Planning Documentation for StockSplosion Buy-Sell Trading Application

# Introduction

This informal document is intended for describing research I plan to undertake for this project. As an I.T professional, I believe it is very important to document plans before undertaking projects. A lecturer I had during my degree taught my class to plan everything before ‘letting my fingers loose’ to code. This is something I try my best to adhere to – with the best results coming from following plans.  
  
While this document is not intended to be formal, I will structure it so that it can be easily understood. This document may also be changed throughout the project – reflecting the constant need for adapting to acquired knowledge.

# Key Areas for Research

While this project may be small, researching key areas of the project is still very essential. The sections below this paragraph describe the core areas I see of this project.

## Appropriate Web-Based Technologies

### The Server

My initial thoughts on this project was not to use a server at all; however, this changed when I realized API calls might not work from HTML files alone. I intend on using Apache 2.2 to serve this project – I already have the correct environment set up, so troubleshooting server-related bugs will be simpler.

An alternative to the Apache server could be the use of Node.JS. I have used Nose.JS on Linux, Windows, and MAC OS machines before with minimal issues, so this may be another server implementation to try, should Apache fail.

### Application Code

#### Bootstrap

To structure the application with minimal time, and higher quality, I intend on using the popular [Bootstrap](http://getbootstrap.com/) library. As I specialize primarily in back-end development, utilizing this library will make it easier to construct an attractive layout.

#### HighCharts

To display stock prices in an attractive, influential, and understandable manner, I intend on using the HighCharts library. I have had great experiences with this library in the past; I find it more comprehensive than other libraries, such as Charts.js.

With this particular project, I am very interested in using the ‘[Compare Multiple Series](http://www.highcharts.com/stock/demo/compare)’ feature of [HighCharts](http://www.highcharts.com). By using this feature, I intend on tracking the stock price average with its price history, together with low/high trend lines.

The buy/sell/wait button dialogs will be placed directly below the chart. The ‘buy’ option will be coloured green, the ‘sell’ option will be coloured blue, with the ‘wait’ option being coloured grey. Via the use of an appropriate algorithm, the recommended option will be highlighted.

#### StockSplosion API

As per the project brief, I will be using the [StockSplosion API](http://docs.stocksplosion.apiary.io/) to source the appropriate stock data.

#### BackBone.JS

To help render the appropriate chart data, I plan to use [Backbone.JS](http://backbonejs.org/). I have just begun to explore this framework, and I’m very impressed by how such a light-weight library can handle API data. This library will play an important part in handling interaction described in the following paragraph.

The user will be able to type in the stock code into a textbox, with options automatically being shown while typing. The user will then be able to: click on one of the options, press the Enter key, or press a ‘Show Stock Pricing History’ button. Each of those options will display the appropriate HighCharts chart.

## Stock Trading

As my knowledge on stock trading is very minimal, I’ll need to conduct some basic research before I proceed with coding. This is especially crucial, as I need to understand what stock trading is in order to formulate a buy/sell/wait algorithm. My ‘current’ knowledge tells me to ‘buy low, and sell high’ – so I at least have a starting point for further algorithm research. The following links point to some resources I will be reading closely:

* <http://pages.stern.nyu.edu/~adamodar/New_Home_Page/articles/introinvphil.htm>
* <http://articles.economictimes.indiatimes.com/2011-01-16/news/28428960_1_stocks-fair-value-volatile-markets>
* <http://www.investmentu.com/article/detail/45816/how-investors-can-determine-when-to-buy-sell-hold>

A basic algorithm I have come up with is as follows:

* Declare array to hold maximum encountered values (in array traversal)
* Initialise empty maximums array to hold same number of values as stock prices array
* Iterate over stock prices array, in reverse order  
  - For each price, in reverse  
  - If current price greater than maximum, maximum equals current price  
  - Else current maximum equals previous maximum
* Begin determining whether to buy, sell, or hold
* Declare count variable for ‘buy’
* Declare count variable for ‘sell’
* Declare count variable for ‘hold’
* For every stock price   
  - If current price less than maximum price, increment ‘buy’   
  - If current price greater than maximum price, increment ‘sell’   
  - If current price equal to maximum price, increment ‘hold’
* If ‘buy’ greater than ‘sell’, and ‘buy’ greater than ‘hold’, return ‘buy’
* Else if ‘sell’ greater than ‘buy’, and ‘sell’ greater than ‘hold’, return ‘sell’
* Else if ‘hold’ greater than ‘buy’, and ‘hold’ greater than ‘sell’, return ‘hold’

In some situations, I believe one might want to allow a small ‘margin’ to determine whether to buy, sell, or hold. An example could be when a stock price is + or - a small percentage over a suggested buy, sell, or hold amount. A way of calculating this percentage would be to sum up the rate of change between each stock price during a specified time span, then divide this sum by the stock price average (within the same specified time span) to obtain this margin value. A basic example is outlined below:



Since the value in this example is negative, this can be the lower-end of this margin. If this value can be converted into the positive equivalent, it could also be the higher-end of this margin. Again, some more research will need to be done.