SEng 462 - Distributed Systems and the Internet

Course Project:

The End-To-End Development and Analysis of a Production Day Trading System

1.0 Important Dates

See http:\\www.ece.uvic.ca/~seng462/ProjectWebSite for the list of due dates for the various project milestones and deliverables.

Note that the on-line log book submissions are due weekly.

2.0 Project Overview

Your consulting group has been contracted by DayTrading Inc. to develop a prototype end-to-end solution to support their day trading clients and services. DayTrading Inc.'s competitive advantage comes from providing its clients with the fastest end-to-end processing times in the industry and it is seeking to improve its competitive advantage in this area through re-implementing its complete internet-based distributed day trading system. DayTrading is soliciting proof-of-concept solutions from multiple consulting groups and will award the final contract to the system that has the minimal transaction processing times, supports all of the required features, and does so at a minimal cost, in terms of up front hardware costs, the development costs, and the on-going maintenance and upgrade costs. DayTrading's client base is continually growing so they are also very interesting in capacity planning issues and in a solution that has the capability to grow incrementally without adversely impacting transaction performance, and where such growth can be supported at minimal cost increments. DayTrading prides itself in providing its clients with the industry leading security; therefore, all bidders must fully vet and document their system with regards to security issues. System down-time is also not acceptable to DayTrading Inc. and system availability is a key design concern. The winning bid will also have its prototype development costs rolled into the overall system development contract and budgets, so full tracking of the work effort required to design, implement, test, analyze and document the prototype is required.

Your group's task is to build a prototype system for DayTrading Inc.'s evaluation and develop the complete documentation for your solution. This documentation must formally cover all issues of concern to DayTrading Inc., who are well known in the industry for having a strong technical review process and for insisting that all claims be formally verified and supported through formal analysis and testing. For example, capacity planning estimates must be supported through solid experimental testing and analysis of the prototype system. It is insufficient merely to claim a certain level of capacity is achievable through extrapolating incomplete testing scenarios. Included in your system must be a prototype mobile client application which both provides a user interface to your backend daytrading system as well as a local store of the current state of the given user's account information.

Outside of the suggested list of software components that DayTrading currently supports in-house, there are no restrictions on the system design (including language choices, database server, we server, system architecture, etc.). Each design team is free to implement the best solution in the manner that they deem best. DayTrading does though insist that all design decisions be fully and formally documented including their rationale as to why the given choice was the best available.

It is important to realize that the scope of this project has been intentionally left quite open to enable each group to explore the design space of the problem and to develop their best solution to the posed problem. Hence, issues such as error conditions that need to be specified, account management processes, etc. have been left purposely undefined within the scope of this document. As such, the project closely mimics the level of project specification and detail available in the real-world in the development of commercial projects for actual clients. It is expected that as the project progresses questions will arise about Day Trading's implementation requirements. Such questions can be, and are expected to be, brought up during class or via email as they arise.

Distributed systems and the internet is a complex topic covering a wide variety of issues and concepts. It is hoped that the course project will form a common focal point within which class discussions about the implementation, development and engineering of such systems can take place. Gaining an understanding of the issues inherent in the development of these types of system requires hands-on experience which the course project is designed to facilitate.

3.0 Group Composition

Each group may consist of a maximum of 4 members. It is advised that you choose the group such that there is a person who has some experience in each of the core project areas, namely: database design, web site development, middle ware, system security. Likely, no one will be experienced within the areas of performance testing and capacity planning of distributed systems. There are core issues within the project as they drive the iterative process of identifying and rectifying performance bottlenecks. Hence, each team member will want to ensure that they understand and gain skills in these areas. A significant portion of the project will be focused on the testing activities and structuring of the tests to elicit the required performance and capacity planning information.

Groups will also be responsible for their own project planning and setting their own internal milestones. The group should assign/elect someone to take the lead on this effort, which includes ensuring that the work load is shared equitably among group members.

