AN INTERACTIVE STOCK MARKET WEB PORTAL WITH ANALYTICAL CAPABILITIES

BY

DONGXIAO WANG

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirement for the Degree of Master of Science

Southern Connecticut State University
New Haven, Connecticut
August 2007
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI Microform 1449390
Copyright 2008 by ProQuest Information and Learning Company. All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346
AN INTERACTIVE STOCK MARKET WEB PORTAL WITH ANALYTICAL CAPABILITIES

BY

DONGXIAO WANG

This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Hrvoje Podnar, Department of Computer Science, and it has been approved by the members of the candidate's thesis committee. It was submitted to the School of Graduate Studies and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

Hrvoje Podnar, Ph.D.
Thesis Advisor

Amal Abd El-Raouf, Ph.D.
Second Reader

Winnie Yu, M.S.
Department Chairperson

Sandra C. Holley, Ph.D.
Dean, School of Graduate Studies

September 10, 2007
Date
ABSTRACT

Author: Dongxiao Wang  
Title: AN INTERACTIVE STOCK MARKET WEB PORTAL WITH ANALYTICAL CAPABILITIES  
Thesis Advisor: Dr. Hrvoje Podnar  
Institution: Southern Connecticut State University  
Year: 2007  

Stock market behavior influences not just activities of financial institutions but also activities of larger population of individuals. To exploit market inefficiency, computer technology and Internet play important role. There are various stock analysis programs available as commercial products and customized tools. Many of them are based on technical analysis approach. They range from chart patterns, databases to artificial intelligence technology. This thesis provides a collection of such tools organized within an online portal. The stock portal relies on stock historical trade data to provide different types of charts presentation with different time frames. The charts provide users with an educated insight into the stock data that can be used to direct the future investments.  

Users can filter and personalize their data views based on their saved portfolios. The portal and the personalized data can be reached from any Internet enabled computer in the world.
ACKNOWLEDGMENTS

First, I would like to thank my family: Kuanping and Sikai, my parents: Guanhua and Beilan, for their help, support and understanding.

I would also like to thank my advisor, Dr. Hrvoje Podnar, for his help and encouragement with this project as well as his help throughout the Master’s program. Without his enthusiastic guidance, it would have been impossible to complete it.

I would like to thank Dr. Amal Abd El-raouf, Dr. Lisa Lancor and Professor Winnie Yu for reviewing my thesis, proposal and giving suggestions on it. Furthermore, I would like to thank all those who gave me the possibility to complete this project and for making this a most education and enjoyable experience.
# TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION ........................................................................................................... 1

1.1 Equity Trading Terminology .............................................................................................. 1

1.1.1 Stock Charts ................................................................................................................. 2

1.1.2 Technical Analysis Approach ......................................................................................... 5

1.1.3 Intelligent Prediction .................................................................................................... 9

1.2 Literature Review ............................................................................................................ 12

CHAPTER 2 OVERALL PROJECT DESIGN ................................................................................. 16

2.1 Motivation ....................................................................................................................... 16

2.2 System Architecture ........................................................................................................ 17

2.3 Development Technologies and Tools ............................................................................ 19

2.4 Conceptual Design .......................................................................................................... 22

CHAPTER 3 FUNCTIONALITY AND WEB SITE STRUCTURE ..................................................... 26

3.1 Application Functionality ................................................................................................. 26

3.1.1 Stock Look-Up ............................................................................................................. 26

3.1.2 Stock Chart ................................................................................................................. 28

3.1.3 Statistical Analysis ..................................................................................................... 30

3.1.4 My Portfolio ............................................................................................................... 31

3.1.5 Help ............................................................................................................................ 32

3.1.6 Administrate ............................................................................................................... 33
LIST OF TABLES

Table 1. ASP.NET application files folders ................................................................. 36
Table 2. ASP .NET files type extensions ........................................................................ 36
Table 3. Internet Applications Comparison .................................................................... 86
LIST OF FIGURES

Figure 1. Three Different Types of a Basic Chart ......................................................... 3
Figure 2. Open-Close-High-Low Bar Charts ................................................................. 4
Figure 3. Relative Comparisons of Two Stocks ............................................................. 5
Figure 4. Head and Shoulders Pattern ......................................................................... 7
Figure 5. Feed-Forward Networks Structure ............................................................... 11
Figure 6. Recurrent Networks Structure ..................................................................... 11
Figure 7. Information Flows ......................................................................................... 18
Figure 8. Development Technologies and Three-Tier Architecture ......................... 21
Figure 9. Interactive Stock Market Web Portal Conceptual Designs ....................... 22
Figure 10. Stock Portal Web Site Home Page ............................................................. 24
Figure 11. User's Roles with the Corresponding Permissions ...................................... 25
Figure 12. Stock Portal Symbol Lookup page ............................................................ 27
Figure 13. Chart Custom Page .................................................................................... 28
Figure 14. Compare Chart Page .................................................................................. 29
Figure 15. Historical Data Displayed in the Tabular Format ........................................ 30
Figure 16. My Portfolio Page ....................................................................................... 31
Figure 17. Chart Help Pop Window ............................................................................ 32
Figure 18. Glossary Page ............................................................................................ 33
Figure 19. Guest Book Page ....................................................................................... 33
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Edit Symbol Data Page</td>
<td>34</td>
</tr>
<tr>
<td>21</td>
<td>Edit Trade Data Page</td>
<td>35</td>
</tr>
<tr>
<td>22</td>
<td>Application Folders</td>
<td>38</td>
</tr>
<tr>
<td>23</td>
<td>ASPNETDB.MDF is Shown in Visual Studio</td>
<td>42</td>
</tr>
<tr>
<td>24</td>
<td>Visual Representation of the ASPNETDB.MDF Database Diagram</td>
<td>44</td>
</tr>
<tr>
<td>26</td>
<td>Visual Representation of the Stock Database Diagram</td>
<td>47</td>
</tr>
<tr>
<td>27</td>
<td>Stored Procedures are shown in Visual Studio 2005</td>
<td>49</td>
</tr>
<tr>
<td>28</td>
<td>Stored Procedure InsertStockData</td>
<td>50</td>
</tr>
<tr>
<td>29</td>
<td>Output of Running InsertStockData Stored Procedure</td>
<td>51</td>
</tr>
<tr>
<td>30</td>
<td>Screen Shot of Yahoo Stock Text File</td>
<td>52</td>
</tr>
<tr>
<td>31</td>
<td>GetYahooDataServer Invokes GetYahooData Method</td>
<td>52</td>
</tr>
<tr>
<td>32</td>
<td>ASP.NET Server Controls</td>
<td>65</td>
</tr>
<tr>
<td>33</td>
<td>StockPortal.master Template</td>
<td>67</td>
</tr>
<tr>
<td>34</td>
<td>Administ .master Template</td>
<td>68</td>
</tr>
<tr>
<td>35</td>
<td>Relationships among the Security Systems in ASP.NET</td>
<td>70</td>
</tr>
<tr>
<td>36</td>
<td>Login Page</td>
<td>72</td>
</tr>
<tr>
<td>37</td>
<td>Roles Established by ASP.NET Web Site Administration Tool</td>
<td>74</td>
</tr>
<tr>
<td>38</td>
<td>Symbol Lookup</td>
<td>76</td>
</tr>
<tr>
<td>39</td>
<td>Real Time Quotes for GOOG</td>
<td>77</td>
</tr>
<tr>
<td>40</td>
<td>Daily View Chart for GOOG</td>
<td>78</td>
</tr>
<tr>
<td>41</td>
<td>Weekly View Chart for GOOG</td>
<td>78</td>
</tr>
<tr>
<td>42</td>
<td>Monthly View Chart for GOOG</td>
<td>79</td>
</tr>
</tbody>
</table>
CHAPTER 1 INTRODUCTION

Investors are always looking for a stock that will outperform the market. The development of the Internet and computer technology has made investing process more effective. Educated investors make their decisions based on the significant amount of data related to the previous performance of the stocks in question. To deal with this enormous amount of data, investors utilize tools to help them in stock selection. A number of different tools categories can be applied including: statistical and analytical tools, scientific visualization charts and artificial intelligence. This project provides investors’ with a set of tools within a web portal allowing them to use historical and current stock data for their analysis.

1.1 Equity Trading Terminology

Stock prices fluctuation is affected by a number of financial and human factors. There are two major analytical approaches to estimate future stock price movements, namely: fundamental analysis and technical analysis (Wikimedia Foundation Inc, 2006a). The fundamental analysis relies on the macroeconomics type of data affected by the economic, social and political forces and industry groups. Technical analysis focuses on the stock behavior itself and studies trading history such as recorded prices and volume movements, in order to forecast future price changes. Because of the large number of unknown variables, technical analysis cannot result in absolute
predictions about the future. Instead, technical analysis can help investors anticipate what is “likely” to happen to prices over time (StockCharts Inc. 2006).

1.1.1. Stock Charts. Investors, who take in consideration the technical analysis approach, make trading decisions on vast amount of available historical data. The presentation of the data is crucial. Data can be presented in a plain tabular format, or even more desirable, using an appropriate graphical charts and diagrams. Various visual representations of the same data can reveal new trends and behavior patterns. Statistical and pattern recognition calculations can be added to clarify and isolate ‘good’ stocks. Investors would then try to identify risk levels and place their trades accordingly.

A number of visual representations have gained popularity over the years including basic stock charts. These charts generally display the information using a two dimensional space with the x-axis representing the historical time intervals. The vertical axis could be used to represent the snapshots of the stock price in question. This basic visual representation can be useful in identifying possible trading trends. Additional information can be added to the y-axes in order to display historical volume data. Each chart has a specific purpose and use. Which one should be used depends upon the personal trading style, the time available to trade and preferences.

