Initial Requirements

Software Systems Scalability

Software Engineering 468

University of Victoria

Spring 2020

Team: Extreme Workload

Kian Gorgichuk V00855771

Devlyn Dorfer V00846516

Goh Sato V00850584

Table of Contents

1.0 Introduction **4**

2.0 Environment Requirements **4**

3.0 Command Requirements **4**

4.0 Data Validation Requirements **10**

5.0 Auditing Requirements **11**

6.0 Web Interface Requirements **12**

7.0 Performance Requirements **12**

8.0 Security Requirements **13**

9.0 Reliability Requirements **13**

10.0 Conclusion **14**

Glossary

System	Refers to the entire Day Trading Application software system.
User	An abstraction of a person who will use the system. Each user is identified by their userid.
Balance	A user's balance is the amount of money a user has.
Internal Server	Refers to a single internal server within the system.
Portfolio	The collection of all the stocks owned by the user.
userid	A unique, alphanumeric string that is assigned to each user.
Reserve account	A temporary account made to reserve the money or stocks that will be used in a future transaction.

1.0 Introduction

The purpose of this document is to outline the technical requirements of the Extreme Workload Day Trading Application as business facing, higher level requirements have been provided. Specifically, this document outlines how business requirements translate into technical requirements.

2.0 Environment Requirements

The following are what is required by the environment in which the system will operate.

Requirement	Details
R1.1	All internal servers of the system must be deployed on Docker containers.
R1.2	All stock quotes are retrieved from the provided legacy quote server.
R1.3	The system must work on the Ubuntu VMs found in the SENG468 lab.

3.0 Command Requirements

This section outlines the requirements for each command. It does not include data validation or logging.

Requirement	Details
All Commands (R2.1)	
R2.1.1	The system must check that a command is a valid user command.
R2.1.2	The system must check that all required command parameters are valid.
R2.1.3	All executions of a command must return a confirmation of their execution or an error.

R2.1.4	The system must check that the command has a userid that is associated with an existing user (with the exception of ADD command and DUMPLOG).
ADD (R2.2)	
R2.2.1	If ADD command is executed with a non existing userid, the system must create a new user with $0 and no stocks in their portfolio.
R2.2.3	If the ADD command is executed on an existing userid, the system must add the amount value to the corresponding user's balance.
QUOTE (R2.3)	
R2.3.1	The system must retrieve the value of the stock associated with the given stock symbol from the legacy quote server.
BUY (R2.4)	
R2.4.1	The system must check that the user has greater than or equal money than the purchasing amount.
R2.4.2	The system must only allow the buying of stocks in whole units.
R2.4.3	The system must present the user with the transaction to be approved (confirm or cancel).
R2.4.4	The system must maintain a 60 second timer to commit or cancel a BUY.
R2.4.5	If the 60 second window elapses the current transaction is cancelled.
R2.4.6	The system must support multiple pending BUY commands.
COMMIT_BUY (R2.5)	
R2.5.1	The system processes the most recent pending BUY command.
R2.5.2	The system must recheck that the user's balance is greater than or equal to the BUY amount.

R2.5.3	The system must purchase the specified stock to the nearest whole number afforded by the BUY amount.
R2.5.4	The system must remove the amount required for the purchase from the user account.
R2.5.5	The system must add the purchased stock amount to the user's portfolio.
R2.5.6	If there is no BUY command to process, the command fails.
CANCEL_BUY (R2.6)	
R2.6.1	The system must process the most recent pending BUY command.
R2.6.2	The system must remove the pending transaction.
R2.6.3	If there is no BUY command to process, the command fails.
SELL (R2.7)	
R2.7.1	The system must check that the user has a greater than or equal dollar value of stock than the amount they're selling. Otherwise, the command fails.
R2.7.2	The system must only allow the sale of full stocks.
R2.7.3	The system must present the user with the transaction to be approved (confirmed or cancelled).
R2.7.4	The system must maintain a 60 second timer to commit or cancel the SELL command.
R2.7.5	If the 60 second window elapses the current transaction cancelled.
R2.7.6	The system must support multiple pending SELL commands.
COMMIT_SELL (R2.8)	
R2.8.1	The system must process the most recent pending SELL command.

R2.8.2	The system must check that the user has the amount of stock to be sold.
R2.8.3	The number of stocks to be sold is the greatest number of whole unit stocks that is below the dollar amount to be sold
R2.8.4	The system must remove the amount of stock from the user's balance.
R2.8.5	The system must add the dollar amount from the sale to the user's portfolio.
R2.8.6	If there is no SELL command to process, the command fails.
CANCEL_SELL (R2.9)	
R2.9.1	The system must process the most recent pending SELL command.
R2.9.2	The system must remove the pending transaction confirmation.
R2.9.3	If there is no SELL command to process, the command fails.
SET_BUY_AMOUNT (R2.10)	
R2.10.1	The system must check that the user's balance is greater than or equal to the buy amount.
R2.10.2	If there is an existing reserve account for the specified stock, the system must return the balance in the reserve account to the regular account and remove any existing triggers for the stock.
R2.10.3	The system must create a reserve account that holds the amount specified by the amount value.
CANCEL_SET_BUY (R2.11)	
R2.11.1	If a SET_BUY_AMOUNT has not been issued for the specified stock, the command fails.
R2.11.2	The system must remove any BUY triggers for the specified stock (set by SET_BUY_TRIGGER).

