TEAM 3

WEB BASED STOCK FORECASTERS

https://sites.google.com/site/group3stockforecasting/

REPORT 1

Peter Zhang

Vincent Chen

Robert Adrion

Syedur Rahman

Robin Karmakar

Mohammed Latif

Manoj Velagaleti

Responsibility Matrix

RESPONSIBILITIES	TEAM MEMBERS: Robert Adrion	Vincent Chen	Robin Karmakar	Mohammed Latif	Syedur Rahman	Manoj Velagaleti	Peter Zhang
Project Management (10)	15%	20%	15%	20%	15%	15%	
Problem Statement (5)				100%			
Glossary of Terms (4)		100%					
Functional Requirements (2)	30%		70%				
Non-Functional Requirements (2)						100%	
On Screen Appearance Requirements (2)					100%		
Stakeholders, Actors, & Goals (2)	50%	50%					
Use Case Casual Description (8)		30%		30%			40%
Use Case Diagram (5)							100%
Use Case Full Description (10)				50%		50%	
System Sequence Diagram (5)		100%					
Preliminary Design (11)					100%		
User Effort Estimation (4)							100%
Concept Definitions (10)	100%						
Association Definitions (5)			100%				
Attribute Definitions (5)			100%				
System Operation Contracts (5)						100%	
Plan of Work (5)	25%	12.5%	25%	12.5%		12.5%	12.5%
TOTAL POINT ALLOCATION	14.35	15.025	14.152	15.525	14.5	14.125	12.825

Allocation of Points

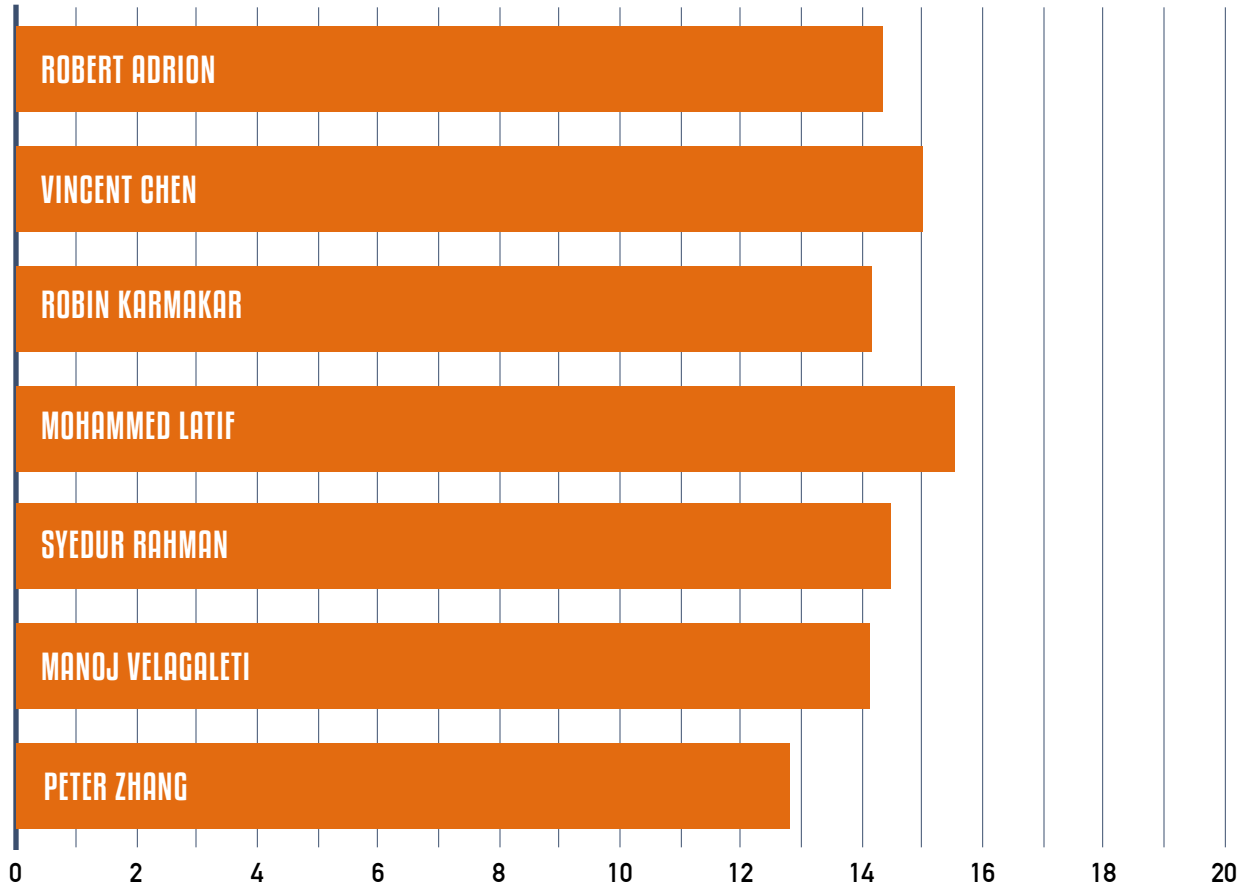


Table of Contents

1. Customer Statement of Requirements

1.1. Problem Statement

From its birth, the stock market has always been a place where businessmen could gain capital buy selling shares of their companies and where investors can purchase these shares in hope to make a profit when those companies prosper. Although the market still serves both these purposes, today it is judged less by what it does for businessmen seeking capital but rather for what it can do for investors seeking gain. In a time of high taxes and low wages, trading stocks seems like a viable and almost effortless way of becoming very rich very quickly. Although this scenario has proven true for many, there are just as many, if not more, who can proclaim the exact opposite. The problem here lies with the fact that the same volatility in the market that can be exploited for gain, can be the cause for loss as well. As both amateur and knowledgable investors, this is what worries as.

Motivation:

For day to day trading, these risks are a major factor in keeping away new, eager yet unaccustomed, traders from the market. They lack the appropriate tools needed for the decision making process when compared to existing professional traders. This scenario is witnessed way to often and can range from the young wanting to have a steady stream of capital without the need to invest in a part time job whilst also being a full time student or even the old who are well established into life attempting to save up for retirement. In both of these cases, these are individuals who would gladly consider trading stocks but many a times shift away from the idea due to the lack of time, recourses, and knowledge in the field.

To the college students amidst us, the idea of “gambling” in the stock market is fear inducing. Today, to turn this “gamble” into at most an informed estimate we would have to go through hours upon hours of research to first acquire the knowledge needed to even begin to understand how trading decisions are made. Understanding is only the first hurdle, however. Once one knows how to read charts, number crunch historical prices, or acquire and comprehend a companies balance sheets amongst many other details, the act of actually performing these tasks remains, something easier said than done. As students, the time and energy needed for such undertakings simply does not exist. This dilemma is one of which anyone foreign to the field of stocks can relate to.

The long-term investors among us, on the other hand, rely on a research heavy way to invest. There methods entail analyzing the financial performance of many companies and choosing the ones that appear to have the best growth potential. Since their prospects are long term, it may seem at first that the volatility of the market can be of no help here. Even these types of investors, however, are looking for as much profit as attainable and therefore do not overlook purchasing a stock at the lowest price possible. Thus, they too rely on the technical analysis techniques of day to day traders. Even with their acumen of the field, however, number crunching and attempting to eyeball trends off of charts for these split second decisions can not only be time consuming and stressful but are also prone to human error.

Most investment advisors suggest maintaining a diverse investment portfolio in order to reduce the risk of investment loss. It is said we should not “put all of their eggs in one basket” and thus should increase the quantity of our investments over various industries. With ten times the stocks, however, arises the ten times the difficulties stated previously. Added on are the problem of organizing and keeping track of each and every investment. For the new and eager day trader among us, if the decision making of one investment seemed fearful how would he/she orchestrate his/her choices when suggested to acquire multiple shares? Without a proper way to declutter and systemize these decisions, these individuals may never move forward from owning one specific type of share at a time and thus be vulnerable to losses due to sudden and unforeseeable events even if their initial investment decision was deemed to be wise.

To summarize, the leading complications that exists with using the stock market as a viable stream of capital is the fact that many newcomers are not knowledgable with the financial concepts needed to make an educated trade decision. Even then, the knowledgeable as well, may find that their techniques are time consuming and stress inducing. When deciding to swallow a set of data and, based on multiple patterns and trends, spit out a conscious commitment that can be the difference between a great profit and a terrible loss, human error is bound to occur. When a myriad of companies are further introduced into the equation, these obstacles accumulate exponentially and the need of quick, clear , and organized information is paramount.

Vision:

What we, as both novel traders as well as long term investors, require is speed and simplicity. Envision being able to search for a particular stock and get a reasonable trading decision within the blink of an eye; a place which not only makes making these decisions effortless, but also provides the information necessary to learn the process behind the decision if we should ever choose to. Computer technology throughout the years has greatly introduced the idea of automation as well as being a learning tool for those that seek it. Everyone nowadays has access to the internet in some way shape or form whether it be a desktop computer, tablet, or even smartphone. Thus, to allow for effortless accessibility to the common user, such as you and I, a web based software service is required.

The first and foremost purpose of the system should be to predict future stock prices. There are many prediction methodologies that fall into two general categories that can and often will overlap: fundamental analysis and technical analysis. Fundamental analysts are concerned with the company that underlies the stock itself. They evaluate a company’s past performance as well as the credibility of its accounts. Technical analysts are not concerned with any of the company’s fundamentals. They generally seek to determine the future price of a stock based solely on the potential trends discovered from observing historical pricing of that stock. This set of past prices can range from dating back anywhere from a week to a couple years. Data used in fundamental analysis for a stock are generated much more slowly than the price and volume data used by technical analysts. Financial statements, for example, are filed quarterly and changes in earnings per share don't emerge on a daily basis. Some even argue there is less reason to analyze a company's fundamentals because these are believe to be

accounted for in the stock's prices. Therefore, considering that we want quick gains without the necessities of research and market knowledge, technical analysis should be primarily used by the proposed system to predict trade decisions for the user.

The use of technical analysis, however, requires the crunching of a myriad of numbers as well as inspecting detailed charts; tasks that we do not want to waste the time even attempting to perform. The use of software should not only automate this process but also produce prediction results with the accuracy of that greater than a human. Each of these results should also come with a reasonable confidence value asserting to the user how probable the given decision of "buy", "sell", or "hold out/sit in" is to be correct. There are various methods to gauge stock trends, and as many as possible should be used to increase prediction accuracy. The result of each model should be displayed to the user with up to date information regarding the current stock, as well as an overall decision based on a weighted average of all models. All of this should be easily accessible at 1-2 mouse clicks away. According to Akamai's *State of the Internet: Third Quarter, 2013 Report*[1], in the United States a resident, on average, has 9.8Mbps broadband connection speed and an average web page takes 2.8 seconds to load. Therefore, following these standards, loading times of this service should be no more than 2.8 seconds on at least a 9.8Mbps connection.

So far, the use of software to aid those of us looking for quick and effortless stock forecasts has been conceptually realized, but what about those that not only want a decision but also aspire to learn the process behind the decision? Some might argue that this will defeat the purpose of not having to be knowledgeable to get reasonable stock predictions. The beauty of this feature, however, is that it is completely optional and thus will serve both sides of the crowd: those that just want quick, easy, and well tested advice to base their decisions on and those that are looking for a little more reasoning behind those decisions. Thus, differing from other designs, each prediction result should have a way to reveal an explanation to the user of the technical analysis used to get to that specific result; an environment for users to learn similar processes used by professionals if he/she chooses to.

With the ability to gain quick prediction for a single stock, what happens if a user would like to broaden his/her investment portfolio with multiple stocks? Again, software can alleviate the issue. With an account based system, the web service should allow for us to track and view multiple predictions all at the same time based on stocks we both own or are simply interested in purchasing. When the system predicts and notices an impending prominent decision, we should be immediately alerted and informed of that decision. With the addition of personal accounts, however, security always becomes a concern. Using statistics gathered from the bank login protection software that runs on 4 million PCs over the last year, security vendor *Trusteer* found that 73 percent of users were using the password for their online bank sites to access at least one other website[2]. Thus, although no real personal data about users would be stored (besides their full name), passwords should always be kept secure. If a particular user is not logged into the service, however, he/she should still be able to search for

---

[1]http://www.akamai.com/dl/akamai/akamai-soti-q313.pdf

[2]http://landing2.trusteer.com/sites/default/files/cross-logins-advisory.pdf

a stock and view the predictive decision associated with it. Furthermore, if no stocks come to mind, we should be able to ask the system to recommend a stock to purchase based on the highest predicted gain for that day. All of these features should be organized into an easy to use modern user interface not only accessible to desktop browsers, but the mobile workspace as well. Through these methods, we will gain the organization and clutter free environment needed to mange multiple stocks. The current method of tracking both owned and interesting stocks is simply by remembering and memorizing. An automated method would ultimately relieve us from this task. Furthermore, making the system mobile friendly, will allow these services to be portable, especially in an era where people are always on the move and attached to smartphones and tablets.

