ECE 16:332:568
Software Engineering of Web Applications
Spring 2017

PredStocks

One Stop Solution To Smart Prediction!

GROUP 8
Anish Grover
Ajinkya Padwad
Chaitanya Kulkarni
Lakshmi Kumar
Sowmya Muralidharan
Tejas Rajput

Instructor
Prof. Shiyu Zhou

Rutgers University
Department of Electrical and Computer Engineering

Responsibility Chart

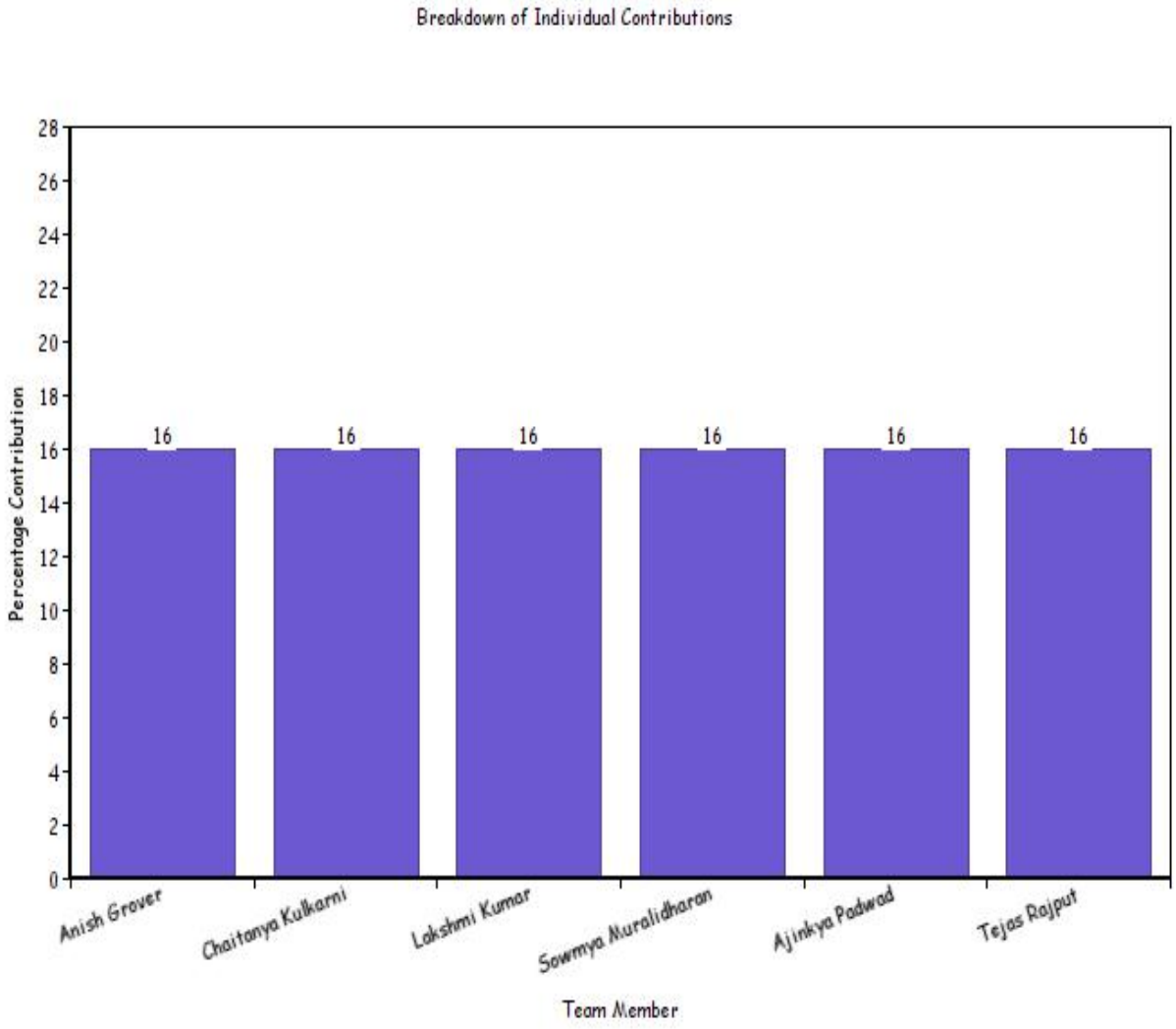


Table of Contents

1. Customer Statement of Requirements:

Stocks are arguably the most popular financial instrument invented for building wealth and are the centerpiece of any investment portfolio. The advances in trading technology has opened up the markets so that nowadays nearly anybody can own stocks. There has thus been an exponential increase in the average person's interest in the stock market over the past few decades.

Stock prices aren't fixed and they tend to rise and fall based on free market forces. It is these ever-shifting market forces that make short-term movements of the stock market so difficult to predict and consequently make short-term stock market investing risky. It therefore becomes important to predict the rise and fall of the stock market, which shares will hit big and when to buy/sell.

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit.

The two main prediction methodologies in the financial market are technical analysis and fundamental analysis. Fundamental analysis attempts to determine a stock's value by focusing on underlying factors that affect a company's actual business and its future prospects. Technical analysis, on the other hand, looks at the price movement of a stock and uses this data to predict its future price movements.

Technical Analysis

It is a method of evaluating securities by analyzing the statistics generated by market activity, such as past prices and volume. It is based on the following three assumptions:

1. The market discounts everything.
2. Price moves in trends.
3. History tends to repeat itself.

1. The Market Discounts Everything

Technical analysis assumes that a stock's price reflects the company's fundamentals, along with broader economic factors and market psychology and therefore removing the need to actually consider these factors separately.

2. Price Moves in Trends

In technical analysis it is assumed that once a trend is established, the price movements in the future, would most likely follow the same trend i.e. be in the same direction.

3. History Tends to Repeat Itself

Technical analysts believe that history tends to repeat itself in terms of price movement, because of market psychology, and analyze these market movements and understand trends using chart patterns.

These assumptions still hold good, in the light of price movements in the market and help in prediction and forecasting. Technical analysis is therefore widely used among traders and financial professionals and is very often used by active day traders, market makers and pit traders.

There are several techniques out there to predict market prices which make use of chart patterns for better visualization and comprehension. Some popular chart patterns include flag and pennants, cup and handle, head and shoulder, support and resistance, trends and channels etc. The technical indicator is a series of data points that are derived by applying a formula to the price data of a security. There are number of well-known indicators moving average, stochastic, momentum, average directional index etc.

1.1 Problem Statement

Investment in the stock market has become much more accessible to the average man, with the widespread use of the Internet in trading. This has made the prediction and forecasting of stock prices essential. As discussed in the previous section, both fundamental and technical analysis are widely used by the various proponents of stock forecasting for maximizing profits. Our focus is on the technical analysis.

Technical analysis deals with the prediction of stock values based on previous stock performance and volume of trade done. This theory is based on the idea that previous stock performance is clear indicator of a company’s performance and analyzing it only provides greater insight which would help the investor/trader make informed decisions.

This process depends on the short-term analysis of the price and volume and is therefore not concerned with long term research into a company's performance. Proponents of the Technical analysis believe that various trends exist in the nature and behavior of stocks and identifying these behaviors would lead to an improved and better way of predicting performance.

Investors may be categorized as short term investors or traders and long term investors. The short-term traders believe in buying and selling stocks for short-term benefits and requirements and thus, they are called the traders. The long-term stock investors invest in the companies for a long-term and thus are called investors. They profit from stock appreciation. Short term traders, on the other hand look forward to buying and selling stocks every now and then in an effort to earn profit by buying low and selling high.

The software designed will provide the customer the ability to predict stocks. The customer must be able to enter the system and choose the stock for which he wants predicted values. The customer must be able to login to the system to save his favourite stocks which he wishes to view frequently. The customer should get a line graph for the predicted values of stock.

2. Glossary of Terms

Artificial Neural Network (ANN): A computer system modelled on the human brain and nervous system. It interconnects systems of neurons that can calculate values for inputs by feeding information through the network.

State Vector Machine (SVM): Supervised learning models with associated learning algorithms that analyse data for classification and regression analysis. It is a non-probabilistic binary linear classifier.

Bayesian Prediction: It is an approach to linear regression in which the statistical analysis is undertaken within the context of Bayesian inference.

Simple Moving Average: It is an indicator calculated by adding closing prices over a time period and then dividing that total by the time periods. Traders can watch how short-term and long-term averages relate to one another and thus analyse trends.

Relative Strength Index: It is a technical momentum indicator - magnitude of recent changes in gain or loss RSI Scale of 100, where 70 and beyond is overbought and 30 and below is oversold.

Back Propagation Network: A common method of training ANN. The algorithm repeats in two phases - propagation and weight update.

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.

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.

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

Opening Price - The price at which a security first trades upon the opening of an exchange on a giving trading day.

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.

Volume: The amount of shares traded during a particular period of time.

3. System Requirements:

3.1 Enumerated Functional Requirements

The various attributes of the system to be implemented as a part of the functional requirements are as follows:

ID	Priority weight	Requirement
REQ 1	5	The system shall allow the user to select a company to predict the stock prices
REQ 2	5	The system shall allow the user to select the duration of prediction.
REQ 3	5	The system shall predict the stock price and indicate a decision of buy/sell/hold to the user.
REQ 4	4	The system should acquire data from Yahoo Finance and needs to be stored in a database.
REQ 5	5	The system should be able to acquire both historical and real-time data.
REQ 6	5	The program will output the stock prices as a graph of price versus date.
REQ 7	4	Understand and implement the indicators for the stock price trends.
REQ 8	4	The stock data stored in the database will be used by the prediction algorithm to calculate the stock's price.
REQ 9	3	The program should have a data structure (portfolio) that holds all the stock tickers that the user is interested in.
REQ 10	3	The website should display the current news articles from reputable sources.
REQ 11	5	Display a list of all companies in the database along with their latest stock price
REQ 12	5	Display the highest stock price of any company in the last ten days.
REQ 13	5	Display the average stock price of any company in the latest one year.
REQ 14	5	Display the lowest stock price for any company in the latest one year.
REQ 15	5	Display the list the ids of companies along with their name which have the average stock price lesser than the lowest of any of the Selected Company in the latest one year.

The major requirements is to acquire the data from Yahoo Finance, both for historical and real-time points which is given as REQ 1 and 2. This should be easily available for the system to use later for the prediction of stock price. The system needs to collect data within the specified duration as specified in REQ 6 to avoid latency errors. If the system doesn't follow the specified time, data points could be lost in real-time.

3.2 Enumerated Non-Functional Requirements

Adhering to the FURPS+ model developed by Hewlett-Packard we intend to qualify software attributes. FURPS entails key attributes such as functionality, usability, reliability, performance and supportability. The + symbol signifies other possible attributes that can be added.

Functionality: The main motivation behind this project is to predict an optimum air fare taking into account several influential factors. The degree to which the software would be able to do this will qualify as the functionality of the system.

Usability: A minimum set of inputs is indispensable for the functioning of the system. A sincere attempt will be made to keep the number of operations to be done by the user at a minimum.

Reliability: The goal is to develop a robust software that never fails. In case of an unfortunate incident, utility is made such that the user is presented with a crash recovery system. The option to start the software freshly or to resume the working of the software from the point of crash occurrence can be made by the user.

Performance: The system inherently has as its core, a complex set of calculations that have to be performed in order to output the required results. Regardless of this, the response time of the system should be kept a minimum.

Supportability: A great deal of importance will be given to make the system easy to understand and the user input fields self-explanatory. A text file explaining in detail every functional aspect of the software will be provided to the user via the 'help' button on the application. Further personal assistance can be availed by the user by using the contact details listed under the 'contact us' field on the application.

