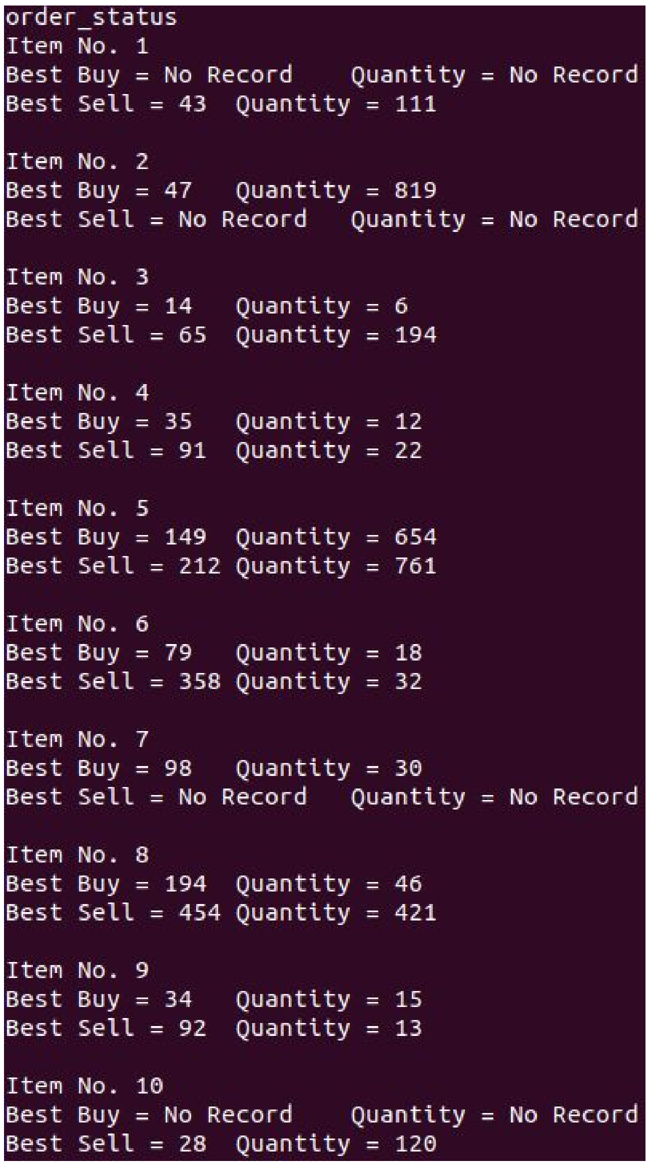
```
sell
Enter item number (between 1 to 10 inclusive)
5
Enter quantity
10
Enter unit price
50
Operation Successful
```



```
sell
Enter item number (between 1 to 10 inclusive)
12
Invalid item number
Available commands:
1. buy
2. sell
3. order_status
4. trade_status
5. logout
```