There are three types of a basic chart depicting a specific stock price behavior over a select period of time. Examples of such chart types are given in Figure 1.
**Figure 1.** Three Different Types of a Basic Chart.

In the scatter and line chart, the y-values represent the closing stock price at the end of the specific point in time. The basic bar chart contains additional information in form of a vertical line indicating the highest and the lowest price for the specific time interval.

More detailed bar charts can be produced, adding more information to existing charts. For example chart can be improved by including the ticks representing opening and closing price of the stock at the certain time. Figure 2 a. depict an example of such a chart where the ticks for opening price are oriented toward the left-hand side, while the closing price ticks point toward the right. If such a chart displays a red bar, it usually means that the closing price is lower than the opening one. Otherwise, any other color indicates the closing price to be higher than the opening one. The open-close ticks can be connected into a thicker bar, to distinguish them from the highest-lowest line. Visually such bars resemble candles with the candlesticks. An example of such ‘candlesticks’ chart is presented in Figure 2 b. Candlestick charts are more popular than other basic chart types highlighting the relationship between the open-close and highest-lowest prices of the selected stock. They are considered to be more visually appealing because the price information can be easily extracted by simply focusing on a single ‘candle’.
Figure 2. Open-Close-High-Low Bar Charts.

The open-close and highest-lowest stock data can be collected over a shorter or longer period of time showing intra-daily, daily, weekly and monthly activity. Such historical data are publicly available and can be retrieved from a variety of sources. Daily and intraday data are usually used by ‘day traders’ to forecast short-term price movements. In contrast, investors to recognize long-term trends and forecast long-term price movements use weekly and monthly data. Comparing past and present data can provide more insight into the types of behavior that occurs in the market.

Charts can be extended by showing a multiple of stocks behaviors in the same chart. Such charts can be used to identify how different stocks interact with each other. One possibility is to compare the relative improvement of multiple stocks: in such charts the x-axis still represents the time intervals, while the y-axis shows the improvement of the stock relative to the start of the time interval. An example of a comparison chart between two stocks (Google™ and eBay®) comparing relative improvements of both stocks is shown in Figure 3.
Figure 3. Relative Comparisons of Two Stocks (Google vs. eBay).

1.1.2. Technical Analysis Approach. Technical analysis studies the historical data-surrounding price and volume movement of stock by using charts as a primary tool to forecast future price movements (Murphy 1999). Investors and traders use charts to make decisions based on various kinds of analysis including trend line analysis, pattern analysis and indicator analysis.

Trend lines are a simple and widely used technical analysis approach to judging entry and exit investment timing. A trend line is a straight line that connects two or more price points of interest, and it extended into the future (x-axis). Such line acts as a line of support or resistance for future investments. Once trend lines are constructed, technical analysis of price trends can be performed. Trend lines are the most basic and the most valuable tool for both trend identification and confirmation. Trends can be classified in tree ways: up, down and range bound. If a trend line connecting two or more points of interest (mostly low points) has a positive slope, it is known as an up trend line. This can be interpreted as a bullish stock indicating the demand being greater than the supply. If a trend line connecting two or more points of interest (mostly high points) has a negative slope, it is know as a downtrend line. This can be interpreted as a bearish stock indicating the supply being greater than the demand. If the forces of supply and demand
are nearly equal, the stock will move sideways keeping its value. Up trend lines are considered be positive (‘support’), while downtrend lines are considered to be negative (‘resistance’). In general, once a trend line is broken, the trend, which was previously in force, is considered over, or at least in pause. It should be noted that when an up trend line or 'support' is broken, it then acts as 'resistance'. Likewise, if a downtrend line or 'resistance' is broken, it then becomes 'support'.

Aside the trend lines additional patterns can be sought. This can be done by a number of pattern analysis techniques. Any general stock chart is a combination of countless different patterns. Detecting the patterns might provide additional information that can be valuable to investors. One can seek a pattern for either classification or generalization purposes. Patterns can be also useful to detect possible investment timeline.

There are two types of patterns that are valued the most: the reversal and continuation pattern. Reversal patterns indicate an important trend changes. Continuation patterns indicate a pause in a trend before the prevailing trend will resume. Another difference between reversal and continuation patterns is in their time duration. Reversal patterns usually take much longer to form. They do represent major changes in a trend. Continuation patterns, on the other hand, are usually shorter-term in duration. They are often classified as intermediate term patterns.

For example: the head and shoulders pattern (see Figure 4) is generally regarded as a reversal pattern and it is most often seen in up-trends. If it is seen in an up-trend it is considered to be rather reliable. Eventually, at the end of the pattern, the market begins to slow down and
the forces of supply and demand are generally considered in balance. Sellers come in at the highs (left shoulder) creating a downside (beginning of the neckline). Buyers soon return to the market and ultimately push through to form a new high (head). Symmetrical triangles can be characterized as continuation pattern. The pattern contains at least two lower highs and two higher lows. When these points are connected, the lines converge as they are extended and the symmetrical triangle takes shape.

![Head and Shoulders Reversal](image)

*Figure 4. Head and Shoulders Pattern.*

Finally, let us consider technical indicator analysis. Technical indicators are mathematical statistics based on historical stock prices. They are used extensively in technical analysis to predict changes in trends or price patterns. Various technical indicators have been developed and presented to investors in a formula format. Some formulae, such as moving averages, are rather easy to understand, while others, such as relative strength index, incorporate complex principles and require additional background knowledge for complete understanding. Regardless of the complexity of the formula, technical indicators can provide unique perspective on the strength and direction of the underlying price action.
Moving Average (MA) is an indicator that calculates the average price of a stock over a specified period of time. Moving average filters out random noise and offers a 'smoother' perspective of the price action. MAs are often used for comparison purposes. For example, different period MAs can be compared to the recent stock price itself. If the current stock price exceeds its moving averages, it indicates an opportunity to buy. On the contrary when the current stock price falls below its moving averages, the owned stocks could decline and should be sold if possible.

Moving averages can be simple (SMA), weighted (WMA) or exponential (EMA). For example: the formula for SMA is rather simple and is given:

\[ SMA = \frac{P_1 + P_2 + \cdots + P_n}{n} \]

Relative Strength Index (RSI) compares the relative strength of price gains. The index represents the ratio of price gains of a single security on days that close above the previous day's close, over the price losses on days that close below the previous day's close (Welles, 1978). The recommended number of time periods is 14. The formula for the RSI is given:

\[ RSI = 100 - \frac{100}{1 + \frac{RS}{RS}} \]

The RSI is usually plotted on a vertical scale from 0 to 100. Movements above 70 are considered to be overbought, while an oversold condition occurs when a move is under 30. The first move to the overbought or oversold region is a warning indicating a need for action. RSI
swings above 70 or below 30 strongly indicate market reversals. If price is low and RSI is high (>70), it might be a signal to buy. If price is high and RSI is low (<30), this might indicate a good time to sell.

The number of metrics and analytical techniques is rather large. In this section we have presented just a simple few. Interested readers can find numerous examples of different metrics and techniques included in ‘Chart School’ on StockCharts.com web-site (StockCharts.com, 2006).

1.1.3 Intelligent Prediction. Using technical analysis to forecast stock future price movement has a long history. However, these methods are based on the assumption that history repeats itself. Unfortunately, the expectation that some historical patterns of a stock price are going to be repeated may not necessarily be true, since market conditions constantly change over time (Wong & Ng, 1994). In order to predict future price behaviors, new techniques need to be used that draw ‘intelligent’ conclusions based on historical data. One way to predict stock prices’ movements is to utilize the area of Artificial Intelligence. Artificial Intelligence (AI) methods are designed to mimic real intelligent behavior. Some of the AI techniques can be easily implemented to run on today’s computers. Since stock prices are nonlinear in nature, some of the nonlinear continuous function approximations can be used for prediction purposes (Cybenko, 1989).

One of the most popular AI methods is Neural Networks (NN). As stated in Wikipedia (Wikimedia Foundation, Inc. 2006b): “Neural Network is an interconnected group of artificial
neurons that uses a mathematical model or computational model for information processing based on a connectionist approach to computation". NN can be used to predict stock prices because its property to being able to closely approximate nonlinear mappings such as stock prices. Neural Networks are based on the concept of Artificial Neuron. A Neural Network has a number of interconnected neurons, and is trained to obtain optimal neuron connections. Each neuron receives inputs from many other neurons and generates a single scalar output that depend only locally available information; either stored internally or via the weight connections (Reed 1998). During the learning phase, the network learns by adjusting the weights so as to be able to correctly predict or classify the output target.

There are two basic types of neural networks structures, namely Feed-Forward Networks (FFN) and Recurrent Networks (RN). A typical neural network consists of a number of neuron layers. In a single layered network there is an input layer of source nodes and an output layer of neurons. A multi-layer network has in addition one or more hidden layers of hidden neurons. Both types of networks are displayed in Figure 5 (a.) a FFN with a single output layer of neurons. (b.) a fully connected FFN with one hidden layer and one output.