ID	Priority weight	Requirement
REQ 16	4	The system should store the acquired data as a csv file.
REQ 17	3	The system should be store the historical and real-time data in two different files.
REQ 18	5	Provide customers with predictive data on the stock market to aid their trading decisions.
REQ 19	4	Website will also display normal data on the stock such as the current price and volume.
REQ 20	3	The database will be updated daily to account for the daily changes to stock values.
REQ 21	4	Site will be simple and clean so that the customer can view everything easily.
REQ 22	5	The prediction will use market data and indicators from the database, such as RSI and stochastics, as input to the predictive algorithm.
REQ 23	3	Users shall be able to access Google search from the website

3.3 On Screen Appearance Requirements:

ID	Priority weight	Requirement
REQ 24	3	The system shall show the latest news on the website.
REQ 25	2	The system shall give a description of the website in 'About Us'.
REQ 26	3	The system shall give a 'Contact Us' page for users if they have any queries.
REQ 27	5	Select a Company: This drop-down menu allows the user to select the company for prediction of stock prices
REQ 28	5	Select a Duration: This drop-down menu allows the user to select the duration for prediction of stock prices
REQ 29	3	The system shall display a marquee with the current price trends of popular stocks.
REQ 30	2	The system shall accept a feedback from users.

3.4 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 tests for REQ1:

AT1.1: Check if the Company the user searches for in the Search bar is suggested by the System.

AT1.2: Once the Company is suggested by the system , the User should be able to select it.

Acceptance tests for REQ2:

AT2.1: Check if the user is able to select a prediction duration from the list that appears.

Acceptance tests for REQ3:

AT3.1: 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 and REQ5:

AT4.1: Refresh the databases in XAMPP to see if the stock data is being updated periodically as desired.

AT4.2: Check if the stock information on our database matches the information on the Yahoo Finance website.

Acceptance tests for REQ6:

AT6.1: Clicking on the submit button should display the predicted price along with the graph. 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.

4. Functional Requirements Specification

4.1 Stakeholders

The stakeholders of our system are:

User: The users are individual investors who don't have the tools of an institutional investor, and will benefit from our web application. These investors have a broad range of assets and can either invest in the short or long term. The individual investor does not use investing as their primary source of income.

Administrator: Maintains and updates website services.

4.2 Actors and Goals

User: The user will take the form of an investor that registers and uses the website

Administrator: The manager that is in charge or keeping the system updated and in working order

Prediction Algorithm: The algorithm(s) that will calculate the prediction.

Database: The database will hold all the user data and information as well as all the stock information. All the user information and their portfolios will be stored in the database.

Price Provider (Yahoo! Finance API): The API is where we will pool all stock data from to store into the database to generate the prediction.

Graph: Will be used to graphically plot the stock data.

4.3 Use Cases:

4.3.1 Casual Description:

UC-1 Search For Stock - (REQ1, REQ2, REQ18):

Allows a user to search for a stock they want prediction for.

UC-2 Build Database- (REQ4, REQ5, REQ11, REQ12, REQ15):

In order to obtain a prediction and generate a graph, we will need to retrieve stock and market data from the Yahoo! Finance API and store it into the database. This is done in the background. If the stock is not already in the database, an algorithm will query Yahoo! Finance for the requested data.

UC-3 Data Acquisition - (REQ4, REQ5, REQ6)

In order to obtain a prediction and generate a graph, we will need to retrieve stock and market data from the Yahoo! Finance API and store it into the database. This is done in the background. If the stock is not already in the database, an algorithm will query Yahoo! Finance for the requested data.

UC-4 Obtain Prediction - (REQ1, REQ2, REQ8, REQ9):

Obtains the input parameters for the prediction algorithm to calculate the prediction. Once the prediction is calculated, a graph will then be generated with both the stock's historical data and prediction.

UC-5 Calculate Price - (REQ3, REQ6, REQ7, REQ13):

From the input parameters, the prediction algorithm will calculate the stock's prediction**.**

UC-6 Display News - (REQ10, REQ14, REQ19):

Users will have access to current events and news regarding different companies directly on our site.

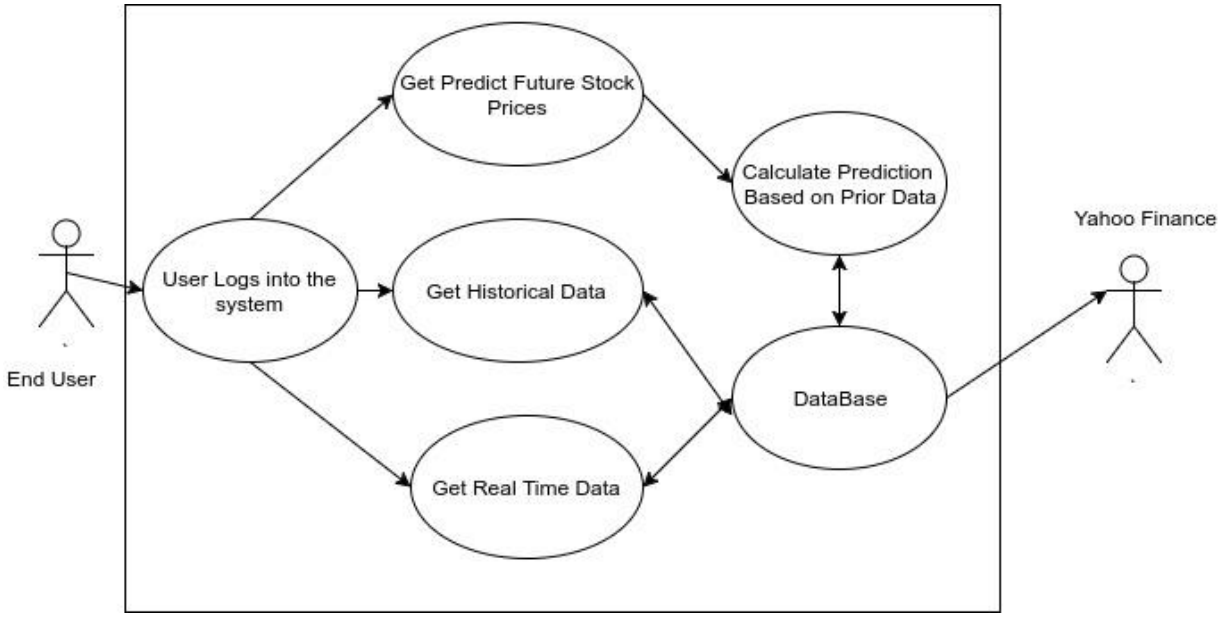
UC-7 Display Webpage - (REQ-12, REQ-13, REQ-14, REQ-15)

Once all the information has been obtained, the system will display the requested information on the stock webpage

UC-8 Get Feedback - (REQ-16)

Once the user is done using the webiste, he will be redirected to a feedback form page, so that he can input his views and help us creating a better user environment.

4.3.2 Use Case Diagram



4.3.3 Traceability Matrix:

Use Cases	Req 1	Req 2	Req 3	Req 4	Req 5	Req 6	Req 7	Req 8	Req 9	Req 10	Req 11	Req 12	Req 13	Req 14	MAX PW	TOTAL PW
PW	5	5	5	4	5	5	4	4	3	3	5	5	5	5		
uc-1	X	X													5	10
uc-2				X	X						X	X		X	5	24
uc-3				X	X	X									5	14
uc-4	X	X						X	X						5	18
uc-5			X			X	X						X		5	19
uc-6										X				X	5	8
uc-7											X	X	X	X	5	30
uc-8														x	5	5

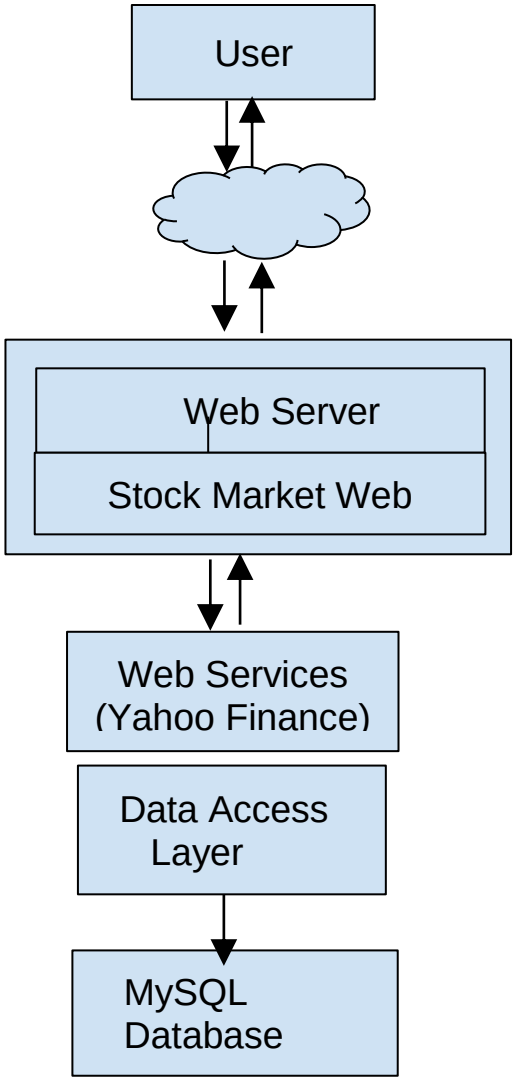
4.3.4 System Sequence Diagram

1. User messages interface (application) to get stock information from web (Yahoo Finance) and keep storing it in the database. Interface messages web to get information of particular stock and forward the information to database. In reply

