Software Engineering of Web Applications

Project Group # 3

Stock Market Personal Investment Assistant

Submitted on May 7th 2008

Benjamin Ross

Prem Kumar Singh

Rushi Rawal

Tathagata Ray

William Taft

Contents

1. Introduction

1.1 Why Predict Stock Prices?

The wish to find methods to predict asset returns has occupied the minds of investors and also academics since the birth of financial markets. Besides the obvious monetary value that comes with the ability to correctly predict financial assets, there has been a growing recognition among economists that the financial variables are closely connected to real economy. One example is the relatively new idea of using interest rates to fight inflation. The liberalization and globalization of world asset markets have caused interest rates, exchange rates, and also other asset markets to be intimately linked, and the need for tools to monitor as well as control risk levels has become obvious both for industrial companies and financial institutions. The question of predictability in the stock markets is therefore important even outside the trading rooms.

1.2 Is the Market Predictable At All?

It is hard to resist the temptation to include the following old joke (cited from Lo, McKinley] about 'an economist strolling down the street with a companion when they come upon a $100 bill lying on the ground. As the companion reaches down to pick it up, the economist says "Don't bother — if it were a real $100 bill, someone would have already picked it up"'. Irrational as it may seem, this way of reasoning is very close to the official academic standpoint until only a few years ago.

Evidence of Predictability

Instead of spending all energy at forcefully defending the Random Walk and the Efficient Market Hypotheses, many researchers have instead started to look at possible reasons why the markets could be predictable and even how some prediction methods could be used to generate "excess" profit. Below are a few examples of effects and "anomalies" that have been observed and reported.

a. Autocorrelation

There has been extensive research on the temporal dynamics of stock returns. For weekly and monthly returns, most results point at negative autocorrelations for individual stocks. A negative serial correlation is consistent with the Stock Market Overreaction and Mean Reversion effects described below. The autocorrelation for portfolios of stocks, and also for indices, are most often reported as positive.

b. Small-Firm Effect

This effect comes from the observation that small capitalized companies over a long time period have had higher rate of returns than the larger companies. The classical argument against trading strategies based on this effect is that the higher returns are balanced by a higher risk attached to small firms.

c. Technical Analysis

Technical analysis, also known as "charting," has been used by practitioners in the financial markets for a long time. So-called "technical indicators," consisting of simple transformations of past stock data, are used to produce buy and sell recommendations for the traders. In the academic community, technical analysis has been at best proclaimed useless, and at worst equated with woodo, astrology or alchemy. One of the reasons for these diverging views is the highly subjective nature of technical analysis. The relation between traded volume and stock prices that always has been obvious for technical traders. Many technical trading rules, when forming their trading signals, include the traded volume as well as the open, close, low, and high price for each day. In contrast, the majority of academic models on the subject works with close prices only. However non-predictable the stock market may be, it is not a correct inference to draw this conclusion from such simplified analyses.

2. Project Overview:

'Stock-Market investor advisory' is the Internet's most comprehensive investment advisor. "Our stock market investor advisory' - it simulates the experience of trading in the stock market."The computer technology enables us to create new ease in investing in the virtual market with precise prediction. This is mainly devised to help new enthusiastic investors/individuals who plan to invest into the market, with or without prior experience.

This project mainly focuses on three aspects:

1. <u>Real-life data feed</u>:

In the project we have used real life data from yahoo stock. So the prediction is based in real data feeding and the history of data, thus making it a realistic prediction advisory.

2. <u>Prediction Parameters</u>:

The Stock Prediction Simulator supports all types of scenario, for long term or short term investment, time period, the amount of money to be invested, and suggests to buy, sell, hold,

.

3. Easy to Use:
The Simulator is useful for everybody from beginners who have never placed a trade to sophisticated investors looking to test out advanced strategies. Use it as an investment challenge.

2.1 Design Overview

Our design is consists of five modules:

1. Web client
2. Prediction algorithm
3. Data collection,
4. Connecting to database.
5. User interface

1. Web client:

Our web service consists of three major functional blocks the framework to interact with the web client, the Prediction Algorithm, the module used to connect and query the database. And the web client is an HTML web form that interacts with the users to get their inputs. The back-end of the form implements a button click event that takes all the user inputs and validates them before proceeding with the further processing. User is prompted to provide the valid inputs for each field. Once the inputs are entered, the Client back-end makes a call to the web service residing on the IIS (Internet Information Server). The client is developed using the code-behind feature provided by .NET framework. The code-behind feature allows users to create the web-forms much like any windows application forms, wherein, the user is just required to create a web-form by drag and drop of all the components required on the form.

2. Prediction algorithm:

The primary prediction algorithm to be used is an autoregressive linear prediction filter. This filter aims to predict the output of a system based on one or more inputs. The desired output prediction is the future price of a stock or security, and the main input parameters are the past stock values. As can be seen below, the current price of a stock, P(t), can be predicted based on N previous past values of the stock P(t-it) multiplied by a weighting function a_i.

$$P(t) = \sum_{i=1}^{N} aiP(t - i\Delta t) .$$

3. Data collection:

The data collection is divided into three sections (described in section 6) namely variable selection, data collection and data Inspection. The data were obtained using a php-based query interface from Yahoo Finance! (http://finance.yahoo.com). The stock market timings are from 9:00am morning to 4:30pm in the evening. During this time the stock data were collected. We chose yahoo finance as it is free source of stock data and it is one of the most reliable sources. However the stock data from yahoo finance are delayed by five minutes. If we had higher budget for this project we could get more recent data for the stocks. Data selection and preprocessing constitute a crucial step in any modeling effort. There are so many stocks that are traded at the stock market. There are different types of stocks that different investors would be interested in depending on what type of investor he/she is. We thought of stocks as companies and businesses instead of stocks. That's an important distinction that's lost on too many people. Stocks are, after all, actual small pieces of an ongoing business, not lottery-like slips of paper. We read financial statements and used P/E ratios along with many other numbers. Read widely about companies that cause interest in the current marker ideally in industries that we has a potential boom in the future. Finally the data was inspected manually to check for errors.

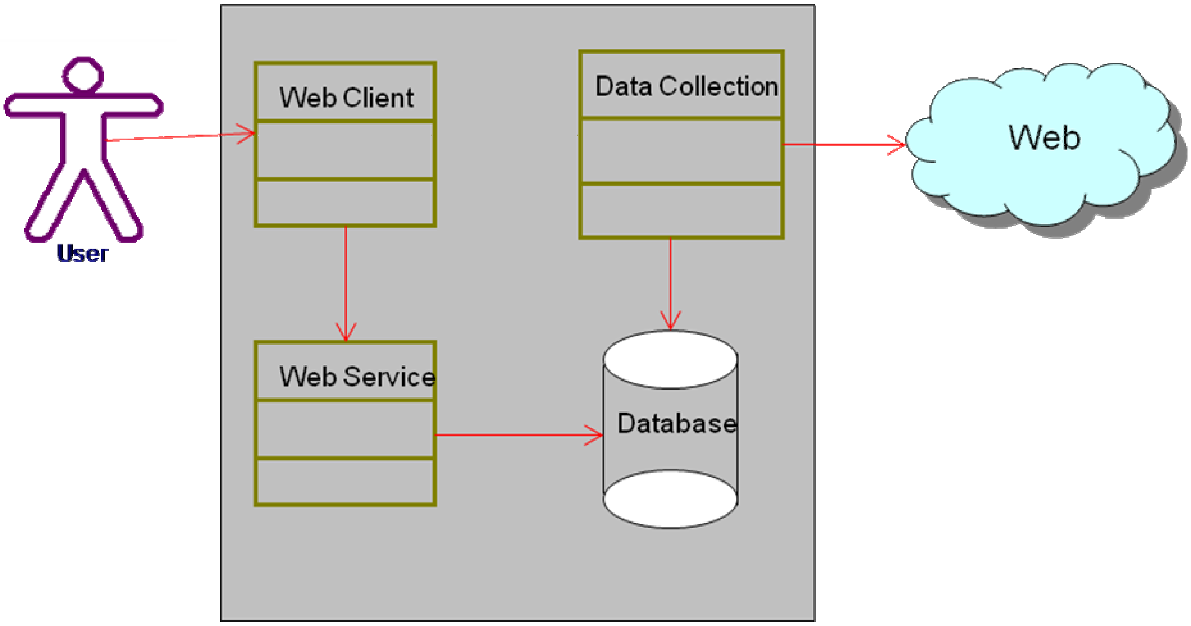
4. Connecting to Database

The purpose of the Connect_Database function is to connect to a server, retrieve specific data off of that server and pass the values to a function which calls it. In the instance of this program, the calling function is the web service, the server it connects to contains a MySQL database of stock data, and the values it retrieves are the prices of a particular stock.

The Connect_Database function is called from the Predictor function and is passed a string representing the ticker symbol. Based on that ticker symbol, the Connect_Database function will query the corresponding table on the server in the database and store the past stock values in an array. It will return these values to the calling function, in this case the prediction function.

5. User interface:

We have considered creating simple user interface since it is easy-to-use and enables the user to learn the system quickly and to use it efficiently. For the user interface we considered the company names, fund available, Time duration, projected value and the suggestion to buy, hold or sell.



System Level Interaction Diagram

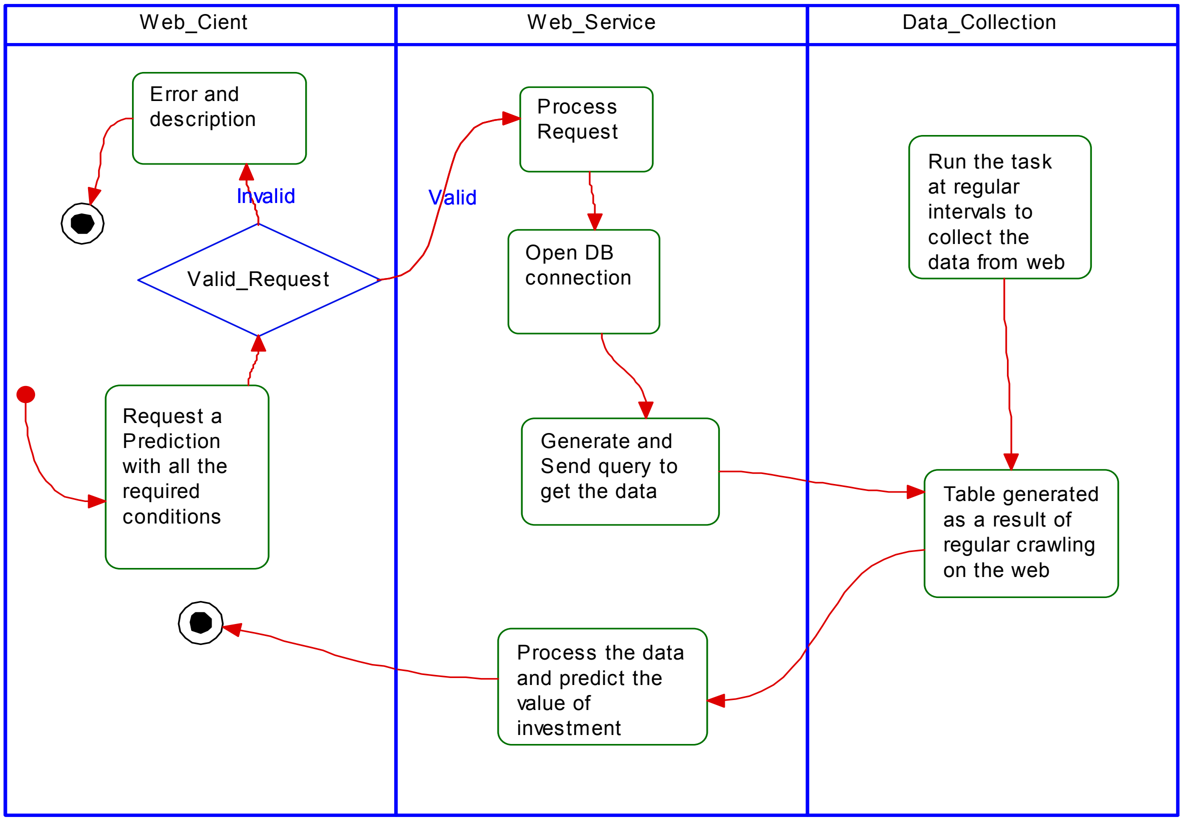

Web_Cient
Web_Service
Data_Collection
Error and description
Invalid
Valid
Process Request
Run the task at regular intervals to collect the data from web
Valid_Request
Open DB connection
Request a Prediction with all the required conditions
Generate and Send query to get the data
Table generated as a result of regular crawling on the web
Process the data and predict the value of investment