A Recurrent Network will usually have feedback connections: reusing the outputs from the network as inputs to the network. Such feedback loops often suffer from a time delay. An example of a RN is given in Figure 6.
Neural Networks need to be carefully designed. In any Neural Net special attention needs to be devoted to selection of inputs, description of outputs and design of network connections. In the case of stock price description, input data could be raw stock data such as prices and volumes and daily changes, but it may also include derived data such as technical indicators (moving averages, RIS). Neural Networks must be trained on pre-determined data. The goal of the NN training is to minimize its error on the training samples. The FNN has been commonly used for stock prediction due to its ability to correctly classify and predict the dependent variable (Vellido & Lisboa & Vaughan 1999).
1.2 Literature Review

Stock prediction is one of few human activities that have been so exhaustively studied during the past fifty years, from so many angles and by so many different sorts of people (Edwards, 1997). This is due to the fact that a successful educated prediction could result in substantial profit. Today, the Internet, Web, and other information technologies have made a dramatic effect on stock market. There are many online trading sites that provide real-time stock market trading data and basic technical analysis charts. Most financial web sites, such as: Bloomberg, Yahoo Finance and Nasdaq.com, provide real-time and archived financial and market data, pricing, trading, news and communications tools.

Bloomberg.com provides users with real-time stock data, charts, analysis tools and real-time radio and video reports (Bloomberg, 2006). Bloomberg’s chart generator is developed using Macromedia Flash technology with some interactivity using sliders. Their web site provides bar and line charts with a possibility of multiple stocks.

Yahoo finance (Yahoo, 2006) provides single stock basic charts, as well as comparison charts between a single stock and a composite stock (such as NASDAQ). Users can download historical data for analysis to their own computers. The download can be performed in either a textual or a spreadsheet format.
Nasdaq.com also provides individual stock charts, together with comparisons charts of a selected stock against the Nasdaq Composite. User can compare the detailed stock chart against the ‘moving averages’ of 20, 50 or 200 days. The moving average of 20 days is the average close price over the 20 days period.

Many additional companies are trying to enter the online analytical tool market with a free and paying-member-only area including: StockCharts.com (2006) and EasyChart (2006). They produce charts with dozens of indicators built in for investors. It is expected that many more will join.

Scientific community has produced a number of research papers that address the significance of scientific visualization in market data analysis. For example: Nesbitt and Barass (2004) conducted experiments that show that use of visual tools in data mining and trading decisions significantly improved their trading performance compared to using a simple rule of chance. Their paper includes a concise overview of graphical charts used in trading together with their cognitive and perceptual values. A number of case studies are included. In one of their prior papers (Nesbitt & Barrass 2002) they discuss a use of three-dimensional displays (volume-price-time) in the decision-making process.

Omrane and Oppens (2004) focus their attention to the technical analysis performed on basic chart patterns. As mentioned in their paper, more than 90% of the foreign exchange dealers
trading in London use technical analysis and more that 60% judge charts to be “as important as fundamentals”. They investigate twelve distinct chart patterns and test their decision-making value. A number of simulations were run to test the specific patterns’ behavior and the technical analysis based on the basic charts.

In their paper “Bimodal Visualization: A Financial Trading Case Study”, Taskaya and Ahmad (2003) perform a study of a visualization computer system that incorporates two different modalities of information: numerical and textual. With the system in place, they conducted a case study to prove the effectiveness of utilization of such a system.

A number of research papers address possible prediction strategies that can be used on stock data. For example, Savin and Zvingelis(2003) focus on “Head-and-Shoulders” price patterns for stock prediction. They stated that “Risk-adjusted excess return to a trading strategy condition on ‘head-and-shoulders’ price patterns are 7-9 percent per year”. In “Neural Networks as A Decision Marker for Stock Trading: A Technical Analysis Approach”, Thawornwong, Enke and Dagli(2003) employ different neural networks modeling to predict stock trend signals by using five technical indicators for input. They found that “neural networks can improve the effectiveness of traditional technical analysis” and “help investors make better investment decision”.

14
Because of the significant implications of such scientific papers in areas of price stock prediction and analysis, the research in this field will continue to attract considerable attention in the future.

In the development of our stock portal, we will use some of the analysis tools and techniques illustrated in this section. We proceed with the description of our motivation and the systems architecture.
CHAPTER 2 OVERALL PROJECT DESIGN

So far, we have introduced some of the main metrics and analytical techniques for stock price forecasting, including stock charts types, different analytical approaches from the literature review. In this chapter we proceed with an overview of our project titled ‘an interactive stock market web portal’. The presentation starts with the motivation for our project.

2.1 Motivation

Today’s investors are utilizing Internet Web technologies and advanced computer algorithms to help them decide which stock to buy and which to sell. Internet Web technologies make investing process easier and faster. Advanced computer algorithms applied to the investing process provide investors with multiple benefits emerging from a wider range of investment choices. The goal of this thesis is to provide investors’ tools within a visually pleasing, functional, and interactive stock web portal.

Our project provides traditional technical analysis techniques, visualization charts, and scientific prediction methods based on the Neural Network approach. Investors can benefit from variety of stock chart types based on historical and current stock data. Such charts can be used to help users identify underlying trends or patterns, with any combination of price, volume and
time-sensitive technical indicators. Built-in Neural Networks based prediction technique provides investors with additional insights to help them to make educated decisions.

Another capability of our portal is to enable users to define their own investment portfolios with their choice of stock companies. Users can test their prospective stock additions to their portfolio utilizing powerful what-if analysis. The portfolio automatically tracks user’s transactions and their possible gains or losses.

2.2 System Architecture

Our application provides a multitude of useful web services such as real time stock quotes, interactive stock charts, technical analysis tools, and on-line portfolios. The main web portal server breaks down the original user’s request and distributes the work. Some tasks might request data from either the local database or from a remote web service. The collected data can be then analyzed and presented to the user in a visual format.

In order to achieve the design goals, the system uses three-tier architecture model by splitting the application’s functionality unit into three logical tiers:

- The *presentation tier* contains the User Interface (UI) elements of the portal, and is composed of dynamic web pages that display output on the screen.

- The *middle tier* receives requests from the presentation tier and returns the obtained result to the presentation tier depending on computational and functional logic it contains. Almost any event that is triggered in the presentation tier results with a middle tier being called.
- The *data tier* is responsible for storing the stock historical data and sending it to the middle tier when requested.

The communication between the presentation tier and the data tier is never direct but through the middle layer. Each tier can be hosted on different networked computer if needed. Figure 7 illustrates how the information flows through the system.

*Figure 7. Information Flows.*

If user is interested in monitoring a specific stock price behavior, she or he would type a specific stock symbol of interest, followed by the request mouse click (Step 1). The presentation tier, which contains the stock symbol text box and the build chart button, would forward the request to the middle tier (Step 2). The middle tier receives the request, understands that the user wants a specific stock symbol chart generation, and handles the request by calling the data tier.
The communication between the middle tier and the data tier is done by a data query for the specific symbol’s historical data (Step 6). The data tier updates the database (Step 4) and eventually returns that symbol data to the middle tier. The middle tier uses the stock raw data to calculate the chart data (Step 5) and then returns the output data to the presentation tier. The presentation tier generates a view of the chart data (Step 6). The final result of such multi-tier execution is an HTML web page with embedded visual information that is returned to the user (Step 7).

This inherent three-tiered architecture makes our application flexible, scalable and reliable. For example, if for some reason the back-end database system needs to be changed, one only needs to update the data tier. The existing presentation and middle tiers do not have to be changed.

2.3 Development Technologies and Tools

Based on our system architecture and requirements, the final choice was given to the .NET Microsoft development environment. Our portal was developed and built using Microsoft .NET 2005 Framework with ASP.NET 2.0. The main language of choice for interactivity, database connectivity and analytical capability was C# version 2.0. The data tier is supported by Microsoft SQL Server 2005.

The Microsoft .NET Framework is very efficient development platform targeted for Web Services, Windows and Web applications. The .NET Framework includes a powerful system of generating Web content dynamically, allowing personalization, security and interactivity.
ASP.NET 2.0 is Microsoft's latest technology for creating dynamic Web applications. It is part of the .NET Framework and there are a lot of features that makes building real world Web applications easier. ASP.NET 2.0 provides efficient code-behind models and structures to separate the application code (the computational and functional layer) from the user interface of a web page (the presentation layer). ASP.NET 2.0 has flexible language option and it supports any one of .NET language from C#, VB.NET, and J# (Microsoft Corporation 2006a).

C# is an object-oriented programming language designed for building a wide range of enterprise applications that run on the .NET Framework. It can be used to code Web Forms' code-behind files for computationally intensive web applications. In our project, a number of computational tasks are implemented such as chart generations and interactive stock analysis and prediction.

Microsoft's SQL Server 2005 is a database platform for large-scale online transaction processing, and e-commerce applications. It can also be used for a business intelligence gathering such as: data integration, analysis, and reporting (Microsoft Corporation 2006b). It easily embeds within the .NET Framework, providing the ability to host the .NET Framework common language runtime (CLR) in the database, and to develop complex SQL queries from any .NET language, especially C#. In our application, we use SQL Server to provide services for our data tier. The communication between the middle tier and the data tier is done through T-SQL (Transact-SQL) language.
The middle tier code programmed in C# will make use of T-SQL stored procedures. Storing T-SQL procedures in data layer, instead of composing them dynamically in C#, is more efficient, secure, and assures data consistency. Once a stored procedure is called from C#, it runs in the data layer; manipulates or accesses the database, gets the results, and returns the result to the middle tier. The C# code can now perform complex mathematical operations based on the received results. Figure 8 illustrates the technologies associated with every tier in the three-tier architecture.

![Diagram](image)

*Figure 8. Development Technologies and Three-Tier Architecture.*

In summary, the presentation tier is mainly developed in ASP.NET, the middle tier is mainly coded in C#, and finally the data tier is developed using SQL.
2.4 Conceptual Design

The conceptual design of our system is depicted in Figure 9, followed by the descriptions of each of the components.

Figure 9. Interactive stock market Web portal conceptual designs.