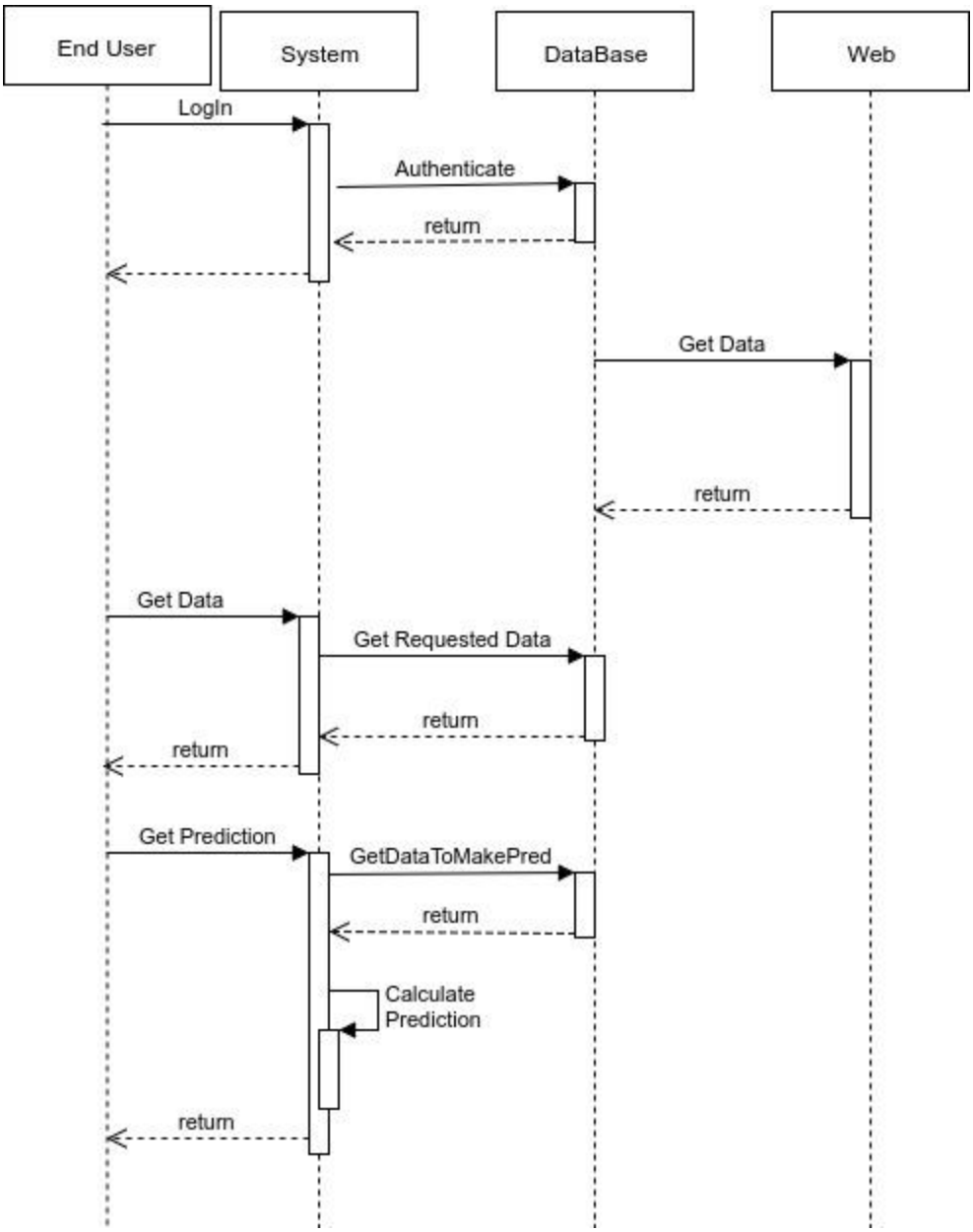
web sends the desired information to database where it gets stored for future use. This process continues in the background as long as application is open.

2. User messages interface to get stock data of a particular stock. Interface messages database and in reply database sends the desired information to interface which is then displayed by interface to user.

3. Background process keeps on taking data from the web to keep its table updated. We get real time data through this process



System Level Interaction Diagram


End User
System
DataBase
Web
LogIn
Authenticate
return
Get Data
return
Get Data
Get Requested Data
return
return
Get Prediction
GetDataToMakePred
return
Calculate
Prediction
return

4.4 Fully Dressed Descriptions of Use Cases

4.4.1 Obtain Prediction:

Related Requirements	REQ-1, REQ-2, REQ-8, REQ-9
Initiating Actors	User
Actors Goals	Obtain the Predicted values out on the Screen for the user.
Participating Actors	Database, Predicting Algorithms
Precondition	The data of the stock should be present in the database.
Success end condition	The stocks page is loaded with the predicted values.
Fail end condition	Prediction could not be displayed because no data was present in the databse.
Flow of events for success scenario	-> User searches for a stock. <- System searches for a stock in the database. -> The system sends the historical data of the stock to the prediction algorithm. <- The prediction algorithm calculates the predicted value. <- System generates a graph for the prediction. <- The Predicted value and graph is displayed on the screen.

4.4.2 Data Acquisition

Related Requirements	REQ-4, REQ-5, REQ-6
Initiating Actors	User, Database
Actors Goals	Obtain the stock data from yahoo finance API.
Participating Actors	Yahoo Finance API.
Precondition	Stock is not present in the database.
Success end condition	The data of the stock was stored in the database.
Failed end condition	The stock was not present in the database, hence the database was not updated.
Flow of events for success main scenario	-> User searches for a stock. -> System searches for the stock in the database If stock not found : -> Database sends a query to yahoo finance requesting the data for the stock. <- Yahoo finance returns the data for the stock.
Flow of events for extensions	If the stock was not found in the Yahoo Finance API : <- Database generates an error message and sends it to the system. <-The error message “Stock not found” is displayed to the user.

5. Effort Estimation

5.1 Preliminary Design

The user interface of PredStocks emphasizes easy to understand graphical representations of financial metrics pertaining to various aspects of trading and the economy in general. In addition, color harmony and adequate space distribution is a priority to provide a pleasant user experience. A UI design built on top of the responsive Bootstrap UI framework was chosen due to its extensive support for many UI components needed in the application. Each primary view is presented below with particular attention having been put into a consistent and uniform user experience. Each view is annotated with applicable use cases so that a sequence of views can be determined for each use case.

5.2 User Effort Estimation

The goal of designing our system was to create a clean and simple to use interface, clearly labeling each feature of the site on the homepage to avoid navigation and provide clarity. The following user effort estimations take the assumption that the user is already logged in and sitting on the home page.

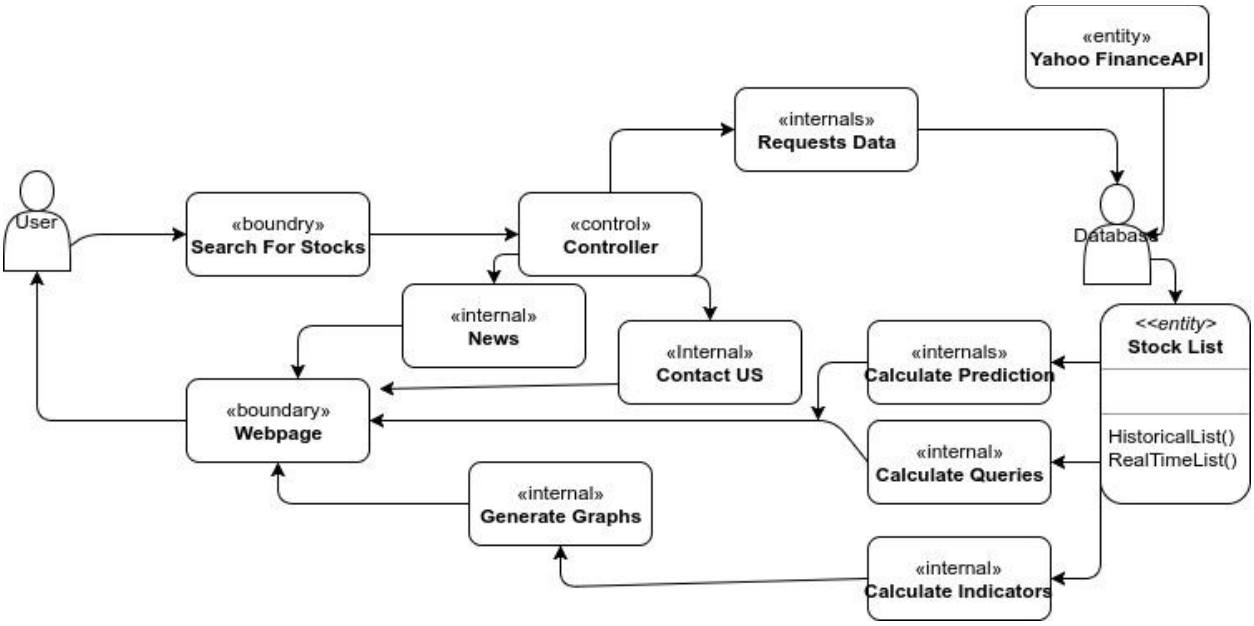
1. Search for a stock prediction/information -
    a. Navigate to the Prediction Page from Home - 1 mouse click
    b. 1 mouse-click on the search field on prediction page
    c. Enter the stock ticker (minimum 2 keystrokes, average would be 3-4)
    d. Hit enter (1 keystroke)
2. Check News Information - (1 mouse click) - Click respective link on navigation bar
3. Check Contact Information
    a. Click respective link on navigation bar - 1 mouse click
    b. Enter feedback - average 10 keystrokes , 1 mouse click to submit
4. Check Home Page - (1 mouse click) - Click respective link on navigation bar

Duration = UCP * 21

UCP = UUCP * TCF * ECF

6. Domain Analysis

In order to build the domain model, we will need to once again analyze the primary use cases. From there we will be able to derive the important concepts of the system. We will first look at the boundary concepts that directly interact with the actors and then afterwards analyze and find the internal concepts of the system.



6.1 Concept Definitions

Table 7-1 will contain all the boundary and internal concept definitions that we obtained from analyzing each actor's responsibility.

6.1.1 Boundary Concepts

In order to analyze the boundary concepts, we must look at all the actors and how they interact with the system. The system interacts with five actors. Three of these actors handle the back end of the system and their primary function is to contain the data and information that the website application needs to output to the human actors, the investors and the administrator. We will first describe and break down responsibilities for each actor.

Let's first look at how the database interacts with the system to create a list of responsibilities and concepts. The database's primary purpose is to store stock information

R1 – Collect stock's historical data (Data Aquisition)

R2 – Generate graph of historical data and prediction (Create Graph)

R3 – Search the system for a stock. (Search For Stock)

After we have obtained the data from the Yahoo! Finance API, we will need to process this data

R4 – Maintain a record of collected data (Stock Database)

R5 – Validates that searched stock is in Stock Database (Find Stock)

R6 – Process numerical data for a given stock (Process Stock Data)

R7 – Calculate a prediction (Calculate Prediction)

R8 – Maintain a record of general stock information (StockInfo)

R9 – Maintain a record of news articles and twitter feeds (News)

R10 – Load Stock page (StockPage)

R11 – Load News page (NewsPage)

R12 – Load Home page (HomePage)

R13 – Load FAQ/Help page (FAQPage)

Below is the summary of the concepts

Responsibility Description	**Type**	**Concept**
Contains all the stock data and basic stock information	**K**	Stock Database
Obtains and collects data from Yahoo! Finances API and stores them in the stock database and stores them in the database	**D**	Request Stocks
Search and find the stock that users request	**D**	Search for stocks
It processes the numerical data from Stock Database and calculates a prediction for the stock. Afterwards it generates a graph of all the historical and predictive data	**D**	Obtain Prediction
Loads all the pages and graphs that the website needs for the user to view and navigate	**D**	Webpage

6.2 Traceability Matrix:

Domain Concepts

Use Cases		Stock Database	Request Stocks	Search for stocks	Obtain Prediction	Create Graph	Webpage
	PW						
UC-1	10	x		x			
UC-2	24	x					
UC-3	14		x	x			
UC-4	18				x	x	
UC-5	19				x	x	x
UC-6	8						x
UC-7	30						x
UC-8	5						x
Max Pw		24	14	14	19	19	30
Total Pw		34	14	24	37	37	62

6.3 Association Definitions

Concept Pair	Association Description	Association Name
Search for Stock <-> Stock Database	Finds the requested stock in the Stock Database	Provides requested data
Data Acquisition <-> Stock Database	The stock database collects data from yahoo! Finance API	Provides and Stores data
Stock Database<-> Obtain Prediction	Uses the data collected and stored to calculate the prediction	Calculates prediction
Stock Database <-> Obtain query results	Uses the data collected and stored to calculate the queries	Calculates Queries
Stock Database <->Obtain indicators	Uses the data collected and stored to calculate the indicators	Calculates Indicators

6.4 Attribute Definitions

Attribute Description	Attribute	Concept
String of the given stock ticker symbol or company name	Company name	Search for Stocks
Data Structures / Linked list that contains: String containing whether historical or realtime data Array of historical data String of ticker symbol String of Company name	Stock Data	Stock Database
An array of stocks historical data / real time data	data	Prediction Calculator
An array of historical data	data1	Queries calculator

6.5 System Operation Contracts:

1) Name: Predict and Notify

Responsibilities: Allows timer to initiate the prediction of future stock prices to be later stored into the database.

