Fall 2015 CSE 4233/6233 – Software Architecture & Design Paradigms

Professional Practice Assignment #2 - Due: 11:59pm Thursday, December 10, 2015

Overview: Iterative implementation of Trade Net requirements to match concurrent architecture development while using version control (GIT)

Assignment: Implement features from Trade Net's software backlog

- Create a GitHub repository for your team. Each team member must have their own account and be listed as a contributor with the appropriate permissions to edit (push/pull) code and project wiki.
  - Complete the GIT tutorial "Getting Started" and "Collaboration" sections here: https://goo.gl/b12cbZ
- Maintain all code and documentation using version control (GIT)
- Review Trade Net Requirements Overview and Scenario Description - PDF
- Setup a Tradier Developer account (https://developer.tradier.com) - one needed per team - for market data
- Develop architecture views for the system (not graded or submitted for this assignment)
- Implement Model-View-Controller (MVC) design pattern
- Implement and document use of a design pattern - see https://sourcemaking.com/design_patterns other than MVC
- Implement Trade Net's client-server market application - Requirements below
  - Establish account management database
  - Implement buy/sell capability for stock transactions
  - Create Trade Management server to access market data

Requirements:

For a more detailed list of requirements see TNet-Requirements_Scenario.pdf

The next software release will implement some of the overall Trade Net functionality. The customer should be able to login to a graphical user interface to execute trades and view their current portfolio. Existing command line bank management application will be used to manipulate customer funds (add, withdraw, transfer). All trade transactions completed using a newly integrated brokerage account type. Funds can be transferred using existing checking/savings accounts.

Extend existing or Create Account Management (AM) Server database to store at minimum:

- Customer information (ID, first name, password, account balance, current profit and loss)
- Transactions (stock, #shares bought / sold [positive / negative], date and time)
- Portfolio (CustomerID, stocks, shares, purchase price)

Create Trade Execution server to get stock information from data provider - Tradier

- Tradiers Market connection for current equity pricing (last price) for any listed equity
- AM Server connection to store buy/sell transactions and update customer account info as needed
- Logic to execute trades

Create Single Client Graphical Interface

- List stock information - input field for a ticker symbol and output area
  - Output Fields: Symbol, Description, Exchange, Closing Price, Daily Net Change & Percentage, Volume, Average Volume, 52-Week High & Low
- Execute buy / sell transactions for a customer ID
  - Fields: CustomerID, Stock ticker, Buy / Sell button, number of shares, account balance
- View Portfolio
  - List stocks held, current price, # of shares, and value (price * shares)
- List Transaction history

Logic in either the client or server to reject trades

- Based on account balance - are funds available to complete the transaction
- For sell transactions, do they own the shares to sell
- Assume properly formatted input always received - no input validation required

Fall 2015 CSE 4233/6233 – Software Architecture & Design Paradigms

Professional Practice Assignment #2 - Due: 11:59pm Thursday, December 10, 2015

Guidance

* Getting Quote Information: https://developer.tradier.com/documentation/markets/get-quotes
* Java Example: https://developer.tradier.com/documentation/examples/java - others available
* Apache HTTP Library Download: http://hc.apache.org/downloads.cgi
* API Endpoint Connection - Request/Response: https://api.tradier.com/v1/
* Will need your authorization token generated by Tradier Dev
* Design Pattern Examples - https://sourcemaking.com/design_patterns

Sample Execution Scenario:

1. Use command line tool to assign brokerage account to customer and add $400,000
2. Login Customer to GUI
3. Customer Brokerage Account Balance = $400,000
4. Get info for FaceBook (FB)
5. Purchase 1000 shares of Facebook (FB)
6. Get info for Apple (AAPL)
7. Purchase 1000 shares of Apple (AAPL)
8. View Portfolio
9. Sell 500 Shares of FB
10. View Portfolio
11. Sell 100 shares of Twitter (TWTR) - Rejected because Customer #1 doesn't own shares in TWTR
12. Buy 1000 shares of Google (GOOG) - Rejected because of insufficient cash - can only use cash available to purchase shares
13. List Transaction History

Constraints

* All code, diagrams, and description should be uploaded and managed using GitHub
* Each team member must contribute some code to the repository
* Code comments should be constructive
* Provide any necessary compilation (required libraries) and execution instructions
* Upload SQLite DB to repository data storage

Grading: Due Thursday, December 10th by Midnight - Early submission OK - Graded when received

Code:

* Use of existing integrated command line tool for fund management (add funds to brokerage account / transfer from checking or savings)
* Execution of code implementing graphical user interface to enable stock transactions - similar to scenario above

Document:

* Graphical representation to match implementation of MVC - no description needed
* Graphical representation and description to match implementation of a design pattern (other than MVC) - can use UML class diagram - https://sourcemaking.com/uml/modeling-it-systems/structural-view/class-diagram
  * Provide a description of the design pattern and how it works for your system

Activity:

* Consistent use of GIT and GitHub for development
* All team members should make at least one code contribution to GitHub (not all members required to do equal coding)
* Provide evidence of progression of development activity - Entire system not dumped to GitHub upon project completion