Since our portal is web based, users do not need to install nor buy any additional software except a web browser. Users’ requests are directed through the Internet to the portal web server. The server coordinates necessary service calls to other servers and communicates the final results back to the users. In our project we utilize Microsoft™ IIS web server, but any other popular web server can be used instead (for example Apache). Originally, the Microsoft SQL Server does not contain any historical data. The data is populated on-demand if necessary. To obtain the data for the data population, the web server connects to the Yahoo! Finance web site and requests the current data of the selected stock symbol. This is done over an Internet connection by using http
session calls. Once the data is obtained, it is formatted and sent to the SQL database server for storage.

The majority of highly computational work is run on the web server computer. We utilize Microsoft’s .NET technology that provides a multitude of web services integrated with a user-friendly developer’s platform. .NET provides developers with the C# programming language that is well suited for computationally intensive web applications.

Majority of data preparation and presentation is done on the server side, with the final result being a simple html document. To control the outlook of such html documents, ASP.NET uses standard controls built into the .NET Framework. The html documents are dynamically generated by the ASP .NET together with the corresponding style sheets. This approach allows the application to be deployed to any platform or operating system with a web browser. The interactivity components are also controlled by ASP.NET, that are also stored as an ASP code. Such ASP code components are pre-compiled into efficient binary files that the server can execute very quickly. This is preferred approach compared to other web scripting languages that are being interpreted every time they are used.

A user’s main point of entrance will be our web portal that resides on our web server. The portal will provide a number of menu selections with seamless navigation. A snapshot of the portal home page can be viewed in Figure 10.
Figure 10. Stock Portal Web Site Home Page.

The users who access our web portal are divided into three groups:

- Anonymous users – they will be able to view stock historical and real time data, use different type of stock analysis tools without login process.

- Registered users – they have registered their account username and password. If they are logged in, they are able to use ‘My Portfolio’ function to save, trace and retrieve their personal stock information.

- Super users – they are pre-assigned the username and password. Through a login process, they will be able to perform on-line maintenance of the portal: including updates, inserts or deletes of records from the database.

Figure 11 shows the permitted activities available to each of the user’s groups.
Figure 11. User’s Roles with the Corresponding Permissions.

At the top of each web page, there are Login and Register buttons, which allow users to login and register at any time. Each web page shows the user’s login information next to the Login button.

This concludes the description of the motivation, system architecture, development technologies and the web portal conceptual design. The following chapters focus on the Web site functionality and structure.
CHAPTER 3 FUNCTIONALITY AND WEB SITE STRUCTURE

As mentioned in chapter 2, the focus of this thesis is to design and develop an interactive stock market web portal that provides an analysis and visualization tools for investors. In this chapter, we will describe the main functions and structure of the web portal.

3.1 Application Functionality

Our application provides the following functionality: Stock Look-Up, Stock Charts, Statistical Analysis, My Portfolio, Help, and Administrate. Users can easily access the corresponding web pages by using the menu bar at the left-hand side of each page.

3.1.1. Stock Look-Up. It is a symbol catalog that alphabetically lists all stock symbols already in the database with the following fields present: symbol, name, exchange, previous trade day’s open price, close price, highest price, lowest price and volumes (as shown in Figure 12).
Figure 12. Stock Portal Symbol Lookup Page.

Clicking on the Quote link will provide users with more detailed real time trading information about the selected symbol. The small chart icon links to a page that displays several essential types of charts related to the selected symbol.

If a visitor needs to search for a symbol in the database, she or he can click Symbol Lookup button located in the lower left corner. Once the button is clicked, a small window will pop up presenting to the user a searchable textbox. The textbox allows users to enter any starting part of the search ticker symbol, or the security, or the company name.
3.1.2. *Stock Chart.* It provides different types of charting tools with a short description. There are four kinds of chart tools available: *Chart Custom, Chart Comparison, Chart Gallery,* and *Chart Group.*

*Chart Customize* tool allows users to customize the chart depends on their selection with different customizable chart settings. There are five types of chart attributes: *periods* (daily, weekly), *range* (1 month, 3 months, 6 months, 1 year, 2 years, 3 years), *chart type* (candlesticks, OHLC bars, HLC bars, line chart), *size* (720, 800, 1000) and *color scheme* (default, black). Users can easily switch between those different values settings through dropdown boxes to customize the appearance and attributes of the chart. The tool also provides users with a selection of four technical indicators: *simple moving average, exponential moving average, RIS (Relative Strength Index), MACD (Moving Average Convergent/Divergence)* (The details will be described in the Chapter 5). Users can easily select any combination of the above listed indicators through the selection of check boxes. Figure 13 shows *Chart Customize* for *GOOG* stock symbol.

![Chart Customize](image)

*Figure 13.* Chart Custom Page.
**Chart Comparison** tool allows users to compare performance of up to five different stocks over the same period of time as shown in Figure 14. Users enter the stock symbols of their interest in already provided text boxes together with a time period. Clicking on the Build Chart button produces a chart that displays the comparison. If user does not know the correct symbol for the stock of his/her interest, the user can utilize the **Symbol Lookup** button located in the top left corner.

![Chart Comparison](image)

**Figure 14.** Compare Chart Page.

**Chart Gallery** tool allows users to view three different charts with different time frames on the same screen. User is required to enter a symbol and click on the Build Chart button. The daily, weekly, and monthly views present the short-, medium-, and long-term view respectively. Clicking on the **Display the Chart Data** link, another window will be opened, showing the corresponding raw historical data displayed in the tabular format (Figure 15). Clicking on the **Draw Chart in another Window** link, a new window will open with the same chart but with
additional tools in it. Once the window is open, user can drag her mouse to draw a trend line on the chart.

![Image of Table]

**Figure 15.** Historical Data Displayed in the Tabular Format.

*Chart Group* tool provides users with parallel overview of charts generated from multiple stocks. Users enter up to four stock symbols of their interest. This tool shows the selected stock charts as a group, and provides users with the capability of quickly scanning the presented charts for patterns.

3.1.3. **Statistical Analysis.** It provides a number of statistical analyses, performance evaluation tools. Statistical analysis tools include: Evaluation of Moving Averages, Momentums,
Stock Correlations, and Analysis of Variance. Performance evaluation tools include: Neural Networks Prediction.

3.1.4. *My Portfolio.* It allows registered members to trace and retrieve saved stock data of their selection (Figure 16). The members are able to create their own stock selections and save it for future use. They are also able to obtain multiple stock symbols quotes at the same time and track their current holdings. The detailed trade data that can be stored include: the number of owned shares, the purchase price of the shares, trade dates, and lower and upper limit prices. Additionally the tracking information is displayed, such as current hold price, hold value, calculations of gains or losses. The lower and upper limit prices can be used as triggers. Once the day’s close price hits the trigger price, an email will be sent to the member.

![Stock Portal](image)

*Figure 16.* My Portfolio Page.
3.1.5. Help. It provides detailed help topics for all chart and analysis tools. Users can browse through the help topics within Chart Help pop-up window (Figure 17). The Glossary page (Figure 18) provides the definitions of the chart and analysis terms. It also contains information about using our Web portal. Finally the Guestbook page (Figure 19) allows users to leave a new message and retrieve feedback left on older messages.

![Chart Help Pop Window](image)

*Figure 17. Chart Help Pop Window.*
3.1.6. Administrate. It is specifically created for super users to perform on-line maintenance of the stock database.
The *Edit symbol data* page (Figure 20) lists all stock tick symbols stored in the database. By clicking the *Edit* or *Delete* button, the selected symbol data can be edited or deleted. It also allows creating a new symbol in our database. To do so, the super user needs to enter the new symbol, together with the correct name and exchange, followed by a click on the *Add Symbol* button.

![Stock Portal](image)

*Figure 20. Edit Symbol Data Page.*

The *Edit Trade Data* page (Figure 21) allows super user to update the trade data table by entering symbol and selecting time period, followed by a click on the *Update* button.
In this section, we have described the main functions provided by our application through a number of Web pages screen shots. In the next section, we will introduce how those web pages are organized within the Web portal.

3.2 Stock Portal Web Site Structure

Our Stock Portal Web site is developed using Microsoft ASP.NET technology. ASP.NET is used for creating dynamic Web pages. ASP.NET page starts running on the Web server side at the moment of user’s request. During a single run, the page can be asked to perform a number of distinct tasks, such as calculating values, reading database information, or calling other programs. At the end of its run, the page dynamically produces markup content and sends it to the client’s browser (Microsoft Corporation 2006a).
Generally speaking, complete ASP .NET project can contain a number of different file types. Some files are supported and managed by the ASP.NET server, while others are supported and managed only by the web server. ASP.Net assumes that certain folders contain specific content types. Table 1 lists such folder names for our web portal application, while Table 2 lists the file type extensions.