Our business plan is to offer the service free and support the operations from commercial advertisement proceeds. Obviously, due to budget and technical limitations, not all existing stocks can be accounted for and thus, only a limited amount can be predicted by this service. These concerns also restrict the frequency of predictions as well as the rate of updating current stock prices. Thus, an administrative account should be created to allow changes to these limits should the service prove to be popular and budget allows for upgrades to superior hardware. The system should log and display to the administrator how long a prediction session has taken so that he/she may make adjustments to timings and/or the list of predicted stocks to compensate. The system should also keep track of what stocks users are searching for display these analytics to the administrator.

1.2. Glossary of Terms

Algorithms: a step-by step procedure for calculations.

Artificial intelligence: the intelligence exhibited by a machine or software, the branch of computer science that develops machines and software intelligence. This intelligence can perceive its environment and take actions that maximize its chances for success.

Closing price: the final price at which a security is traded on a given trading day. It represents the most up-to-date valuation of a security until the next trading day.

Database: an organized collection of data that are typically organized to model relevant aspects of reality in a way that supports process rewiring this information.

Data mining: The computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning statistics, and data base systems.

Fundamental analysis: a method of evaluating stocks. It tries to measure the value relating to economic, financial and other qualitative and quantitative factors to produce a value that an

investor can compare with current stock prices. With this they can decide what action to take with the security.

Long term investment: an account on the asset side of a company's balance sheet that represents the investments that a company intends to hold for more than a year. They may include stocks bonds, real estate and cash

Machine learning: A branch of artificial intelligence, concerned with the construction and study of systems that can learn from data.

Neural network: conceptually based off the central nervous system, it interconnects systems of neurons that can calculate values for inputs by feeding information through the network.

Short term investment: An account in the current assets section of a company's balance sheet. This account contains any investments that a company has made that will expire within one year. For the most part, these accounts contain stocks and bonds that can be liquidated fairly quickly.

Stocks: A type of security that signifies ownership in a corporation and represents a claim on part of the corporation's assets and earnings.

Technical analysis: a method of evaluating securities by analyzing the statistics generated by the market today. It uses charts and other tools to identify patterns (such as past price and volume) that can suggest future activity of that security

Technical indicators: look to predict the future price levels or simply the general price direction of a security by looking at past patterns.

Time Horizon: the length of time over which an investment is made or held before it is liquidated. It can range from seconds-decades.

User interface: the space where interaction between humans and machines occur.

"Watch List": a list of tracked stocks stored within the system for a particular registered user.

Web service: is a method of communicating between two electronic devices over the World Wide Web

2. System Requirements

2.1. Enumerated Functional Requirements

IDENTIFIER	PW	REQUIREMENT
REQ1	5	The system shall allow the administrator to add and remove stocks from the list of stocks that predictions are made on.
REQ2	5	The system shall continuously gather the time series of the current market data (stock prices, trading volumes, etc) for a set number of companies.
REQ3	5	The system shall periodically apply prediction algorithms or models on the obtained data and store the results to a central database.
REQ4	5	The system shall allow for users to search for a particular company and display the predicted trade decision associated with that stock.
REQ5	4	The system shall allow users to request a stock that is predicted to have a large gain in the near future.
REQ6	4	The system shall obtain and display a confidence value for each prediction given to the user.
REQ7	3	The system shall support registering new investors by providing a real-world email, which shall be external to our website. Required information shall include a unique login ID and a password that conforms to guidelines, as well as users first and last name. Upon successful registration, the system shall set up an account for the investor.
REQ8	3	The system shall allow for registered users to track stocks that he/she owns or is interested in purchasing. Registered users shall be alerted via various methods chosen by that user, if an impending prominent prediction is made for a stock he/she has chosen to track
REQ9	2	The system shall allow a method for users to learn the professional analysis used for each prediction method if he/she chooses to.
REQ10	2	The system shall allow for the administrator to alter the rate at which current stock prices are updated and the rate at which predictions are made.
REQ11	2	The system should log and display to the administrator the time taken for previous prediction sessions.
REQ12	1	The system should track what stocks users are searching for and log and display these analytics to the administrator.

IDENTIFIER	PW	REQUIREMENT
REQ13	1	The system should provide the user with options to troubleshoot various issues such as, but not limited to, a FAQ, tutorial system, or methods to contact technical support (i.e send an email to the admin) and show solutions to similar issues other users had.

Table 1: Enumerated functional requirements for a web based stock forecasting system.

In the customer statement of requirements, our customer explained the priority weights (PW) as follows. Obviously, the main solution this system is going to be proposing is to have an automated process of predicting future stock prices. Naturally, not all listed companies can be accounted for. The World Federation of Exchanges has a total of 42,417 listed companies as of January 2014[1] and without sophisticated hardware, even approaching this number of daily predictions is unfeasible. Thus, before predictions can even be made, someone must select the range of stocks that the system will handle. Therefore, REQ1 is given highest priority. Furthermore, the core function of the system is to forecast stock prices, regularly update current prices so accurate predictions can be made, and to allow users to be able to search for those predictions. Thus, REQ2, REQ3, and REQ4 respectively are also given highest priority.

Requirement REQ5 allows users to suggest for stock that is predicted to have a very high gain and REQ6 allows users to be reassured of predictions prescribed to them by means of a confidence value. These are added features and not essential (although highly desirable) to the core functionality of the system as stated before, and thus REQ5 and REQ6 were given a lower priority.

Requirement REQ7 deals with allowing users to register for an account with the system and REQ8 allows those registered users to be able to track stocks and add them to his/her "watch list". These are given an even lower priority because users will not need to be registered to be able to use essential features of the system. Furthermore, users can track certain stocks by other means such as remembering ticker names and performing individual searches of each stock. A "watch list" serves as a means of convenience for users.

Requirement REQ9 allows users to learn the decision making behind each prediction. This is given an even lower priority because not all users will be interested in this feature. For those that are, however, other means of "learning" are available such as researching through books or the internet. This feature again only serves as a convenience for the user by giving them quick and easy information all in one place. REQ10 and REQ11 are given similar priorities. REQ12 and REQ13 are desirable but are given lowest priority and will only be implemented if development time allows.

The requirement REQ5 does not specify the number of stocks that will be suggested to users. We may introduce an option for the user to choose an arbitrary number or, for ease of

[1] http://www.world-exchanges.org/statistics/monthly-reports

use, will make that decision for them. Furthermore, REQ8 is the compound of two similar requirements:

> REQ8a: The system shall allow for registered users to track stocks that he/she owns or is interested in purchasing.
>
> REQ8b: Registered users shall be alerted via various methods chosen by that user, if an impending prominent prediction is made for a stock he/she has chosen to track.

Since the second depends heavily on the first (and the only reason why an account is needed in the first place), it was thought to be more useful to combine these into one requirement. For testing purposes, however, it may be useful to separate the requirement into its two sub parts.

Finally, requirement REQ12, was suggested by the customer to further allow the administrator to satisfy user needs based on their searches. This is better said than done, however. The combination of available characters to input for a search keyword are limitless and thus one must take into consideration how the system will handle storing and keeping track of these almost infinite combinations. One solution may be to somehow validate a keyword to being meaningful or not (i.e. does it relate to a World Federation of Exchanges listed company?) and only track of those that are.

Some of the above requirements are vague and may need to be further detailed into sub requirements. Note that these will not be listed as requirements on their own as fragmentation can cause clutter and a lack of simplicity.

IDENTIFIER	REQUIREMENT
REQ3a	(As REQ3 in Table 1)
REQ3b	The system shall only predict prices for stocks up to a week ahead.
REQ4a	(As REQ4 in Table 1)
REQ4b	The system shall only load and display at most 5 results at a time.
REQ8a	(As REQ4 in Table 1)
REQ8b	The system shall allow for registered users to remove stocks from their “watch lit”
REQ8c	The system shall allow for registered users to set their method of notification (i.e. email, text, or none).

Table 2: Extended list of functional requirements from Table 1.

Requirement REQ3b introduces a forecast “length” business policy:

> **BP01**: The system shall only predict prices for stocks up to a week ahead.

There are many options when it comes to stock forecasting as time frames can range from weeks to months and even years. A real world implementation can take advantage of varying prediction models for each time frame but for simplicity we will only deal with a week where, because of the volatility of the market, we feel prediction is most needed. This will also leave room for further upgrades to the system in the future where the policy allows for predictions to span a greater number of time frames.

REQ4b is intended to keep loading times low (since charts are needed to be created for each result) for the user and introduces another business policy:

BP-02: The system shall only load and display at most 10 results at a time.

The number of displayed results for a search may differ depending on the acceptable amount of time needed for loading those results and thus may be altered in the future.

Furthermore, REQ8b gives the option to registered users to remove tracked stocks from their "watch list" as well as set their method of notification. We have limited these methods to two (i.e text and email) but a real world implementation might choose to enact more such as a telephone call or Facebook message.

2.2. Enumerated Non-Functional Requirements

IDENTIFIER	REQUIREMENT
REQ14	The system shall ensure that if a copy of the database were to be acquired unlawfully, all user passwords shall remain secure.
REQ15	The system shall be able to run through all prediction models for each and every listed stock within 1 hour.
REQ16	The system shall allow for users to get a prediction for a stock within at most 2 mouse click.
REQ17	The system should insure that any features that do not require a user to be logged in are not hidden to unregistered users.
REQ18	The system shall have loading times no greater than 2.8 seconds with at least a broadband speed 9.8Mbps.
REQ19	The system should be able to run with core functionality intact from a smartphone/tablet workspace.

Table 3: Enumerated non-functional requirements for a web based stock forecasting system.

Since the system will be storing user accounts, REQ14 needs to be specified for security purposes. REQ15 and REQ18 insure that latency are kept to a low since the system will be a

web service where a fraction of a second in loading can be the difference between user satisfaction and disappointment. Furthermore, the customer intends for the service to be as easily accessible as possible and thus REQ17 insures visitors to the service are not locked out of key features and REQ16 insures user efforts for the most prominent feature (searching for predictions) is kept to a low. REQ19 finally insures that the system is able to be used for not only desktop browsers but also smartphones and tablets as well. This requirement is of the least priority because we will be working on the functionality for modern desktop web browsers first and then optimize for mobile if development time allows.

2.3. On Screen Appearance Requirements

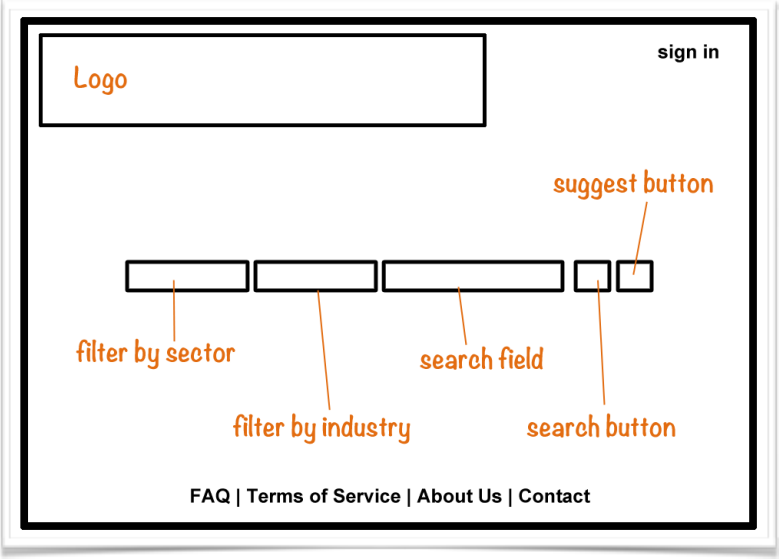


Figure 1: Customer's sketch of homepage

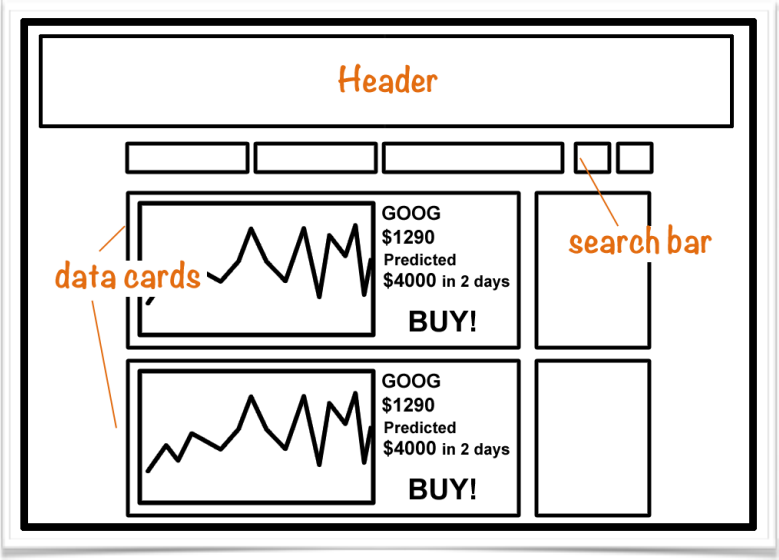


Figure 2: Customer's sketch of "results" page showing the organization of data.