The project accounts for 40% of the course mark, so the required work effort is commensurate with this percentage. If your group leaves the project to the last minute then expect that your marks will reflect this choice. Project grades will reflect the work effort, thoroughness, and commitment of the group to developing a well engineered solution to the posed problem.

The work load require to excel at this project is substantial. Work on the project will be required throughout the term. It is strongly recommended that you being the project early through the rapid prototyping of a basic end-to-end solution. The system can then be expanded through testing and refining as the term progresses. You should expect that significant portions of your solution will likely need to be reworked in order to achieve high transaction throughputs.

4.0 Day Trading Activity

The basic structure of the day trading system is to support a large number of remote clients through a centralized transaction processing system. Each client logs in through a web browser and then performs a number of stock trading and account management activities such as:

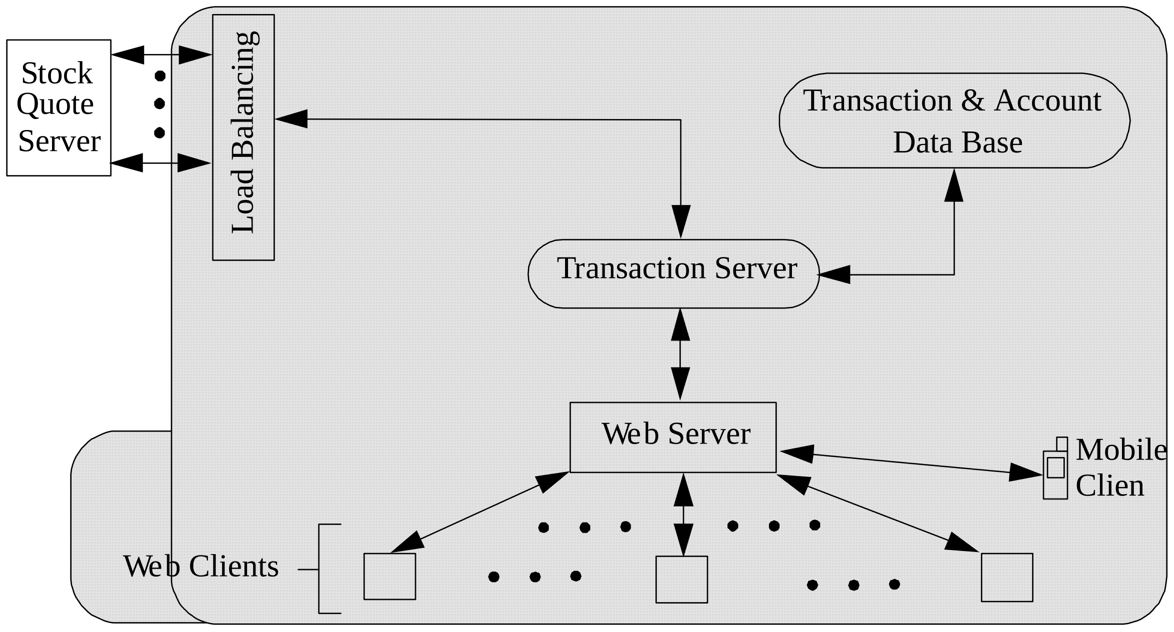
- View their account
- Add money to their account
- Get a stock Quote
- Buy a number of shares in a stock
- Sell a number of shares in a stock they own
- Set an automated sell point for a stock
- Set a automated but point for a stock
- Review their complete list of transactions
- Cancel a specified transaction prior to its being committed
- Commit a transaction

In addition to the client activities, DayTrading requires full auditing capabilities; hence, complete transaction logs must be able to be produced on demand that detail of all client activities in the system (including timestamps of all transactions), a record of each individual transaction, each transaction's processing time information, and all account state changes within the system. This log must be dumped from the system as an ASCII text file when it receives the "DUMPLOG" command.

For a complete list of Client and System Commands see Appendix A.

5.0 Base Architecture

The base architecture for the day trading system is shown in the block diagram below.



There is no requirement that the architecture your group chooses to implement follows this basic structure. Other architectures are possible with different trade-offs regarding performance, fault tolerance, etc. This architecture is meant merely as a baseline and it does not address a number of the design issues that you will need to consider in developing you solution.