**Table 1**

**ASP.NET application files folders**

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>App_code</td>
<td>Contains source code for utility classes and business objects</td>
</tr>
<tr>
<td>App_Data</td>
<td>Contains application data files including MDF files. The folder is used to store an application's local database, which can be used for maintaining membership and role information.</td>
</tr>
<tr>
<td>images</td>
<td>Contains images used in the application</td>
</tr>
<tr>
<td>Javascript</td>
<td>Contains Javascript files</td>
</tr>
<tr>
<td>Style</td>
<td>Contains style sheet files</td>
</tr>
<tr>
<td>UserControl</td>
<td>Contains ASP.Net user control files</td>
</tr>
</tbody>
</table>

**Table 2**

**ASP .NET files type extensions**

<table>
<thead>
<tr>
<th>File Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.aspx</td>
<td>These are ASP.NET Web form files (pages) that contain the user interface and presentation tier. Users request or navigate directly to one of these pages to start using our web portal.</td>
</tr>
<tr>
<td>.ascx</td>
<td>These are ASP.NET user control files. User controls are similar to web pages, but can not be accessed directly. Instead, they must be hosted inside an ASP.NET web page. User controls are used for defining a small piece of user interface and reused in as many web forms as needed.</td>
</tr>
<tr>
<td>File Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>.asmx</td>
<td>These are ASP.NET web services files that contain classes and methods. Unlike the ASP web pages, web services do not produce presentable HTML code, but they still share the same application resources, configuration settings, and memory.</td>
</tr>
<tr>
<td>web.config</td>
<td>This is the XML-based configuration file for our Stock Web portal application. It includes settings for customizing security, state management, memory management, and much more.</td>
</tr>
<tr>
<td>.master</td>
<td>These are master page files that define layouts for other Web pages in the application.</td>
</tr>
<tr>
<td>.sitemap</td>
<td>This is a site-map file that contains the structure of our Web site. It is used for navigational controls displayed within a Web page.</td>
</tr>
<tr>
<td>.aspx.cs</td>
<td>These are code-behind files that contain C# code. They are used to separate the application from the user interface.</td>
</tr>
<tr>
<td>.ascx.cs</td>
<td>These are C# class source-code files that contain application logic. They are used for middle tier implementation and located in App_Code subdirectory.</td>
</tr>
</tbody>
</table>

Figure 22 shows the complete list of ASP .NET application folders. Only one folder is opened (named UserControl) showcasing the list of files contained within the folder.

In all ASP.NET pages (aspx extension), user interface programming is divided into two components: the visual and the logic component. The visual component consists of a file containing static markups such as HTML or ASP.NET server controls or both. The logic component consists of code that runs ‘behind the scenes’ is used for interaction with the page.

ASP.NET Web pages are pre-compiled into a dynamic-link library (.dll) file. The first time user browses an .aspx page, ASP.NET automatically generates a .NET class file that represents the page and then compiles it using the dll library.
Figure 22. Application Folders.

ASP.NET Web Forms have the .aspx extension and are the standard way to provide web functionality to users. Once a request to an .aspx resource is made, an ASP file is executed and the results are being composed as a HTML page. The resulting page is then sent back to the client. ASP.NET 2.0 provides two ways for structuring the code of the ASP.NET pages: code-
inline model and code-behind model. Our preference was given to code-behind model, which allows for code separation of the page's business logic from its presentation logic. In this model, the presentation logic is stored in an .aspx file, whereas the business logic piece is stored in a separate class file: .aspx.cs, which is also considered to be a part of the Web Form.

The described functionality and structure is well suited for our portal application. Each of the three tiers utilizes the presented file and directory structure. We proceed with the in-depth specifications of each tier, starting with the data tier as the backbone of our application.
CHAPTER 4 DATA TIER IMPLEMENTATION

The project's architecture, development technologies, and the functionalities were presented in Chapter 2. In this chapter we continue with the discussion of the implementation of the three-tier architecture. The presentation starts with the data tier implementation details.

4.1 Data Analysis and Database Design

The data tier is responsible for storing the application's data and sending it to the middle tier when requested. Almost every user's request results in the data tier being queried for information, so it is important to have a fast and reliable database system.

Today there are two types of databases widely deployed: relational and hierarchical. A hierarchical database has a tree-like structure. The structure allows repeating information using parent/child relationships: each parent can have many children but each child only has one parent. All attributes of a specific record are listed under an entity type. In such a database, an entity type is the equivalent of a table; each individual record is represented as a row and an attribute as a column (Wikimedia Foundation, Inc. 2006c). A relational database consists of tables that are related to each other, either directly or indirectly. Hierarchical databases are usually very fast due to the directly linked data structures; but they are not good for modeling complex relationships.
Our Web portal application uses Relational Database Management System (RDBMS) to store, access and manipulate the information. The leading RDBMS products are: Oracle, IBM’s DB2 and Microsoft’s SQL Server. In our project, we use Microsoft’s SQL Server 2005.

Essentially, a relational database is made up of data tables and the relationships that exist between them. In order to analyze and understand the data, the data needs to be stored using an optimal data-base design. According to the functionalities provided by the portal application, the information that needs to be stored and manipulated falls into the two large categories: stock data information and users’ data information.

Stock data information includes stock ticker symbol, company name, stock exchange name, and trade history information. Currently, the number of stock ticker symbols is far beyond one thousand and will probably increase in the future. For example: NASDQ includes more than 5000 companies. The trade history information includes trade date, open and close price, highest and lowest price, and volumes. All trade history date can be collected on daily, weekly and monthly basis.

Users’ data can be broken into two categories: users’ account information and user stock portfolio information. Users’ account information is used for users’ authentication and identification. Such data includes: users’ ID, users’ role, password and e-mail. User stock portfolio information is used for stock data manipulation. Such information includes: users’ watching symbol, owned stock symbol, the number of shares and price per share.
As mentioned before, authentication is based on the stored user account information. Our project makes use of the newly available ASP.NET membership system. ASP.NET 2.0 includes a number of services that store state in database. The membership system provides standardized means for system user account implementation. The system uses a predefined SQL server database format stored in ASPNETDB.MDF database. When we first build and run our application on Visual Studio 2005, it creates this database file dynamically that is used to store user’s login and role information for the application. Figure 23(a) shows the ASPNETDB.MDF is in the App_Data folder. The ASPNETDB.MDF database file is in the Visual Studio Server Explorer, as shown in Figure 23(b)

![Diagram](image)

**(a)**

**(b)**

*Figure 23. ASPNETDB.MDF is Shown in Visual Studio.*

42
This database includes multiple tables for membership details, roles and profiles. The tables are:

- **ASPNET_Application** table: includes application name, id, and description.
- **ASPNET_Users** table: includes application id, user id, user name, and last activity date.
- **ASPNET_UsersInRoles** table: includes user id, role id.
- **ASPNET_PersonalizationPerUser** table: includes id, Path id, user id, page setting, and last updated date.
- **ASPNET_Membership** table: includes application id, user id, password, password format, password salt, email, password question, password answer, isApproved, isLockOut, createdDate, LastLoginDate, lastPasswordChangedDate, lastLockOutDate, failedPasswordAttemptCount.

Figure 24 shows visual representation of the ASPNETDB.MDF database diagram.

In order to store stock data and users’ stock portfolio information, we create a StockDatabase database within the SQL server. (Shown in Figure 25)
Figure 24. Visual Representation of the ASPNETDB.MDF Database Diagram.

Figure 25. Creating a new SQL Server Database using Visual Studio 2005.
StockDatabase includes the following tables:

- **SYMBOL_CATEGORY** table: includes stock symbols, names, exchanges, stock trade information of the latest corresponding stock data update that includes opening price, highest price, lowest price, closing price and trading volume.

- **USER_SYMBOL** table: holds registered users’ watching symbol that includes ID, symbol, and user id.

- **USER_HOLD** table: includes user’s stock portfolio information such as user id, stock symbol, price, and volumes.

- **SYMBOL_DTH** (Daily Trade History) tables hold the day-trading raw data for each stock. The data fields include opening price, highest price, lowest price, closing price and trading volume. Each stock will have its own table with the table name composed of the symbol name followed by _DTH. For example: GOOG_DTH holds the GOOG stock daily data.

- **SYMBOL_WTH** (Weekly Trade History) tables hold the week-trading raw data for each stock. The data fields include opening price, highest price, lowest price, closing price and trading volume. Each stock will have its own table with the table name composed of the symbol name followed by _WTH.

- **SYMBOL_MTH** (Monthly Trade History) tables hold the month-trading raw data for each stock. The data fields include opening price, highest price, lowest price, closing price and trading volume. Each stock will have its own table with the table name composed of the symbol name followed by _MTH.
Based on the database tables' descriptions, our stock database can be modeled as a dimensional database. Dimensional model is a specialized adaptation of the relational database model often used to represent data in a time dependent data-warehouse. The dimensional model takes advantage of the OLAP (On Line Analytical Processing) queries to summarize the data. In any dimensional model, database consists of a single large table of facts that are described using dimensions and measures. Such model is especially useful in an application that involves time series and enables users to analyze large amounts of data with very fast response time (Wikimedia Foundation, Inc. 2006c).

Our stock database dimensional model is implemented as a dimensional database consisting of one large facts table (*SYMBOLCATEGORY*). The surrounding tables contain the dimensions: Daily Trade History for each symbol, Weekly Trade History for each symbol and Monthly Trade History for each symbol. The user's information might be represented by multiple tables and the relationships that exist between the *SYMBOLCATEGORY* table and historical symbol tables. Figure 26 shows visual representation of the Stock Database diagram.
Figure 26. Visual Representation of the Stock Database Diagram.

4.2 Communicating with the Database

In order to retrieve data from a database, our application needs to know how to communicate with one. SQL (Structured Query Language) is the language used for such communication with modern Relational Database Management Systems (RDBMS). Most database systems support a particular dialect of SQL, such as T-SQL (Transact-SQL) for SQL Server and PL/SQL (Procedural Language extensions to SQL) for Oracle.
In our application, the communication is done by using T-SQL commands. The basic and most important T-SQL commands are SELECT, INSERT, UPDATE, and DELETE. The SELECT statement is used to query the database and retrieve selected data that match the specified criteria. The INSERT statement is used to insert or add a row of data into the table. The UPDATE statement is used to modify existing data. The DELETE statement is used to delete a row of data from the table. However, these commands can be sent either directly from the middle tier to SQL server or can be centralized and saved as stored procedures as a part of the database. In our application we gave preference to the stored procedures because of the following advantages:

- Storing SQL code as a stored procedure usually results in better performance because the database can optimize the data access plan used by the procedure and cache it subsequent reuse.