IDENTIFIER	REQUIREMENT
REQ20	Figure 1 shows the design is for the first page a user comes across when accessing the web page. A search bar should be displayed with filtering options as well as a both a “search” and “suggest” button. This is also where a user will be allowed to sign in and visitors to register for an account. Various links should be displayed on the bottom of the page such as FAQ, About Us, Terms of Service, Contact, etc.
REQ21	Figure 2 shows the design for the page after logging in, after making a search, or after requesting a suggestion. All three of these pages will share similar layouts since they all will display similar information (where logging in will show tracked stocks). A persistent search bar is displayed at the top while stock data are given in cards filled with a chart of historical prices, current price, predicted price, confidence value, and predicted decision (buy, sell, hold/ sit-out). Most pages will use this “card” format to display information and thus there isn't any need to provide requirements for secondary pages (i.e. contact, registration, about us, etc).

Table 4: On screen Appearance requirements for a web based stock forecasting system.

2.4. Acceptance Tests

Acceptance tests that the customer will run to check that the system meets the requirements are as follows (not all requirements will be elaborated on due to limited time):

Acceptance test cases for REQ1:

ATC1.01 Use the administrative account to add a stock that does not already exist within the system. (pass: stock shows up on the list of stocks currently being tracked by the system)

ATC1.02 Use the administrative account to add a stock already being tracked by the system. (pass: stock does not show up twice on the list of stocks currently being tracked by the system)

ATC1.03 Use the administrative account to add an invalid stock (random letters). (pass: invalid stock does not show up on the list of stocks currently being tracked by the system)

ATC1.04 Use the administrative account to remove a stock from within the system. (pass: stock does not show up on the list of stocks currently being tracked by the system)

Acceptance tests for REQ2:

ATC2.01 View the current price of any stock within the system. Wait a time equivalent to the update timer and view the current price of the same stock. (pass: the two viewed prices should be different)

It should be noted there is a possibility that the price will not change after a successful update has been made but this possibility is very low considering that prices change continuously every time a trade goes through in real time and thus will be ignored.

Acceptance tests for REQ3:

ATC3.01 View the predicted price for a given stock and the day the stock is predicted to get to that price. Wait until stated day. Compare predicted price and actual price. (pass: predicted price and actual price have a small difference)

Since the system is working with predictions there really is no simple way to test that these predictions are working besides actually comparing predicted prices to actual prices. Using the given confidence value's associated with a prediction should give the customer an idea of how close a predicted price should be with its actual counterpart.

Acceptance tests for REQ4:

ATC4.01 Search for a stock known to be within the system using its ticker symbol. (pass: known stock's related data are shown)

ATC4.02 Search for a stock known to be within the system using a company name or part of a company name. (pass: known stock's related data are shown)

ATC4.03 Filter a search by a sector or industry and search without a keyword. (pass: all stocks within the given sector and industry are shown)

ATC4.04 Search by entering a random sequence of keys. (pass: no significant results should be found)

For test case 4.04 the customer should make sure that his/her random sequence of keys does not relate to any known company name or ticker symbol as that might trigger a valid search.

Also for test case 4.02, more than one result might be shown if the customer enters a vague search term such as “micro” in which the system might interpret the keyword to mean both “Microsoft Corp.” and “Micron Technology Inc.” It should be noted that for test case 4.03 the number of results loaded at a time will be limited by BP-02.

Acceptance tests for REQ5:

ATC5.01 Click the suggest button. (pass: stocks with high predicted gains are shown as a result)

ATC5.02 Select a sector and industry and click the suggest button. (pass: stock from the given sector and industry with high predicted gains are shown as a result)

Acceptance tests for REQ6:

ATC6.01 Search for a particular stock or get a suggestion from the system. (pass: a confidence value should be shown for every prediction)

Due to limited time, only the highest priority requirements were given acceptance tests. The customer can formulate his/her own tests for the remaining lower priority requirements.

3.Functional Requirements Specification

3.1. Stakeholders

Three Stakeholders can be identified:

1. Users: any user of the web service registered or not.
2. Advertisers: provider of advertisements for the web service.
3. Administrator: maintains and updates website services.

3.2. Actors and Goals

Eight actors can be identified:

1. Registered User: a registered user.
2. Visitors: any unregistered user
3. User: both a registered user and visitor (will be used for diagrams and descriptions where both a registered user and a visitor can interact with the system)
4. Database: records of stock information (i.e historical prices, prediction results, confidence value, etc), user data (i.e. username, password, tracked stocks, email, etc), and system data (timers, search logs, prediction time logs).

5. Price Provider (i.e. Yahoo! Finance): Provides the current pricing of a stock of interest
6. Timer: Tells the system when predictions should be made and when current stock prices should be updated.
7. Grapher (i.e. Google Charts): Provide visual charts from raw data.
8. Administrator: a special case User that maintains and updates website services.

3.3. Use Cases

3.3.1. Casual Description

The summary use cases are as follows:

UC-1: Login — Allows user to access their account and view information for their tracked stocks. [Derived from REQ7]

UC-2: AddStock — Allows the admin to add a stock to the list of stocks predictions are made on. («include» login). [Derived from REQ1]

UC-3: EditPredictionTimer — Allows for the admin to alter the rate at which prediction are made. («include» login). [Derived from REQ10]

UC-4: EditUpdateTimer — allows for the admin to alter the rate at which current stock prices are gathered («include» login). [Derived from REQ10]

UC-5: RemoveStock — allows the admin to remove a stock from the list of stocks predictions are made on («include» login). [Derived from REQ1]

UC-6: Search — allows a user or visitor to search for a particular stock through keywords and filter by sector or industry and view its corresponding prediction . [Derived from REQ4]

UC-7: Suggest — allows a user to request for the system to suggest a stock that is predicted to have the highest gain. [Derived from REQ5]

UC-8: Track — allow users to add a certain stock to their "watch list". («include» login, «include» search or «include» suggest). [Derived from REQ8]

UC-9: RemoveTrackedStock — allow users to remove a certain stock from their "watch list" [Derived from REQ8]

UC-10: Register — allow a visitors to fill out the a registration form and become a user. [Derived from REQ7]

UC-11: Logout — allow users to log out of the system. [Derived from REQ7]

UC-12: Update - allows timer to initiate the loading of current stock prices from the price-provider into the database. [Derived from REQ2]

UC-13: PredictAndNotify — Allows timer to initiate the prediction of future stock prices to be later stored into the database. Registered Users are then notified if an change in prediction is made for a stock he/she has chosen to track. [Derived from REQ3, REQ6, REQ8, REQ11]

UC-14: Learn — allows user and visitors to learn the decision making process behind a prediction by clicking on that prediction. [Derived from REQ9]

UC-15: Support — allows user to access technical support and send emails to the admin. [Derived from REQ13]

Some alternative use cases may be considered. For example, Use Case UC-13: PredictAndNotify can be broken up into two separate use cases (one for the prediction process and one for the notification process). This would require for the Timer actor to keep track of yet another time period. Since the customer stated that alerts should be immediately sent out as soon as a change in prediction for a user's tracked stock is detected, we concluded it would be best to have the notification process right after predictions are made. Thus, only one initiating actor is needed for both.

Furthermore, Use Case UC-6: Search, further elaborates on searching methods such as having the option to filter that were not strictly implied by REQ4. This is to give users added convenience as well as let them be able to "play" with the systems search functionality without having to name specific companies.

It should be noted that the different prediction models are not stated as extension use cases. This is because during the analysis process we still do not know all of the models we will be using and are still researching viable prediction methods. In addition, going into these models will lead into talk about implementation which is not desired in the analysis phase.

3.3.2. Use Case Diagram

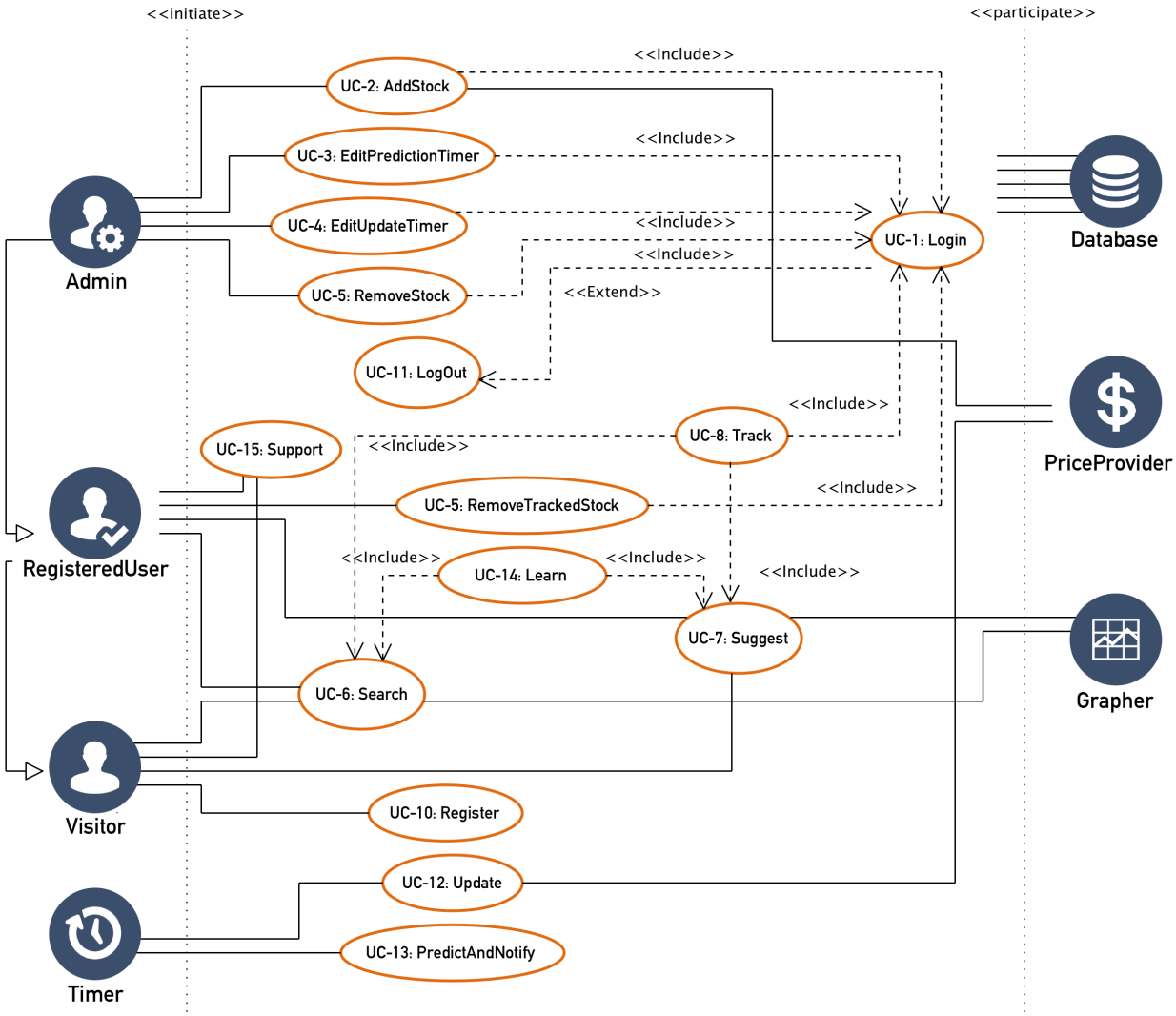


Figure 3: Use case diagram for our proposed web based stock forecasting system.

The use case diagram is shown in Figure 3. As shown by the diagram a registered user is a specialization of a visitor and an administrator is a specialization of a registered user. A visitor can search or get a suggestion from the system. A registered user can accomplish all that of a visitor but also is allowed to track stocks. Tracking a stock is an extension of a search or suggestion because the registered user must first retrieve the stock before he/she can track it. Similarly for UC-14: Learn, a user must first get a prediction before he/she can learn the involved analysis behind it and thus it must be an extension of a search or suggestion. It should be noted that although an administrator has similar rights to that of a registered user (search, track, get a suggestion), the diagram does not show these connections because these are not ultimately the administrators goals. The database is a participant in all use cases, and thus its connections are not shown to reduce clutter.

3.3.3. Traceability Matrix

REQ	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12	UC-13	UC-14	UC-15
REQ1	5		X			X										
REQ2	5												X			
REQ3	5													X		
REQ4	5						X									
REQ5	4							X								
REQ6	4													X		
REQ7	3	X									X	X				
REQ8	3								X	X				X		
REQ9	2														X	
REQ10	2			X	X											
REQ11	2													X		
REQ12	1						X									
REQ13	1															X
MAX PW		3	5	2	2	5	5	4	3	3	3	3	5	5	2	1
TOTAL PW		3	5	2	2	5	6	4	3	3	3	3	5	14	2	1

Figure 4: Traceability matrix mapping the use cases to system requirements. Priority weights (PW) are given in Table 1.