Expectations: Predictions for stocks are made in a certain time period, and every time 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.

Post conditions: 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.

2) Search

Responsibilities: Takes users inputted stock name, and match it to the database and retrieve the data and predictions for that stock.

Expectations: Stock name must be found in the database or added to database if it doesn't exist.

Preconditions: Access to internet is required to search for stocks,

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

6.6 Mathematical Models

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

ROC = [(Close - Close n periods ago) / (Close n periods ago)] * 100

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

$$RSI = 100 - \frac{100}{1 + RS}$$

Simple Moving Average (SMA): A simple moving average (SMA) is an arithmetic moving average calculated by adding the closing price of the security for a number of time periods and then dividing this total by the number of time periods.

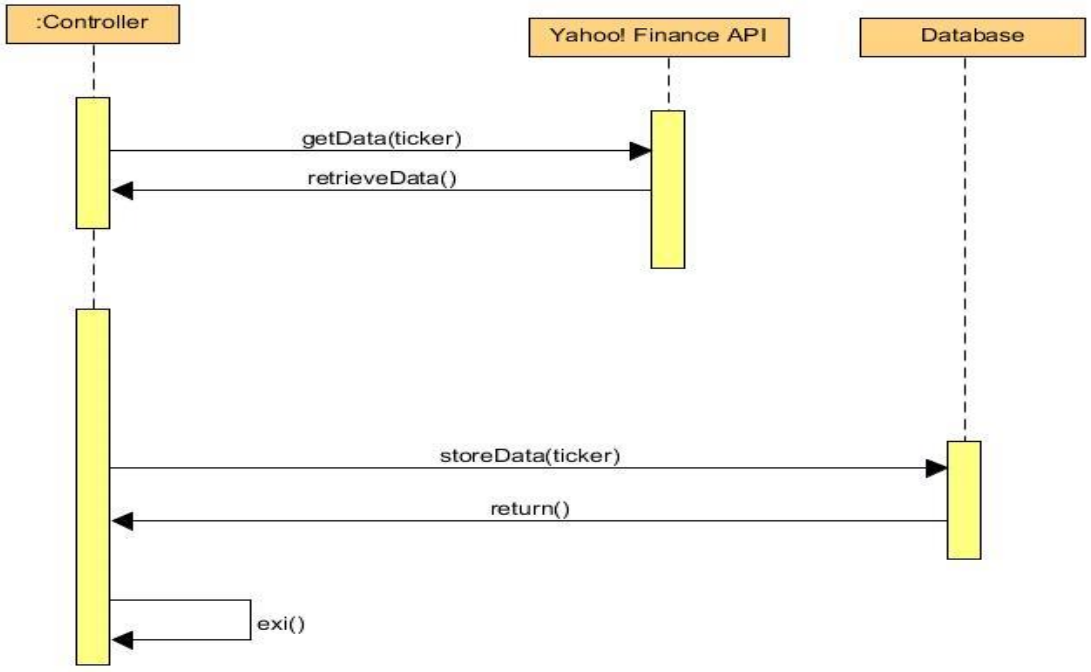
$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n}$$

Pm = *prices of the stock at a certain point*

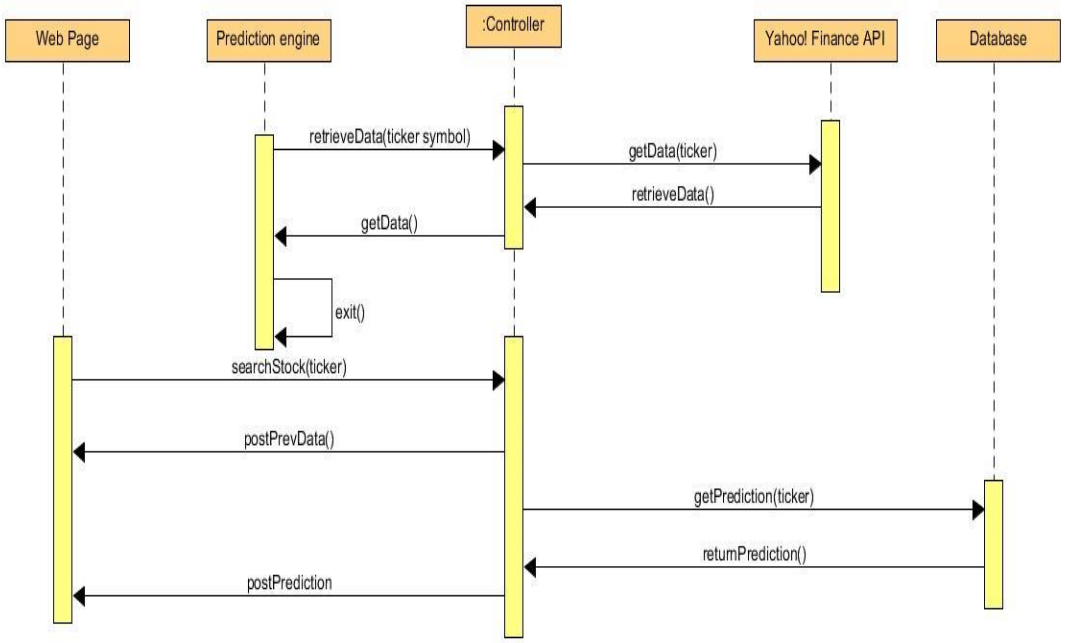
n = *period of days the data is across*

7. Interaction Diagrams

Use Case UC2: Build Database

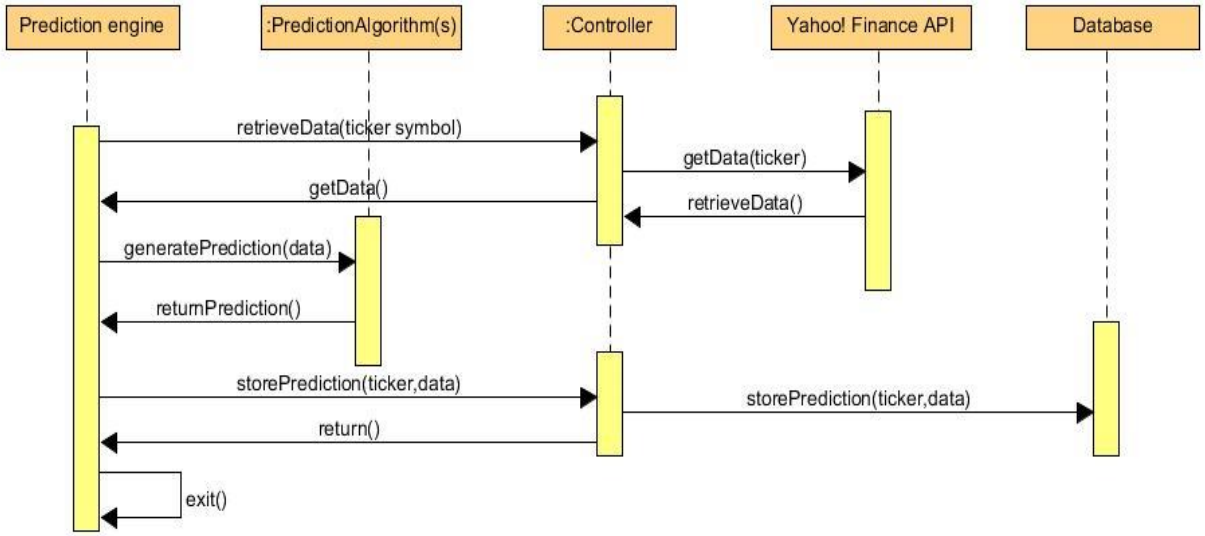


Use Case UC3: Data Acquisition

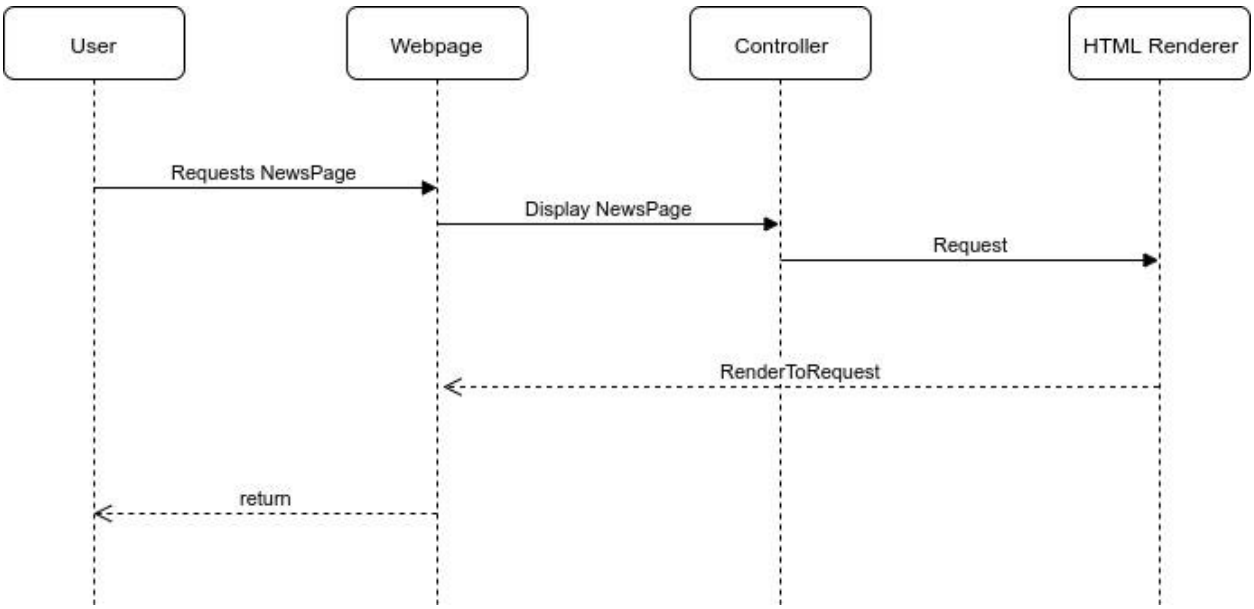


The High Cohesion Principle is exemplified in this diagram since there is a lot of communication between the objects. All the data is being moved from location, to location to be stored. Therefore, there needs to be lots of communication.