*Fig-1.*Activity Diagram

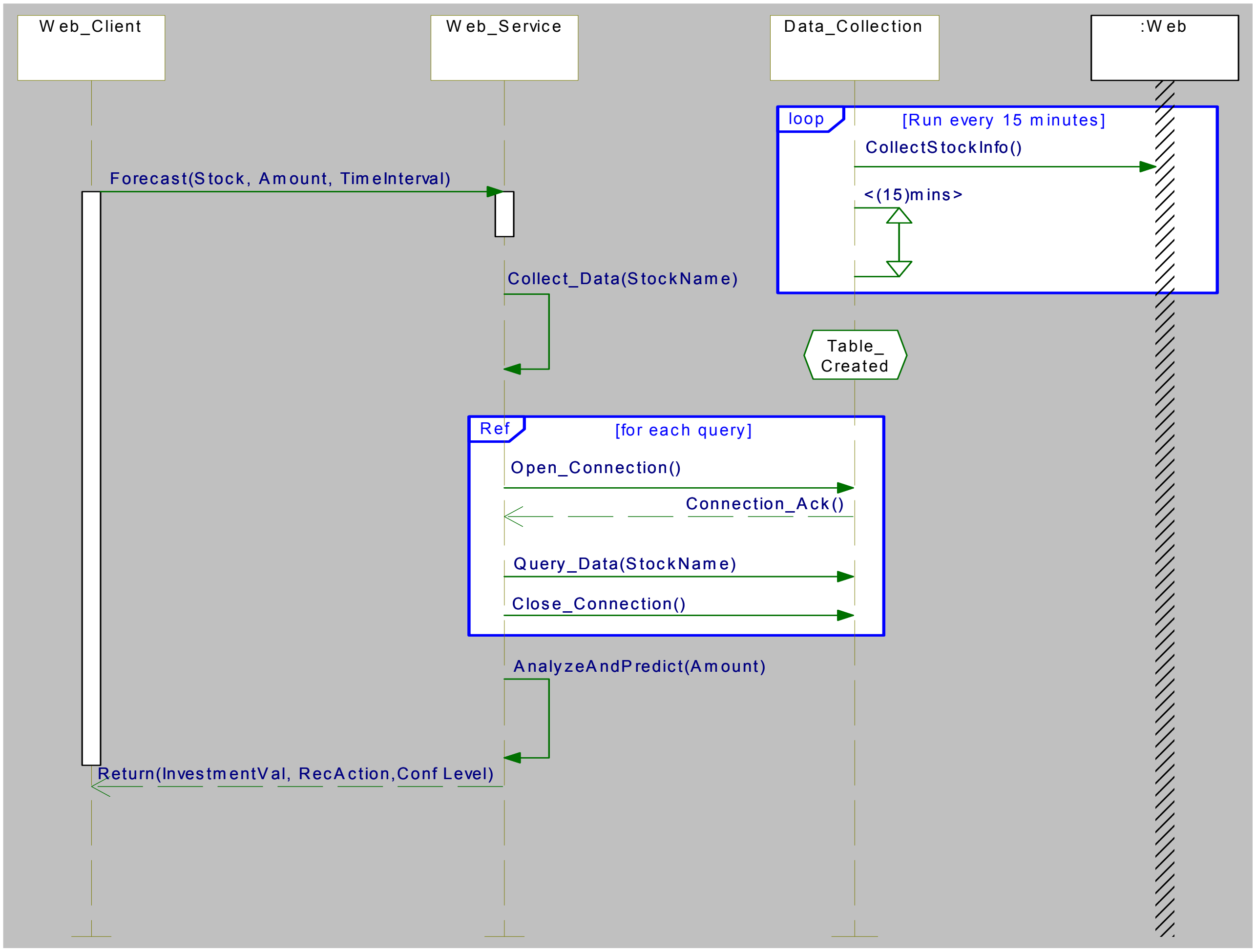


*Fig-2.*Sequence Diagram

3. USE CASES

ACTORS:

1. Forecaster
2. Investors
3. Database

3. B. ACTORS AND GOALS

ACTOR	ACTORS GOALS	USE CASE
Forecaster	Select stock (if the inquiry requested selection from or "any"),Prediction: price trend or numeric value at time *t* in the	UC1
Forecaster	Recommendations about placing a protective sell or buy Order.	UC3
Investor	Tries to get information about Stock(s) to consider: *individual* (specified by ticker symbol), *select-one-for-sector* (sector specified by a standard category), *any* (select the best candidate),Informs the forecaster about Trade to consider: *buy*, *sell*, *hold*,	UC1
Investor	Informs the forecaster about Time horizon for the investment: integer number	UC1
Investor	Informs the forecaster about Funds available: integer number for the capital amount/range	UC1
Database	It provides/stores necessary information regarding the investors and history of stocks.	UC2,

USE CASE	**USE CASE NAME**	**Description**
UC1	Query (for required parameter)	The system asks the investor for required parameter.
UC2	Analysis	The system analyzes the given parameter for optimum prediction
UC3	Forecasting	After analysis the system forecasts about *buy*, *sell*, *hold*, OR Time to consider: , *any time horizon*

3.1 USE CASE DESCRIPTION

Use Case UC: UC1: Query (for required parameter)

Initiating Actor: Forecaster

Actor's goal: Query for required parameter

Participating Actors: Forecaster, Investor

Precondition: The investor logged in with required information

Post conditions: The parameters received by the forecaster, in proper format

Main success scenario:
-> 1. System prompts the investor to provide the parameters it requires to analyze.
<- 2. The investors provides the parameters asked by the system.
-> 3. The investors also provide additional information about the time horizon or the preferred stocks, he is in think of, like long time investor or short time investor.

Alternative Scenario:
-> 2(a) The investor provides wrong types of data in wrong field, and
<- 2(b) the system will prompts for correction.
<- 2(c) System prompts the actor to re-type the given field

Use Case UC: UC2: Analysis

Initiating Actor: Forecaster

Actor's goal: Analyze the data provided by the investor and observe his requirement

Participating Actors: Forecaster, Database

Precondition: The system is provided with all the information from the investor

Post conditions:

Main success scenario:
-> 1. System prompts the investor to provide the parameters it requires to analyze.
<- 2. The system analyzes the all data collected from investor,

Alternative Scenario:
-> 2(a) The investor provides wrong types of data in wrong field, and
<- 2(b) the system will prompts for correction.
<- 2(c) System prompts the actor to re-type the given field

Use Case UC: UC3: Forecasting

Initiating Actor: Forecaster

Actor's goal: Forecast about the stock after recognizing the pattern.

Participating Actors: Forecaster,

Precondition: The pattern of the given stock price is recognized from the history of that stock

Post conditions: The stock price is predicted for a given period of time.

Main success scenario:
-> 1. System prompts the investor to provide the parameters it requires to analyze.
<- 2. The investors provide the data asked by the system.
-> 3. The investors also provide additional information about the time horizon or the preferred stocks; he thinks of, like long time investor or short time investor.

Alternative Scenario:
-> 2(a) The investor provides wrong types of data in wrong field, and
<- 2(b) the system will prompts for correction.
<- 2(c) System prompts the actor to re-type the given field

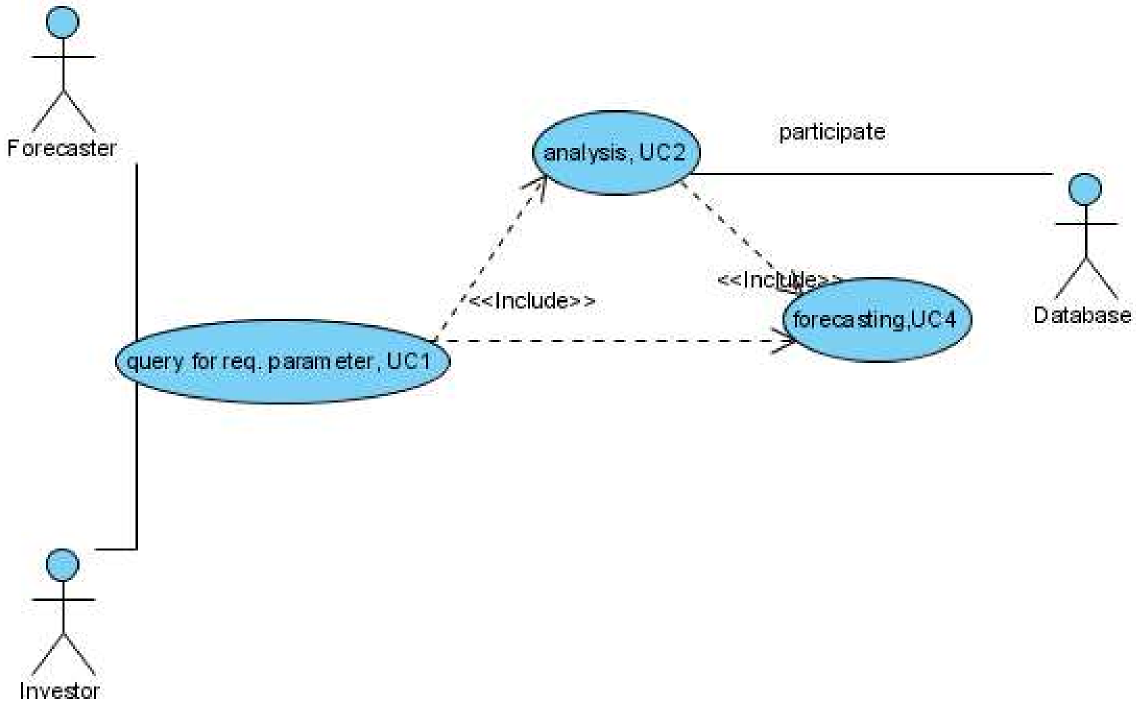


Fig-3 USE CASE DIAGRAM

4. Web service:

Web Client:

The web client is an HTML web form that interacts with the users to get their inputs. The back-end of the form implements a button_click event that takes all the user inputs and validates them before proceeding with the further processing. User is prompted to provide the valid inputs for each field. Once the inputs are entered, the Client back-end makes a call to the web service residing on the IIS (Internet Information Server).

The client is developed using the **code-behind** feature provided by .NET framework. It includes the .aspx and .aspx.cs files. The code-behind feature allows users to create the web-forms much like any windows application forms, wherein, the user is just required to create a web-form by drag and drop of all the components required on the form. The **.aspx** file gets created with the required HTML code. The **.aspx.cs** file consists of all the code required to handle the events occurring on the web form. Since, the processing logic is separated from the HTML code; it makes the maintenance much easier by improving thereadability.(http://en.wikipedia.org/wiki/ASP.NET, http://support.microsoft.com/kb/303247)

The web client uses SOAP messages over HTTP to communicate with the web service. However, in the MS .NET 2.0 the messaging between the web forms and services is abstracted from the developer.

The web form consists of a **StockUI** class implemented in the .aspx.cs file. This class implements the button_click event.

Description of Method:

The button_click() event gets generated when the user interacting with the web form, inputs all the required data and clicks the "submit" button. All the values entered by the user in web form get captured into the local variables in the event handler. The event handler then makes a call to the published web service "Forecast" by passing in the required inputs entered by the user. The handler is also responsible for receiving inputs from the web service, do the necessary processing on the received values before sending the responses back on the web form. In our case, the algorithm returns the value of investment after number of days. The actual prediction algorithm is described later in the document. Based on the response from the prediction algorithm for the "Projected Value" of investment, the Web Service compares it with the user entered "Funds Available".

```
If (Projected Value > Funds Available)
{
    Recommended Action = "Buy";
}
Else If (Projected Value < Funds Available)
{
    Recommended Action = "Sell";
}
Else
{
    Recommended Action = "Hold";
```

Input:

These inputs to the handler are not passed by the conventional method of calling the function; however, they are accessible to the local variables declared inside the method.

CompanyName(string): The name of the company the user is interested in investing.
AmountInvested(int32): The amount of money available to invest. (USD).
TimeDuration(int32): The time duration for which the amount could be invested. (Number of days).

Output:

Again, the outputs are not the conventional returns by the function; they are assigned to the components available on the web form to show the output.

ProjectedValue(string): The projected value returned by the Forecast web method.
RecommendedAction(string): The "buy", "sale", "hold" advice based on the comparison of the AmountInvested and the ProjectedValue.
Projected stock value (double)
Confidence level (integer) Value return by prediction algorithm(described in document)
Failure: Error message with the error description.