The above traceability matrix shows how use cases map to system requirements. With calculated priority weights the use cases can be prioritized by the following:

UC-13 > UC-6 > UC-2, UC-5, UC-12 > UC-7 >…
… UC-1, UC-8, UC-9, UC-10, UC-11 > UC-3, UC-4, UC-14 > UC-15

We will select the 5 use cases with highest priority for further elaboration. UC-5: RemoveStock, although a high priority, only deals with clearing the database for given stocks and thus will not be elaborated on. In its place we will further elaborate on UC-8: Track.

3.3.4. Fully Dressed Description

Use case UC-2: AddStock, allows the administrator to add stocks into the system. This is the first logical step necessary to instantiate the system since out of the box it will contain no stocks for predictions to be made on. It is up to the administrator to pick and choose companies he/she views customers will want to get forecasts for. As stated before, there are over 42,000 companies that the administrator can choose from and having a simple check box for each is unfeasible. He/She should rather enter desired stocks into a text box and have the system verify that the company exists. Since each and every stock already has its own unique identifier, known as a ticker symbol, the symbol will used for searches rather than terms susceptible to vagueness such as company names.

USE CASE UC-2: ADDSTOCK

Related Requirements: REQ1
Initiating Actor: Administrator
Actor's Goal: to add stocks to the list of stocks within the database (can be empty) that predictions are made on.
Participating Actors: Price Provider, Database
Preconditions: Administrator is logged in.
Success End Condition: All of the chosen stocks are added to the list of stocks within the database that predictions are made on. Historical prices for the given stocks are retrieved from the Price Provider and stored within the database.
Failed End Condition: The administrator is notified of all stocks out of the entered stocks that were not found in the Price Provider's database.

Flow of Events for Main Success Scenario:

Include::Login(UC-1)

1. **Administrator** enters ticker symbols for the stocks he/she wishes to add into an "Add Stocks" text field. If multiple ticker symbols are inserted, they should be separated by commas.
2. **Administrator** clicks the "Add Stocks" button.

(loop: for each ticker symbol entered)

1. **System** verifies the stock currently does not already exist within the **Database**.
2. **System** creates a request URL for historical prices using the ticker symbol.
3. **System** requests historical prices from the **Price Provider**.
4. **Price Provider** returns historical prices for the given stock.
5. **System** creates a request URL for current price using the ticker symbol.
6. **System** requests current price from the **Price Provider**.
7. **Price Provider** returns current price for the given stock.
8. **System** stores the historical prices for the given stock within the **Database**.

(end loop)

9. **System** notifies **Administrator** that all stocks were successfully added.

Flow of Events for Alternate Scenarios:

5a. Price Provider returns an invalid URL.

1. **System** records the ticker symbol.
2. **System** gets the next ticker symbol entered by the **Administrator** and goes to step (3) of the main success scenario. Once the loop has ended step (8) of the main success scenario is not executed.
3. **System** notifies **Administrator** of all stocks that have failed to be added.

Note that in alternate scenario 5a, an invalid URL is detected which in turn corresponds to an invalid ticker symbol (most likely caused by a typo). The system simply records the invalid symbol, and moves on to the next stock to be added. At the end, the admin is notified of all stocks that have failed to be added. Another version of this alternate scenario can have the admin be notified immediately as a stock fails to be added rather than waiting until the end of the sequence. We see no significance in choosing one over the other as both scenarios will ultimately return the same information.

In use case UC-6, it is assumed that a user can enter any combination of characters into the search field. Since searches (valid or invalid) need to be recorded for analytics, the system must first verify that an invalid search for a stock, although not located within our database is still a stock and not a random set of characters. This process of connecting keywords to known ticker symbols proves to be very difficult and thus will be ignored by our system, introducing another business policy:

BP-03: All search keywords (valid or invalid) will be logged and tracked

We believe that searches of gibberish will not be a frequent matter and thus will weed themselves out by having very low reoccurrences within the database (searches for "Microsoft" on the other hand may occur 10-100 times and thus will be at the top of analytics charts). Therefore, frequently clearing the database of any of keywords with low occurrences can be a solution to this problem. In the future, we may choose to invest in the time to create a way to connect certain keywords to ticker symbols. This will reduce the amount of "junk" stored in the database at a given time but in itself may be considered another software project.

USE CASE UC-6: SEARCH

Related Requirements: REQ4, REQ12
Initiating Actor: Visitor, Registered User (the word "User" will be used to represent both)
Actor's Goal: to find related information for a particular stock he/she has in mind.
Participating Actors: Database, Grapher
Preconditions: none
Success End Conditions: For the searched stock the visitor is provided with prediction results, confidence values, current price, and a line graph of the changing prices over time.

Failure End Condition: Search does not result in any stock stored within the database. A message indicating that system is currently not tracking his/her specified stock is displayed to the user and he/she is prompted to try another search.

Flow of events for the main success scenario:

1. **User** types in the name or ticker symbol of a stock he/she would like to search for inside the "search" text field.
2. **System** verifies that the given stock exists within the **Database**.
3. **System** increments the "search" counter for that stock within the **Database**.
4. **System** retrieves historical prices of that stock from the **Database**.
5. **Systems** sends historical prices to the **Grapher**.
6. **Grapher** returns a line graph of the change of prices over time to the **System**.
7. **System** retrieves prediction results for the given stock along with the confidence value for each prediction and the current price of the stock from the **Database**.
8. **System** displays the graph, prediction results, confidence values, and current stock price to the **User**.

Flow of Events for Alternate Scenarios:

2a. The given stock does not exist within the **Database**

1. **System** checks if the failed keyword exists within the **Database**.
   1a. Exists: **Systems** increments the "search" counter for the failed keyword
   1b. Doesn't Exist: **System** adds the failed keyword to the **Database**.
2. **System** displays a "no results found" message to the **User** and he/she is prompted to try another search.

Use case UC-8: Track, requires that a user be logged in and initiate a search or request a suggestion from the system.

USE CASE UC-8: TRACK

Related Requirements: REQ8
Initiating Actors: Registered User
Actor's Goal: Add a specific stock to a "wish list" so that the stock information is provided on the user's profile for easy access.
Participating Actors: Database
Preconditions: Registered User must be logged in and initiated a successful search (UC-5) or suggest (UC-6)
Success End Conditions: The selected stock the Registered User want to track will be added to his/her "watch list".
Failure End Conditions: no meaningful failure end conditions.

Flow of Events for Main Success Scenario:

Include::Login(UC-1)

Include::Search(UC-6) or Include::Suggest(UC-7)

1. **Registered User** clicks the "track" button for the stock he/she wishes to track.
2. **System** verifies stock does not already exist within the "watch list" stored for the Registered User within the **Database**.
3. **System** stores the stock within the "watch list" for the user in the Database.
4. **System** notifies **Registered User** the stock of interest has been added to his/her "watch list".

Flow of Events for Alternate Scenario:

2a. System detects the stock already exists within the "watch list" stored for the Registered User within the Database.

1. System notifies Registered User the stock of interest already exists within his/her "watch list"

An alternate scenario 2a is given where the stock happens to be already added to the registered user's "watch list". This scenario can be taken out altogether since the user's goal (to add a stock to his/her "watch list") is already accomplished. We thought it would be important to give the user feedback to his/her actions and thus decided to keep this alternate scenario.

Use case UC-12: Update deals with updating prices within the database and is similar to UC-2: AddStock in that it also requests current and historical pricing from the Price Provider. In this case, however, the possibility of requesting invalid stock information is removed since the system is specifying the requests rather than a human. This use case must not only deal with updating current day stock prices but also any day in-between the last stored price and the current day (If the system were to be shut down for a period of time longer than 24 hours).

USE CASE UC-12: UPDATE

Related Requirements: REQ2
Initiating Actors: Timer
Actor's Goal: To update pricing for the current day and any missing days in the Database for all stocks.
Participating Actors: Database, Price Provider
Preconditions: There is at least one stock within the database.
Success End Conditions: Prices stored in the Database are up to date for all stocks.
Failure End Conditions: no meaningful failure end conditions.

Flow of Events for Main Success Scenario:

1. **Timer** notifies the **System** to start updating.
2. **System** retrieves list of all stocks from the **Database**.
3. **System** retrieves current date.

(loop: for all stocks)

4. **System** retrieves date of last stored price within the **Database**.
5. **System** verifies the last retrieved date and current date match.
6. **System** creates a request URL using the ticker symbol and current date for the stock.
7. **System** requests current price from **Price Provider** using the request URL
8. **Price Provider** returns an up to date price for the stock.
9. **System** updates the pricing (corresponding to the current day) in the **Database**.
10. **System** waits two seconds.

(end loop)

11. **System** resets the **Timer**.

Flow of Events for Alternate Scenario:

5a. **System** notices last retrieved date and current date do not match.

1. **System** creates a request URL using the ticker symbol, current date, and last retrieved date for the stock.
2. **System** requests all prices in-between the current and last retrieved date from the **Price Provider** using the request URL.
3. **Price Provider** returns all prices in-between the current and last retrieved date.
4. **System** adds the missing historical prices in the **Database**.
5. Flow is restored to step 7 of Main success scenario.

It should be noted that in alternate scenario 5a, the request for both historical and current prices can possibly be combined together. We have separated these steps because by researching ahead, we've discovered that our Price Provider does not include a current price when returning historical prices. Also, to prevent the Price Provider from blocking requests we have decided to add a delay of two seconds between each request.

The final elaborated use case is also the most important. UC-13: PredictAndNotify deals with running prediction models for every existing stock within the database, calculating confidence values, calculating gain/loss, as well as notifying registered users of prediction changes. An internal timer is also used to log the amount of time a prediction session takes for analytics to be later displayed to the administrator. Since this process is all automated without any human interaction, no significant failure conditions exist.

USE CASE UC-13: PREDICT AND NOTIFY

Related Requirements: REQ3, REQ6, REQ8, REQ11
Initiating Actors: Timer
Actor's Goal: To initiate the prediction of future stock prices, store the predictions within the Database, and notify Registered Users of any prediction changes.
Participating Actors: Database
Preconditions: At least one stock exists within the database.
Success End Conditions: Predictions are made for all existing stocks and stored within the Database. Registered Users are alerted if any prediction changes are detected.
Failure End Conditions: no meaningful failure end conditions.

Flow of Events for Main Success Scenario:

1. **Timer** notifies the **System** to start making predictions.
2. **System** resets and starts an internal timer.
3. **System** retrieves list of all stocks from the **Database**.

(loop: for all stocks)
(loop: for all prediction models)

4. **System** predicts the future price of the stock and calculates a confidence value.
5. **System** stores the prediction and confidence value within the **Database**.

(end loop)

6. **System** retrieves results for all prediction models from within the **Database**.
7. **System** calculates overall prediction and gain/loss.

(end loop)

8. **System** stops internal timer.
9. **System** stores the time taken to make all predictions in the **Database**.
10. **System** retrieves list of all **Registered Users** from the **Database**.

(loop: for all Registered Users)

11. **System** retrieves list of all stocks tracked by the Registered User and his/her suggested means of alert (i.e. text, email) from the **Database**.

(loop: for all tracked stocks)

12. **System** retrieves current and previous predictions for the stock from the **Database**.
13. **System** verifies that the current and previous predictions are different.
14. **System** alerts Registered User through his/her suggested means of alert.

(end loop)
(end loop)

Flow of Events for Alternate Scenario:

10a. **System** notices current and previous predictions are the same

1. **System** retrieves the next stock and the flow is returned to step (9)

3.3.5. Acceptance Tests for Use Cases

Test cases for UC-2: AddStock include, but are not limited to:

TEST CASE IDENTIFIER: TC-2.01	
Use Case Tested: UC-2: AddStock — Alternate scenario 5a **Pass/Fail Criteria:** The test passes if the invalid stock is not added to the database and the administrator is notified of the error.	
TEST PROCEDURE:	**EXPECTED RESULT:**
Setup: The administrator logs into the system. Step 1: The administrator enters ticker symbols separated by commas into the "add stock" text box. At least one of the ticker symbols does not relate to a stock and is simply an input of random characters. Step 2: The administrator clicks the "add" button	 System notifies the administrator: [ticker symbol] was/were failed to be added.

Test cases for UC-6: Search include, but are not limited to:

TEST CASE IDENTIFIER: TC-6.01	
Use Case Tested: U6-2: Search — Alternate scenario 2a **Pass/Fail Criteria:** The test passes if the invalid search is added to the database if it doesn't already exist, or its counter incremented if it does it exist, otherwise it fails.	
TEST PROCEDURE:	**EXPECTED RESULT:**
Setup: User enters a keyword relating to a stock not being tracked by our system.	