6.0 Overall Architecture Goals:

The overall goals of the developed system and architecture are:

- Minimum transaction processing times.
- Full support for required features.
- Reliability and maintainability of the system.
- High availability and fault recoverability (i.e. proper use of fault tolerance)
- Minimal costs (development, hardware, maintenance, etc.)
- Clean design that is easily understandable and maintainable.
- Appropriate security
- A clean web-client interface.

6.1 Documentation

DayTrading Inc. requires a complete set of documentation to be delivered at the end of the project in addition to the working prototype.

- Full documentation of architecture including complete analysis of design choices.

- Full documentation of test plans, testing results, and test analyses.
- Full documentation of work effort required to build prototype, including weekly individual log book entries signed by each member of the design team.
- Complete capacity planning and transaction time documentation, including experimental results and extrapolations.
- Full security analysis.
- Full project planning and execution documentation for prototype development effort.
- Full analysis of system capacity and capacity planning documentation.

All claims within the project documentation must be supported through appropriate experimental testing and analysis

All documents must be clear, concise, and correct with respect to english usage and grammar. Documentation that is not comprehensible, overly verbose, rambling, etc. will be viewed by DayTrading Inc. as indicative of the design team's general level of care and attention, and therefore will reflect negatively on the team's overall evaluation.

6.2 Group Web Site

Each group must maintain a group web site during the course of the project. This web site must consist of a public section, used for providing on-going performance evaluation information, and a private section (available only to the group members and the course tutotial instructor) to maintain copies of the project documents and access to the source code.

Verified transaction performance is obtainable by submitting a transaction log file to the verification server (as detailed on the project web site). The required transaction log file format is detailed at http://www.ece.uvic.ca/~seng462/ProjectWebSite/ExampleLog.html.

7.0 Project Planning & Work Effort

DayTrading Inc. will engage in no project planning or work effort enforcement activities. It is left to the individual design teams to structure and enact their own project management plans, including the assignment of individual tasks to group members.

Complete project plans must be submitted to DayTrading Inc. and are to include group development milestone dates as well as an assessment of the risk factors associated with the project.

Each team will track their project progress on their group's project web site. Of particular interest to DayTrading Inc. is the tracking of improvements in transaction processing speeds. This project tracking portion of the web site must consist of a public portion that includes a tracking to the milestones at a daily level, and experimentally confirmed current transaction processing speeds. Additionally, the private portion of the web site, may be used to track documentation and development efforts in more detail, including detailed project plans and task assignments.

DayTrading Inc. will only track the overall performance of each team in meeting the project milestones. All group members will be viewed as equally culpable for missed milestones project.

DayTrading does require each team member to submit individual electronic weekly log book entries. This entry must specify all the work the team member contributed to the project during the given week, as well as the time devoted to each of the tasks listed. The log book must be electronically signed off by all other team members prior to its submission to DayTrading Inc. Log book entries can be made at https://secure.engr.uvic.ca/~seng462/logbook

8.0 Supported Component Software

DayTrading Inc. supports the following software components:

- The most current list of supported software can be found at http://www.engr.uvic.ca/~seng462/ProjectWebSite/index.shtml

The software listed on this web site are the only components supported by the company and any implementation that utilizes software different from this list must be justified. Additionally, unsupported software components add significantly to DayTrading Inc. maintenance costs as new personal must be trained in the systems and more resources must be added to facilitate on-going maintenance of additional technologies. Additions to the above supported software lists may be possible but must be negotiated with DayTrading Inc. and be formally shown to provide superior value and low additional maintenance costs. Implementations that remain within the above list will be viewed more favorable by DayTrading Inc. in the final analysis, unless there are substantial gains provided by alternate components that justify their total cost.