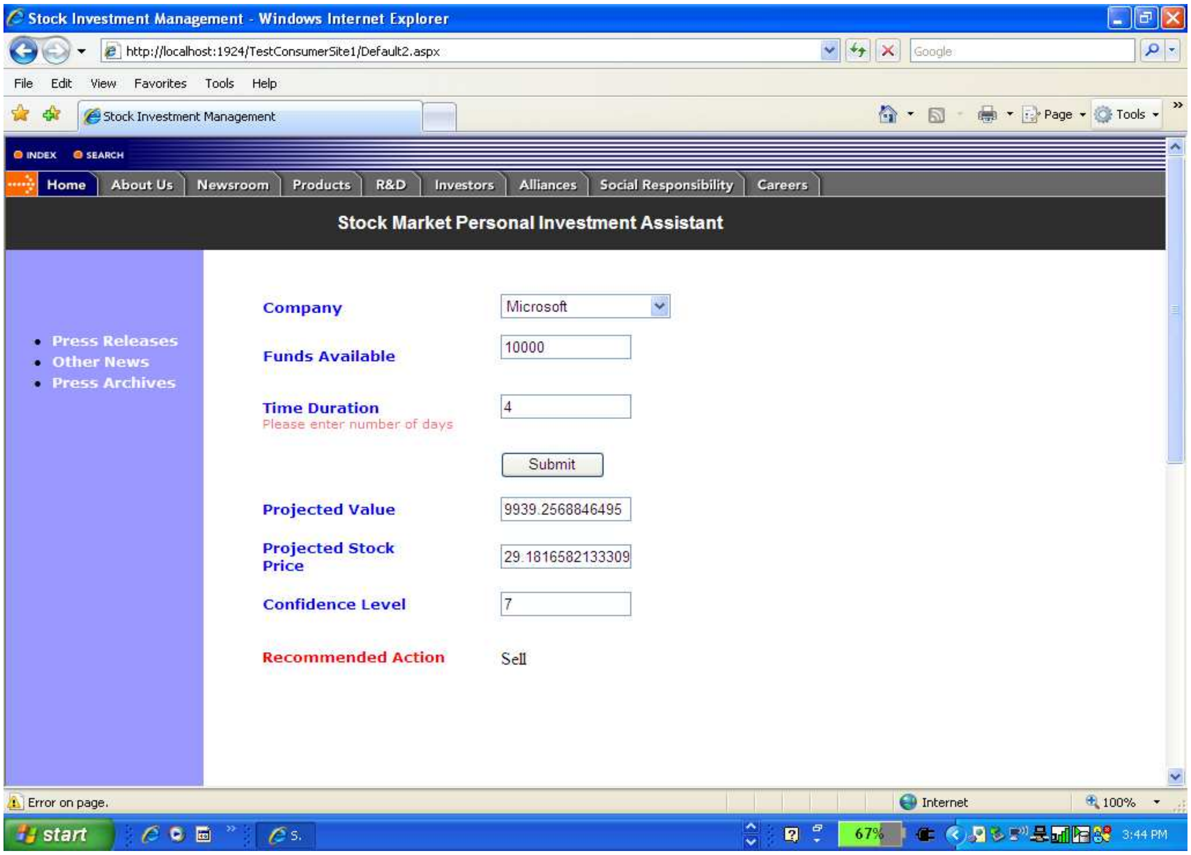
****

```
protected void Button1_Click(object sender, EventArgs e);
```

****

Following message will be shown right next to the predicted investment value on the website to expose the limitation of the prediction technique to the user (ref: *section 5 Prediction algorithms, page19, paragraph 1*)."(disclaimer: The prediction technique used in the application defaults the prediction to 10 days if number of days entered are >=10)"

Here is the screen capture of how the Web Form looks like as of now:


Stock Investment Management - Windows Internet Explorer
http://localhost:1924/TestConsumerSite1/Default2.aspx
Google
File Edit View Favorites Tools Help
Stock Investment Management
Page Tools
INDEX SEARCH
Home About Us Newsroom Products R&D Investors Alliances Social Responsibility Careers
Press Releases
Other News
Press Archives
Submit
Error on page.
Internet
100%
start
67%
3:44 PM

Stock Market Personal Investment Assistant

Company	Microsoft
Funds Available	10000
Time Duration Please enter number of days	4
Projected Value	9939.2568846495
Projected Stock Price	29.1816582133309
Confidence Level	7
Recommended Action	Sell

Screenshot 1

Web Service:

The web service consists of three major functional blocks:

1. The framework to interact with the web client
2. The Prediction Algorithm
3. The module used to connect and query the database

This section focuses on the framework that publishes the web service and communicates with the client application. Prediction Algorithm and the database connection modules are described later in the document.

Since the SOAP messaging through http is abstracted from the developer in MS .NET 2.0 framework, the attributes defined in web.config file are used to tell the Common Language Runtime (CLR) to treat particular class and its methods as a web service. Based on these specifications, the CLR auto generates necessary SOAP, HTML modules.

The class "Service" is defined as follows, to declare it as a web service. The class has to be declared public in order for it to be utilized by other clients. The methods inside this class should be declared as web method.

```
public class Service : System.Web.Services.Webservice
{
   [WebMethod]
   public int Forecast()
   {
   }
}
```

The web service could be hosted on a local machine using IIS server (which is what our current implementation does.) and could be advertised on a specified URL using .disco file. .disco is a XML based discovery document that advertises the service residing at a particular URL. It has two types of reference nodes contractRef and dicoveryRef. contractRef contains references to the WSDL files of the webservice while the discoveryRef contains references to the other web services.

The .NET framework generates the WSDL description of the web service which can be viewed by building the project and clicking on the link "Service Description".

Our web service makes the Forecast method public, i.e. could be used by the clients as a web service, whereas the database connection module remains private by design.

The client looks for the web services available by browsing the following resources:

1. Services available in a particular solution
2. Services available on a local machine

3. UDDI servers on the local network.

When choosing the one of the above mentioned options, it shows the list of available services. The client could pick the one it intends to use and add as “App_Webreferences”. This makes the connection complete. When the client is built again after this procedure the web form is successfully able to communicate with the web service.

Code Package Description:

The implementation of above mentioned modules consists of the following two packages:

1. ConsumerSite (Client)	-Folder
App_Data	-Folder
App_WebReferences	-Folder
Service3	-Folder
Service.disco	-Web service discovery file
Service.discomap	-DISCOMAP file
Service.wsdl	-Web Service Description Language
Default2.aspx	-ASP.NET server page
Default2.aspx.cs	-Visual C# source file
Web.Config	-XML Configuration file

2. WebSite (Web Service)	-Folder
App_Code	-Folder
Service.cs	-Visual C# source file
App_Data	-Folder
Service.asmx	-ASP.NET web service
Web.Config	-XML Configuration file

Here are the SOAP messages in XML:

```
POST /TestWebSite1/Service.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <Predictor xmlns="http://tempuri.org/namespace">
      <tickers>string</tickers>
      <principal>int</principal>
      <day>int</day>
    </Predictor>
  </soap12:Body>
</soap12:Envelope>

HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <PredictorResponse
xmlns="http://tempuri.org/namespace">
      <PredictorResult>
        <double>double</double>
        <double>double</double>
      </PredictorResult>
    </PredictorResponse>
  </soap12:Body>
</soap12:Envelope>
```

5. Prediction Algorithm:

Predictor Method:

Description:

This function is called from the server upon request from the user. The user passes in necessary parameters, including a ticker symbol for a stock of interest and a principal investment amount. This function in turn calls a secondary function "Connect_Database," which returns to the Prediction function an array of stock prices and trade volumes. After this array of stock values is returned from "Connect_Database," the next 10 predicted stock values are calculated, along with the expected investment value at a time x days into the future (which is specified by the user).

Input Parameters:

double Principal(double): The amount of principal that the investor can invest in a stock.
int day(integer): The length of time (in days) that the investment is expected for.
string Stockname(string): The name of the stock of interest. This name will be converted to a stock ticker symmbol and used when calling the function "Connect_Database"

Output Values:

double [](double): The function will return a double precision array which contains:

An expected investment value at the end of the investment period.
A numerical indicator of confidence.

****
double Predictor (double Principal, int day, string Stockname);

****

The overall flowchart of the prediction algorithm is shown in next page:

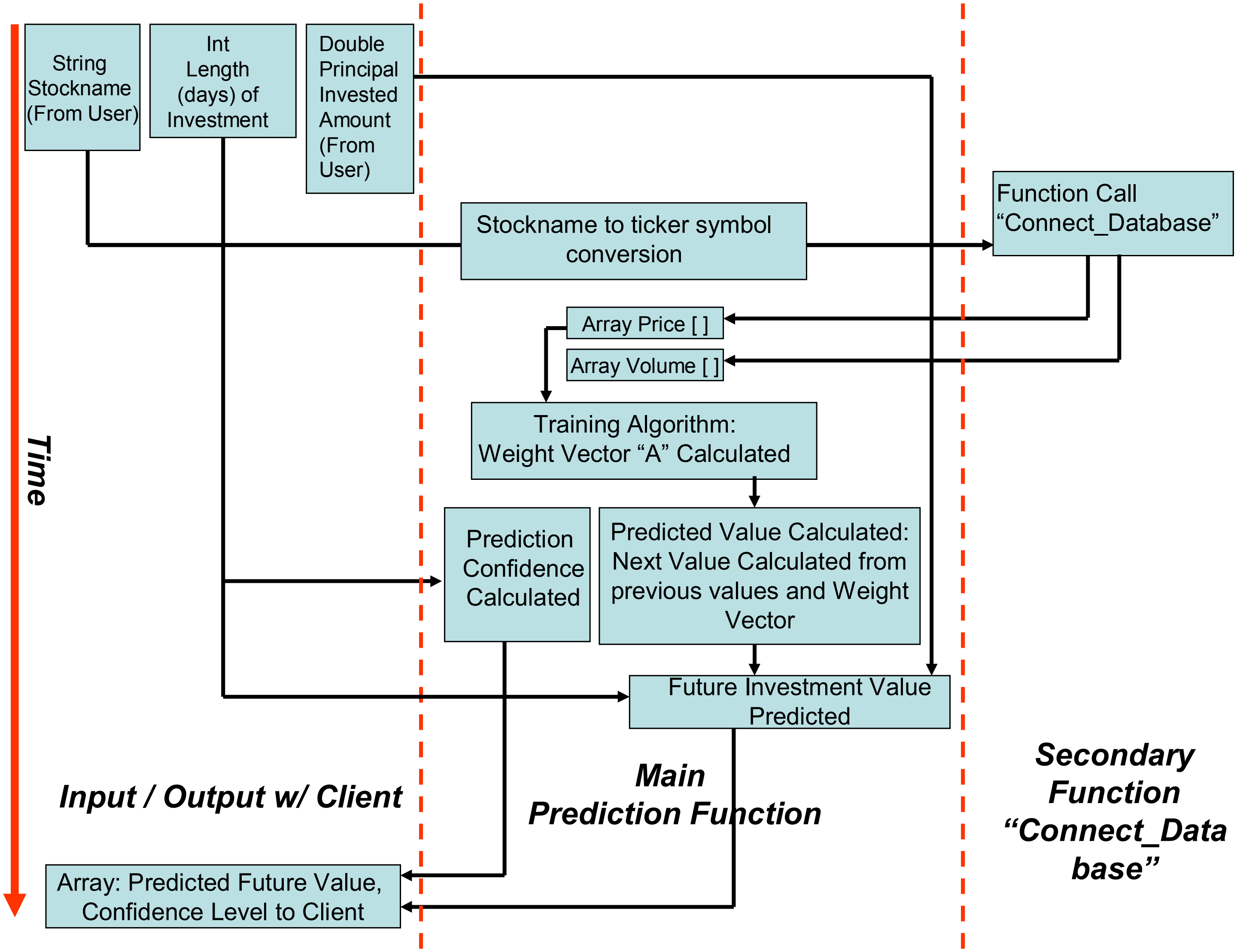

String Stockname (From User)
Int Length (days) of Investment
Double Principal Invested Amount (From User)
Stockname to ticker symbol conversion
Function Call "Connect_Database"
Array Price [ ]
Array Volume [ ]
Training Algorithm: Weight Vector "A" Calculated
Prediction Confidence Calculated
Predicted Value Calculated: Next Value Calculated from previous values and Weight Vector
Future Investment Value Predicted
Array: Predicted Future Value, Confidence Level to Client
Time
Input / Output w/ Client
Main Prediction Function
Secondary Function "Connect_Database"

Fig-6 flowchart of the prediction algorithm

The function Predictor is called from the web service, and is passed 3 values: A stock name, a principal invested value, and an expected length of investment, in days. The stock prediction algorithm loses accuracy as a prediction is made further and further into the future. Because a total of 10 previous values are evaluated in the prediction calculation (this is explained below), 10 days is the furthest future value that can be predicted with any accuracy. If the user inputs a value greater then 10 days, the function will automatically default to 10 days and return the expected investment value at this point in the future. After having these values passed in from the web service, the prediction algorithm calls a secondary function called "Connect_Database." This function then passes an array to the prediction algorithm of stock price values sampled at even time spaces (i.e., the collection program runs continuously, so there will be an even amount of time between each stock value recorded). The array of price values is used to calculate future stock values based on an autoregressive linear prediction filter, which will be described in detail below.

