|  |  |
| --- | --- |
|  | Technical Report |
|  |  |
|  | Chai Chek Ju  C++ Stock Application  11/6/23 |

Contents

[Before the program run (Client and Server) 2](#_Toc150158066)

[Program Runtime (Screenshot and user & server detail) 3](#_Toc150158067)

[Program Algorithm Detail (Source Code and Algorithm) 22](#_Toc150158068)

[Assumption 34](#_Toc150158069)

[Improvement 34](#_Toc150158070)

[Conclusion / Personal Thought 35](#_Toc150158071)

# Before the program run (Client and Server)

To setup the entire environment can refer to the user manual.

Server script doesn’t have much thing need to input or run and just simple double click it will been start at the port 12345, then client or spammer script can be connected into the server. It used to store the loading csv, data storing, logic and important algorithm, message that reply to client, buy or sell stock matching and buy or sell stock pending.

Spammer script doesn’t have listen function so it just all the way spams the task and exit the server even when server reply thing it won’t be received

Client script is having listen and send function so it can communicate to the server in both send and received. Further action such as view available stock (name, symbol, volume and price) buy stock, sell stock, view the pending list and view the holding inventory are done by both way communication. Other than view the holding inventory, other script and code was done at the server side, don’t have client local calculation and error checking.

Hence, to increase the security of the entire program, client will just open the client.exe to do the designed action and server will not respond with any action or useful information if client do more than that.

# Program Runtime (Screenshot and user & server detail)

**Initial Startup for both program**

**Client :** Prompt to insert the user name

A screenshot of a computer

Description automatically generated

**Server :** It will read the csv file and if read successfully it will startup the socket

A screenshot of a computer

Description automatically generated

**User joins connect into the server**

**Client :** User can do from 1 to 5 as specified, and can type “quit” to exit the program

A screenshot of a computer

Description automatically generated

**Server :** Server will handle the client and at this time, since user don’t have any action or command it wont be display out anything

A screenshot of a computer

Description automatically generated

**User insert “1” to the command prompt**

**Client :** Client can see the available stock that can be buy or sell in later, and the volume and latest price

A screenshot of a computer

Description automatically generated

**Server :** Server will handle the command and reply back the client with the entire stock list also will print out the username and the action that done by user

A screenshot of a computer

Description automatically generated

**User insert “2” to the command prompt**

**Client :** Client are continue to ask to insert the stock detail that want to sell such as symbol, quantity and price

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Server :** Server will handle the command and using the matching algorithm, if not match it will put at the pending list and print out the client detail such as username, action, stock symbol, quantity and price

A screenshot of a computer

Description automatically generated

**User insert “3” to the command prompt**

**Client :** Client are continue to ask to insert the stock detail that want to buy such as symbol, quantity and price. In this case client insert symbol: AEP, quantity:100, price 74

A screenshot of a computer

Description automatically generated

**Server :** Server will handle the buy command and using the matching algorithm, if not match it will put at the pending list and print out the client detail such as username, action, stock symbol, quantity and price.

In this case, the order didn’t match any same symbol and same price in the server

A screenshot of a computer

Description automatically generated

**User insert “4” to the command prompt**

**Client :** Able to see all the pending buy list and pending buy sell list.

In this case, we place the buy and sell order from the above so it appear here since no one was match to the price.

A screenshot of a computer

Description automatically generated

**Server :** Server will handle the buy command and reply back with the pending buy list and pending buy sell list

A screenshot of a computer

Description automatically generated

**User insert “5” to the command prompt**

**Client:** Able to see all the stock that purchased and sold at here

In this case, we don’t have any matching stock so it display nothing at here but we will clearly see the holding stock for later

A screenshot of a computer

Description automatically generated

**Server :** Server will not handle or reply back any respond to the client for this command

A screenshot of a computer

Description automatically generated

**User insert “quit” to the command prompt**

**Client:** Able to quit the program without any hesitation or bugging.

A screenshot of a computer

Description automatically generated

**Server :** Server will respond to the client with bye <username> and server here will display the output

**A screenshot of a computer

Description automatically generated**

**Exception for symbol checking during buy and sell stock**

**Client:** username: Client, trying to buy stock for AAPL which is not listed in the database

A screenshot of a computer

Description automatically generated

Sell AAPL stock

A screenshot of a computer

Description automatically generated

Buy AAPL stocks

A screenshot of a computer

Description automatically generated

**Server :** Server will respond the symbol not valid and print out the user action

A screenshot of a computer

Description automatically generated

**Matching Order (Both buy and sell with same price and same quantity)**

Username: Client01

Goal: Sell to Goat with AEP, 100, $74

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Current AEP price is at $74.93

A screenshot of a computer

Description automatically generated

Server reply with : SELL ALL Stock sold mean the stock is match and fully been sold

A screenshot of a computer

Description automatically generated

Currently the volume increases 100 and price become latest price $74

**Matching Order (Buy and sell same price and higher the buy order quantity)**

Username: Client02

Goal: Sell to Client01 with ADP, 100, $214

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

It will buy the50 of the client01 stocks and another 50 will go into the pending list waiting for new buyer