R2.11.3	The system must return the balance in the reserve account to the user's regular account.
SET_BUY_TRIGGER (R2.12)	
R2.12.1	If a SET_BUY_AMOUNT has not been issued for the specified stock, the command fails.
R2.12.2	If the trigger price is greater than the reserved amount, the command fails.
R2.12.3	The system must create a trigger at the specified price and for the specified stock.
R2.12.4	If there is an existing trigger in place for the specified stock, the system must clear the existing reserve and create a new reserve.
R2.12.5	When the specified stock's price is lower than or equal to the trigger price, the system will buy the max amount of the specified stock using the amount in the reserve account. The system must add the purchased stock to the user's account.
R2.12.6	After the transaction occurs, the system must return the remaining balance in the reserve account to the user's regular account.
R2.12.7	After the transaction occurs, the system must remove the trigger for the specified stock.
SET_SELL_AMOUNT (R2.13)	
R2.13.1	The system must check that the user's portfolio has the specified amount of stock to sell.
R2.13.2	The system must save the sell amount value to be used when a trigger is set.
R2.13.3	If there is an existing reserve for the specified stock, the system must return the stock to the user's regular portfolio and remove any existing triggers for the stock.

SET_SELL_TRIGGER (R2.14)	
R2.14.1	If a SET_SELL_AMOUNT has not been issued for the specified stock, the command fails.
R2.14.2	The system must reserve the maximum number of stocks that can be sold at the trigger price given the amount set by the SET_SELL_AMOUNT.
R2.14.3	The system must create a trigger at the specified price and for the specified stock.
R2.14.4	If there is an existing trigger in place for the specified stock, the system must return the reserve stock and create a new reserve.
R2.14.5	When the specified stock's price is greater than or equal to the trigger price , the system will sell all the stocks in the reserve.
R2.14.6	After the transaction occurs, the system must remove the trigger for the specified stock.
CANCEL_SET_SELL (R2.15)	
R2.15.1	If a SET_SELL_AMOUNT has not been issued for the specified stock, the command fails.
R2.15.2	The system must remove the SELL trigger for the specified stock (set by SET_SELL_TRIGGER).
R2.15.3	The system must return the reserved stocks to the user's portfolio if applicable.
DUMP_LOG (R2.16)	
R2.16.1	The system must write a complete log file of all transactions to the file specified in the command.
R2.16.2	If the specified file does not exist, the system must create a file with the name of the specified file and write the log to it.

R2.16.3	If userid is provided, the system will only write logs to the file for commands made by the user corresponding to the userid.
DISPLAY_SUMMARY (R2.17)	
R2.17.1	The information will be delivered as JSON.
R2.17.2	The system will return a summary of the given user's transaction history, account, and triggers.

4.0 Data Validation Requirements

This section includes the constraints on the data in the system.

Requirement	Details
R3.1	Every userid in the system must use alphanumeric characters.
R3.2	The value of all “amount” parameters are in dollars. This includes the sell transactions.
R3.3	The amount of money specified in commands must be greater than zero.
R3.4	All stock symbols in the system are between one and three capitalized alphabetic characters.
R3.5	Money specified internally in the system must be represented in cents.
R3.6	Any quote received from the legacy quote server is valid for only 60 seconds.
R3.7	A user’s balance must not be negative.
R3.8	The amount of a stock in a user’s portfolio must not be negative.
R3.9	The amount specified by the user will be in dollars and cents.

5.0 Auditing Requirements

This section specifies the events that must be logged and what is to be included in each log.

Requirement	Details
R4.1	Every response from the legacy quote server must be logged.
R4.2	Every command requested by a user must be logged.
R4.3	Every outgoing request from an internal server must be logged.
R4.4	Every incoming request from an internal server must be logged.
R4.5	Every 10 seconds the status (alive/dead) of all internal servers must be logged.
R4.6	Every time a user's balance is modified (add/remove), the system must log the change.
R4.7	Every time a user enters an invalid command or parameter the system must log the error.
R4.8	The system logs must be queryable by userid.
R4.9	All system logs must use the Unix timestamp format for timestamps in milliseconds.
R4.10	The log files outputted by the system must adhere to the XML format specified by the project requirements.
R4.11	The system must log the cryptographic key returned by the legacy quote server for each request to the legacy quote server.

6.0 Web Interface Requirements

This section specifies the requirements for the graphical web interface.

Requirement	Details
R5.1	A web interface for users to interact with the system must be provided.
R5.2	The web interface must display all the errors that arise for invalid user commands.
R5.3	The web interface must support all the commands in the Commands Requirements section.

7.0 Performance Requirements

Here are the performance requirements around the system.

Requirement	Details
R6.1	The system must be able to handle 1250 users in a simulated workload.
R6.2	The system must cache the legacy quote server’s responses to avoid unnecessary requests on the legacy quote server.

8.0 Security Requirements

There are the security requirements for the project.

Requirement	Details
R7.1	Request from the web client to the web server must be encrypted using SSL.
R7.2	The DUMPLOG request to obtain logs for all users requires administrator privileges.

9.0 Reliability Requirements

Here are the reliability requirements.

Requirement	Details
R8.1	If an individual internal server of the system becomes overloaded within the expected user workload, that requesting server must not drop any requests.
R8.2	Data stored in the system must be stored redundantly. No data is to be stored in a single location.
R8.3	If an internal server goes down, the system must be able to redirect the request to a backup internal server.
R8.4.1	If a request is made from one internal server to another server and the second server is unresponsive the first server must continuously retry the request with a timeout period.
R8.4.2	If the second server continues to be unresponsive for a given period of time, the first server must propagate the error to the user.

10.0 Conclusion

Upon completion of the above requirements, the system will meet the requirements of the Day Trading Application project. Any further technical requirements that manifest as the system is implemented will be added to this document.