After 10 future values are predicted, the final value of the invested principal can be easily calculated. Using the desired investment time period (in days), the invested principal is simply multiplied by the ratio of the final stock value (the value of the stock on the specified number of days in the future) divided by the current value of the stock (the value of the stock currently). The value of the investment at the time in the future specified by the user is passed back to the web service through an array, which also contains a numerical indication of confidence in the prediction. This confidence calculation is a 1 – 10 number, with 1 representing the lowest confidence, and 10 representing the highest.

The primary prediction algorithm to be used is an autoregressive linear prediction filter. This filter aims to predict the output of a system based on one or more inputs. The desired output prediction is the future price of a stock or security, and the main input parameters are the past stock values. As can be seen below, the current price of a stock, P(t), can be predicted based on N previous past values of the stock P(t-it) multiplied by a weighting function a_i.

$$P(t) = \sum_{i=1}^{N} aiP(t - i\Delta t)$$

Written in vector notation:

$$P(t) = \begin{bmatrix} a1 & a2 & ... & aN \end{bmatrix} * \begin{bmatrix} P(t-1) \\ P(t-2) \\ . \\ . \\ . \\ P(t-N) \end{bmatrix}$$

It was decided that the future stock values would be calculated based on a weighted average of 10 previous values. If too few values are used in this calculation, the future predicted value will have an under dependence on historical prices; if too many values are used, the prediction will be based too heavily on past values, and thus not “see” quickly changing trends. Several sources consulted (see number 14 in list of references at end of report) suggested using 10 previous values in a prediction calculation.

In order to predict a future value P(t+1), we will need to substitute the currently predicted value P(t) for P(t-1); to predict a value P(t+2), we will need to substitute the currently predicted value P(t+2) for P(t-1), and so forth. As predictions are made for further and further into the future, the accuracy of the predicted values decreases, as predictions are made less on true historical data and instead on predicted values, which have error. This is demonstrated below:

$$P(t+1)=[a1\ \ a2\ \ ...\ \ aN]*\begin{bmatrix} P(t) \\ P(t-1) \\ . \\ . \\ . \\ P(t-(N+1)) \end{bmatrix},\rightarrow P(t+2)=[a1\ \ a2\ \ ...\ \ aN]*\begin{bmatrix} P(t+1) \\ P(t) \\ . \\ . \\ . \\ P(t-(N+2)) \end{bmatrix}\rightarrow P(t+N)=[a1\ \ a2\ \ ...\ \ aN]*\begin{bmatrix} P(t+N) \\ P(t+(N-1)) \\ . \\ . \\ . \\ P(t) \end{bmatrix}$$

As can be seen, by the time the prediction is made N time samples into the future, the prediction is based solely on previously predicted values. Because each of these values has error, the prediction P(t+N) will also contain error. Thus, the confidence in these predicted values deteriorates for times t > 0.

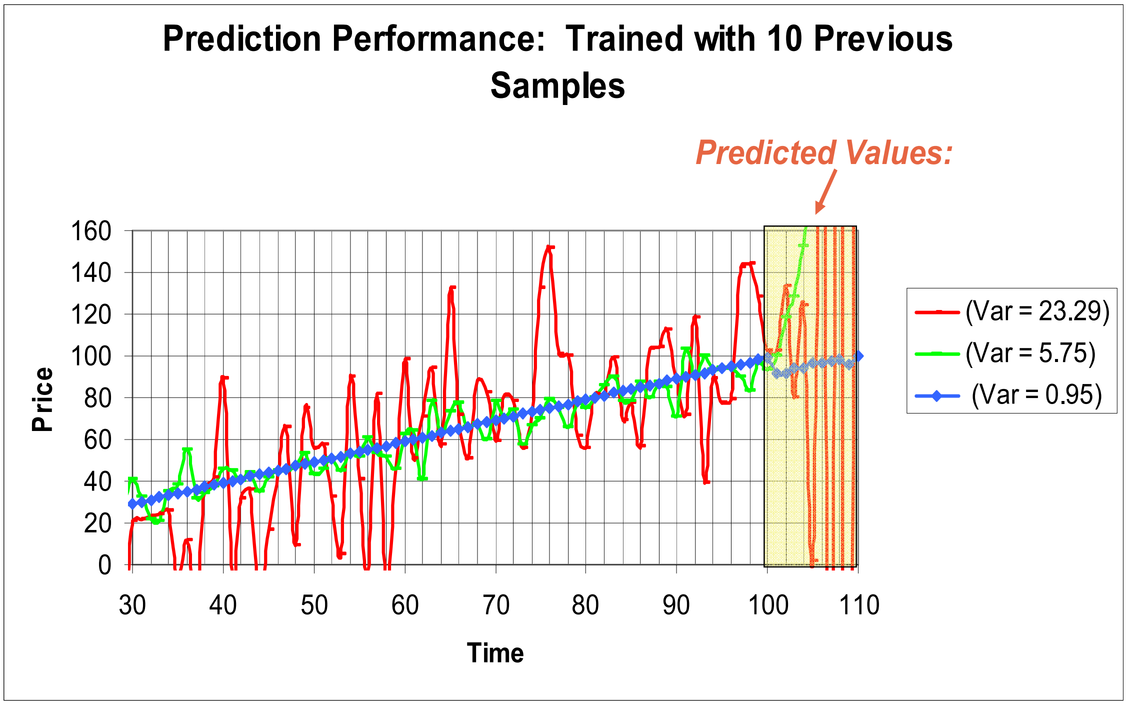
In order to use this model to predict future values, the vector “a” of weighting functions must be calculated. This can be done by “training” the system with a current stock value P(t) and N previous values P(t-1), P(t-2)… through P(t-N). Consider the below matrix equation:

$$\begin{bmatrix} P(t0-1) & P(t0-2) & .... & P(t0-N) \\ P(t-1-1) & P(t-1-2) & .... & P(t-1-N) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ P(t-J-1) & P(t-J-2) & .... & P(t-J-N) \end{bmatrix} * \begin{bmatrix} a1 \\ a2 \\ . \\ . \\ . \\ aN \end{bmatrix} = \begin{bmatrix} P(t0) \\ P(t-1) \\ . \\ . \\ . \\ P(t-J) \end{bmatrix}$$

As can be seen, the training algorithm involves calculating a vector of N weights based on J sets of input / output data for the stock. Selecting the number of J training sets to be used is an important problem that must be solved. If too few sets are used, the calculated vector of weights will be very noisy, and thus, future values that are predicted using these weights will be especially error prone. However, too many weights cannot be used, as we have only a finite amount of data available, and also, weights based on very old data will yield inaccurate future results.

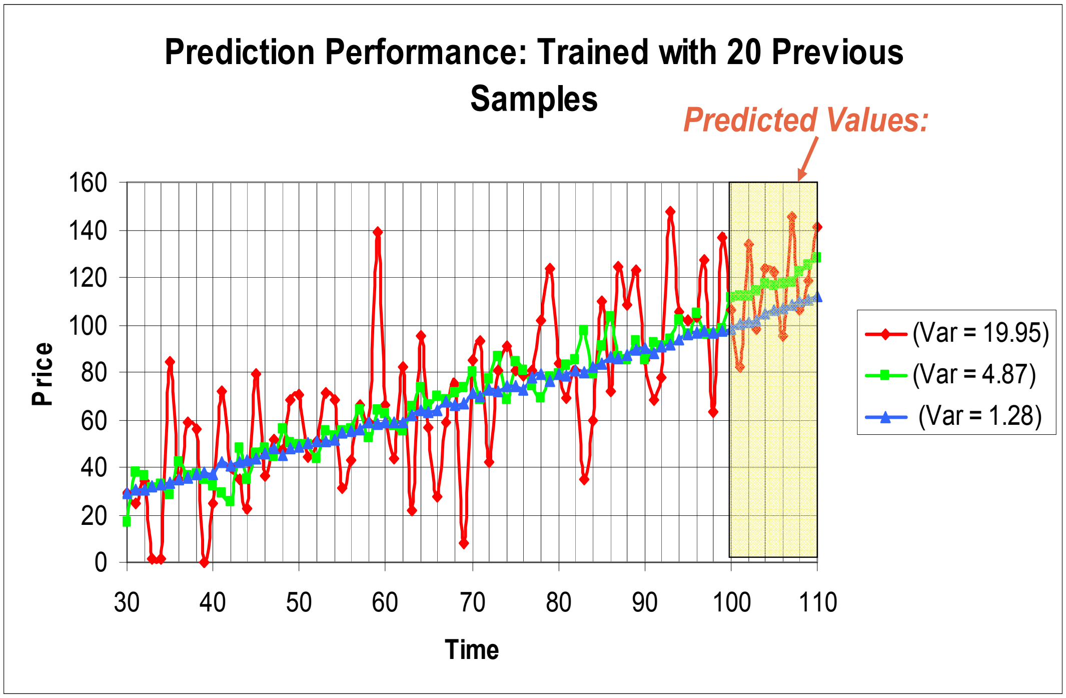
In order to solve this problem, vectors of test data are created in C#. The test case created is a continuation pattern, where a stock is bullish and the price is on an upward trend, despite “noise” which makes the stock price fluctuate away from the trend. The ability of the autoregressive model to predict this upward trend in presence of “noise” can be used as a figure of merit for deciding the optimal weight and training combination. The amount of noise in the price data can be quantified by the mean variance in the price data from a perfectly linear trend. The mean error of the prediction algorithm is calculated by the variance in predicted values from this same perfectly linear trend. The quality of the prediction can be quantified as the ratio of mean error to stock price variance, e.g., the predicted values for a stock which has a high variance (very noisy) would be expected to be more error prone then predicted values for a low variance stock (straight line). Thus, it makes sense to present the quality of the prediction as the ratio of prediction error to stock price variance.

The above test case was performed for a vector of 10 weights, trained with 10, 20, and 30 previous price samples. The below charts demonstrate this, with generated test vectors of 3 different variances. The test vector plotted below with the lowest variance appears relatively linear, while the test vector with the highest variance appears very random and stochastic. Thus, both the generated test vector of price data, as well as the 10 future predicted values, are shown on the below charts, as well as the ratio of mean prediction error to test vector variance.



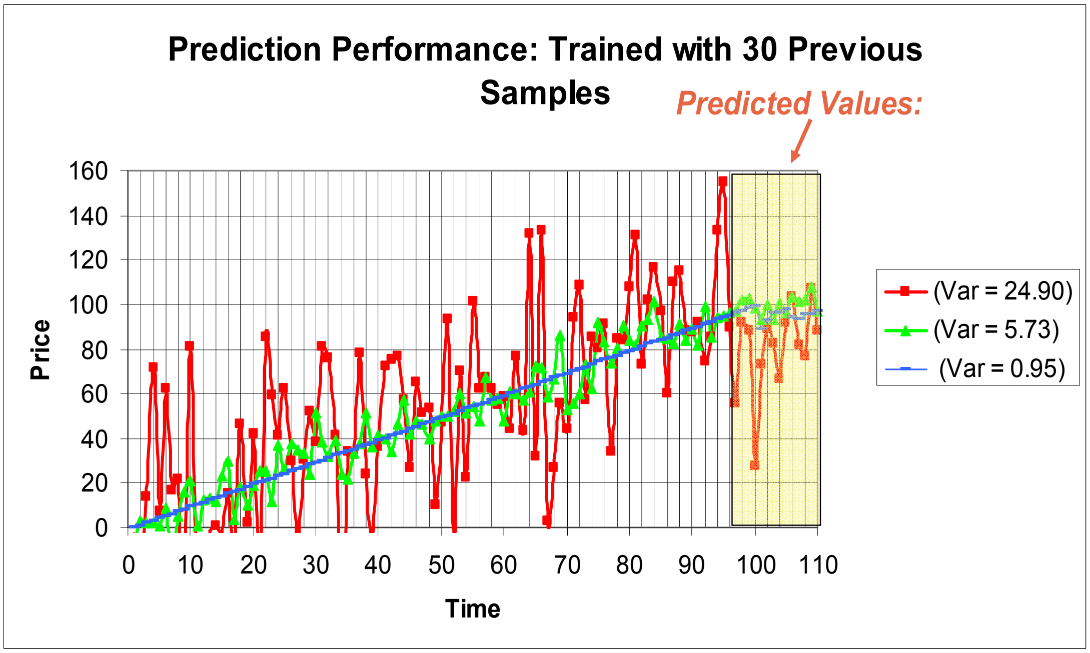
Mean Test Vector Variance:	*Mean Prediction Error:*	*10*LOG (Mean Error / Mean Variance):*
23.2943	457.017	12.93
5.751156	139.3164	12.84
0.956596	9.906232	10.15

It is clear from looking at the above chart that training with 10 previous data sets is insufficient. For the high variance data set, we can see that the predicted values oscillate wildly, yielding no valuable information. The ratio of mean error to mean variance, at almost 13 dB, says that the predicted values are 20 times noisier then the observed values, making this prediction worthless. The predicted values for the medium variance test vector is not much better; although the predicted values do not oscillate, they show an unrealistically rapid increase predicted. Again, the ratio of mean error to mean variance is almost 13 dB, indicating that a very poor prediction is being made.