Step 1: User hits enter or clicks the search button.	System checks to see if the searched keyword relates to any existing stocks within the database. No relation found: System checks to see if the keyword exists within the “searched for” database If keyword exists: System increments the counter associated with the keyword. If keyword does not exist: System adds the keyword to The user is given a “No results found” message and is prompted to try another search.

Test cases for UC-8: Track include, but are not limited to:

TEST CASE IDENTIFIER: TC-8.01	
Use Case Tested: UC-8: Track — Main success scenario **Pass/Fail Criteria:** The test passes if the selected stock is added to the registered user’s “watch list”, otherwise it fails.	
TEST PROCEDURE:	**EXPECTED RESULT:**
Setup: A registered user logs into the system and conducts a search or requests a suggestion from the system. Step 1: The registered user clicks the “track” button on the card displayed for the stock he/she wishes to track.	 System checks to see if the stock already exists within the registered user’s “watch list”. Stock does not exist: Registered user is given notification that the stock has been successfully added to his/her “watch list”

The above test cases are only a sample of the many that can be derived from all use cases success scenarios as well as the infinite amount of alternate scenarios.

3.3.6. System Sequence Diagrams

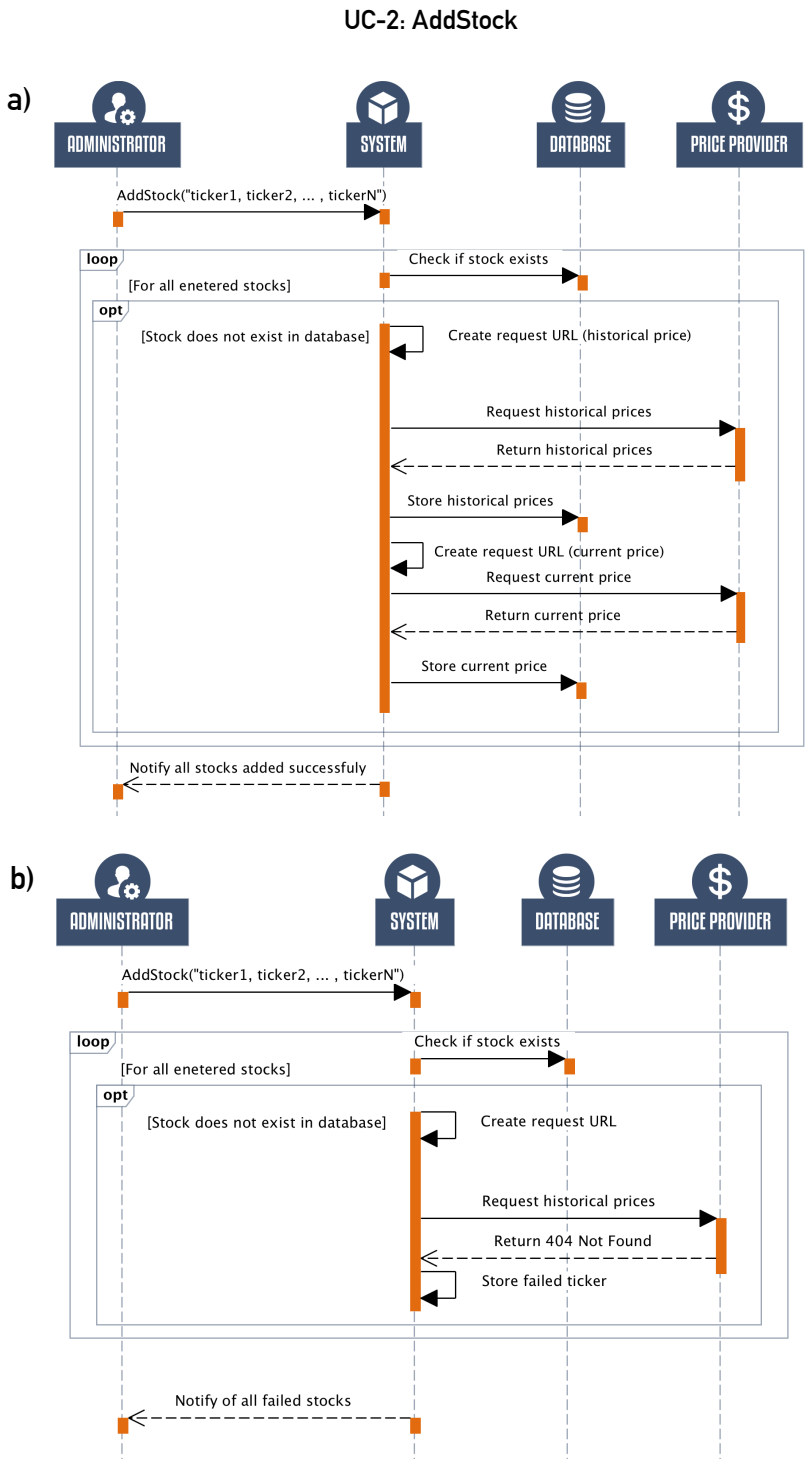


Figure 5: System sequence diagram from UC-2: AddStock. a) Main success scenario. b) An alternate scenario where the request URL does not exist.

Figure 5a shows the system sequence diagram for the main success scenario of UC-2: AddStock. A string of ticker symbols separated by commas is inserted by the administrator which initiates a loop within the system for each ticker symbol. The system first makes sure that the stock does not already exist within the database (to prevent from creating duplicates). If the stock does not exist, the system creates two request URL's using the given ticker symbol: one to request historical prices from the Price Provider and the other to request a current price. The system receives this data and stores it within the database. Once this process is ran for all stocks needed to be added, the administrator is notified of a success. Figure 5b, on the other hand, shows an alternate scenario where the administrator might have made a typo when entering a a specific ticker. Upon creating a request URL and then using that URL to request prices, the system is returned a 404 Page Not Found error. The failed ticker is remembered by the system and the loop moves on to the next stock. Upon looping through all entered tickers, the administrator is notified of the failed ticker.

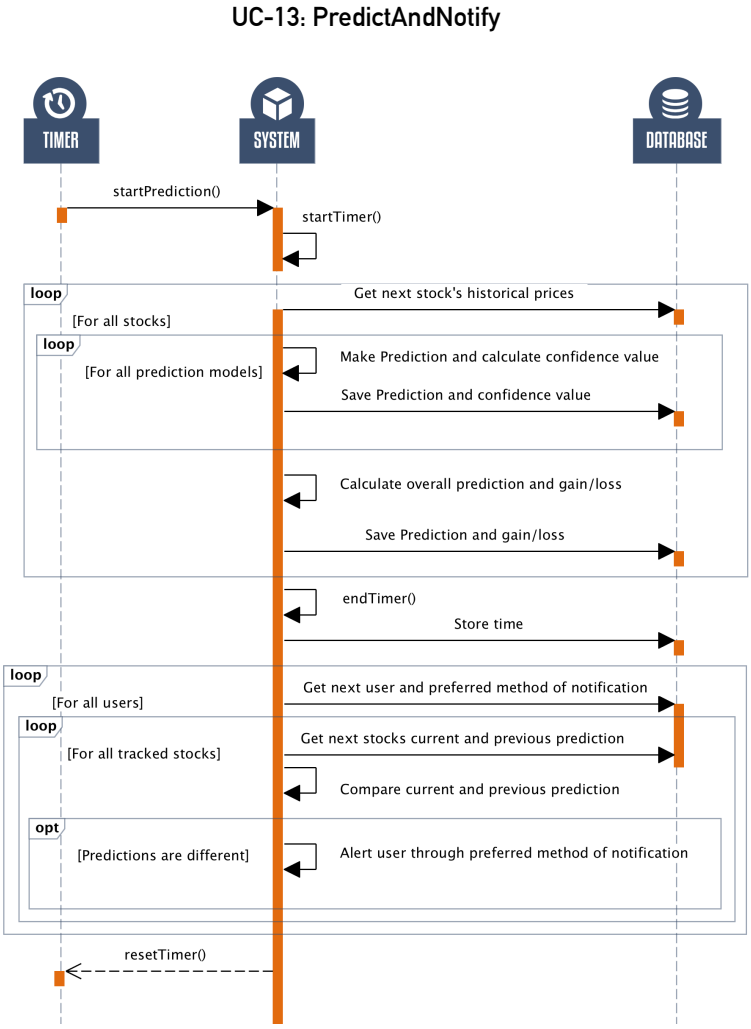


Figure 6: System sequence diagram for UC-13: PredictAndNotify

Figure 6 shows the system sequence diagram for UC-13: PredictAndNotify. The timer initiates this use case and a second internal timer within the system is also started. A loop begins for all stocks being tracked by the system. For each stock, prediction models are run on it's historical prices and confidence values are generated by testing previous predictions to current prices. Once all prediction models are ran, an overall prediction is generated along with the predicted gain/loss for that particular stock. All this information is stored within the database and the internal timer is stopped and the time is logged to be later displayed to the administrator. A second loop is then ran for all registered users. For each of their tracked stocks, a comparison is made between the previous prediction and the current prediction. If the two decisions differ, then a notification is sent out to that user. Finally, the initiating timer is reset.

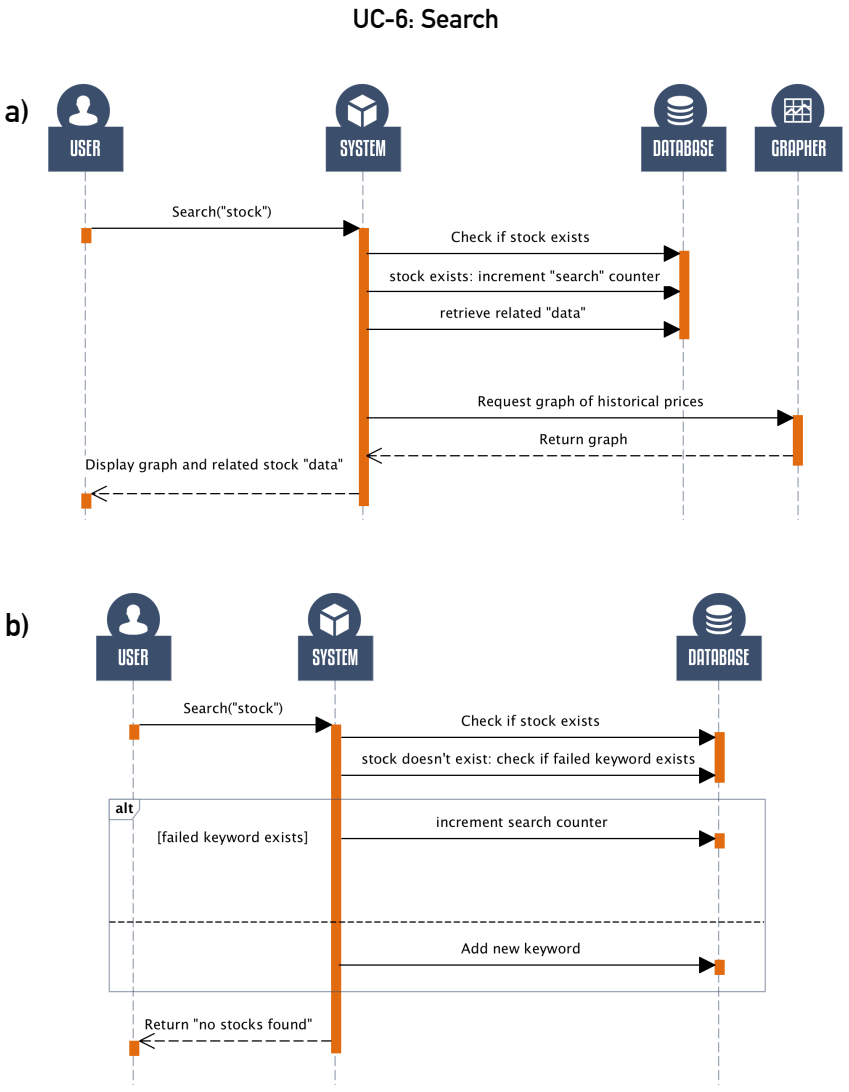


Figure 7: System sequence diagram for UC-6: Search. a) Main success scenario. b) An alternate scenario where a search fails to find any results.

Figure 7a shows the system sequence digram for the main success scenario of UC-6: Search. A user enters a keyword into a search bar and clicks the search button, thus initiating this use case. The system first checks if the keyword can be related to any existing stock within the database. If it exists a search counter is incremented for that stock for analytics and all of its related data is retrieved. This data includes historical prices, prediction results, confidence values, company name, ticker symbol, and current price. The historical data is sent to the grapher which then returns a visual chart from the raw data. The chart along with the other relevant data are then displayed to the user. Figure 7b shows an alternate scenario where the keyword does not relate to any stock being tracked by the system. The system will then check if the keyword exists within the database. If so, the keywords search counter is incremented. If not, the keyword is added to the database. The user is then returned a “no results found” message and is prompted to try another search.

