**Stock Market Analysis Software Abstract & System Software Design**

**Interim software name: PyStock**

**Abstract:**

Long term: Create a software analysis tool to analyze stock market data that is asynchronously pulled using promises via various sources (scrapy, beautifulsoup) that is constantly evolving, using machine learning. The purpose of the program is to be a lightweight console program that displays raw data in a Comma-Separated Value file or into either CassandraDB, MongoDB, PostgresSQL. With the raw data inserted, use Tableau or Ploty to display data within the console, or eventually through a GUI via Tkinter. This stock market analysis will use a basic machine-learning multi-agent algorithm that will forecast the appreciation of the stock, profitability, performance, and account for any business news regarding the specific stock with account for any governmental accountability, publicity decisions, and internal agent news. The software will not only create recommendations using various standard error statistical equations such like SEp and SEx etc. The goal of the software is produce a standard **70% confidence interval,** and using a neural network to **create self-improving progress and self-increasing CI**.

Short term: Create a software analysis tool to analyze stock market data. The point of the software is monitor closely the price of the software, while taking in the account of the history of the price and performance to create a future expectation of the company. Using <https://www.tradestation.com/> or <https://www.oanda.com/> API to create RESTful HTTP requests to automatically sell or buy when a certain price is reached, depending upon configuration of user. Create the base code to be able to implement algorithm to alert and/or automatically purchase plane tickets.

**Short term software & modules required and/or comparison:**

In order to properly develop and engineer a full stack software, consider the major components of such software that require the needs to automate tasks and goals. Since the short term goal is the base code for the long term goal, we must consider the longevity of such plugins and modules. In short, the three things I am considering in choosing with module is required to best optimize the program is: ***length of support & updates, lengthy & in depth documentation, community support.*** If all three requirements are a sufficient level, regardless of how arbitrary sufficient level is defined, it will be chosen as best module for its state currently. The software should and will be designed with in mind the need to swap an integral component at a moments notice. How I define sufficient level is searching for quick start guides, common issues with similar patterns, contemporary and relevance to end users utilizing said modules.

**Requirements for completed Short Term:**

* Functioning program, that is able to produce accurate data and numerical numbers
* Configurable & Robust
* Entire full stack is packaged so software is independent.
* Able to successfully make RESTful HTTP requests with fail safes
* Able to buy/sell stocks with configuration of automatic mode or alert mode
* Able to successfully be implemented as base code for alert/auto purchase of plane tickets

Programming Language:

* Python 3.5.2
* Most up to date, documented, and stable version released with lots of developed modules including required web scrapers and crawlers, as well as integration with MongoDB, CassandraDB, and SQLite. Hands down, executive decision.

Web scraping: **Scrapy**, BeautifulSoup, lxml and Requests

* Scrapy: <https://scrapy.org/>
* Stackoverflow result: <https://www.google.com/search?q=stack+overflow+scrapy&rlz=1C1GGRV_enUS762US762&oq=stack+overflow+scrapy&aqs=chrome..69i57j0j69i64.3595j1j4&sourceid=chrome&ie=UTF-8> (yielding 78,700 results)
* Forum result: <https://www.google.com/search?safe=active&rlz=1C1GGRV_enUS762US762&q=scrapy+forum&oq=scrapy+forum&gs_l=psy-ab.3..0j0i20k1j0i22i30k1.30113.32681.0.32919.14.13.1.0.0.0.209.1573.6j5j2.13.0..2..0...1.1.64.psy-ab..0.14.1586...35i39k1j0i67k1j0i131i20k1.Xt64HMomKdA> (yielding 319,000 results)
* Documentation: <https://doc.scrapy.org/en/latest/news.html> (latest release 5-18-2017), documentation is thorugh
* BeautifulSoup https://www.crummy.com/software/BeautifulSoup/ (Note: it is Pythonic! +1)
* Stackoverflow result: <https://www.google.com/search?q=stack+overflow+beautifulsoup&rlz=1C1GGRV_enUS762US762&oq=stack+overflow+beautifulsoup&aqs=chrome..69i57j0j69i64.3412j1j4&sourceid=chrome&ie=UTF-8> (yielding 94,600 results)
* Forum result: <https://www.google.com/search?safe=active&rlz=1C1GGRV_enUS762US762&q=forum+beautifulsoup&oq=forum+beautifulsoup&gs_l=psy-ab.3..0i8i30k1.29064.29595.0.29714.6.6.0.0.0.0.130.502.3j2.5.0....0...1.1.64.psy-ab..4.2.243...0i8i7i30k1.d-h_M1nmejg> (yielding 67,300)
* <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
* [http://bazaar.launchpad.net/~leonardr/beautifulsoup/bs4/files 2-22-2017](http://bazaar.launchpad.net/~leonardr/beautifulsoup/bs4/files%202-22-2017)

After analyzing the results, it appears Scrapy is not as popular regarding stackoverflow as BeautifulSoup, but Scrapy rocks forum results. Not only does this tell me that the documentation is thorough, therefore amount of questions regarding use is less needed. Scrapy is compatible with PostgreSQL and could also be used a data mining tool, where therefore allows further expanding and provides better infrastructure for future development. It will be chosen as the web crawler/scrapper and data mining implementation.