- Using stored procedures breaks down complex tasks into smaller and more logical modules for better maintainability of data access and code manipulation.

- Security is enhanced because SQL Server sets different security permissions for each individual stored procedure.

- Stored procedures can reduce network traffic.

We create a number of different SQL queries as stored procedures to perform a variety of tasks such as access, insertions, updates and deletions (see Figure 27).
Figure 27. Stored Procedures are shown in Visual Studio 2005.

Stored procedures are stored on the database server. Once a store procedure is created, it needs to be given a name. Much like normal functions, stored procedures accept input and output parameters and have return values. When using stored procedures, we pass the name of the procedure together with the values of any parameter it might have. For example, "InsertStockData" is one of the stored procedures shown in Figure 28.
Figure 28. Stored Procedure InsertStockData.

The main purpose of ‘InsertStockData’ procedure is to insert a new stock trade data into database. The procedure takes the following arguments: stock trade date, open price, highest price, lowest price, close price and volume. It also requires the table name that SQL command will be performed on. It returns the number of affected rows if the call is successful. Otherwise it returns -1. Figure 29 illustrates the output of InsertStockData stored procedure as reported by Visual Studio 2005.
Figure 28. Output of running InsertStockData Stored Procedure.

4.3 Data Acquisition and Access

Yahoo! Finance web site offers historical stock data for any symbol. The most recent real-time data that can be obtained is delayed by 15 minutes. The quotes can be obtained in both text and .csv (comma separated values) format. A screen shot of Yahoo stock text file including historical GOOG data is given in Figure 30.
Figure 30. Screen Shot of Yahoo Stock Text File.

Originally our database is empty. To populate the database, a super-user can run a C# code named *GetYahooTradeDataServer* that will connect to Yahoo! Finance web site and request historical data for a selected symbol. The code is using an http session call with the following form data: a valid ticker symbol, start time, end time and frequency query parameters.

Figure 31 shows *GetYahooDataServer* user front that will invoke *GetYahooData* method.

Figure 31. GetYahooDataServer Invokes GetYahooData Method.
As mentioned in Chapter 3, the *Edit Trade Data* (Figure 3.10) page provides another form for updating the stock data. When a user enters *symbol*, selects *period* and clicks *update* button, GetYahooTradeDataServer is called from the presentation tier. The presentation tier passes the symbol, time period and frequency parameters to its GetYahooData method to connect to Yahoo web server in order to get the corresponding raw data. Once the data is obtained, it is wrapped within a stock class and sent to the SQL database server. Once the class is sent a stored procedure `InsertStockData` is called by function `InsertStockData(String tableName, Stock myStock)` located in the middle tier. If the data insertion is successful, the function will return a message that will be shown on the *Edit Trade Data* page of the presentation tier.

In this chapter we covered the data tier implementation. In the next chapter, we will describe the methodology and implementation of the middle tier.
CHAPTER 5 MIDDLE TIER IMPLEMENTATION

The middle tier manages the web portal’s logic. Its code is developed in C#; an object-oriented programming language that can access .NET Framework library. The description of the implementation details of data access, stock data structure, indicator calculation, stock charts generation and Neural Networks Prediction follows.

5.1 Data Access

.NET technology provides user-friendly database access extensions that can be used in C#. Such extension is known as ADO (ActiveX Data Objects) .NET. ADO.NET enables applications to connect to data stores and manipulate data contained. In our project, we implement four classes that make extensive use of ADO.NET for data access:

- *Generic DataAccess* class provides generic database operations and database access commands, such as opening connection to the SQL Server database, performing the needed operations within the database and closing the connection to the database. For example, *Create Command* creates a new *DbCommand* object on a new connection. *ExecuteSelectCommand* executes a command and returns the results as a data table object. *ExecuteNonQuery* (*DbCommand command*) executes an update, deletion, or insertion and returns the number of affected rows by the query. We use

54
GenericDataAccess class to store these common database access operations. The class also implements basic error-handling and logging functionality.

- CatalogAccess class provides the stock catalog and stock trade history data access and updates methods that will retrieve, insert, delete, and update stock data information. It is used for execution of selected stored procedures. Given the input parameters, the stored procedures return the retrieved stock data, or stock data updates.

- UserAccess class provides user stock portfolio data access and update methods that will retrieve, insert, delete, and update users’ portfolio information.

- DongxiaoConfiguration class provides easy access to various configuration settings, such as the database connection string, data provider name and web configuration settings.

All the classes are declared static. This improves the performance, because static classes do not need to be re-initialized each time a new visitor makes a new request (Darie, 2006).

5.2 Stock Data and Indicator Calculations

The main purpose of our stock portal is to manage and analyze stock data. The data is either in the raw format gotten directly from the third tier database or Yahoo! Finance web site including open and close price, highest and lowest price, and volumes, or it is derived from the raw data including technical indicators and predictors.
Technical indicators are mathematical statistics derived from stock trade historic data. They are used in technical analysis for prediction of changes in trends or price patterns. Various technical indicators have been developed and presented to investors in a formula format. The determination of which indicator is used to capture the buy or sell signal is based on the users’ cumulative experience. Four popular technical indicators are Simple Moving Average (SMA), Exponential Moving Average (EMA), Relative Strength Index (RSI) and Moving Average Convergent/Divergence (MACD). A brief description and formula for each of the indicators are provided.

5.2.1. Simple Moving Average (SMA). Moving Average (MA) is an indicator that shows the average value of a stock price over a period of time. Simple MA is the un-weighted mean of \( n \) stock data points over \( n \) time periods. Usually it is calculated by calculating the mean of the closing price. In other words, if the closing prices are \( P_1 \) to \( P_n \) then the formula for SMA is:

\[
SMA = \frac{P_1 + P_2 + \cdots + P_n}{n}
\]

In technical analysis there are various popular values for \( n \), such as 10 days, 20 days or 50 days. The selected period depends on the kind of movements one is focusing on, such as short or long term. Short-term averages respond ‘quickly’ to the changes in the price of the underlying stock, while long-term averages are ‘slower’.
In any case, a moving average lags behind the latest price action, because of its smoothing historical nature. An SMA can lag to an undesirable extent, and can be influenced too much by old prices (Wikimedia Foundation, Inc. 2006d). This behavior can be corrected by associating extra weight to recent prices. One such mean is presented next.

5.2.2. Exponential Moving Average (EMA). The Exponential MA applies weighing factors for each day. The corresponding factors decrease exponentially. EMA emphases on more recent data, hence assigning less weight on past data. It can be calculated in two ways - as a percent-based EMA or as a period-based EMA. A percent-based EMA has a percentage as its single parameter while a period-based EMA has a parameter that represents the duration of the EMA (StockCharts.com, 2006). The formula is:

\[ EMA_{current} = (Price_{current} - EMA_{prev}) \times \alpha + EMA_{prev} \]

- In the percentage-based EMA[\( \alpha \)], \( \alpha \) is equal to the EMA’s specified percentage.

- In a period-based EMA[\( n \)], \( \alpha = \frac{2}{n + 1} \) where \( n \) is the specified number of periods. For example, a 10-period EMA’s multiplier is calculated as follows:

\[ \alpha = \frac{2}{11} = 0.1818 \ (18.18\%) \]

In our project, we only use the period-based EMA.

5.2.3. Relative Strength Index (RSI). It compares the relative strength of price gains. The index represents the ratio of price gains of a single security on days that close above the previous day’s close, over the price losses on days that close below the previous day’s close (Welles,
For each day an upward change \((U)\) or downward change \((D)\) amount is calculated. On an ‘up-day’ (today’s close higher than yesterday’s close) changes are:

\[
U = \text{close}_\text{today} - \text{close}_\text{yesterday}
\]
\[
D = 0
\]

Conversely on ‘down-day’ (today’s close lower than yesterday’s close).

\[
U = 0
\]
\[
D = \text{close}_\text{yesterday} - \text{close}_\text{today}
\]

The ratio of those averages is the Relative Strength

\[
RS = \frac{\text{EMA}[n]\ of\ U}{\text{EMA}[n]\ of\ D}
\]

This is converted to a Relative Strength Index between 0 and 100 using the following formula:

\[
RSI = 100 - \frac{100}{1 + RS}
\]

Wilder recommended a smoothing period of \(N=14\). This uses EMA smoothing with \(\alpha=1/14\). Wilder considered a stock to be overbought if the RSI reached the 70 level, meaning that the investor should consider selling. Conversely a stock is ‘oversold’ if it is at the 30 level.

5.2.4. *Moving Average Convergent/Divergence (MACD)*. MACD is a trend following indicator for the trend change identification (Wikimedia Foundation, Inc, 2007e). It shows the difference between a fast and slow closing price EMA. Fast EMA is calculated based on a short period average, while slow EMA is based on a long period one. The standard periods are \(n=12\) and \(n=26\) days. The MACD formula is:
MACD = EMA[12]– EMA[26]

A signal line is then formed by additional EMA. The standard period for the signal is 9 days:

\[ \text{Signal} = \text{EMA}[9] \]

The basic MACD trading rule is to sell when the MACD falls below its signal line, and to buy when the MACD rises above it.

We implement two programming classes that implement the technical indicators. The implemented classes are:

- *Stock* class implements the data structure that wraps stock trade data.
- *StockTable* class constructs a table used to store historic stock trade data together with various technical indicators that are calculated from the stored data.

The code for both classes is available in the Appendix.