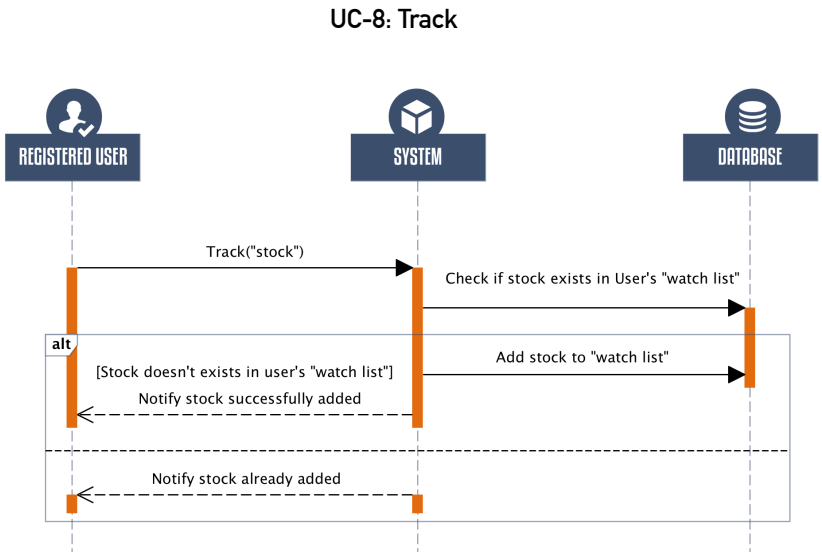


Figure 8: System sequence diagram for UC-8: Track

Figure 8 shows the system sequence diagram for UC-8 Track. A registered user must first trigger a search or request a suggestion from the system. Once data is shown for a given stock, the registered user clicks on the “track” button associated with that stock. The use case is then initiated. The system checks to see if the stock is already being tracked by the registered user. If so, he/she is notified that the given stock has already been added. If not, the stock is added to his/her “watch list” and he/she is notified of a success.

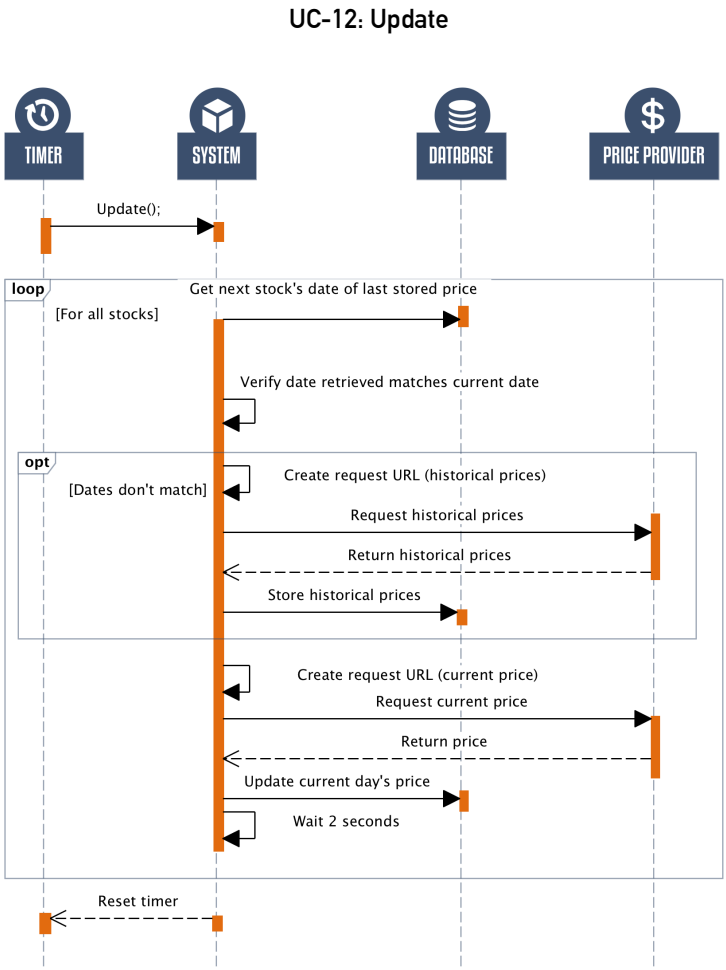


Figure 9: System sequence diagram for UC-12: Update

Figure 9 shows the system sequence diagram for UC-12: Update. This use case is initiated by the timer. The system loops through each and every stock being tracked by the system. For each stock, the date of the last stored price is retrieved and compared with the current date. If the dates don't match, the system creates a request URL for all historical prices in-between the two dates. Using the request URL, historical prices are requested from the Price Provider. Once returned, the data is saved to the database. For any case, a request URL for the current price is made by the system. The current price of a stock is requested from the Price Provider using the request URL. Once returned, the data is stored into the database. The system waits two seconds to prevent the Price Provider from blocking future request due to too frequent accesses. Once all stocks are intreated through, the initiating timer is rest.

4. User Interface Specification

4.1. Preliminary Design

Figure 10: Preliminary design of the home screen.

Upon entering the site for the first time, the user is presented with the splash screen area as shown by figure 10. Here the user can log in, register, search for a stock, or request a suggested stock from the system.

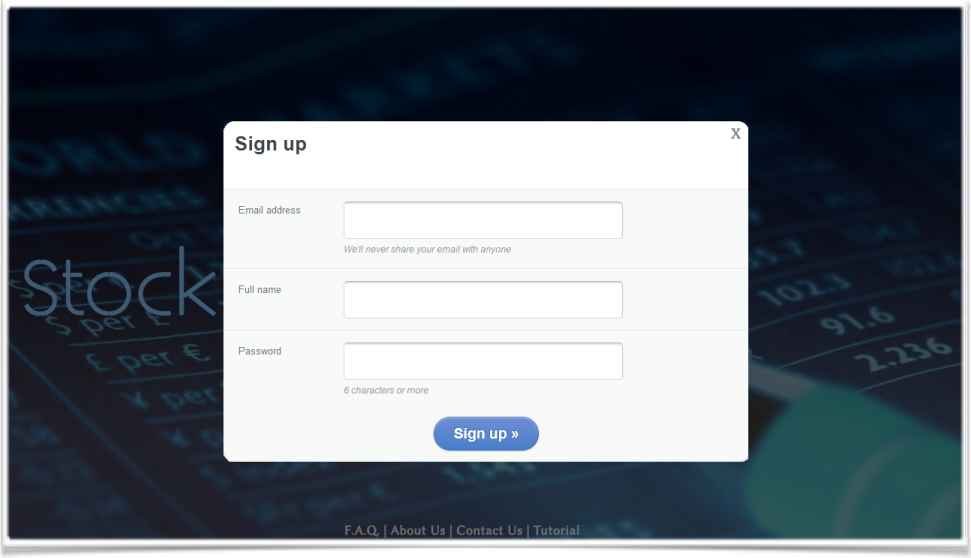


Figure 11: Registration page.

Upon clicking the registration button, the user is presented with the registration page overlaying the original page as shown by Figure 11. This is not limited to only the splash page so the registration page can overlay a variety of pages. However, as there is no need to show the overlay on the other pages, only the splash screen registration will be demonstrated in the picture above.

Figure 12: Dashboard upon logging in.

If the user logs in, the user will be taken to his or her dashboard as shown in Figure 12. Here the user will be presented with his or her tracked stocks and other information. The details have not yet been fully discussed so many of this can change in the future. The logout button will take the user back to the splash page and the arrowhead will activate the search. Clicking on any of the 'cards' such as the Google inc. stock information card, will open up a more detailed interface.

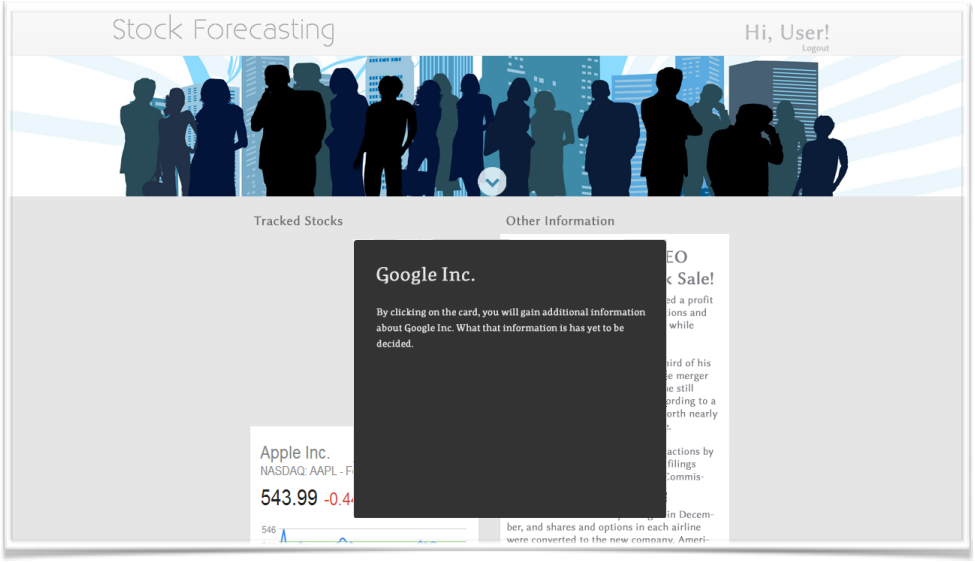


Figure 13: Clicking on a card

The card 'flips' to show more details about the stock itself upon clicking it. Information about individual prediction models can be shown here.

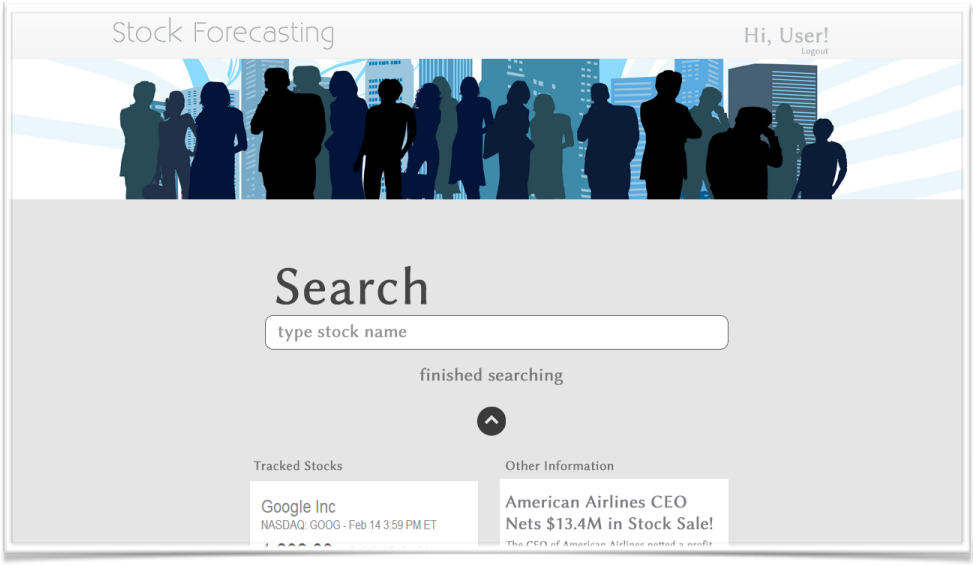


Figure 14: Clicking on arrow head

As shown in figure 15 the arrow head 'pulls' down the search from the banner. Clicking the arrow head again will pull it back up. As the user will see, the 'pull' effect drops the 'cards' lower as well.

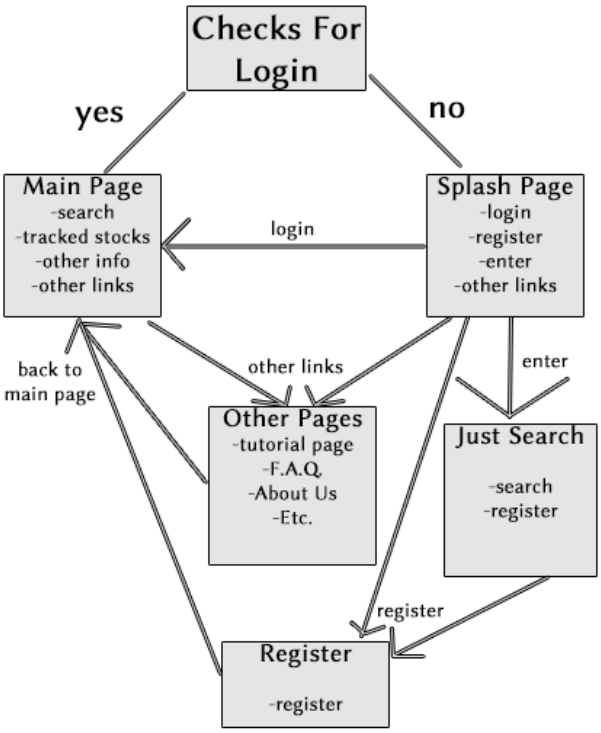


Figure 16: Navigation flow.

Figure 16 shows a basic navigation flow. Subject to change. Logout will always push the user back to the splash page. It should be noted that these are only preliminary designs and are

subject to change throughout the development lifecycle.