**Matching Order (Buy and sell same price and higher the sell order quantity)**

Username: Client02

Goal: Buy to ABNB with Goat, 50, $116

Before

A screenshot of a computer

Description automatically generated

After

A screenshot of a computer

Description automatically generated

**Matching Order (With Prior Orderid)**

Username: Goat7

Goal: Buy to ADP, 50, $214

A screenshot of a computer

Description automatically generated

After: the stock that release by client02 will be process first, since it have thet lower orderid compare to client01

A screenshot of a computer

Description automatically generated

**Matching Order (With multiple Orderid/ with multiple different order with same price)**

Username: Client01

Goal: Buy to ABNB, 150, $214

Before:

A screenshot of a computer

Description automatically generated

After:

A screenshot of a computer

Description automatically generated

It will appear at the option “5”, inventory there

A screenshot of a computer

Description automatically generated

# Program Algorithm Detail (Source Code and Algorithm)

**Server.cpp**

A screen shot of a computer program

Description automatically generated

The program have two different user define data struct with one is stock is use for reading the csv and one is order that use for the order transaction later

A screen shot of a computer program

Description automatically generated

This is the readstock class that will been call after this to use to read the stock\_info.csv and will convert the csv data into the stock class for accessing it.

A computer screen shot of code

Description automatically generated

Stockserver is a class that will call for the readstock class and will startup the server after the stock been read.

A computer screen shot of code

Description automatically generated

This is the code that run to startup the listener from the client and it will print the “StockServer is listening for incoming connections on port” at the command prompt when it successfully run the readstock class and load it into the compiler for processing.

A screen shot of a computer code

Description automatically generated

This is the code when user press “4” to see the pending order,, it will start to clean up the stock that with quantity==0 then will loop through the vector (pendingSellOrders & pendingBuyOrders)A black background with many small lights

Description automatically generated with medium confidence

A screen shot of a computer program

Description automatically generated

This is where the match function is. This is the matching function that match the buy order to the sell order so its name is matchSellOrders with taking order into it and loop through the pendingSellOrders vector to find the same symbol and same price. In here, the vector was loop through ascending order, so the small orderid will come first and if there is match, the stock data (user option ”1”) will be update the quantity and the price.

A screen shot of a computer program

Description automatically generated

This is where the remaining logic for the function, it will check the remaining user order quantity and check the order in the sell pendinglist if it is 0 then the order is match if there is outstanding for the user order quantity, then it will been push back to the pendingbuylist when it end of theloop but if there is not remaining for the user order but sell pendinglist order still remaining then it will deduct the quantity that user bought and update the vector. Else, if there is not if statement trigger mean that the loop will continue to search next order in the pendinglist for the same order and same symbol. If nothing was found in the end or there is still remaining quantity for user order it will been push to the pending buy vector.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

matchBuyOrders is matching the buy order to the sell order.

It using same logic as matchSellOrders. It will first search for the same symbol and same price then proceed the order by the logic if don’t have match or there is still user order quantity left then it will push into the sell order pending list.

A computer screen with text

Description automatically generated

A function that use to check stock symbol exist or not, if not then it will return false

A screen shot of a computer screen

Description automatically generated

Private function is storing the respond to the user and calling the public function to process.

A screen shot of a computer program

Description automatically generatedIt use to process the message that send by the client also it pack the message and send to the client so that client can understand.

To separate the other message than sell/buy successful message here using “|” instead of “:”

A computer screen with text on it

Description automatically generated

Main function is just running the stockserver with port 12345 and start it

**Client.cpp**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Client.cpp is quiet simple script mostly with cin and separate each data with “:” and send to the server

A screen shot of a computer program

Description automatically generatedit is using while loop to run after each time complete of the cin (action) it will send to the server and wait for the respond

A screenshot of a computer program

Description automatically generated

This is where the inventory holding been proceed. If the buffer received SELL:message:orderid:symbol:quantity:price or BUY:message:orderid:symbol:quantity:price

Then it will auto push to the vector <order>holding

# Assumption

* No actual balance or money was in flow or process during the time.
* Program might be fail due to the overflow (not happen during final testing and product demonstration)
* Program might need the environment setup before running. Exe file can direct run on the window cmd
* Program was well tested and well perform in the window environment, but might not well perform in the unix system due to the library used
* There is no authentication for the user
* There is no balance check and unlimited trade stock even user came in without holding any stock still can buy and sell

# Improvement

* GUI concept
* Replace the used of the socket, can use more efficient and effective library
* Implement user authentication and authorization
* Implement the balance
* Implement the edit and delete the stock if user regret to place the order (if not match)

# Conclusion / Personal Thought

I had the most enjoyable assessment experience recently! It was a real surprise because I've always been passionate about designing and creating a stock market application, but I never got around to making one. This time, I decided to take on the challenge, and to my amazement, I managed to design and implement the entire source code in just one week.

I've had a lot of experience with Python and Java over the past year, but C++ was a language I hadn't touched in quite some time. When the assessment required me to use C++, I was taken aback. It was like rekindling an old flame, as I had to reacquaint myself with C++ and brush up on all the knowledge I had forgotten.

The most challenging part of the assessment for me was implementing the order matching functionality. It took me around four days to design and implement, but it didn't work initially. The issue stemmed from my choice to use a hash map to store the data. There was some conflict between the hash map and the specific actions I needed to perform.

Despite the initial setback, I persevered, made necessary adjustments, and ultimately, I managed to overcome the issues with the hash map. It was a fantastic learning experience, and I'm thrilled that I was able to tackle the challenge and create something I'm truly passionate about – a stock market application.