Mean Test Vector Variance:	*Mean Prediction Error:*	*10 *LOG (Mean Error / Mean Variance):*
19.95251	18.36571	-0.36
4.867474	12.65057	4.14
1.287883	1.204104	-0.29

It is clear that there is a large improvement in increasing the number of training sets from 10 to 20; especially for the high variance case. For the highest variance data set, we the ratio of mean error to mean variance drop by over 13 dB. There is also an improvement in the mid variance case, with an 8 dB drop in mean error to mean variance. What is even more telling is a quick analysis of the above chart: As can be seen, the future values (shown shaded) show a reasonable relationship to the previous values; the very predictable values (shown in blue) have a well predicted continuation. The medium variance predicted values (shown in green) show a good prediction as well, it should be noticed that a jump around time 108 is predicted, which correlates well to the known previous value around time 98.



Mean Test Vector Variance:	*Mean Prediction Error:*	*10*LOG (Mean Error / Mean Variance):*
24.90513	19.46865	-1.07
5.731456	5.827956	0.073
0.953391	10.56933	-0.34

The above chart shows the prediction results when 30 training data sets are used to calculate the vector “a” of weights. For both the high and mid variance data sets, we again see an improvement in the mean error to mean variance ratio. As can be seen, there is a minimal improvement in performance when increasing the training sets from 20 to 30. Thus, 30 training sets will be used in the actual program. Thus, training the prediction algorithm requires a total of 40 values; 30 to complete the J x N matrix A, and 10 to complete the N x 1 matrix B. Then, the 10 element weighting vector can be calculated.

Once these training sets are used to calculate the weight vector “a”, future stock values can be predicted by multiplying these N weights by N previous values, as described above.

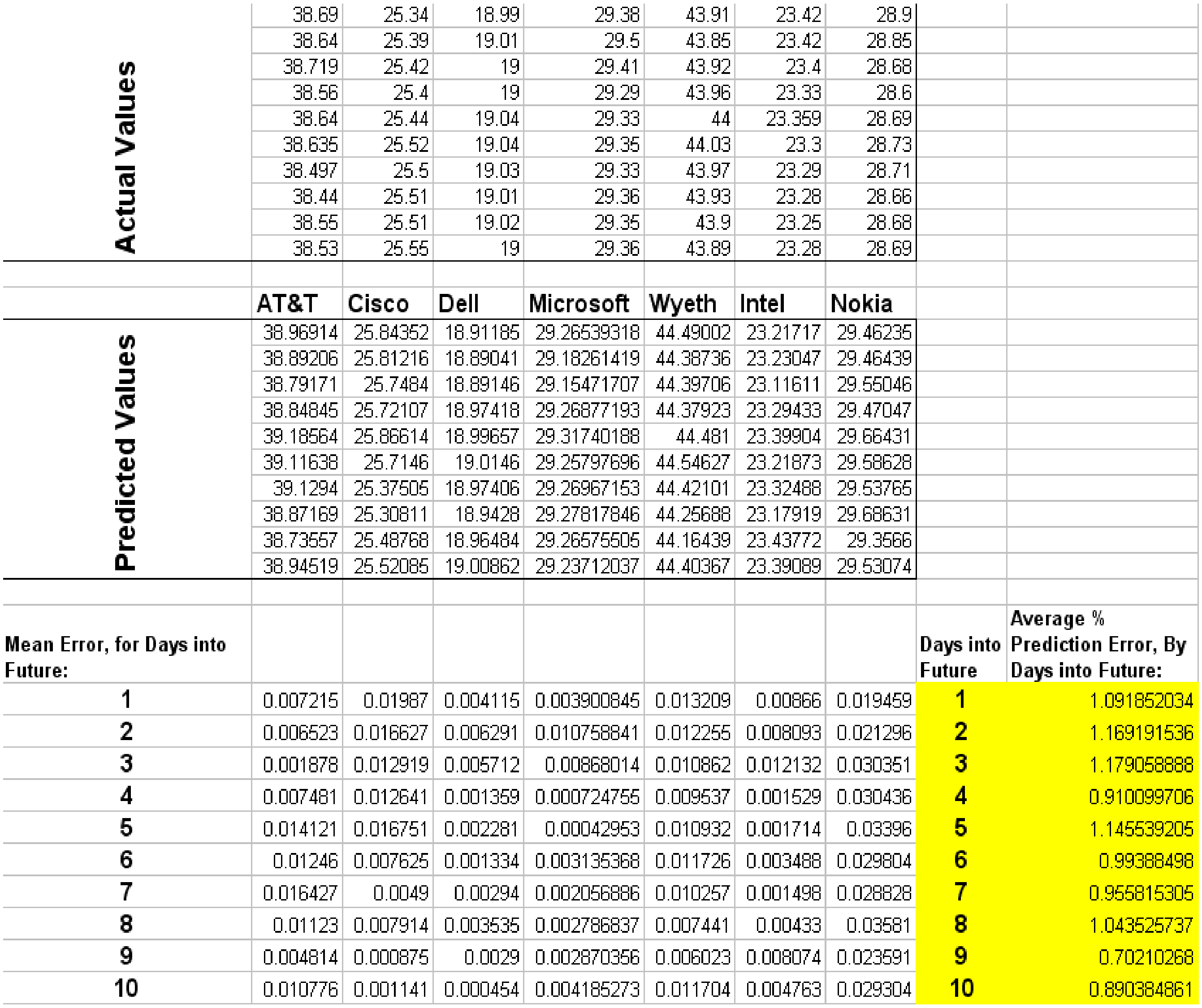
$$P(t) = \begin{bmatrix} a1 & a2 & \ldots & aN \end{bmatrix} * \begin{bmatrix} P(t-1) \\ P(t-2) \\ . \\ . \\ . \\ P(t-N) \end{bmatrix}$$

The confidence a user can have in the predicted values is heavily dependent on two factors; the length of time into the future the user requests a prediction for, and the variance in the stock price analyzed. The first factor needs little explanation; as can be seen above, the prediction of future values (at, say, a time (t+10)) will require calculations based on *other* predicted values (times (t+1) through (t+9)). Obviously, these future values are not completely accurate, and thus, the predicted value at (t+10) will in turn loose accuracy. The second factor, variance of a given stock, is also intuitive; the higher the variance of the stock, the less predictable it becomes. Both of these factors are taken into account with a weighted average, and a numerical indication of confidence is returned to the web service. The formula used is roughly:

$$Confidence = .8 * (10 - (days)) - .2(Var / MaxVar)$$

This formula for calculating confidence is not based on a reference, but is a logical assessment of the confidence of the prediction algorithm. Effectively, there is a 1 for 1 correlation between the number of days into the future we are asked to predict, and the number of past values included in the prediction. Thus, using a -1 to 1 slope between days into the future and confidence appears sensible, and this is the meaning of the $.8 * (10 - (days))$ argument in the above equation. However, the quality of the predicted prices will be related to the variance of the stock price data, as a stock with high variance in price will be more difficult to predict then one with low variance. Thus, the formula also includes a reduction in confidence for a high variance stock, and this is represented by the $-.2(Var / MaxVar)$ in the above equation. The final take away from this is that a stock with low variance, that is predicted only 1 to 2 days into the future, can be predicted with good confidence, while a stock that is "jumpy" (i.e., has high variance) cannot be predicted with much confidence far into the future.

The prediction algorithm was tested by using arrays of price data for 7 different companies. Instead of using data from (t-40) to (t-1) to predict stock values for times t>0, values (t-50) to (t-10) are used to predict stock values for times (t-10) to (t-1). Because these "predicted" values are known, the error of the predictions can be compared to actual values, and the mean error of the algorithm calculated. As will be seen, the mean error was relatively low, hovering around 1 % regardless of the number of days into the future the prediction is made for. In the table below, we see the 10 actual values (for each company), the 10 predicted values (for each company), and the mean error calculated:



Actual Values	38.69	25.34	18.99	29.38	43.91	23.42	28.9		
	38.64	25.39	19.01	29.5	43.85	23.42	28.85		
	38.719	25.42	19	29.41	43.92	23.4	28.68		
	38.56	25.4	19	29.29	43.96	23.33	28.6		
	38.64	25.44	19.04	29.33	44	23.359	28.69		
	38.635	25.52	19.04	29.35	44.03	23.3	28.73		
	38.497	25.5	19.03	29.33	43.97	23.29	28.71		
	38.44	25.51	19.01	29.36	43.93	23.28	28.66		
	38.55	25.51	19.02	29.35	43.9	23.25	28.68		
	38.53	25.55	19	29.36	43.89	23.28	28.69		
	AT&T	**Cisco**	**Dell**	**Microsoft**	**Wyeth**	**Intel**	**Nokia**		
Predicted Values	38.96914	25.84352	18.91185	29.26539318	44.49002	23.21717	29.46235		
	38.89206	25.81216	18.89041	29.18261419	44.38736	23.23047	29.46439		
	38.79171	25.7484	18.89146	29.15471707	44.39706	23.11611	29.55046		
	38.84845	25.72107	18.97418	29.26877193	44.37923	23.29433	29.47047		
	39.18564	25.86614	18.99657	29.31740188	44.481	23.39904	29.66431		
	39.11638	25.7146	19.0146	29.25797696	44.54627	23.21873	29.58628		
	39.1294	25.37505	18.97406	29.26967153	44.42101	23.32488	29.53765		
	38.87169	25.30811	18.9428	29.27817846	44.25688	23.17919	29.68631		
	38.73557	25.48768	18.96484	29.26575505	44.16439	23.43772	29.3566		
	38.94519	25.52085	19.00862	29.23712037	44.40367	23.39089	29.53074		
Mean Error, for Days into Future:								**Days into Future**	**Average % Prediction Error, By Days into Future:**
1	0.007215	0.01987	0.004115	0.003900845	0.013209	0.00866	0.019459	**1**	1.091852034
2	0.006523	0.016627	0.006291	0.010758841	0.012255	0.008093	0.021296	**2**	1.169191536
3	0.001878	0.012919	0.005712	0.00868014	0.010862	0.012132	0.030351	**3**	1.179058888
4	0.007481	0.012641	0.001359	0.000724755	0.009537	0.001529	0.030436	**4**	0.910099706
5	0.014121	0.016751	0.002281	0.00042953	0.010932	0.001714	0.03396	**5**	1.145539205
6	0.01246	0.007625	0.001334	0.003135368	0.011726	0.003488	0.029804	**6**	0.99388498
7	0.016427	0.0049	0.00294	0.002056886	0.010257	0.001498	0.028828	**7**	0.955815305
8	0.01123	0.007914	0.003535	0.002786837	0.007441	0.00433	0.03581	**8**	1.043525737
9	0.004814	0.000875	0.0029	0.002870356	0.006023	0.008074	0.023591	**9**	0.70210268
10	0.010776	0.001141	0.000454	0.004185273	0.011704	0.004763	0.029304	**10**	0.890384861

Because this prediction algorithm requires very matrix intensive computations, a set of C# functions (operating under the namespace "Mapack") which is useful for creating and modifying matrices is used. Mapack is a .NET class library for basic linear algebra computations, and is available online free of charge, from the website http://www.aisto.com/roeder/dotnet. The class library contains the functions:

```
public class CholeskyDecomposition

public class EigenvalueDecomposition

public class LuDecomposition

public class QrDecomposition

public class SingularValueDecomposition

public class Matrix
```

Of these functions, the function `public class Matrix:` is the only one called from the main function Prediction, and is used in setting up all matricies. The Prediction function uses multi-dimensional C# arrays to store and display all stock data, and then calls the Matrix function to perform all mathmatical operations.