Use Case UC4: Obtain Prediction

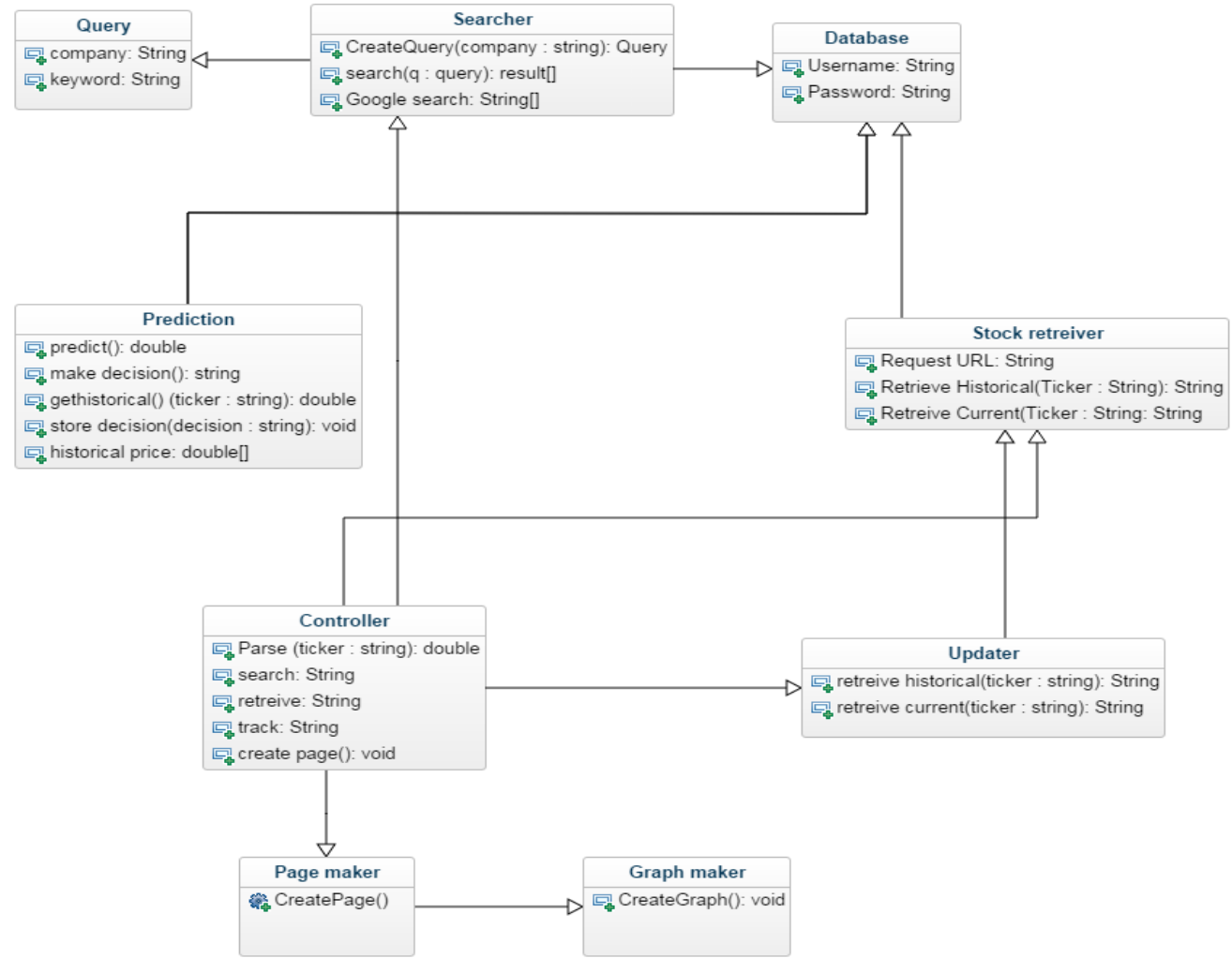


The design principle employed in this diagram is the Expert Doer Principle because there is a decent amount of communication in between the objects, but it is short and focused. Therefore, since the communication chains are shortened between objects, this principle works best for this design.



8. Class Diagram and Interface Specifications:

8.1 Class Diagram



8.2 Data Types And Operation Signatures

The detailed specifications of each individual class is show in in UML notation in Figures 3.2, Figures 3.3, Figures 3.4 and Figure 3.5. The classes are separated into four packages: UserInterface, Database, Prediction, and Webpage. We will explain each class's functions and operations in package order.

8.2.1 User-Interface Package

The classes Login and Register are very self-explanatory and do not need clarification. So we will not go in depth with these two classes because they are not unique to project itself.

1. UserAccess

UserAccess controls when the user is able to access the system. The class will check if the user is currently logged in or logged out and will adjust the access accordingly. In order to achieve this, the class itself will have two parameters. One parameter will be a boolean flag and the other one is simply the username. Every time the user logs into the website (and is properly authenticated), the login function will call UserAccess and set the Boolean flag to 1, making it true. When that subsequent user logs out of the website, the Boolean flag will be set to 0, making it false. UserAccess will actually be linked to the User Database and the Boolean flag will be stored with the user information. This is where the second parameter comes in; it is used to check if the user already has access to the web application (currently logged in) or they haven't logged in yet. UserAccess looks for the username in the Database and checks if the Boolean flag has been asserted. The UserAccess class also handles the login and register requests of the user.

2. Search For Stock

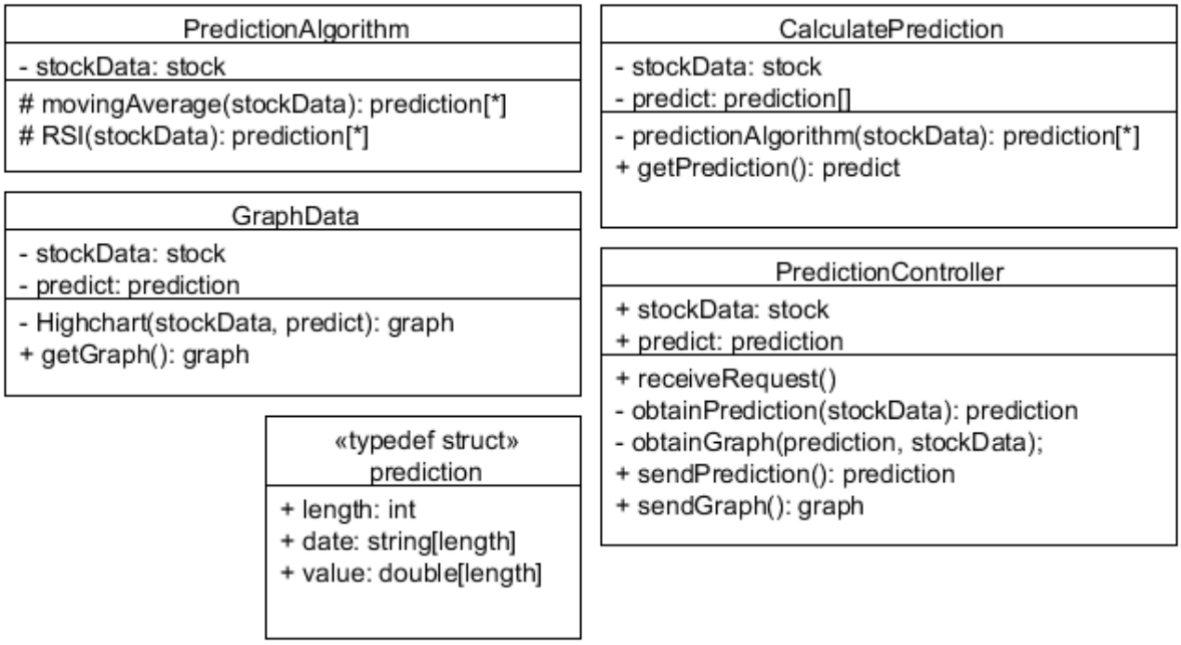
The class is the key feature of our system; it enables users to search the Database for a given stock using either the stock ticker or the actual company name. The class will send this request to the Database and have the stock's information, prediction, and graphs be returned. Once all the information is returned it will send the information to the Controller so that the stock's webpage may be created and loaded for the user.

3. Get News

This class obtains news articles using RSS feeds from reputable websites and Twitter to obtain news about a certain stock.

4. UserController

As the name suggests, this is the controller for the user interface. The controller receives requests to the Database from the functions within the user interface and sends them to the Database. It is essentially the communicator between the interface, the database, and the web-page.



8.2.2 Database Package

The classes and objects in this package store, obtain, and maintain all of the user, portfolio, and stock data. First we will talk about the classes that define objects before delving into the classes that maintain functionality in the Database.

1. **Stock:** This class defines and contains all the information that defines a stock. This information is: stock ticker, company name, a 2D array of the stock's historical data. This two-dimensional array will contain the stock values in a range of dates.
2. **User:** User is a class that is similar to stock in that it defines and contains all the information that defines a user. It will hold the user's username, password, email, and portfolio. The portfolio is a separate class that will be defined below. Therefore the User class contains the Portfolio class.
3. **StockDatabase:** The StockDatabase is similar to the UserDatabase. It is a class that contains a list of all the stocks acquired from the Yahoo! Finance API. It has functions such as findStock, addStock, updateStock. The StockDatabase uses the class DataAquisition to add new stocks and update current stocks in the database.

4. **DataAquisition:** DataAquisition is the class that queries stock information from Yahoo! Finance API. It has 2 operations, aquireStock and queryNewData. The aquireStock operation obtains new stock data form Yahoo! Finance if the StockDatabase does not have that specific stock. The operation queryNewData, is what is used to update and add new stock values into the already stored stocks in the StockDatabase.
5. **DatabaseController:** This class is similar to the UserController and is the controller for the database. It communicates with the UserController and ObtainPrediction to send and receive requests from the functions in the database

8.2.3 Prediction Package

The prediction package holds all of the classes that are needed to obtain a prediction**.**

1. Prediction Controller

This class is the controller for the Prediction package. It receives requests from the database to calculate a prediction and then conveys the request to Calculate Prediction.

2. Calculate Prediction

This class obtains an algorithm from the Algorithm class to calculate the prediction.

3. Prediction Algorithm

Each operation in this class is a specific prediction algorithm.

4. Graph Data:

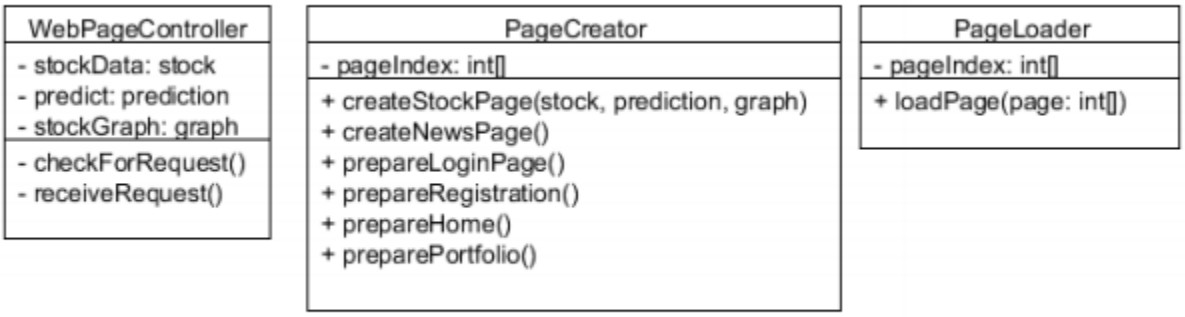
This class calls Highchart to create a graph of the historical data. It also obtains the prediction from Calculate Prediction to include the prediction in the graph.

8.2.4 Webpage Package

The webpage package contains the classes the will ultimately prepare the requested webpage and then load that generated webpage.

1. **PageCreator** This class prepares the webpages that the user requests. It obtains all the parameters that will be on the webpage.
2. **PageLoader** After the webpage has been prepared by the PageCreator class, it tells the PageLoader class to load the prepared webpage.

3. **WebPageControlle**r This class handles all the requests to prepare and load a webpage. It obtains the information to be placed on the webpage and then sends those attributes to the PageCreator.



9. System Architecture and System Design

9.1 Architectural Styles

Our software uses several architectural styles. They follow:

1. Client/Server
2. Event-driven
3. Rule-based system
4. Database-centric

1. Client/Server Architecture

Client/Server is our main architectural style; it separates our system requirements into two easily programmable systems. First, the client, which acts as the User Interface, requests data from the server, and waits for the server's response. Secondly, the server, which authorizes users and processes stock data into information the user can use. It then sends this processed information to the client to display to the user.

2. Event-driven Architecture

Our system will only need to execute its functions after some major state change. It has no real time components like a video game. Instead, we'll have two event emitters, the user and the timer. Both will drive the application to execute relevant operations though the execution of different events. These events include login, adding new stocks, deleting stocks, and requesting an updated lists of stocks for the user; and a time-based update for the timer.