5.3 *Charts Build*

Generating stock charts is one of the most important features of our stock web portal. The charts are dynamically created in the middle tier according to users’ requirements. We use GDI+ (Graphics Device Interface-plus) which is an all-purpose drawing model for .NET applications. Using GDI+ methods to draw graphics requires more real time than having pre-generated static
images. However, using GDI+ provides higher level of flexibility and interactivity since the graphs are generated dynamically. The heart of GDI+ programming is the “System.Drawing.Graphics” class. The Graphics class encapsulates a GDI+ drawing surface, whether it is a window, a print document, or an in-memory bitmap (Microsoft Corporation 2006c).

When user enters a symbol and clicks on Build Chart button on any Chart tools page, the query string is sent from presentation tier to middle tier. In order to build the chart, the middle tier first communicates with the data tier to retrieve the corresponding historical data, second, wraps and calculates necessary chart meta-data and, finally third, draw the chart and chart image is returned in the response stream as binary data to displays on the presentation tier.

There are three classes in the middle tier used to implement the chart generation:

- *ChartData* class contains several properties and methods. As its name explains, it is used to wrap the necessary data to be used for drawing stock chart. The properties include a symbol string, a data table necessary to hold trade data, maximum highest price, minimum lowest price, and maximum volume for calculating chart legends. The data table consists of five columns (date, open price, high, low, close price and volume), and has the same structure as the trade history data table stored in the database. The *LoadDataFromDatabase* method is used to import data into the table
from our stock database; LoadTableFromYahoo method is used to import data into
the table from Yahoo’s Web service.

- **Mag** class is a static class that provides methods for calculations of the grid intervals
  for chart price’s and chart volume’s legends. The price legends’ scalars are based on a
  range between maximum highest price and minimum lowest price, while the volume
  legends’ scalars are based on the maximum volume only.

- **ChartBuild** contains several properties and methods for generating the dynamic stock
  charts. One of its methods creates in-memory bitmap containing the chart to be
  displayed. The chart can then be drawn using different chart types, bitmap image
  sizes and color scheme based on the user’s input.

The source code for all three classes is available in the Appendix.

5.4 Neural Networks Prediction

Neural Network prediction plays a significant role in our stock Web portal. We employ a
neural network to predict stock trends by using historical stock data together with the technical
analysis indicators.

There are many types of neural networks. Our preference was given to multilayered
Feed Forward Neural Networks (FFN). This is the most common network architecture for
financial predictions. The training of such networks makes use of the back propagation principle.
Back-propagation process propagates errors from the output layer towards the input layer during
training. As the errors are propagated, the connection weights are changed. Training a neural network involves presenting input training patterns to the network with the goal of minimizing the error. Training continues until the output errors are sufficiently small to be accepted.

Our portal provides two classes necessary for Neural Network training, and prediction:

- *NeuralNetData* class provides a data structure for storing corresponding data necessary for Neural Network training, and evaluation. The data is obtained from the database tier. The data is adjusted in order to be fit to the Neural Network.

- *NeuralNet* class provides methods necessary for the Neural Network training and prediction. The main methods related to the Neural Network algorithm are *InitializeNeuralNet, ForwardPropagation, BackPropagation, and adjustWeights.*

The complete code can be viewed in the Appendix.

The behavior of our Neural Network depends on the number of parameters including: the number of layers, the number of neurons in each layer, the learning constants, and the training sequence.

After preliminary tests, the best results were achieved with 3 layers of neurons having 15, 8, 3 neurons per layer respectively. The selected learning constant was 0.001; the training procedure focused on data consisting of a sequence of 15 days of price increase or decrease at the end of the day. If the price of the stock increased, it is denoted by +1. If it decreased it is
denoted by -1. No change is denoted by 0. The total number of consecutive days that considered for input was 100.

After testing the neural network’s success rate was about 70%. Such prediction can be certainly beneficial to the investor.
CHAPTER 6 PRESENTATION TIER IMPLEMENTATION

The presentation tier is responsible for the final communication with the users. The tier makes use of middle tier objects based on the users’ selection. The screenshots from previous chapter already show the presentation tier’s output. In this chapter, we introduce the presentation tier implementation. The implementation is developed using ASP.NET technology.

6.1 Components of ASP.NET Web Pages and Server Controls

ASP.NET relies on re-use of elements of visual functionality known as server controls. Each server control has the runat = “server” attributes, which means that when the pages are requested, each element is processed at the server, in order to generate HTML to be sent to the client.

ASP.NET provides two distinct types of server controls: HTML and Web server controls. HTML server controls are presented in Figure 32 (a), while Web server controls are depicted in Figure 32 (b).
The HTML server controls correspond to the standard HTML elements such as radio buttons, text areas and check boxes. Web server controls map to specific functionality on the ASP.NET pages. Web server controls include traditional form controls such as buttons and text boxes as well as complex controls such as tables. They also include controls that provide commonly used form functionality such as displaying data in a grid, choosing dates, displaying menus, and so on. The determination of which type of control is to be added to a page depends
on the required functionality. We use ASP.NET's standard web server controls for the majority of server-side controls and static HTML elements for layout.

In order to programmatically add a control to a page, the following steps should be implemented:

1. Create an instance of the control and set its properties.
2. Add the new control to the Controls collection of a container already on the page.

In addition to using Web server controls in our Portal Web pages, we also created our own custom, reusable user controls (as shown under UserControl folder in Figure 22). Web User Controls are developed from basic Web controls and HTML content. And they are usually prepared to be used multiple times. They are always embedded within an ASP page container and can not be directly accessed by a user. Web User Controls are stored in .ascx files and can be included within any other Web Form, with the parent Web Form becoming the container of the control.

In order to include a user control in an ASP.NET Web page, the following steps should be completed:

1. In the containing ASP.NET Web page, create an @Register directive that includes a TagPrefix, TagName and Src that is associated with this user control.
2. In the body of the Web page, declare the user control element inside the form element.
6.2 Master Pages

Master pages are used for creating a consistent layout for a group of pages within an application. A master page defines the common look, feel and standard behavior for this group of pages. Master Page is defined by a Web page layout, complete with all the usual details such as header, footer, and menu bar. The advantages of using master pages are as follows:

- The ability to define a portion of a page separately and reuse it on multiple pages.
- The ability to create a locked-in layout that defines editable regions. Pages that reuse this template are then constrained to adding or modifying content in the allowed regions.
- The common functionality of the pages can be easily updated.

We create two master pages for our Stock Portal Web site: StockPortal.master (see Figure 33) and Administ.master (see Figure 34). The first one is used for all common pages of our Web site. The last one is just for administrators’ pages.

![Image of Stock Portal](image)

**Figure 33.** StockPortal.master Template.
6.3 Site Navigation

Navigation is a fundamental component of a Web site. The Stock Portal provides multiple functionalities over a specifically designed multitude of ASP pages. It is necessary to create a site menu bar that links each page available on the site. To create such a menu bar the following steps need to be implemented:

1. Create an XML file named Web.sitemap that hierarchically organizes all the pages within the site.

2. Create a user control file named menu.ascx that uses ASP.NET 2.0 Menu control and specify its contents by binding it to the Web.sitemap data source.

3. Add the menu user control to the StockPortal.master page.
CHAPTER 7 APPLICATION SECURITY

Today’s on-line applications are required to provide heighten secure environments. Our application was written with this in mind. In this chapter, we provide steps and security policies to be taken in order to make our application secure.

7.1 Security Concerns

Basically, creating a secure Web application involves three steps: First, we need to create a secure architecture and design for our application. Second, we have to adhere to creating secure code as well. Finally, we should run application within a secure environment.

For most Web applications, the fundamental tasks for creating such a secure Web application are always the same:

- **Authentication.** Authentication is the process of attempting to verify the user’s identity and ensuring the authenticity of this identity. In other words, it is a way to ensure who the user claims to be.
- **Authorization.** Authorization is the process of determining the rights and restrictions assigned to an authenticated user.
- **Confidentiality.** Confidentiality is aimed at preventing private and sensitive information disclosure.
- **Integrity.** Integrity means that the data transmitted between the client and the server can not be changed by unauthorized users.

Figure 35 illustrates the relationship among the security system in ASP.NET application (Microsoft Corporation, 2006d).

![Diagram of Security System in ASP.NET](image)

**Figure 35.** Relationships among the Security Systems in ASP.NET.

As the illustration shows, all Web clients communicate with ASP.NET applications through Microsoft Internet Information Services (IIS). IIS authenticates the request if required and then locates the requested resource such as an ASP.NET application. If the client is authorized, the resource is made available (Microsoft Corporation 2007d).

### 7.2 Authentication

ASP.NET provides additional level of security though the already provided built-in ASP.NET security features. In any ASP.NET application, authentication can be implemented through one of three possible authentication systems: Windows authentication, Forms authentication, and Passport authentication (Microsoft Corporation 2007d). In each of these,
users provide credentials when logging in. The user's identity is tracked in different ways depending on the type of authentication. Windows Authentication treats the user identity supplied by Microsoft Internet Information Services (IIS) as the authenticated identity in an ASP.NET application. Forms authentication uses a login form to receive data with basic user information to authenticate the user. Passport authentication provides information about the centralized authentication service provided by Microsoft Windows Server that offers a single login and core profile services for member sites.

ASP.NET security settings are configured in the Machine.config and Web.config files. The Machine.config file establishes the default setting created with the .NET Framework installation. The Web.config establishes the site-specific and application-specific setting overriding the settings from Machine.config. It is build and stored in the Web site root and application root directories.