6. Data collection

Data selection and preprocessing constitute a crucial step in any modeling effort. There are so many stocks that are traded at the stock market. As a stock prediction developer we need to choose wisely the range of stocks that we would advise our investors to invest in. There are different types of stocks that different investors would be interested in depending on what type of investor he/she is. However for our system we employed the following thought process:

We thought of stocks as companies and businesses instead of stocks. That's an important distinction that's lost on too many people. Stocks are, after all, actual small pieces of an ongoing business, not lottery-like slips of paper.

Next, we did not make the mistake that many people make -- basing a decision to choose stocks in a company on just one or two factors. For example, some people see what they think is a low price-to-earnings ratio (P/E) and think that's enough. Or maybe they read an article about a company that made it seem irresistible.

Instead, take the time to learn how to evaluate companies. We read financial statements and used P/E ratios along with *many other* numbers. Read widely about companies that cause interest in the current marker ideally in industries that we has a potential boom in the future.

Over all we come across many things to consider when evaluating a company after our research. The more we learn, the better our investment decisions and performance are likely to be.

Based on the above thought we chose the following stocks for our system :
1) Apple
2) Microsoft
3) Intel
4) Nokia
5) Oracle
6) Dell
7) Canon
8) Texas Instruments
9) EMC Corporation

10) Motorola
11) AMD
12) Johnson & Johnson
13) Pfizer
14) Glaxosmithkline
15) Novartis
16) Merck
17) Astra Zeneca
18) Northrop
19) Amgen
20) Sybase
21) Abbot Labs
22) Wyeth
23) Genentec
24) Assisted Living Concepts
25) Citigroup
26) Bank of America
27) HSBC
28) Berkshire Hathway
29) Agilent
30) JP Morgan
31) Allianz
32) Wachovia
33) China Life Insurance Corporation
34) Goldman Sachs
35) Barclays Capital
36) AXA
37) Exxon Mobile
38) General Motors
39) Chevron
40) Barnes Group
41) Occidental Petroleum
42) Imperial Oil
43) China Mobile
44) T Mobile
45) Vodafone
46) Google
47) Walmart
48) Verizon
49) Akamai Technologies
50) Chile Fund

The phases of data preparation can be broadly classified into three distinct areas: variable selection and data collection and data inspection.

a) Variable Selection: We selected to get the following variable variables to get as they are freely available. Some of these could be used for future development when the need be.
1)Stock ID
2)Stock Value
3)Change Points
4)Open Value
5)Intra day top
6)Intra day bottom
7)Date
8)Time

b) Data Collection: The data were obtained using a php-based query interface from Yahoo Finance! (http://finance.yahoo.com). The stock market timings are from 9:00am morning to 4:30pm in the evening. During this time the stock data were collected. We chose yahoo finance as it is free source of stock data and it is one of the most reliable sources. The stock data from yahoo finance are delayed by five minutes.

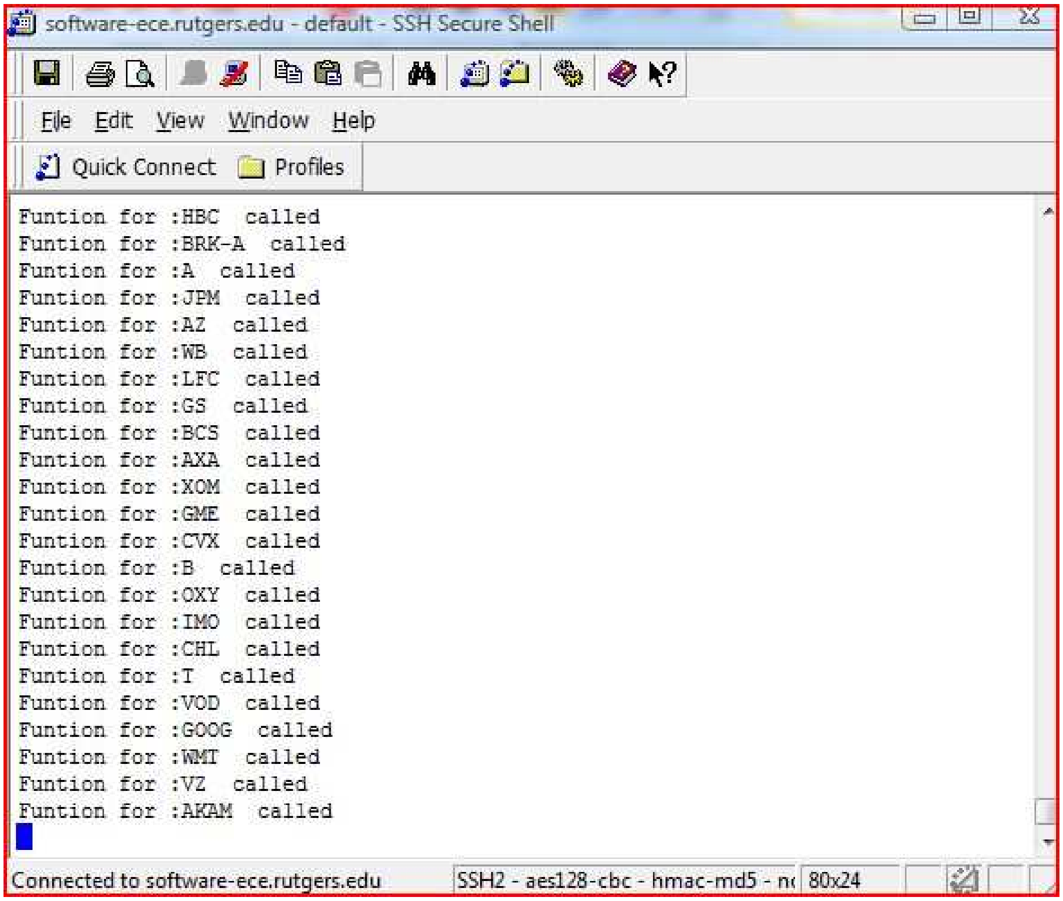
The methodology used to collect the data is described as follows. After the yahoo_stocks class is initiated we use the function $stocks->my_connect(); to connect to the SQL server where our database group803 contains the table "STOCKS". Two loops are used to get the data from yahoo finance. When the stock market is closed the program runs in the outer loop where no action is performed and the outer loop is repeated continuously. The status of the stock market is determined from the time() function in Unix. When the stock market opens the program goes into the inner loop the php query gets initiated and the following functions are called

```
$apple = $stocks->get_stocks("AAPL");

$cisco = $stocks->get_stocks("CSCO");

$msft = $stocks->get_stocks("MSFT");
```

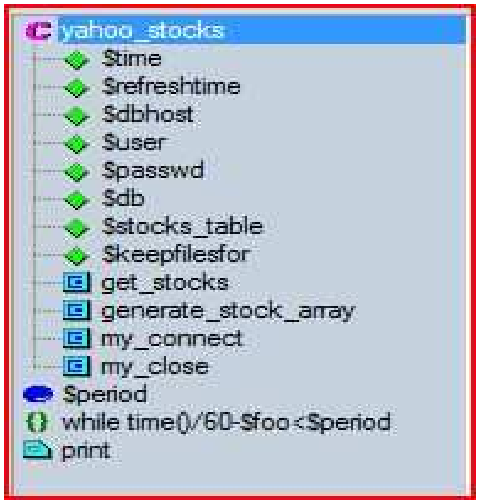
ect. for all the 50 stocks as shown in the figure in next page.


software-ece.rutgers.edu - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to software-ece.rutgers.edu
SSH2 - aes128-cbc - hmac-md5 - n
80x24

```
Funtion for :HBC  called
Funtion for :BRK-A  called
Funtion for :A  called
Funtion for :JPM  called
Funtion for :AZ  called
Funtion for :WB  called
Funtion for :LFC  called
Funtion for :GS  called
Funtion for :BCS  called
Funtion for :AXA  called
Funtion for :XOM  called
Funtion for :GME  called
Funtion for :CVX  called
Funtion for :B  called
Funtion for :OXY  called
Funtion for :IMO  called
Funtion for :CHL  called
Funtion for :T  called
Funtion for :VOD  called
Funtion for :GOOG  called
Funtion for :WMT  called
Funtion for :VZ  called
Funtion for :AKAM  called
```

Fig-7 List of the stocks

The function get_stocks($stock) internally calls the function
$return = $this->generate_stock_array($stock) to connect to the yahoo server.
The generate_stock_array($stock) function internally uses function fopen( ) to get the array of stock parameters as shown below.
$open=fopen("http://finance.yahoo.com/d/quotes.csv?s=$stock&f=sl1d1t1c1ohgv&e=.csv", "r");
It reads the appropriate parameters from the received array and returns these values.Then these values are written in the table "STOCKS" from the database group803 on the ECE server at Rutgers University. In pictorial form the functions can be represented as shown in figure.



```
yahoo_stocks
    $time
    $refreshtime
    $dbhost
    $user
    $passwd
    $db
    $stocks_table
    $keepfilesfor
    get_stocks
    generate_stock_array
    my_connect
    my_close
$period
while time()/60-$foo<$period
print
```

Fig 8. The Functions and the variables used in pictoral form

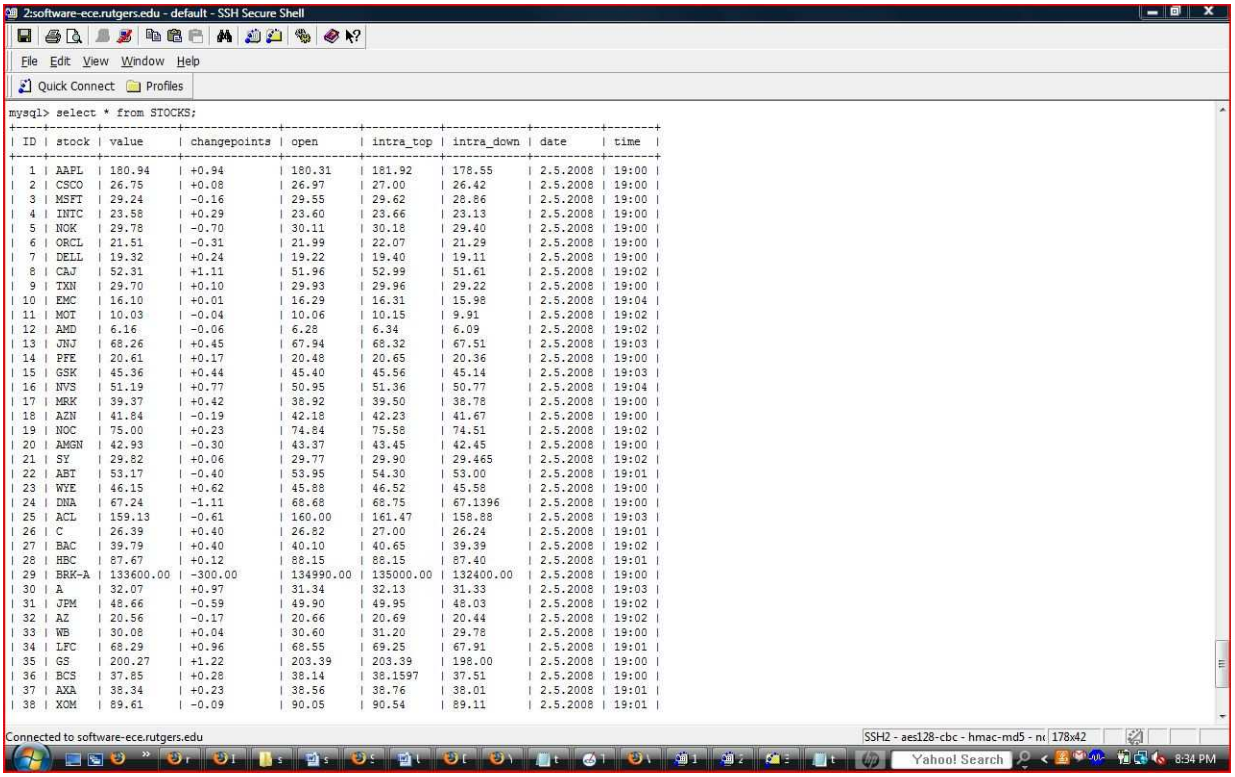
The database schema and the actual database are shown in figure 9 & 10:

mysql> describe STOCKS;