3. Rule-Based System

Our application will be rule-based. In other words, the system will use a set of rules that we determine it to analyze the stock information it gathers. These rules comprise a semantic reasoned which makes decisions for the application and the user. It uses a match cycle act cycle to deduct which stocks will be best to buy and which stocks would be best to sell. Then, it outputs these results to the user-interface.

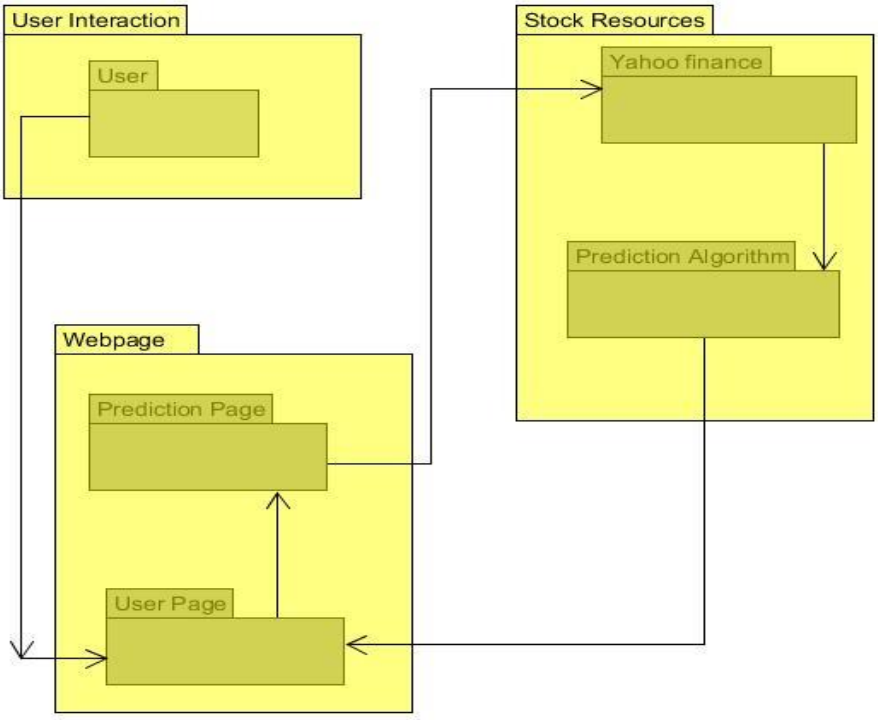
4. Database-centric Architecture

Our system relies heavily on its database, both the store relevant stock data and to analyze the data we give it. The database-centric architecture offers:

1. A standard relational database management system. This means the data will be stored away from the client side application.
2. Dynamic table-driven logic. We need to update the tables every time stock prices change.
3. Stored procedures running on database servers to analyze our data.
4. A shared database for communication between parallel processes

In short, it's a good way of managing a large amount of data

9.2 Identifying Subsystems



Our design will be structured similarly to the MVC model (Model, View, and Control). Model deals with storage and managing data, View is how the user interacts with our product, and Control handles the main algorithm. The subsystems that are shown in the UML package diagram are the User Interaction Subsystem, Data Mining Subsystem and the Data Analysis Subsystem.

The user interaction subsystem deals with the web interface and basic input/outputs and encompasses any class that assists in such matters.

Stock subsystem has a dependency on the Yahoo Finance subsystem. This is because the info in the Yahoo Finance subsystem needs to be used for all of the things the Stock subsystem uses. Highchart also needs this information to generate its charts. It also relies on the Prediction Algorithm subsystem to show the prediction. It holds the responsibility of implementing prediction models on the previously mined data and storing those predictions for later use.

The different tags next to each line are for different reasons. <<import>> is placed where information is shared from system to system publicly, and <<access>> is for when this data transfer is to be private, like when sharing passwords, or other sensitive information.

9.3 Mapping Subsystems to Hardware

Our software employs the use of a client/server system. The client–server model of computing is a dispersed application structure that divides tasks between the provider of a resource or service, called the server, and an entity making a request, called the client. This establishment can be made with a network connection between host (the server) and client or it is possible for this relationship to exist on the same system, sharing hardware.

The web-based stock forecasting design we presume to execute will need to be accessible anywhere that web access exists. This means the client in our client/server relationship will be a web client. A web browser is an example of a web client, and can remotely access requested documentation from the server via HTTP. This web client/server model will need to be run across multiple computers, or subsystems.

A web browser will be used to request the various data from our server, as well as retrieve stock information and updates that can be sent and retrieved via communication with that server. The GUI, which will run on the web browser, will be executed client side while the process itself is handled by the server's web service. The web service will ensure for proper transmission of user data between the client and the server.

9.4 Persistent Data Storage

For the stock price prediction, the system does need to save data that will outlive a single execution of the system. A sample of the data that should be saved is stock tickers and their corresponding company names and stock data.

The data will be stored using a relational database, specifically MySQL. MySQL was chosen for the job due to its open source nature and has all the functions required for the job. Also, several team members have experience with this database and feel that it is the best option. The data will be stored in three separate tables.

The first table have stock ticker and the corresponding company name. The second and third databases store information about historical and real-time stock data from the Yahoo! Finance website in separate tables for each of the companies in the list. The tables hold information about the date, time, opening price, close price and volume traded.

Table 1

Stock

```
Create Table stock_company(
Ticker varchar(100),
Company_Name varchar(255)
);
```

Ticker	**Company_Name**

Table 2

Sample for 1 Company

```
CREATE TABLE Amazon(
date CHAR(40),
price FLOAT,
volume INT
);
```

Date	Price	Volume

9.5 Network Protocol

Simply, our software will communicate with a single main database. This database will use PHP scripts to both send data to our user's localized systems and call data from Yahoo stocks for analysis. The data itself will be managed by SQL software, and the PHP will output HTML to user's systems.

The components will be connected in the following way:

1. PHP requests for raw stock data from Yahoo
2. The data is stored by using SQL
3. Using SQL, we will apply our stock analysis algorithms
4. PHP waits for prompt from user's systems.
5. When prompted, PHP converts analyzed data into HTML
6. User's system converts HTML to working UI

We decided to use PHP because it is standard in creating dynamic web pages. Furthermore, it works well with relational database management systems. We chose SQL because it is the standard RDMS used to manage and manipulate large amounts of data. Lastly, we use HTML because, with the release of HTML 5, HTML has become one of the most powerful and simple markup languages for developing web pages.

9.6 Global Control Flow

Execution Orderliness: The system can generally be defined as event-driven; it will wait for a user to make an action before processing data. The user's interaction will characterize their visit and the control structure will wait for the user's request, remaining idle until it receives such information. This allows for a user to sequence their actions upon a visit in different patterns without confusing the system. Some actions may require multithreading in order for updating to be accomplished thoroughly.

Time dependency: The software will make use of timers to keep current, up-to-date, information regarding stocks in our database at all times. This is a real-time system that will update the database at exact defined times throughout a given day.

Time dependency: The software will make use of timers to keep current, up-to-date, information regarding stocks in our database at all times. This is a real-time system that will update the database at exact defined times throughout a given day.

9.7 Hardware Requirements

There are really three instances that need to be addressed when looking at hardware specification; the hardware to run the server, the hardware to access the website, and what type of hardware is needed to use any supplemental mobile technology.

The server requires a processor that has at least one 1.4 GHz single core (64-bit) or a 1.3 GHz dual-core, 2 Gigabytes of RAM, and has at least 10 gigabytes of hard drive available. The server computer must have networking capability allowing access to a router either via network cable or wirelessly. The router should be an UPnP-certified device; however this is not a requirement.

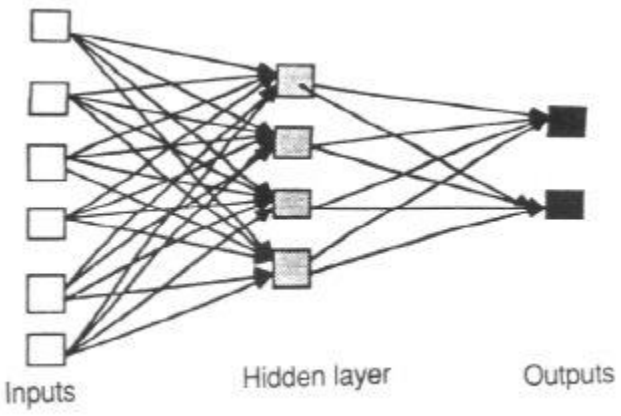
This web application should primarily be used for computers can also be extended to Tablets and/or phones. If there is enough time and resources, the web application would be extended to other devices as well.

10. Algorithms and Data Structures

10.1 Algorithms

10.1.1 Artificial Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process.



These networks are excellent for designing the behavior of more complicated structures because of their ability to learn. A major advantage in using this method for what we seek to accomplish is that they can be used to model a system of events, stock trends, that are totally unknown and how it will react to distortion and interference or in our case the imperfections of the stock market.

The behaviour of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

- linear (or ramp)
- threshold
- sigmoid

For linear units, the output activity is proportional to the total weighted output.

For threshold units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

10.1.2 Bayesian Probability

Bayesian probability is an interpretation of the concept of probability, in which, instead of frequency or propensity of some phenomenon, probability is interpreted as reasonable expectation representing a state of knowledge or as quantification of a personal belief.

The Bayesian interpretation of probability can be seen as an extension of propositional logic that enables reasoning with hypotheses, i.e., the propositions whose truth or falsity is uncertain. In the Bayesian view, a probability is assigned to a hypothesis, whereas under frequentist inference, a hypothesis is typically tested without being assigned a probability.

Bayesian probability belongs to the category of evidential probabilities; to evaluate the probability of a hypothesis, the Bayesian probabilist specifies some prior probability, which is then updated to a posterior probability in the light of new, relevant data (evidence). The Bayesian interpretation provides a standard set of procedures and formulae to perform this calculation.

Our goal is to implement the Bayesian Polynomial Curve Fitting formula to predict the stock price of any company, given a range of historical data, at a time N+1. Having implemented the Bayesian curve fitting in code and using it to predict the Adjusted Stock Price at Closing, it is important to check if the performance of the prediction algorithm is good.

Bishop, in his book on Pattern recognition and Machine Learning, points out that in the maximum likelihood approach, the performance on the training set is not a good indicator of predictive performance on unseen data due to the problem of over-fitting. If data is plentiful, then one approach is simply to use some of the available data to train a range of models, or a given model with a range of values for its complexity parameters, and then to compare them on independent data, sometimes called a validation set, and select the one having the best predictive performance. If the model design is iterated many times using a limited size data set, then some over-fitting to the validation data can occur and so it may be necessary to keep aside a third test set on which the performance of the selected model is finally evaluated.