9.0 Legacy Software: Stock Quote Server

DayTrading Inc. accesses stock quote information through a contract with ACME StockSystems Inc. via legacy server code. This interface must continue to be supported through the new day trading system. It is suggested that load balancing be implemented in accessing the stock quote server to minimize the transaction request times. For cost and contractual reasons DayTrading Inc. is unwilling and unable to change this server to either a faster server or away from the current implementation; hence, this legacy server must be used as is within the prototype implementations.

Each stock quote provided by ACME StockSystems Inc. incurs a $5^{¢}$ charge which is passed directly on to DayTrading Inc.'s customers. Obviously, it is important to DayTrading Inc. and their customers that any IT trading solution minimizes the charges incurred from ACME Stock Systems Inc.

Given a stock symbol and user id this server returns the stock's current value, the user ID, and a cryptographic key. This key is required for auditing purposes and must be included in the client's transaction history as an additionally field associated with the QUERY command recorded in system transaction database.

When transaction histories are dumped from the database this key must be included with each query transaction written.

As the stock server is a legacy system, it provides little in the way of error or bound checking. A JAVA based example of a basic client server program is available on the course project web site. The design teams are free to re-implement the client if they so choose.

Inputs and outputs of the Stock Quote Server are ASCII strings of the following format:

Server Command Format: Stock Symbol, USER NAME

Server Quote Return Format: “Quote, Stock Symbol, USER NAME, CryptoKey”

10.0 Project Marks

Percentage	Deliverable
5%	Transaction and cost performance relative to other the groups
5%	Overall Architecture and Documentation (inclusive of initial documents)
5%	Security Design and Documentation
5%	Test plan and Test Documentation
5%	Capacity Planning Measurements, Analysis and Documentation
5%	Fault Tolerance Analysis and Documentation
5%	Performance Analysis, Testing and Documentation
5%	Project Presentation
-1%	Each missing or late Log book entry (to a max. of -5%)
-1%	Each missed milestone Date (to a max. of -5%)

- All documentation (including the web site and presentation text) must be clearly and concisely written. Marks will be deducted based on the quality and professionalism of the submitted material. In particular, areas of text that are patently unclear will not be marked.

Appendix A:

System Commands and Required Formats

All system commands are formed as ASCII strings with the following formats and are of the format "COMMAND, USER NAME, Parameter 1, Parameter 2,..., Parameter N"

Stocks are bought and sold in terms of the nearest whole number of shares that can be bought for the given amount of money specified in the BUY or SELL commands.

DUMPLOG with a user name provides a dump of the given users complete transaction log (which must include the cryptokey supplied by the quote service)

DUMPLOG without a user name provides a complete dump of all the system transactions for all of the users.

COMMIT and CANCEL commands are used to commit or cancel the immediately proceeding buy or sell command of the specified user. BUY or SELL commands must be committed for the account balance to be updated. Day Trading Inc. allows the user a 1 minute window in which to commit their buy or sell transaction after the requested buy or sell transaction is presented for their confirmation. If the user does not commit the transaction in this time window then the transaction at the current stock price is voided and the user is informed of the new stock price for the transaction and re-prompted to either confirm or cancel the transaction. Day Trading Inc. guarantees a 15 second commit time once the user has issued the commit command for a buy or sell transaction.

Any command that changes the system state must be followed by either a COMMIT or a CANCEL.

All BUY commands inherently generate a QUOTE command the results of which are returned to the user such that they can confirm or cancel the sell command at the stock's current price.

All SELL commands inherently general a QUOTE command the results of which are returned to the user such that they can confirm or cancel the sell command at the stock's current price.

The SET_BUY_AMOUNT and SET_SELL_AMOUNT commands cause the specified amount of money or stock to be held in reserve to execute the triggered sell or buy command (i.e. one cannot go into a negative account balance through the execution of a buy or sell trigger event)

All commands must return confirmation to the user of their execution or an error message indicating the cause of their failure to complete.

All changes to system state must be recording in a transaction log and associated with the unique USER Command that initiated them.

Money within the system is in terms of dollars and cents.

The complete set of user commands and their syntax can be found at: http://www.ece.uvic.ca/~seng462/ProjectWebSite/Commands.html