Field	Type	Null	Key	Default	Extra
ID	int(11)		PRI	NULL	auto_increment
stock	varchar(64)				
value	varchar(16)				
changepoints	varchar(16)				
open	varchar(16)				
intra_top	varchar(16)				
intra_down	varchar(16)				
date	varchar(10)				
time	varchar(6)				

9 rows in set (0.00 sec)

Fig 9. The Database Schema


2:software-ece.rutgers.edu - default - SSH Secure Shell
File
Edit
View
Window
Help
Quick Connect
Profiles
SSH2 - aes128-cbc - hmac-md5 - n(
178x42
Yahoo! Search
8:34 PM

```
mysql> select * from STOCKS;
+----+-------+-----------+--------------+-----------+-----------+------------+----------+-------+
| ID | stock | value     | changepoints | open      | intra_top | intra_down | date     | time  |
+----+-------+-----------+--------------+-----------+-----------+------------+----------+-------+
|  1 | AAPL  | 180.94    | +0.94        | 180.31    | 181.92    | 178.55     | 2.5.2008 | 19:00 |
|  2 | CSCO  | 26.75     | +0.08        | 26.97     | 27.00     | 26.42      | 2.5.2008 | 19:00 |
|  3 | MSFT  | 29.24     | -0.16        | 29.55     | 29.62     | 28.86      | 2.5.2008 | 19:00 |
|  4 | INTC  | 23.58     | +0.29        | 23.60     | 23.66     | 23.13      | 2.5.2008 | 19:00 |
|  5 | NOK   | 29.78     | -0.70        | 30.11     | 30.18     | 29.40      | 2.5.2008 | 19:00 |
|  6 | ORCL  | 21.51     | -0.31        | 21.99     | 22.07     | 21.29      | 2.5.2008 | 19:00 |
|  7 | DELL  | 19.32     | +0.24        | 19.22     | 19.40     | 19.11      | 2.5.2008 | 19:00 |
|  8 | CAJ   | 52.31     | +1.11        | 51.96     | 52.99     | 51.61      | 2.5.2008 | 19:02 |
|  9 | TXN   | 29.70     | +0.10        | 29.93     | 29.96     | 29.22      | 2.5.2008 | 19:00 |
| 10 | EMC   | 16.10     | +0.01        | 16.29     | 16.31     | 15.98      | 2.5.2008 | 19:04 |
| 11 | MOT   | 10.03     | -0.04        | 10.06     | 10.15     | 9.91       | 2.5.2008 | 19:02 |
| 12 | AMD   | 6.16      | -0.06        | 6.28      | 6.34      | 6.09       | 2.5.2008 | 19:02 |
| 13 | JNJ   | 68.26     | +0.45        | 67.94     | 68.32     | 67.51      | 2.5.2008 | 19:03 |
| 14 | PFE   | 20.61     | +0.17        | 20.48     | 20.65     | 20.36      | 2.5.2008 | 19:00 |
| 15 | GSK   | 45.36     | +0.44        | 45.40     | 45.56     | 45.14      | 2.5.2008 | 19:03 |
| 16 | NVS   | 51.19     | +0.77        | 50.95     | 51.36     | 50.77      | 2.5.2008 | 19:04 |
| 17 | MRK   | 39.37     | +0.42        | 38.92     | 39.50     | 38.78      | 2.5.2008 | 19:00 |
| 18 | AZN   | 41.84     | -0.19        | 42.18     | 42.23     | 41.67      | 2.5.2008 | 19:00 |
| 19 | NOC   | 75.00     | +0.23        | 74.84     | 75.58     | 74.51      | 2.5.2008 | 19:02 |
| 20 | AMGN  | 42.93     | -0.30        | 43.37     | 43.45     | 42.45      | 2.5.2008 | 19:00 |
| 21 | SY    | 29.82     | +0.06        | 29.77     | 29.90     | 29.465     | 2.5.2008 | 19:02 |
| 22 | ABT   | 53.17     | -0.40        | 53.95     | 54.30     | 53.00      | 2.5.2008 | 19:01 |
| 23 | WYE   | 46.15     | +0.62        | 45.88     | 46.52     | 45.58      | 2.5.2008 | 19:00 |
| 24 | DNA   | 67.24     | -1.11        | 68.68     | 68.75     | 67.1396    | 2.5.2008 | 19:00 |
| 25 | ACL   | 159.13    | -0.61        | 160.00    | 161.47    | 158.88     | 2.5.2008 | 19:03 |
| 26 | C     | 26.39     | +0.40        | 26.82     | 27.00     | 26.24      | 2.5.2008 | 19:01 |
| 27 | BAC   | 39.79     | +0.40        | 40.10     | 40.65     | 39.39      | 2.5.2008 | 19:02 |
| 28 | HBC   | 87.67     | +0.12        | 88.15     | 88.15     | 87.40      | 2.5.2008 | 19:01 |
| 29 | BRK-A | 133600.00 | -300.00      | 134990.00 | 135000.00 | 132400.00  | 2.5.2008 | 19:00 |
| 30 | A     | 32.07     | +0.97        | 31.34     | 32.13     | 31.33      | 2.5.2008 | 19:03 |
| 31 | JPM   | 48.66     | -0.59        | 49.90     | 49.95     | 48.03      | 2.5.2008 | 19:02 |
| 32 | AZ    | 20.56     | -0.17        | 20.66     | 20.69     | 20.44      | 2.5.2008 | 19:02 |
| 33 | WB    | 30.08     | +0.04        | 30.60     | 31.20     | 29.78      | 2.5.2008 | 19:00 |
| 34 | LFC   | 68.29     | +0.96        | 68.55     | 69.25     | 67.91      | 2.5.2008 | 19:01 |
| 35 | GS    | 200.27    | +1.22        | 203.39    | 203.39    | 198.00     | 2.5.2008 | 19:00 |
| 36 | BCS   | 37.85     | +0.28        | 38.14     | 38.1597   | 37.51      | 2.5.2008 | 19:00 |
| 37 | AXA   | 38.34     | +0.23        | 38.56     | 38.76     | 38.01      | 2.5.2008 | 19:01 |
| 38 | XOM   | 89.61     | -0.09        | 90.05     | 90.54     | 89.11      | 2.5.2008 | 19:01 |
```

Connected to software-ece.rutgers.edu

Figure 10. The Database

After the functions for all the stocks are called the inner loop goes in for sleeping for period equal the value of the variable $interval. After $interval minutes of sleep the inner loop functions are repeated and the program execution stays in the inner loop. When the stock market closes for the day the execution goes into the outer loop and this process repeats.

Sequence of Activities:

Initiating Actor: The initiating actor is the Timer, which is set to initiate stock update request on a regular basis.

1.Actor's Goal: To gather stock quote information from http://finance.yahoo.com and store the results in Stock Database on our local machine.

2.Participating actors: Stock List, Stock Attribute File, Stock Database, Yahoo! Finance Library

3.Preconditions: The Timer Interval has been set, and the Stock List and the Stock database been defined. The Stock Database exists and implements the expected schema.

4.Trigger: The Timer times out.

Post conditions: The Stock Database is updated. The Timer is reset.
→ 1. Timer initiates STOCKS Database update.
← 2. System reads Stock List.
← 3. System reads Stock Attribute File.
← 4. System calls Yahoo! Finance Library.
← 5. System copies the information returned by the Yahoo! Finance Library into the Stock Database.
← 7. System resets Timer.

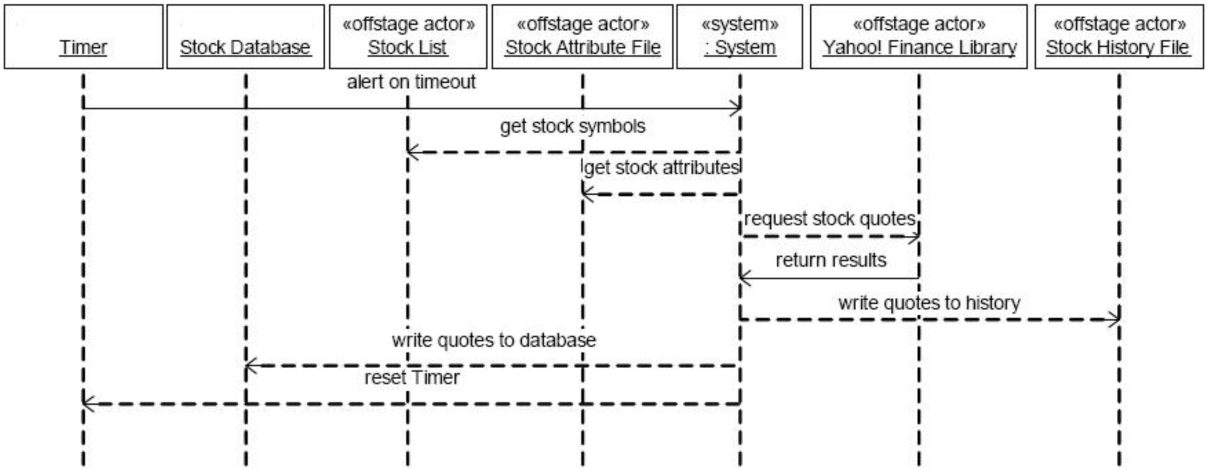


Figure 11. The order of the activities

c) Data inspection
Even though the data were obtained from a reliable source, the data was manually checked of errors and it was reported that the data retrieved was fairly correct.

The Data Collection System has been developed with easy to adopt changes from the system administrator. The system administrator could just change the value of $interval which is globally defined to get data for different time interval.

Set Timer Interval

□ Initiating Actor: Server Administrator
□ Actor's Goal: Change the Stock Database update interval
□ Participating Actors: Timer
□ Preconditions: None.
□ Post conditions: The Timer Interval has been set changed

7. Connect_Database Method:

Description:

The purpose of the Connect_Database function is to connect to a server, retrieve specific data off of that server and pass the values to a function which calls it. In the instance of this program, the calling function is the web service, the server it connects to contains a MySQL database of stock data, and the values it retrieves are the prices of a particular stock.

The Connect_Database function is called from the Predictor function and is passed a string representing the ticker symbol. Based on that ticker symbol, the Connect_Database function will query the corresponding table on the server in the database and store the past stock values in an array. It will return these values to the calling function, in this case the prediction function.

Input Parameters:

The input parameters are passed in from the calling function, in this case from the stock prediction algorithm.

String Tickersymbol(string)**:** This input is the ticker symbol for the stock of interest. It is used as a reference to select the specific stock in the MySQL database.

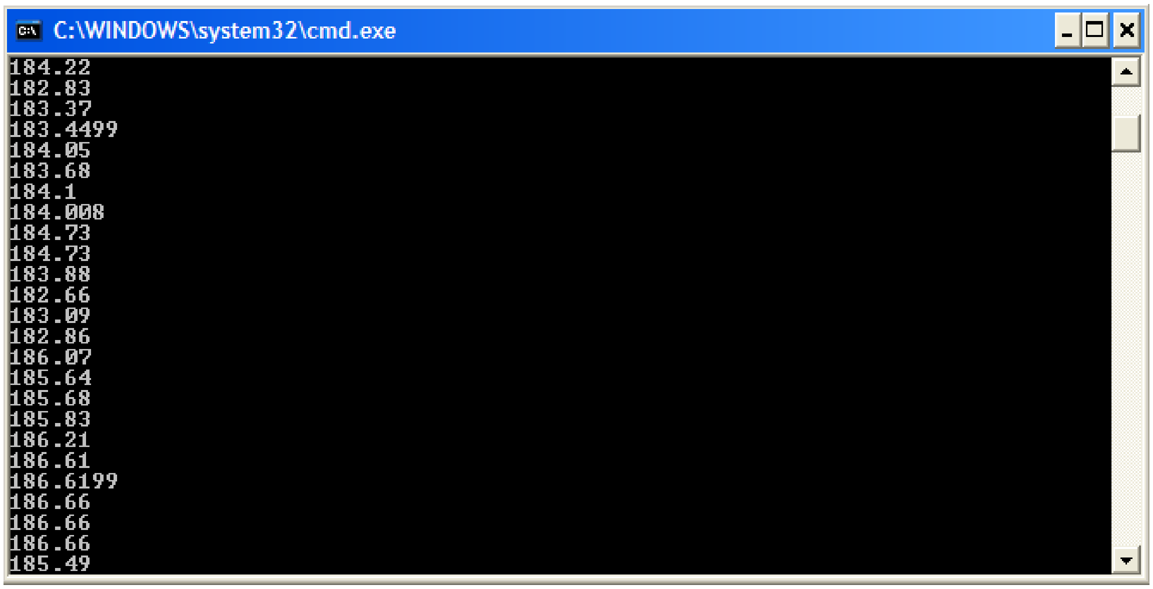
Output Parameters:

The output parameter is an array that is returned to the calling function. It contains the retrieved stock data from the data collection algorithm.