In the curve fitting problem, we are given the training data x and t, along with a new test point x, and our goal is to predict the value of t. We therefore wish to evaluate the predictive distribution $p(t|x, \mathbf{x}, \mathbf{t})$.

$$p(t|x, \mathbf{x}, \mathbf{t}) = \int p(t|x, \mathrm{w}) p(\mathrm{w}|\mathbf{x}, \mathbf{t}) \, \mathrm{dw}.$$

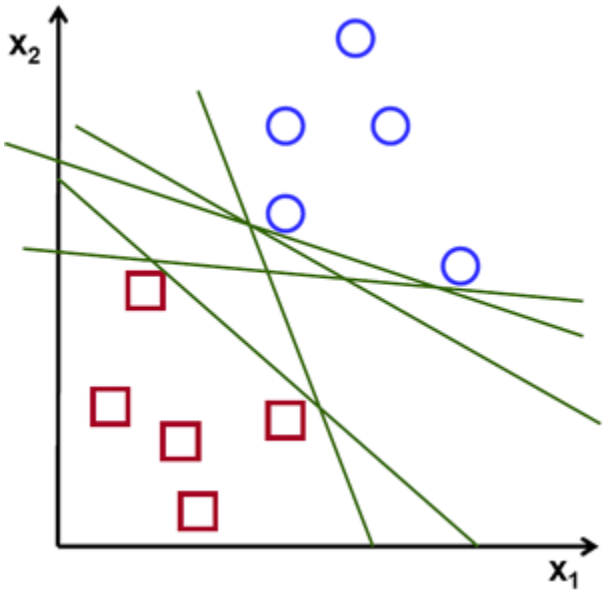
10.1.3 Support Vector Machine

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

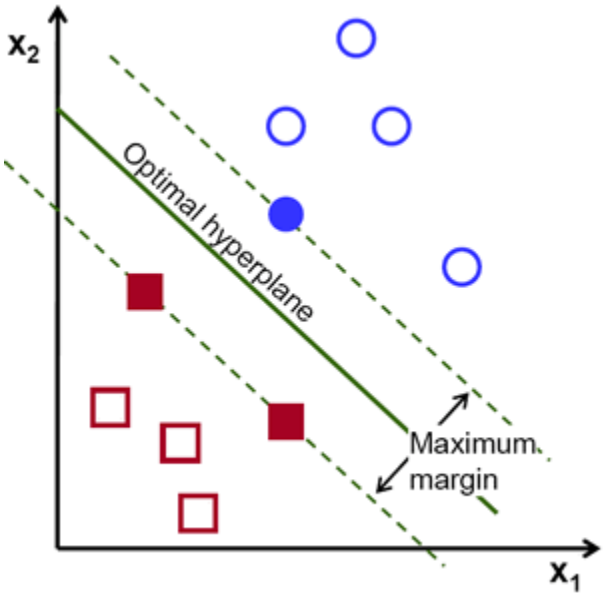
For a linearly separable set of 2D-points which belong to one of two classes, find a separating straight line.



In the above picture you can see that there exists multiple lines that offer a solution to the problem. Is any of them better than the others? We can intuitively define a criterion to estimate the worth of the lines:

A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, our goal should be to find the line passing as far as possible from all points.

Then, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of **margin** within SVM's theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data.



10.1.4 Moving Average Model

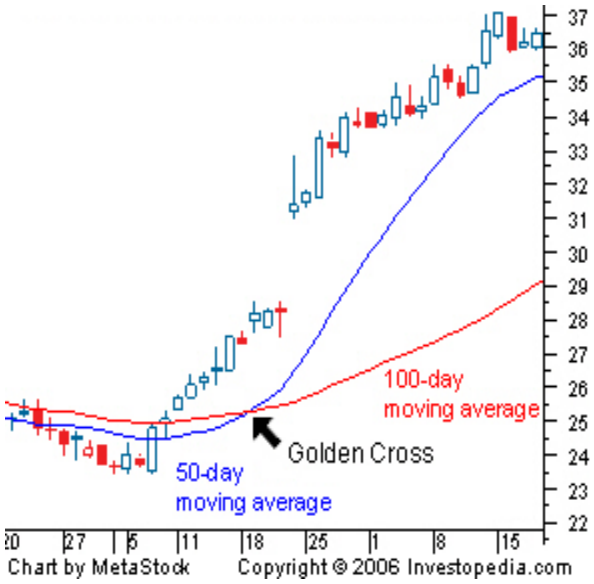
Moving average is a statistical process that creates a set of averages for many small subsets of the full dataset. By taking the averages of mini-subsets of the dataset, it helps smoothen out the trend of the data by showing the long-term lengths. It usually shows the price fluctuation of a stock for a certain period, but can also be used for longer periods. The moving average is a lagging indicator since the predication is made on past prices.

The two main ways to implement is the simple moving average model and the exponential moving average. The difference is that the exponential moving average gives more weight to more recent prices. In this project we used the simple moving average and used the period as 200 days. So, the average price is taken over the past 200 trading days. For instance, the first average price is taken from day 0 to day 200 the second average is taken from day 1 to day 201. This trend is continued until the current price is taken into account. The period is always kept constant at 200 days. This number was chosen from the research the group did which lead us to believe that 200 days will provide a good balance between placing emphasis on past prices as well as more recent prices.

$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n}$$

Pm = *prices of the stock at a certain point*

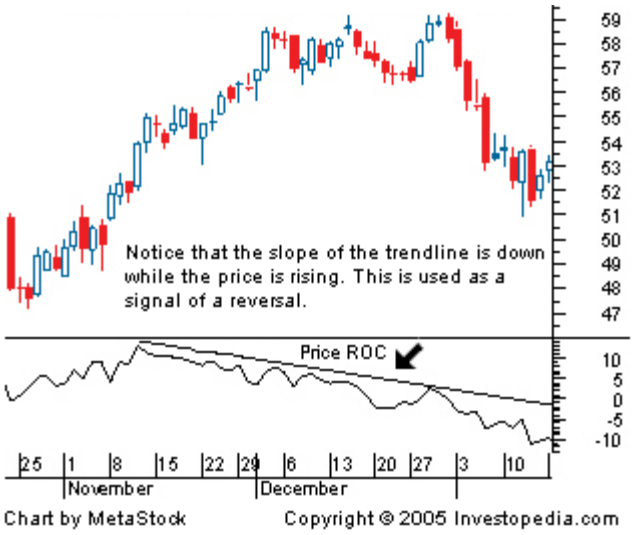
n = *period of days the data is across*

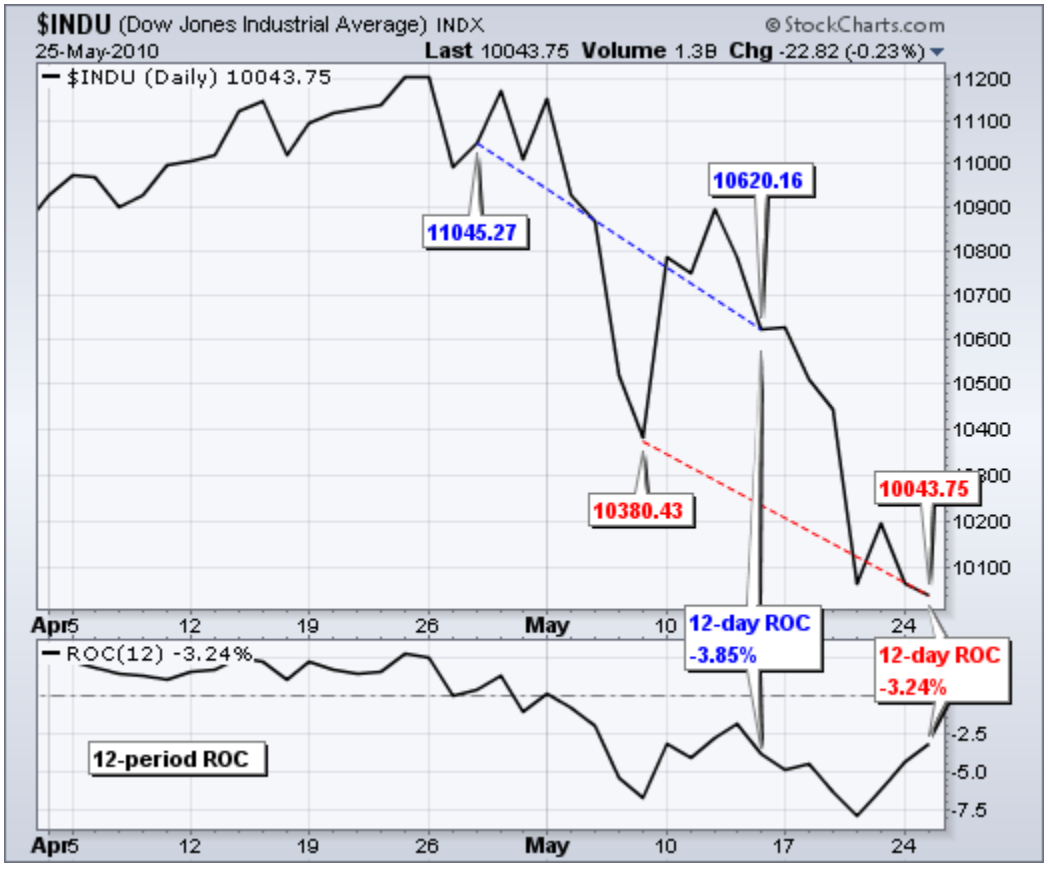


10.1.5 Rate of Change

The Rate-of-Change (ROC) indicator, which is also referred to as simply Momentum, is a pure momentum oscillator that measures the percent change in price from one period to the next. The ROC calculation compares the current price with the price "n" periods ago. The plot forms an oscillator that fluctuates above and below the zero line as the Rate-of-Change moves from positive to negative. As a momentum oscillator, ROC signals include centerline crossovers, divergences and overbought-oversold readings. Divergences fail to foreshadow reversals more often than not so this article will forgo a discussion on divergences. Even though centerline crossovers are prone to whipsaw, especially short-term, these crossovers can be used to identify the overall trend. Identifying overbought or oversold extremes comes natural to the Rate-of-Change oscillator.

ROC = [(Close - Close n periods ago) / (Close n periods ago)] * 100


Notice that the slope of the trendline is down while the price is rising. This is used as a signal of a reversal.
Price ROC
November
December
Chart by MetaStock
Copyright © 2005 Investopedia.com


$INDU (Dow Jones Industrial Average) INDX
©StockCharts.com
25-May-2010
Last 10043.75 Volume 1.3B Chg -22.82 (-0.23%)
$INDU (Daily) 10043.75
11045.27
10620.16
10380.43
10043.75
12-day ROC
-3.85%
12-day ROC
-3.24%
ROC(12) -3.24%
12-period ROC
Apr
May

10.1.6 Relative Strength Index

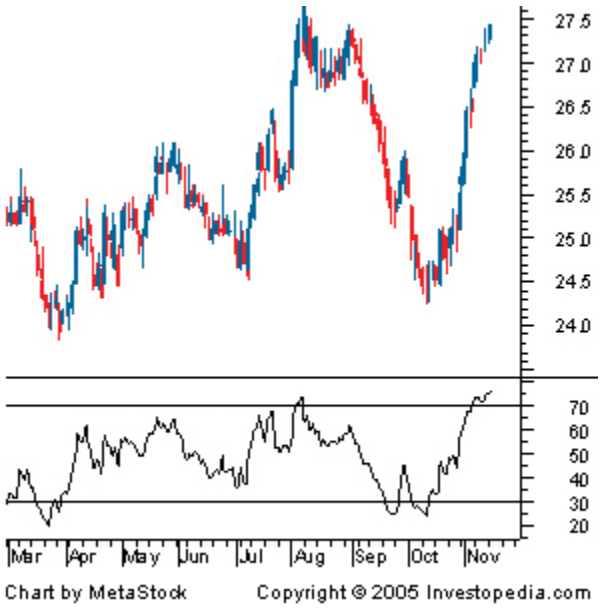
RSI, in practice, is an oscillator from 0-100 showing the momentum of a given stock. The equation for the RSI is calculated by the following formula:

RSI = 100 - 100/(1+RS)

where RS = Average of x days' up close/Average of x days' down close

This formula will either be incorporated into the prediction algorithms, or we may pull the data from a website. When the oscillator goes below the 30 line, it is an indication that the stock has a good chance of going up. Conversely, when the oscillator climbs above 70, there is a high chance the stock price will fall soon. Using that information, the prediction algorithm will detect when a stock has passed either threshold, and adjust its prediction model accordingly.