- order_status

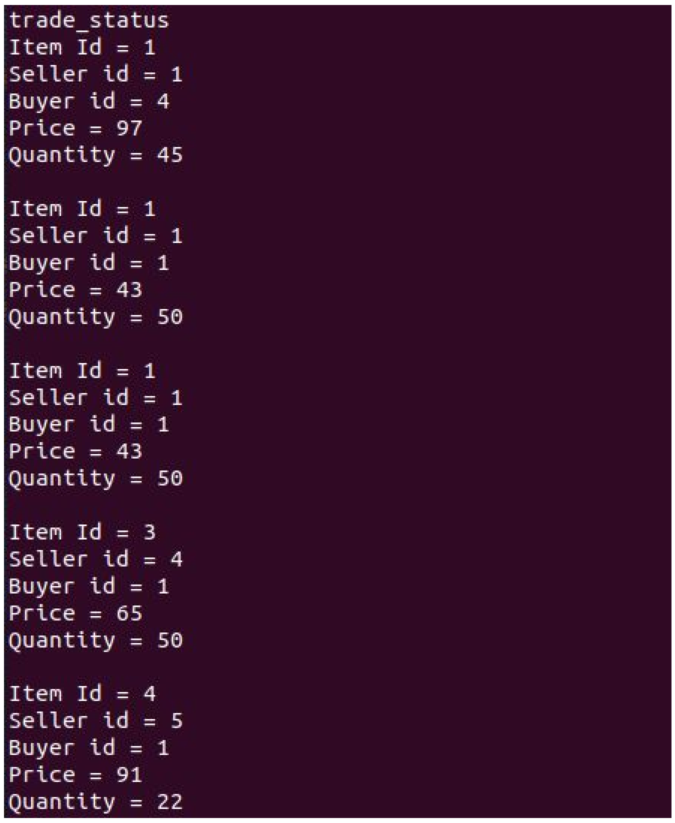
  This option shows the current best sell (least price) and best price (max price) for all items and then returns back to the main menu.



```
order_status
Item No. 1
Best Buy = No Record    Quantity = No Record
Best Sell = 43  Quantity = 111

Item No. 2
Best Buy = 47   Quantity = 819
Best Sell = No Record   Quantity = No Record

Item No. 3
Best Buy = 14   Quantity = 6
Best Sell = 65  Quantity = 194

Item No. 4
Best Buy = 35   Quantity = 12
Best Sell = 91  Quantity = 22

Item No. 5
Best Buy = 149  Quantity = 654
Best Sell = 212 Quantity = 761

Item No. 6
Best Buy = 79   Quantity = 18
Best Sell = 358 Quantity = 32

Item No. 7
Best Buy = 98   Quantity = 30
Best Sell = No Record   Quantity = No Record

Item No. 8
Best Buy = 194  Quantity = 46
Best Sell = 454 Quantity = 421

Item No. 9
Best Buy = 34   Quantity = 15
Best Sell = 92  Quantity = 13

Item No. 10
Best Buy = No Record    Quantity = No Record
Best Sell = 28  Quantity = 120
```

- trade_status

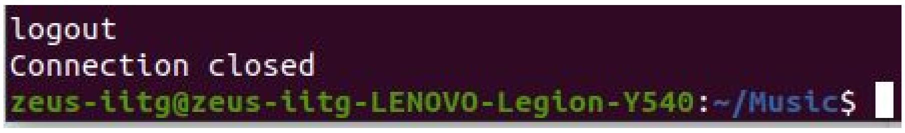
  This option displays the list of all trades which involved the current trader.



```
trade_status
Item Id = 1
Seller id = 1
Buyer id = 4
Price = 97
Quantity = 45

Item Id = 1
Seller id = 1
Buyer id = 1
Price = 43
Quantity = 50

Item Id = 1
Seller id = 1
Buyer id = 1
Price = 43
Quantity = 50

Item Id = 3
Seller id = 4
Buyer id = 1
Price = 65
Quantity = 50

Item Id = 4
Seller id = 5
Buyer id = 1
Price = 91
Quantity = 22
```

- logout

  This option provides the user to log out of the current session, thereby closing the connection between client and server.



```
logout
Connection closed
zeus-iitg@zeus-iitg-LENOVO-Legion-Y540:~/Music$
```

IMPLEMENTATION DETAILS

- Every buy/sell request is stored in a struct container having 3 attributes namely trader_id, price, quantity.
- Every trade is stored in a struct container having 5 attributes namely item_id, seller_id, buyer_id, price, quantity.
- The buy requests are stored in a max heap since every seller would like to sell his item to the buyer giving the highest price. Max heap supports insert and extractMax operations in O(log n) and getMax in O(1). Using a max heap ensures fast run time of the operations which our application supports.
- The sell requests are stored in a min heap since every buyer would like to buy items from the seller seeking the lowest price. Min heap supports insert and extractMin operations in O(log n) and getMin in O(1). Using a min heap ensures fast run time of the operations which our application supports.
- The trades are stored in an array since it facilitates storage of trades in an increasing order from earliest to newest.
- Every item has 2 different priority queues - 1 for buy requests and 1 for sell requests.
- Every trader has a different trade array for storing his trades.
- The username and passwords are stored in **“`traders_auth.txt`”**.
- The structures and their associated functions are stored in **“`server.h`”** header file.
- Multiple clients are handled by the server using **select()**. It provides the following functionalities:
  - Select command allows us to monitor multiple file descriptors, waiting until one of the file descriptors becomes active.
  - For example, if there is some data to be read on one of the sockets select will provide that information.
  - Select works like an interrupt handler, which gets activated as soon as any file descriptor sends any data.
- We have provided no option to close the server since servers generally keep running at all times. It can be terminated by using **Ctrl-C** on the terminal.
- An array named logged_in stores the session indicators for all users. The value of the indicator tells whether the corresponding user has an active session somewhere or not.