Double [] price(double)**:** This output is an array that contains past prices of a stock for a particular ticker symbol.

```
********************************************************************
    public static double[] connect_database(string ticker)
********************************************************************
```

Below is a screen capture. What is depicted in the image is the output of the double array containing the stock values. The most recent value is added to the bottom of the list. These values correspond to the stock “APPLE” with symbol “AAPL”.



```
184.22
182.83
183.37
183.4499
184.05
183.68
184.1
184.008
184.73
184.73
183.88
182.66
183.09
182.86
186.07
185.64
185.68
185.83
186.21
186.61
186.6199
186.66
186.66
186.66
185.49
```

The Connect_Database function is written in C# and takes advantage of the MySQL connector. The MySQL connector is an interface that enables connections to be made to a MySQL server via a .NET programming language such as C#. It can be downloaded from http://dev.mysql.com/downloads/connector/net/1.0.html. This connector contains a library of functions that can be accessed, so long as the programmer types *using MySql.Data.MySqlClient;* before defining any classes. Below is a brief outline of the commands available from the MySQL connector in C#.

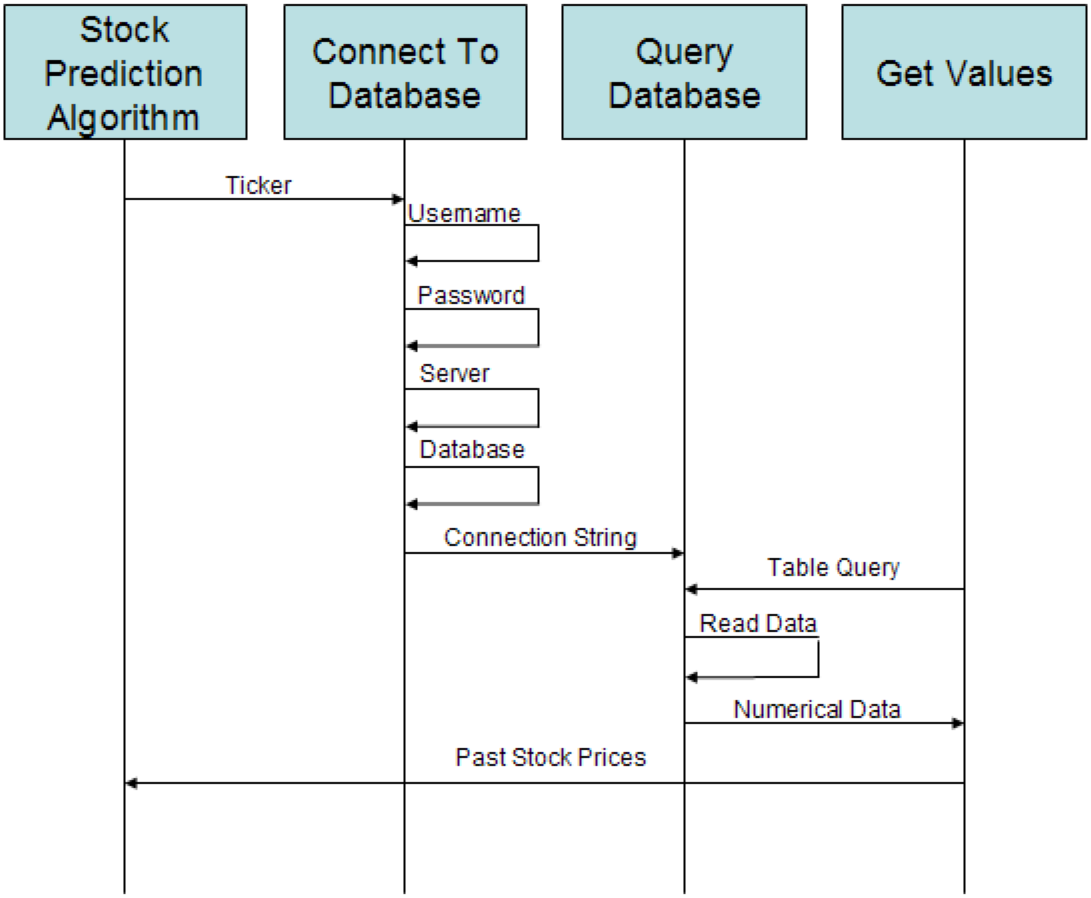
MySqlConnection: Enables a new instance of a mysql connection and saves it to a variable. It takes in four arguments: Userid (the user name for the server), database (the database intended to be used on the server, typically called from the use command in mysql), server (the IP address of the server), and password (the password corresponding the user name for the server).

MySqlCommand: Enables programmer to input MySQL commands directly into C# code and save it to a string. It takes in two arguments: the first being the mysql command string and the second being the mysqlconnection instance variable.

MySqlDataReader: Enables programmer to execute MySQL commands and save the data output query to a variable or array.

We begin our stock data query by creating a MySQL connection. The server providing this data to us is saved on the Rutgers ECE Server at software-ece.rutgers.edu (a ping command in DOS yields IP 128.6.29.169). The user name on this server is group803, the password is SE#287, and the database we use is named group803. In the database group803 there is a table called STOCKS which contains the for each stock. We make two queries one for an array of stock prices for a particular stock.

The way this function works is outlined the following data-flow diagram.



As shown above the stock prediction algorithm sends a ticker symbol into the database connection function. The function will supply the appropriate username, password, server, and database to use. It will then open up a connection.

```
// Enter in Database Information
        string username = "group803";
        string password = "SE#287";
        string server = "128.6.29.169";
        string database = "group803";

// Connect to Database
        string connString ="Userid=" + username + ";database=" +
database + ";server=" + server + ";password=" + password + "";
        MySqlConnection conn = new MySqlConnection(connString);
        conn.Open();
```

Here, the connection variable “conn” contains the instance of the mysql connection and will be used when applying mysql commands. We now wish to query the database. We will need the connection variable “conn” as well as the Table, Record, and column from we wish to query.

```
// Query the Database
```

```
string table = "STOCKS";
string record = "stock";
        string column = "value";
        string cmdString = "select " + column + " from " + table + "
        where " + record + "=\"" + ticker + "\";";
        MySqlCommand comm;
        MySqlDataReader dr = comm.ExecuteReader();
```

We now must read in the data returned from this query. One thing to keep in mind is that this data is of type object, and it needs to be converted to type double[] before returning to the prediction algorithm for compatibility.

```
// Create array
        int j = 0;
        double[] data = new double[i];
        while (dr.Read())
        {
            if (j <= i)
            {
                data[j] = Convert.ToDouble(dr.GetValue(0));
            }
            j = j + 1;
        }
```

Finally, we must close the connection to the database and return the values to the prediction algorithm.

```
// Disconnect from Database
        dr.Close();
        conn.Close();

        // Return values
        return data;
```

8.HARDWARE REQUIREMENT:

The minimum hardware requirements of our system include the following:

- **Screen resolution** 600*800,
- **RAM** 128 Mb,
- **Hard disk** 10 GB,
- **Processor**- 450 MHz, 66 MHz FSB, 64 KB cac nhe,
- **Multimedia keyboard**.
- **56 kbps internet**.
- **Mouse**.
- **Scanner**.
- **Printer**.

9. HISTORY OF WORK

#	NAME	DURATION	START	FINISH	DEADLINE
1	Project group meeting 1	1d	1/22/2008	1/22/2008	
2	Started working on project proposal	3 d	1/22/2008	2/7/2008	
3	Downloading stock information and storing it in a MySQL database	10 d	2/17/2008	02/28/2008	02/28/2008
4	Project Presentation #1	5 d	03/02/2008	03/06/2008	03/07/2008
5	Project group meeting 2	1d	03/07/2008	03/07/2008	
6	Understand how a webservice work	3 d	03/17/2008	03/20/2008	
7	Interim Project Report		03/20/2008	03/28/2008	03/28/2008
8	Developing the skeleton of the Web service	10d	03/19/2008	03/29/2008	03/31/2008
9	Review report 1	3 d	04/07/2008	04/10/2008	
10	Start work for Demo 1	6 d	04/10/2008	04/16/2008	
11	Project group meet 3	1 d	04/16/2008	04/16/2008	
12	Coding for final demo	15 d	04/16/2008	05/01/2008	05/05/2008
13	Project group meet 4	1d	04/20/2008	04/20/2008	
14	Group meet 4.1	1d	04/30/2008	04/30/2008	
15	Group meet 4.2	1 d	05/02/2008	05/02/2008	
16	Project Group meet	1 d	05/06/2008	05/06/2008	
17	Writing final report	7 d	04/28/2008	05/07/2008	05/07/2008
18	System testing	1 d	05/02/2008	05/02/2008	05/06/2008
19	System Fine tuning	7 d	05/06/2008	05/06/2008	05/06/2008
20	System Fine tuning	1 d	05/07/2008	05/07/2008	05/06/2008
21	DEMO	1 d			05/09/2008

10. CONCLUSION AND FUTURE WORK:

Technical challenges encountered: The primary role of software engineering is to address the development of systems as an assembly of components, the development of components as reusable entities, and the maintenance and upgrading of systems by customizing and replacing such parts. This requires established methodologies and tool support covering the entire component and system lifecycle including technological, organizational, marketing, legal and other aspects.

This project was about development of “Stock market investment advisory” a stock market prediction advisor. During this project we encountered many technical challenges for the website development. The website was basically developed using the following:
HTML(For the website designing)
MatLab(for coding of prediction algorithm)
C#(For the web service and for connecting the Mat Lab code with the webservice and data connector).

We had to use the internet resources and go through several books to become familiar with these languages. Working with HTML and SQL was not very difficult but using C# was a real challenge because it was the first time we were dealing with it. Also we worked with “ssh” and we were unfamiliar about how to use it, but with passage of time we made ourselves comfortable with it.

The second challenge was learning and getting information about financial economics. Since we hardly had any information about how the financial market works, we had to go through and refer a lot of financial web sites and grasp knowledge about their working and functioning. We had to gather a lot of information about the functioning of stock exchange markets and then use that information in the development of our project.

The third challenge was regarding making decisions. Since we were working in groups, so we were not supposed to make independent decisions and so we conducted regular meetings and discussed various issues and accordingly took decisions. We had to make sure that our website was impressive enough to attract the advertisers as well as simple (user friendly) enough so that different kinds of users can use it without difficulty. We had
to take care about how social our website is because the socialization of media and applications has emerged as a trend of socializing things which are not inherently social such as photo, videos and text.

To deal with these challenges, we analyzed problem statements, asked questions, conducted further research, discussed with team members during meetings and plenary sessions and then developed solutions in different forms.

We got a lot of classroom support with the help of which we got rid of a lot of problems and doubts. Our class almost looked like an academic classroom but actually we were in a real work-like environment with a team of people who were all working together to accomplish same goals. On the other hand, we were introduced to more software lifecycle management in the class which helped us in the development and establishment of our project.

In future, we can try our best to make our project practical and usable. We could try to make it a real world working website by using more rigorous algorithm .

11. REFERENCES

1. Franklin Allen and Risto Karjalainen Using genetic algorithms to nd technical trading rules
2. Technical report, Rodney L White Center for Financial Research 1995
3. William Brock, Josef Lakonishok, and Blake LeBaron Simple technical trading rules and the stochastic properties of stock returns Journal of Finance 47(5):1731:1764
4. Andrew McCallum and Kamal Nigam A comparison of event models for naive bayes text classification In AAAI98, Workshop on Learning for Text Categorization 1998.
5. http://volume.technicalanalysis.org.uk/ThSy00.pdf
6. Software Engineering by Ivan Marsic, Rutgers University
7. www.wikipedia.com
8. www.uml.org/tutorials
9. Visual Paradigm for UML – Enterprise edition (Help)
10. www.omondo.com/tutorials
11. EclipseUML 3.3.0.20071003 (Help)
12. EclipseUML 3.3.0.20072003 Studio (Help)
13. Borlands UML tutorial
14. Introduction to OMG UML by Dr. John Siegel
    Lo, Andrew W., Harry Mamaysky, Jian Wang Foundations of Technical Analysis: Computational Algorithms, Statisitcal Inference, and Emperical Implementation. The Journal of Finance, Volume 60, No. 4, August 2000
15. O'Brien, Paul. Chapter 9. Linear Prediction Filters and Nural Networks. www-ssc.igpp.ucla.edu/personnel/russell/ESS265/Ch**9**/**linear**_predict/index.html - 38k,