The creator of the Relative Strength Index was a guy named John Wilder; he created the arbitrary thresholds of 70 and 30 since that indicates moderate momentum in either direction. These values are used as thresholds by most people who rely on this algorithm so we decided to use these values as well. This threshold can obviously be changed in our program to let's say 80,20 or 90,10 to show a stronger momentum. While our program is set by default to figure out when a stock goes above 70 or below 30 but in the future we could in the future allow the user to choose his or her own threshold since people are looking for different qualitative trends with different types of stocks.



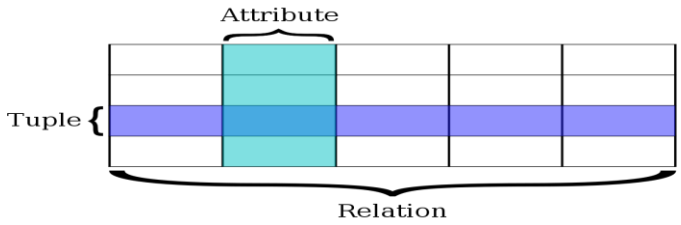
As we can see from the above figure we can see the actual market movement as seen in the top portion of the graph correlated with the RSI prediction model in the bottom portion of the graph.

The formula for R Pivots, support, and resistance could also be used as input as they can be important price points to the market, and it becomes less likely that a stock will move below support or above resistance the more times that resistance/support was tested and held up as a significant price point.

10.2 Data Structures

Database:

The main database will hold all of the stock data implemented with a relational table. A relational table is the ideal structure to hold the stock data as we need to store many different numerical values all related to a single stock ticker stored as a string. For example (CompanyName, TickerSymbol, OpenPrice, ClosePrice, High, Low, etc.) where all the values in a row correspond to a given stock.



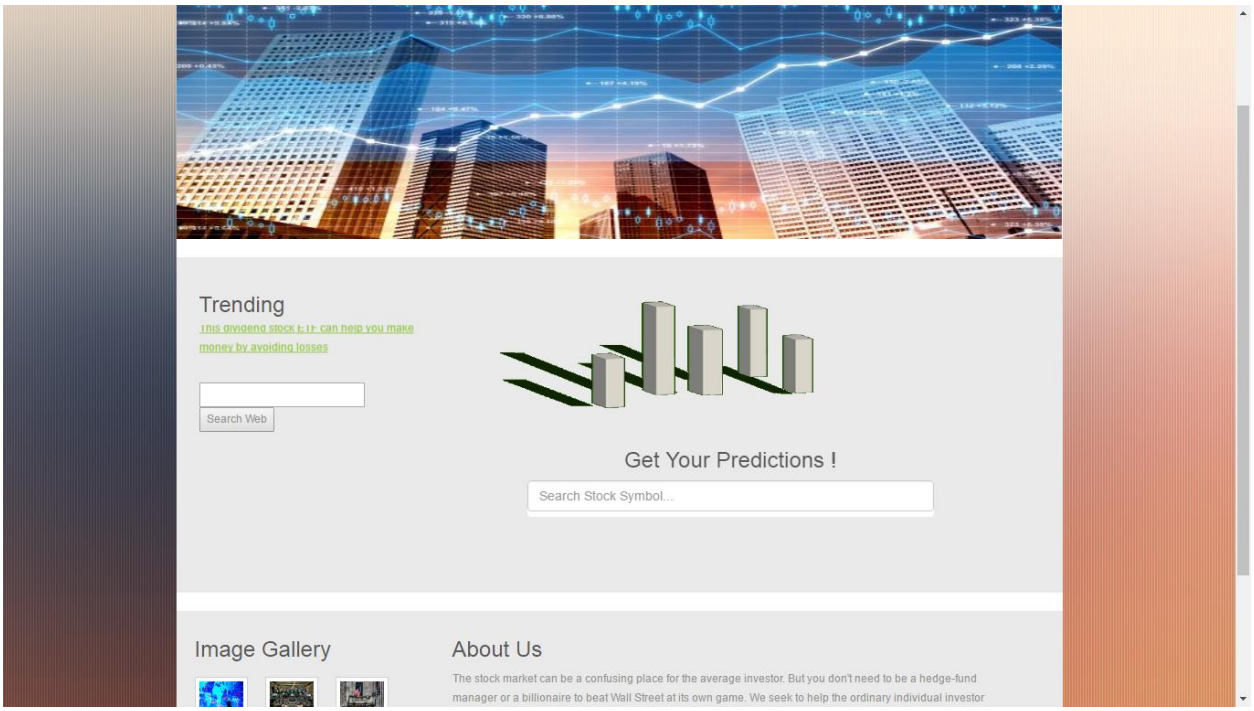
Ticker Symbols:

We will need a smaller database which contains all of the company names and their ticker symbols. This is best implemented with a hashmap where each key of the map is either the company name or ticker symbol and the values are the corresponding ticker symbols. This will let a user search for a stock by either company name, like “Google,” or by ticker symbol, like “GOOG.”

Word Frequency:

To implement our neural network, we will likely use a hashmap to store the frequency of found words. A hash map is a good implementation because it can be easily searched with efficient search methods like quicksort and its key-value setup is similar to the needed word-frequency setup for analysis with the neural network

11. User Interface Design and Implementation



PredStocks

One stop solution to smart prediction !

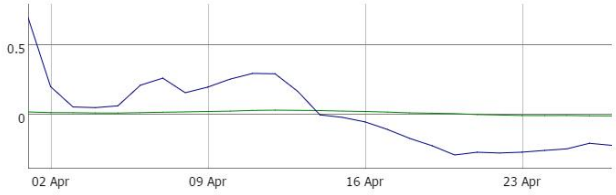
Home | Prediction | News | Contact Us

PredStocks Forecast Chart

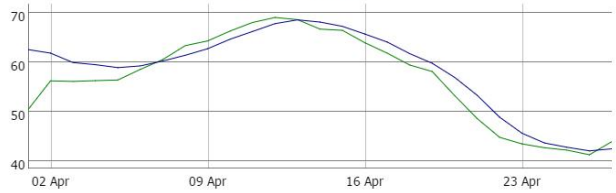
#	Duration	Bayesian Curve Fitting	Artificial Neural Network	Support Vector Machine
1	5 minutes	65.684	66.19	68.17
2	10 minutes	73.239	66.35	69.67
3	15 minutes	72.372	67.4	70.1
4	2 hours	71.642	70.0	74.2
5	4 hours	68.573	66.22	71.52
6	6 hours	75.204	66.08	67.4
7	1 day	75.204	66.37	69.03

Stock Trends

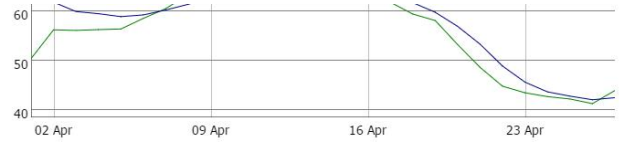
Rate Of Change



Relative Strength Index



Moving Average



Moving Average

PredStocks Verdict : BUY !

Buy Now >

12. Design of Tests

12.1 User Interface Testing

Use Case 1 – Search for Stock This is an important concept in our domain analysis and it enables the user to search for a stock in our database. Once the stock is found the system will request the stock's prediction and load the stock page. The test for this use case is to primarily ensure that the search function is working and the input entered by the user is read by the system properly.

Test 1 – Controller Received and Sent Input Request This test will see if the input entered by the user is received and sent by the controllers in the system. It will make sure that the user controller can read input entered by the user. If the user controller cannot obtain this input then it will not be able to convey this information to the database controller and an error will occur internally and the stock page will not load for the user. This test will also see if the database controller is able to obtain the input request from the user controller. This test is to see if the stock that is searched by the user is sent properly to the system.

Test 2 – Database Connection This test will make sure that the database controller sends the request to the database properly. It will also make sure that the information that the database sends back is the same information that the controller requested.

12.2 Stock Page

Test 1 – Load Page This test will make sure that after the stock has been searched that the correct page is loaded. It will also make sure that the various components of the page is also loaded successfully. This will include the graphs

Test 2 – Graph This ensures that the stock's graph on the page is correct and loads properly. There will be two graphs, one will have a dependency on yahoo finance data to populate and the other will have data that will be populated from our database. We will be able to identify issues due to the fact that we have two graphs pulling data from different sources. This could be useful because if one graph's data isn't loading properly we can pinpoint it on that graphs data source which will aid in debugging.

12.3 Web Pages

Test 1 – Load Page Like the stock page, we will run a test for all the other pages on the website and make sure they load properly. These pages are Portfolio, Login, Registration, Logout, Home, News, and FAQ

12.4 Prediction Algorithm Testing

The prediction algorithms will be tested using previous data. This has not been implemented yet in terms of actual testing, however manual testing has begun that compares the result of the moving average based off of historical data. This testing can be automated using programs that will test for validity of both the algorithm itself as well as the validity of the code. We will implement these tests for all of the algorithms that we intend on implementing, at least

the Moving average and RSI for now.

12.5 Integration Testing

Our integration testing will be done using big bang integration. Prior to compiling each individual sub part, we will conduct unit testing on individual pieces of software in order to validate methods and strategies so that we can eliminate logical errors and concentrate on technical errors that may occur as a result of integration. There aren't too many external dependencies (the only is yahoo! finance api) and everything is relatively modular since everything has its own purpose and is independent of other aspects of the website so pinpointing issues shouldn't be too much of an issue. We will design larger test

13. History of Work

13.1 Key Accomplishments

	NAME	DURATION	START	FINISH	DEADLINE
1.	Project group meeting 1	1 day	02/13/2017	02/13/2017	
2.	Downloading stock information and storing it in a MySQL database	9 day	02/20/2017	03/01/2017	03/03/2017
3.	Design Report	2 days	03/01/2017	03/02/2017	03/03/2017
4.	Project group meeting 2	1 day	03/10/2017	03/10/2017	
5.	Understand how a webservice work	3 days	03/11/2017	03/13/2017	
6.	Finalizing Web tools and softwares	2 days	03/15/2017	03/16/2017	
7.	Final Project preliminary Presentation	3 days	03/18/2017	03/20/2017	03/24/2017
8.	Developing the skeleton of the Web service	10 days	03/31/2017	04/09/2017	
9.	Project group meet 3	1 day	04/11/2017	04/11/2017	
10.	Coding for final demo	10 days	04/12/2017	04/21/2017	
11.	Writing final report	12 days	04/13/2017	04/27/2017	04/27/2017
12.	System testing	1 day	04/26/2017	04/26/2017	
13.	System Fine tuning	2 days	04/26/2017	04/27/2017	
14.	Demo	1 day			04/28/2017

13.2 Future Work

1. The current system does not yet support user portfolio management, which we seek to implement in the future.
2. Dynamic forecasting at user homepage and text alert facility.
3. Android app for PredStocks realtime forecasting
4. Enhanced user interface for more stock indicators
5. Support for larger stock databases
6. Efficient pattern recognition facility

14. References:

1. http://www.investopedia.com/terms
2. http://en.wikipedia.org/wiki/Stock_market_prediction
3. http://www.dummies.com/how-to/content/stock-investing-for-dummies-cheat-sheet.html
4. https://en.wikipedia.org/wiki/Stock
5. http://en.wikipedia.org/wiki/Internet_access#Technologies
6. http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf
7. http://www.investopedia.com
8. http://www.world-exchanges.org/statistics/monthly-reports