***Web scraper: Scrapy version 1.4.0***

Data science/analysis: Numpy, Scipy, Matploblib, Pandas

* This one is automatically NumPy, due to familiarity and longevity of the module and its constant development.

***Data science/analysis: NumPy version 1.31.1***

Machine Learning: TensorFlow, scikit-learn

* TensorFlow because Google

***Machine Learning: TensorFlow version r1.3***

Data Mining: Pattern, **Scrapy**

* Scrapy because it will already be used.

Databases: MongoDB, CassandraDB, SQLite, PostgreSQL

* The need to research into each database is meaningless in this case, albeit could provide useful information. But as

PostgreSQL

* Requires <https://github.com/psycopg/psycopg2>

***Databases: PostgreSQL 9.6.5, SQLite 3.0***

Version Control

* Github, initially as a private repository until short term is completed, then convert to public repository as open source.

Operating System: ***CentOS 7***

SCRUM: Trello

https://trello.com/b/9gc3D017/software-development

**Deadlines & Timeline:**

Note: Since the short term abstract of stock market analysis tool is in essence, the base code of the long term, using basic software engineering principles as listed here <https://www.amazon.com/Software-Engineering-Hans-van-Vliet/dp/0470031468> we will build off the base code. Start from bottom-up method of software engineering and design <http://www.cs.cornell.edu/courses/cs2110/2012sp/lectures/07-SoftwareDesign_6up.pdf>

*Short term abstract completion of financial analysis research scrum stories overview:*

1. Be able to read stock graphs
   1. X = time
   2. Y = price
   3. Trendlines/Channels: Finding patterns. Highest shows high point of dips a good time to sell, blue line shows lowest, a good time to buy.
   4. Candlestick graphs, red shows closed less than, green shows growth. Long sticks, very volatile. <https://www.investing.com/equities/google-inc-candlestick>
2. Fundamental analysis
   1. P/E ratio: Price-earnings ratio. <https://en.wikipedia.org/wiki/Price%E2%80%93earnings_ratio> http://www.investopedia.com/university/peratio/ P/E ratio measures market price of a company’s stock relative to corporate earnings. Calculated stock’s current share price divided by its earning per share
   2. Earning per share = portion of company’s profit allocated to each outstanding share of common stock. <http://www.investopedia.com/terms/e/eps.asp> It is calculated as EPS = (Net income – dividends on preferred stock) / average outstanding shares. Highest EPS in S&P 500: PCLN, AZO, RE. Lowest EPS in S&P 500: PRGO, HES, FE
   3. Outstanding shares: refers to all company’s stock held by shareholders.
   4. Cash flow per share: a ratio, CFPS = (Operating Cash Flow – Preferred Dividends) / Common Shares Outstanding
   5. Important to note that P/E ratio can be overpriced! <http://www.investopedia.com/ask/answers/05/lowperatiostocksbetterinvestments.asp>

*Short term abstract completion of technical scrum stories overview:*

1. Operating System Spin up (node-01.local)
   1. Virtual Machine, CentOS 7, minimal
      1. 16GB of RAM
      2. 500GB of Hard Drive Space, SSD
      3. Console only, no GUI
   2. Automated snapshot, 12hr interval, 4 week max
   3. modules and software into /(software\_name)/:
      1. Scrapy 1.4.0
      2. NumPy 1.31.1
      3. TensorFlow r1.3
      4. Python 3.5.2
      5. PostgreSQL 9.6.5
      6. SQLite 3.0
      7. Github: <https://github.com/kernx86/PyStock> MIT License
   4. Potential modules and software
      1. CRON
      2. AWS for scaling
2. Installer script into /software\_name/
   1. Install modules and software from 1.c
   2. Write in bash
   3. Validator bash
   4. Directory structure:
      1. Dist
         1. Logs
            1. Pystock\_error.log
            2. Pystock\_info.log
            3. Pystock\_warning.log
            4. Pystock.log
         2. Log\_mgr.py
         3. \_\_init\_\_.py
      2. docs
      3. src
         1. lib
            1. http\_request.py
            2. db\_mgr.py
            3. \_\_init\_\_.py
            4. Main.py
            5. Cruncher.py
            6. Agent.py
         2. pystock.config
         3. setup.py
      4. test
         1. benchmarks
         2. e2e
         3. unit
      5. tools
         1. Scrapy
         2. NumPy
         3. TensorFlow
         4. Python
         5. PostgreSQL
         6. SQLite
         7. Amanda (remote backup)
      6. LICENSE
      7. README
3. Main.py
   1. I: None
   2. O: None
4. Log\_mgr.py
   1. I: log level, byte size max, delete log limit
   2. O: none
5. http\_request.py
   1. I: url
   2. O: return crawler object, HTTP code & method, response, selector, title, graph, current price
6. db\_mgr.py
   1. I: http\_request.py O
   2. O: none
7. Cruncher.py
   1. I: DB location, table, row, column
   2. O: (TBD