4.2. User Effort Estimation

Visitor or Registered User attempting to search for a specific stock:

a. Click inside the search text field
b. 4-10 Keystrokes to either enter a company name or a ticker symbol
c. Click “search button”

Total: 2 Clicks and 4-10 Keystrokes

Visitor or Registered User attempting to get a suggested stock:

a. Click “suggest” button

Total: 1 Click and 0 Keystrokes

Administrator attempting to add a stock to the list of stocks predictions are made on:

a. Click on username text field
b. 7-20 Keystrokes to enter username
c. Click on password text field
d. 7-20 Keystrokes to enter password
e. Click on “login” button
f. Click on “Add Stocks” text field
g. 4-30 Keystrokes to enter ticker symbols of stocks to add
h. Click on “Add” button

Total: 5 Clicks and 18-70 Keystrokes

Registered User attempting to add a stock to their “watch list”

Minimum Effort:

a. 1 Click and 0 Keystrokes to get a suggested stock (as explained above)
b. Click to track stock

Total: 2 Clicks and 0 Keystrokes

Maximum Effort:

a. 2 Clicks and 4-10 Keystrokes to search for a specific stock (as explained above)
b. Click to track stock

Total: 3 Clicks and 4-10 Keystrokes

5. Domain Model

5.1. Concept Definitions

The concepts and their definitions are discussed below.

Website:
Definition: A hypertext document connected to the World Wide Web.
Responsibilities:
- Display HTML document that shows the actor the current context(K)
- Shows what actions can be taken through buttons(K)

Motivation: This is probably the most important function, because satisfying the user is always the at the utmost priority. The user requires a functioning website that can meet their needs os providing stock analysis. To do this a dialogue must be created, therefore the presence of inputs and buttons is necessary. The website cannot simply be a reading page.

Entry:
Definition: Users way to input their intentions.
Responsibilities:
- Many different areas of inputs will be given to the user(K)

Motivation: All inputs(navigation,searches,etc.) will provide the user with a great sense of satisfaction. Therefore, the more area covered the better. It is of-course part of the website, but perhaps the most important part which the website must be in constant contact with to know if any input has been received. Time is short, therefore having a separate division of resources dedicated to recieving and transfer inputs is necessary.

Display Pages:
Definition: Focuses on the context of the website
Responsibilities:
- Display all graphs and data along with buttons(K)

Motivation: The website must not only look pleasing but also have capabilities of displaying the graphs. The graphs are the heart of the prediction model and without it how can the user truly trust any advice the system gives simply from words? Graphs are a simple and quick way to prove a strong point, and in this particular problem the point is to predict the future. So graphs are a great fit because most people can relate to time as part of a domain for a graph.

Gatekeeper:
Definition: keeps track of internal time of the system.
Responsibilities:
- Validate user actions through checking system data(K)
- Knows when to query for new stock information(K)

Motivation: The timer must have some way to tell the system of requested timing data since this can significantly change the behavior of the system. Not only does the behavior change upon the rate edit but the timer needs to change at times as well. Another similar responsibility arises which is how to deal with user inputs. This is a dynamic mechanism too and therefore must be treated well with logical conditions, such as no more that two seconds apart between inputs or only two mouse clicks.

Tutor:
Definition: Teach user on past mistakes and lessons
Responsibilities:
- From collected past data create knowledge and generate advice(K)

Motivation: There is a diverse range of users that can use this program. And our program must seek to provide its resources in a clear and understandable fashion for everyone. Therefore a tutoring system with a low learning curve is desirable.

Controller:
Definition: Directs or regulates the requests made from user or another concept.
Responsibilities:
- Access account creation (D)
- Retrieves information from Data Renderer and passes to Website (K)
- Coordinate decisions based on the specific use case (D)
- Passes information to and from Watcher to Data Handler(D)

Motivation: Without the controller, most other concepts fail to perform their tasks. Communication is key between concepts, otherwise how can one know what is going on in different areas of functionality? And one concept should care to know because it affects its own functionality.

Renderer:
Definition: Generates display inputs ultimately for website
Responsibilities:
- Must be able to display text, numbers and graphics for website environment(D)

Motivation: Data and images simply cannot come to website in a quick and easy fashion. There must be a transformation or parsing of "raw" local data that can be manipulated to fit the website environment. The reason for doing this dates to the fact that a web-site is necessary to accommodate a large spectrum of users.

Data Renderer:
Definition: Generates text and non-graphical context for Website
Responsibilities:
- Must be able to display text & numbers for the website environment(D)

Motivation: Input has collected and analyzed, now it is time to give some feedback to the user. Our text or data will in most part reiterate what our models or graphs say. This is crucial since most users need a definitive answer especially when dealing with risk of losing or making money.

Graph Renderer:
Definition: The process of displaying graphs onto the Website
Responsibilities:
- Transfer graphs of worked financial data to Website smoothly(D)

. Generate appropriate images, backgrounds, colors and other graphics(D)

Motivation: Graphs are very important as stated before. But they are useless if they only exist as a ghost or on the local system and not on the web. Time must be invested in discovering how to translate a graph to a website and be placed on a good location next to text or whatsoever, so that a clear and effective feedback is given.

Fortune Teller:

Definition: Generate stock predictions.

Responsibilities:

· Apply prediction algorithms to data(D)

Motivation: Without prediction models this program should not even exist. Prediction is the mind of the program. One prediction is no longer good enough, there is a necessity for a various amounts of models stocks are volatile and one model is not sufficient to establish the confidence of a user interested in investing. Not only is data provided for graphing but also a calculation of past performance that ultimately outputs a confidence level.

Fisher:

Definition: Collects data from Internet Fetch stock prices and re by querying the PriceProvider.

Responsibilities:

· Retains momentary stock data from external websites and passes to Data Handler(D)

Motivation: Without water there is no life, just as without financial stock data there is no program. If prediction is the mind of the program then data collection is the heart of the program. A simple task but with a grave responsibility. There can be nothing done without data.

Watcher:

Definition: A tool for monitoring user's stocks of interest.

Responsibilities:

. Pass user decisions to Gatekeeper

· Place respective priority weights on registered user's stock (K)

Motivation: The people closest to us are the ones that care the most. So to show the user that the system is available to help every step of the way in their grandiose adventure a mechanism to record all activities that can help focus investors on favorite stocks and most profitable stocks.

System Database:

Definition: An organized collection of data used by the system

Responsibilities:

· Store timers (K)

· Store invalid searches (K)

Motivation: A database is crucial to primary operations. In other words memory is important. Memory that holds system characteristics such as a prediction timer constant, update timer constant, and system error prompting.

Stock Database:
Definition: An organized collection of stock data
Responsibilities:
- Store information about the stock market (K)

Motivation: Memory must be granted to stocks profiles collected. Since it is a waste of effort and time to continuously collect data from an external source. It is better to have it saved and recall it when needed.

User Database:
Definition: An organized collection of user data
Responsibilities:
- Store information about the users (K)
- Store the users preferences (K)

Motivation: What good is a program if it cannot show the user that we care enough to have you part of the family? Once registered key information and past actions can be utilized to only help in future stock forecasting. Plus why would anyone desire to be a stranger to a familiar scenario or place (that is assuming they visit it many times)?

Addresser:
Definition: Handles Administrator communication with the system for a diverse range of messages
Responsibilities:
- Notify of user's comments and questions(D)
- Notify of any system problems or errors(D)

Motivation: Sometimes errors occur throughout the system, and code has been placed to catch such errors. However that is not enough. To implement a system that can provide error checking and fixing is one that requires communicating that error to the administrator, so that he/she can forward that message to developers who can spend time to fix them.

Authenticator:
Definition: A way to prove to a computer system that you really are who you are.
Responsibilities:
- Determine the validity of a logon request (K)
- Provide help or tips to a user attempting a logon (D)
- Terminate malicious threats (D)

Motivation: To provide security, users must be authenticated before allowed access to various resources of the program. It would be a terrible mistake to allow the public access to account holders' stock data. Not only would this violate many ethical principles but it simply is not common sense.

AccountCreator:
Definition: Create new users.

Responsibilities:

- Obtain information and update user database(D)

Motivation: There must be a place to start for everyone. This concept is built for users who trust and/or are interested in the services the program can provide.

DataHandler:

Definition: Handle data transfer between concepts and databases.

- Obtain information from and update user database(D)

5.2. Traceability Matrix

	PW	Web	entr.	tutor	user mess.	controller	addresser	fortune teller	Watcher.	Gatekeeper	Fisher.	Authenticator	Graphical Renderer	Data Renderer	Creator	Display Pages	System Database	Stock Database	UserDatabase	Data Handler
1	3	x	x			x						x							x	x
2	5		x															x		x
3	2									x							x			x
4	2									x							x			x
5	5		x															x		x
6	6	x	x															x		x
7	4	x	x	x		x		x					x					x		x
8	3	x	x		x	x			x					x		x		x	x	x
9	3	x	x																x	x
10	3	x	x												x				x	x
11	3	x	x			x						x							x	x
12	5									x	x									
13	14									x								x		x
14	2	x		x																
15	3					x	x													

5.3. Associations

CONCEPT PAIR	ASSOCAITION DESCRIPTION	ASSOCIATION NAME
Controller<->Authenticator	Controller obtains key information from Authenticator Motivation: Controller needs to verify user before he is given certain privileges	Obtain

CONCEPT PAIR	ASSOCAITION DESCRIPTION	ASSOCIATION NAME
Controller<->User Messager	Controller passes information for Messenger to display, and vice versa Motivation: There needs to be an external communication in case user cannot be reached through website	Provides data
Data Handler <-> Stock Database	Data Handler retrieves valid stock information	Conveys requests
AccountDatabase <-> Account	Account database stores user Account data	Conveys requests
Data Handler <->System Database	Data Handler retrieves valid system information	Conveys requests
Watcher<-> Authenticator	Watcher will request information to Authenticator of which user's to watch Motivation: Watcher needs to know which user it should pay attention to	Requests notify
Website<->Controller	Website passes user's inputs to Controller Motivation: Data inputs need to be processed by other concepts otherwise there is no interaction	Pass Information
Renderer<->Display Page	Motivation: Display context is crucial to Website otherwise user cannot receive important feedbac	Prepare
Controller<->Addresse	Motivation: User administrator also needs feedback	Provides dat
Authenticator<->Data Handle	Motivation: Authenticator needs data from database therefore data handler is contacte	Conveys request
Website<->Creato	Motivation: New accounts are desired since users want profile and stock information to be logged and processed	Provide
Creator<->Data Handle	Motivation: New user information must be placed in Database therefore memory must be allocate	Requests memor
Controller<->Watche	Motivation: Watcher must know what information current user has inputted however it must first receive validation from gatekeepe	Passe
Watcher<->Gatekeepe	Motivation: Not all of user inputs may be valid therefore gatekeeper keeps validates user inputs sends information to watcher	Provides dat

CONCEPT PAIR	ASSOCAITION DESCRIPTION	ASSOCIATION NAME
Gatekeeper<->DataHandle	Motivation: System information needs to be updated if desire	Provides dat
Gatekeeper<->Fishe	Motivation: Information must be queried at certain desired times	Notifier
Fisher<->Data Handle	Motivation: Data must be updated at certain times for new analysi	Requests & update
FortuneTeller<->Data Handle	Motivation: Analysis is required of data otherwise there is no user feedbac	requests & obtain
Controller<->Rendere	Motivation: Controller provides new data that need to be processed for website upload	Provide

5.4. Attributes

Tutor has MaxSuggestions. This is to target visitors, since once he/she stops receiving feedback, it will be an incentive to create an account.

Data Handler has MaxNumofEntries which all the databases share. Memory is not free and abundant, therefore this constant that needs to be closely followed. If it is passing a threshold it should generate an alert that will eventually reach Addresser.

Website has status. This means the website might be in maintenance, online, or it is simply shut downed.

Controller has MaxNumofAttempts. This is for validation purposes, a virus or a malicious user wants to gain access. This is simply a constant for blocking a user. It can also report to addresser when such events were happening.

Total system attributes such as timer intervals will be stored in the system database.

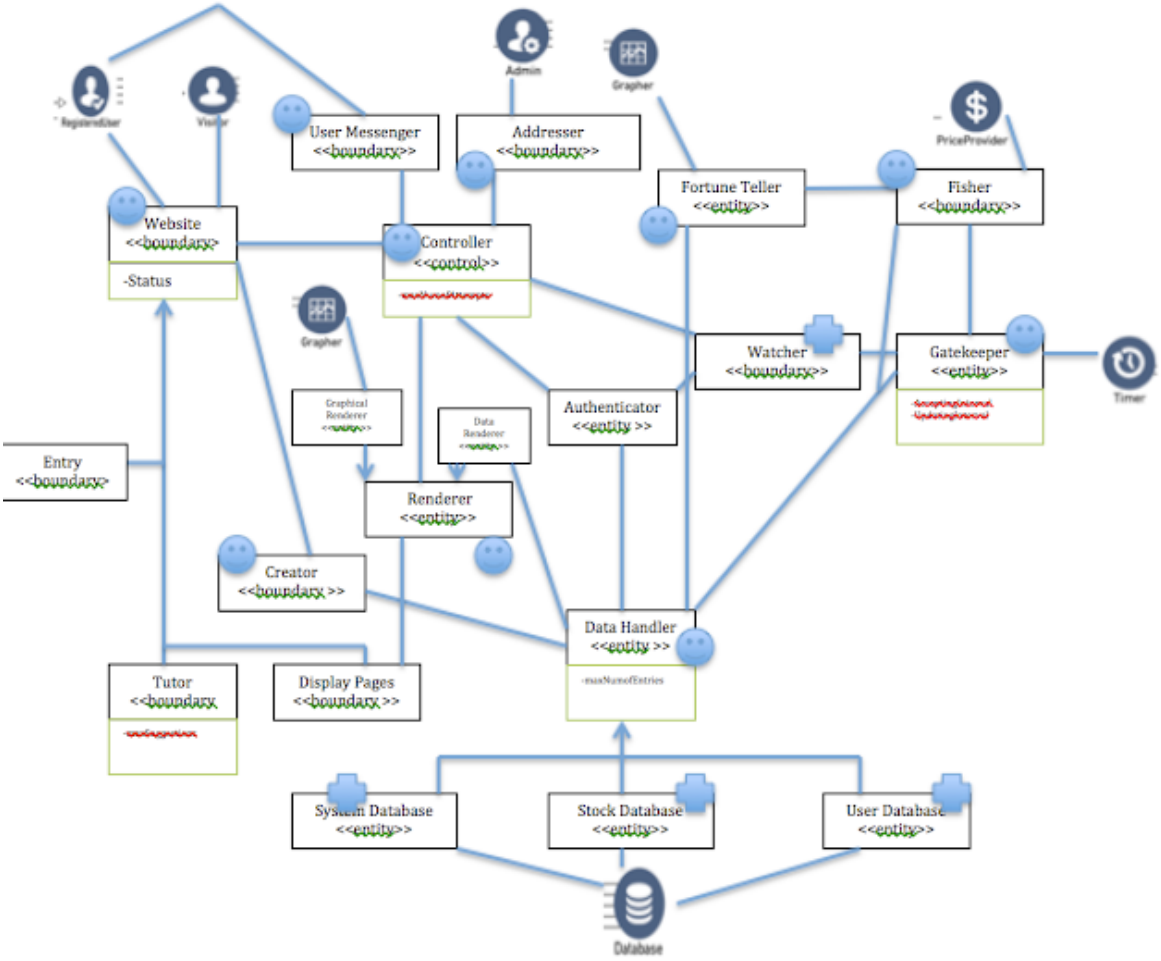


Figure 17: Domain model diagram.

5.5. System Operation Contracts:

1) Name: **AddStock**
Responsibilities: Allows user to add a stock to database that doesn't already exist and then stock predictions for this specific stock will be notified to the user's account.

Cross References: User Cases(1,2,10,13). Login, Search, Register, Predict and Notify

Expectations: If user adds stock then user will get predictions about stock, user must initially have an account for them to add a stock to view predictions.

Preconditions: Must be connected to the internet, must have a profile.

Postconditions: Stock predictions are displayed and notified to the specific user account.

2) Name: **Search**
Responsibilities: Takes users inputted stock name, user name or other means of filtering to a specific stock and match it to the database and retrieve the data and predictions for that stock.

Cross references: User cases: (1,2,10)Login, AddStock, Register

Expectations: Stock name must be found in the database or added to database if it doesnt exist.

Preconditions: Access to internet is required to search for stocks online, User must also have an account to have access to search for a stock.

Postconditions: A stock that the user searches for is found and its previous data as well as future predictions are given to the user.

3) Name: **Track**
Responsibilities: Add a specific stock to a “wish list” so that the stock information is provided on the user’s profile for easy access.

Cross references: User cases: (1,2,6,10)Login, AddStock, Search, Register

Expectations: For access to a wish list, the user must first have an account..

Preconditions: The stock must first be existent in the database, if not then user must add stock to the database.

Postconditions: The stock that the user wants to track will be displayed on his wish list and with easy access to the stock information such as data and predictions.

4) Name: **Update**
Responsibilities: Allows timer to initiate the loading of current stock prices from the price-provider into the database.

Cross references: User cases: (3,4)EditPredictionTimer, EditUpdateTimer

Expectations: The stock data from the price-provider must be of easy access so that the website is updated in a timely manner so that the users can conveniently view and edit their stock information.

Preconditions: There needs to be a link from the price-provider to the website. Also algorithms that are used to predict the stock changes in the future must also be updated constantly so that accurate information is given to the users.

Postconditions: When a user goes to website to view their stocks data or information about the future predictions, the information will be accurate and will be displayed with the data refreshing and updating.

5) Name: **Predict and Notify**
Responsibilities: Allows timer to initiate the prediction of future stock prices
to be later stored into the database. Registered Users are then notified if an change in
prediction is made for a stock he/she has chosen to track.

Cross references: User cases: (8,12)Track, Update

Expectations:Predictions for stocks are made in a certain time period, and everytime the new stock predictions are made through algorithms and other ways, then it is displayed to user. So user must have internet access, for constant updates as well as an account for the information to be displayed into.

Preconditions: The algorithms and link between the price-provider and websites database must be accurate. The users also must have selected which stocks they want to get predictions from by using the wish list.

Postconditions: The predictions are made using the required material and then displayed to the user. So the stocks that the user selects in the wish list will be updated and provided for the user to view.

5.6. Mathematical Models

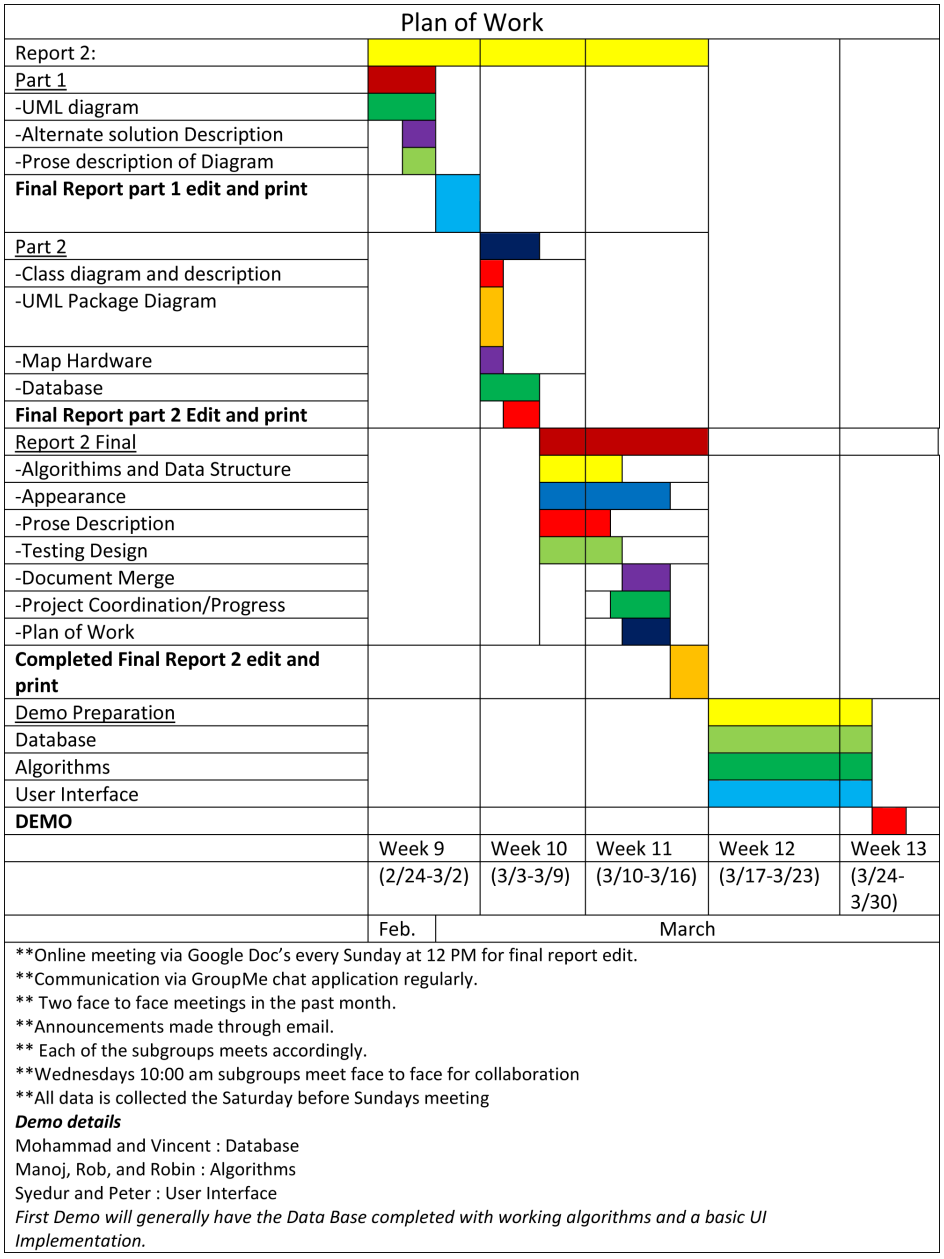
ARIMA time series analysis - auto regressive integrated moving average. The basic principle of this is that with each collected data point the average changes. The average is what predicts the future.

Exponential Moving Average (EMA) : More weight is given to recent values though not all older data sets are not discarded.

Rate of Change (ROC) : Current prices are ratioed with the price n days aways. n is usually 5 to 10 days.

Relative Strength Index (RSI): Check the magnitude of the upward trends against the downward trends within a specific time interval (usually 9 – 14 days).

6. Plan of Work



**Online meeting via Google Doc's every Sunday at 12 PM for final report edit.
**Communication via GroupMe chat application regularly.
** Two face to face meetings in the past month.
**Announcements made through email.
** Each of the subgroups meets accordingly.
**Wednesdays 10:00 am subgroups meet face to face for collaboration
**All data is collected the Saturday before Sundays meeting

Demo details

Mohammad and Vincent : Database
Manoj, Rob, and Robin : Algorithms
Syedur and Peter : User Interface

First Demo will generally have the Data Base completed with working algorithms and a basic UI Implementation.

6.1. Product ownership:

Our team has decided that the most efficient way to handle the functionality of

our proposed software is to split into sub groups. Since most of the team shares similar expertise and knowledge, the skills of each member will be the determining factor of when and how work may be allocated to another team member. This is to say that there will be adherence of product capabilities and awareness among multiple members of the team in the event that the customer seeks information about a specific task. That does not mean that the team in entirety will not know details about certain functionality of the software. As our software's capabilities will be tied to one another and work as a whole based on the correlative nature of the elements. Nevertheless, there will be ownership of the different aspects of design to a particular subgroup of the team.

(Vincent Chen and Mohammed Latif) will be responsible for the data mining and storage aspect of this project. Data mining is a term used in computer science for seeking patterns in large sets of data using different techniques. This team will use historical data from a 3rd party supplier such as Yahoo Finance in our database, among other techniques (e.g. collected data will be received and overwritten into our database at frequent intervals). This team will be solely responsible for the integration of the data collection and storage in a 2 week span implementation.

(Robert Adrion, Robin Karmakar, and Manoj Velagaleti) will take responsibility for the prediction algorithms given the completed database of historical values and any other information attained in the database. This team will also be responsible for researching the most viable machine learning technique that can be implemented via software. In these regards, this generally refers to A.I. (Artificial Intelligence) algorithms that can and will learn from the data in our predefined database. This team will also handle the research and implementation of as many secondary technical indicators (algorithms for trend and pattern detection) as possible.

(Peter Zhang and Syedur Rahman) will design and implement the user interface and graphical functions. The display will provide a very organized, user-friendly and modern experience for the user. This team will query the database for relevant information and show them to the user. User accounts will also be handled by this team as well as any specific user related storage such as the "watch list". This includes all the normal management functions an account has associated with it, such as account creation, deletion and password implementation. A tutorial system for new users can also be used.

7. References

7.1.1. Useful Information

1. Software Engineering by Ivan Marsic, Rutgers University
http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf

2. Background knowledge of stock market prediction:
http://en.wikipedia.org/wiki/Stock_market_prediction

3. Useful definitions:
http://www.investopedia.com

4. Information on internet speeds and page loading times:
http://www.akamai.com/dl/akamai/akamai-soti-q313.pdf

5. Information on password reuse statistics:
http://landing2.trusteer.com/sites/default/files/cross-logins-advisory.pdf

6. Listed companies within the World Federation of Exchanges as of January 2014:
http://www.world-exchanges.org/statistics/monthly-reports

7. Mathematical Models:
http://www.vatsals.com/Essays/MachineLearningTechniquesforStockPrediction.pdf
http://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

7.1.2. Images