Our application uses form authentication because it does not require additional Windows authentication services. When an anonymous user requests a page available only to members or administrators, such as My Portfolio or Edit symbol data page, the ASP.NET runtime verifies whether the form authentication data is available. If it is not, ASP.NET automatically redirects the user to a login page. The following steps provide set-up information for forms authentication:


2. Configure IIS to allow anonymous access to the virtual directory, and configure ASP.NET to restrict anonymous access to the web application.
3. Create a user login page (see Figure 36) that collects and validates a user name and password and then interacts with the forms authentication infrastructure for creating the ticket.

![Image of Stock Portal login page]

*Figure 36. Login Page.*

For user management, our application utilizes the ASP.NET built-in membership to validate and store user credentials. The membership supports the following functions:

- Create new users and passwords.
- Store membership information (user names, passwords, and supporting data) in Microsoft SQL Server.
- Authenticate users upon their Web access.
- Manage passwords: including password create, change, and reset.

To set-up the membership administrator needs to complete the following steps:

2. Define user accounts for membership.

3. Programming the Login control

7.3 Authorization

Authorization, in contrast to authentication, provides an effective way to control access to resources by granting or denying specific permissions to an authentication identity. In ASP.NET, there are two ways to authorize access to a given resource: File authorization and URL authorization. The File authorization checks the access control list (ACL) of the .aspx or .asmx handler file to determine whether the user should have access to the file. ACL permissions are verified by checking user’s Windows System identity. The URL authorization maps users and roles to URLs in ASP.NET application.

Since we use Forms authentication module, URL authorization is the preferred way of setting security permissions for individual Web pages. ASP.NET role management capability provides administrators with the ability to specify which resources are available to specific groups of users. Role management treats groups of users as a unit by assigning users to roles. The following steps are needed to perform the role management task:

1. Create roles for application. As mentioned in Chapter 2, the users who access our web portal are divided into three groups: anonymous users, registered members and administrators. Anonymous access is given non-authenticated users. Authenticated users are broken down into two main role categories: members and administrators (see Figure 37).
2. Set access rules to map users and roles for URL authorization described in the web.config file.

Figure 37. Roles Established by ASP.NET Web Site Administration Tool.

7.4 Confidentiality and Integrity

The final two tasks, Confidentiality and Integrity, can be controlled by the already described ASP .NET mechanisms. Additional security layer can be introduced by using encryption. The sensitive information managed by our application includes user names, passwords and database connection strings.
All the security measures described in this Chapter are required for any ASP .NET application. Upcoming operating systems and updated APS .NET libraries will only tighten the application security.
CHAPTER 8 CASE STUDY

We proceed with a case study of Google™ stock (Symbol: GOOG). A number of charts and statistics are presented.

We start with Symbol Lookup page (Figure 38 Symbol Lookup) by clicking on the Symbol Lookup tab located on the left navigation bar. The page displays the information for all symbols that currently saved in our stock database. The information includes: the ticker symbol, name, exchange, previous trading date, open price, high price, low price, close price and volume for that day. We can sort by the Symbol, Name, Open, High, Low, Close, Volume items easily, by clicking the corresponding tabs at the top of the list. The Quote link and the chart icon on the left of each row will take us directly to the real time Quote page and Charts Gallery View page correspondingly.

![Symbol Lookup](image)

To look up any stock symbol on our site, use this symbol catalog.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Exchange</th>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quote</td>
<td>AMZN</td>
<td>Amazon.com, Inc.</td>
<td>NASD</td>
<td>2-16-2007</td>
<td>39.90</td>
<td>40.44</td>
<td>39.97</td>
<td>40.23</td>
</tr>
<tr>
<td>Quote</td>
<td>DELL</td>
<td>Dell Inc.</td>
<td>NASD</td>
<td>2-16-2007</td>
<td>24.27</td>
<td>24.58</td>
<td>23.85</td>
<td>24.39</td>
</tr>
<tr>
<td>Quote</td>
<td>EBAY</td>
<td>eBay, Inc.</td>
<td>NASD</td>
<td>2-16-2007</td>
<td>93.71</td>
<td>93.78</td>
<td>93.53</td>
<td>93.62</td>
</tr>
<tr>
<td>Quote</td>
<td>EMC</td>
<td>EMC Corp</td>
<td>NYSE</td>
<td>2-16-2007</td>
<td>14.69</td>
<td>14.69</td>
<td>14.46</td>
<td>14.60</td>
</tr>
<tr>
<td>Quote</td>
<td>F</td>
<td>Ford Motor Co.</td>
<td>NYSE</td>
<td>2-16-2007</td>
<td>8.54</td>
<td>8.76</td>
<td>8.51</td>
<td>8.53</td>
</tr>
<tr>
<td>Quote</td>
<td>GE</td>
<td>General Electric Co.</td>
<td>NYSE</td>
<td>2-16-2007</td>
<td>36.07</td>
<td>36.10</td>
<td>35.87</td>
<td>35.87</td>
</tr>
<tr>
<td>Quote</td>
<td>GOOD</td>
<td>Gladstone Commercial Corp.</td>
<td>NASD</td>
<td>2-18-2007</td>
<td>20.51</td>
<td>20.51</td>
<td>20.17</td>
<td>20.29</td>
</tr>
<tr>
<td>Quote</td>
<td>GOOG</td>
<td>Google, Inc.</td>
<td>NASD</td>
<td>2-16-2007</td>
<td>462.90</td>
<td>470.15</td>
<td>462.06</td>
<td>469.94</td>
</tr>
</tbody>
</table>

Figure 38. Symbol Lookup.

76
Clicking on the **Quote** link for symbol GOOG, the real time quote for GOOG obtained on 2/20/2007 directly from Yahoo! Finance web site (with 15 minutes delay) is displayed (Figure 39).

![Stock Portal - Real Time Quote](image)

*Figure 39. Real Time Quotes for GOOG.*

Clicking on the **icon** for symbol GOOG, produces the resulting charts obtained by the chart gallery tool that provides a view of three different charts with different time frames within the same browser page (Figures 40 through 41). The daily (six months), weekly (one year), and monthly (two years) views present the short-, medium-, and long-term view respectively.

In Figure 40, presenting the GOOG Daily View, we provide a candlestick chart view for six-month daily trading records of GOOG. There are two types SM (Simple Moving Average) drawn on the chart: green line SM (10) for 10 days and blue line SM (20) for 20 days. When the
SM (10) line is above the SM (20) line, the price is likely to go up; otherwise, the price is going down.

*Figure 40.* Daily View Chart for GOOG.

*Figure 41.* Weekly View Chart for GOOG.
Figure 42. Monthly View Chart for GOOG.

Using the information in Figures 37 and 41, if an investor is interested in buying GOOG stock, she/he will look for signs of existence of longer black ‘candles’.

Next we demonstrate the usefulness of the custom chart tool. The tool provides dropdown boxes and radio buttons (shown in Figure 43) containing different type of chart attributes and indicators for user’s selection.

Figure 43. Chart Custom Menus.
Typing “GOOG” in stock symbol box and clicking on the BUILD CHART button will display the default chart for GOOG. Note that the default chart is the candlesticks chart with daily period, six months range, 720-pixel size and light gray background. User can use dropdown box to change these setting, and use radio button to add indicators. Figure 44 displays the Line chart, Figure 45 displays the HLC Bar chart, and, finally, Figure 46 shows Candlestick Chart with RIS indicator.

Figure 44. Line Chart for GOOG.
Figure 45. HLC Bar Chart for GOOG.

Figure 46. Candlestick Chart for GOOG with RIS Indicator.
In Figure 46, RIS indicator is displayed below the Volume Chart. When RIS is reaching around 70 level, the price would be going down indicating a sell signal. When RIS is reaching around 30 level, the price would be going up indicating a buy signal.

Adding Annotation to the Chart tool allows users to draw a trend line on top of a chart. Drawing a trend line requires a new window entitled ChartNotes to be open. To do so user needs to click on the ‘click me’ label. Clicking on two data points on the price plot will draw a trend line (Figure 47). In the Figure 46, we have drawn two trend lines: the top one, also known as the support line, and the bottom one, known as resistance line. Usually the price of the stock is between these two lines. If some anomaly is happening, such as the price suddenly dropping below the resistance line, it signifies the change of the trend. In the latter case this activity indicates that it would be beneficial to sell.

Figure 47. Trend Line Analyses for GOOG.
Statistical Analysis tool provides users with a tabular format view for historical trading data records and 4 types of indicators: Simple Moving Average (MA), Exponential Moving Average (EMA), Relative Strength Index (RSI) and Moving Average Convergent/Divergence (MACD). Figure 48 displays such a table for GOOG for 20 days.

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>MA10</th>
<th>MA20</th>
<th>EMA10</th>
<th>EMA20</th>
<th>RSI(14)</th>
<th>MACD</th>
<th>DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/22/2007</td>
<td>492.50</td>
<td>480.84</td>
<td>492.65</td>
<td>478.50</td>
<td>5436300</td>
<td>492.32</td>
<td>478.23</td>
<td>469.76</td>
<td>478.63</td>
<td>54.75</td>
<td>-8.44</td>
<td>60.25</td>
</tr>
<tr>
<td>1/23/2007</td>
<td>460.79</td>
<td>479.05</td>
<td>464.75</td>
<td>477.39</td>
<td>4665500</td>
<td>461.97</td>
<td>480.04</td>
<td>488.63</td>
<td>478.65</td>
<td>59.97</td>
<td>-8.47</td>
<td>58.66</td>
</tr>
<tr>
<td>1/24/2007</td>
<td>484.42</td>
<td>499.07</td>
<td>489.54</td>
<td>482.29</td>
<td>4059000</td>
<td>493.23</td>
<td>492.16</td>
<td>493.52</td>
<td>480.77</td>
<td>72.69</td>
<td>-6.72</td>
<td>58.01</td>
</tr>
<tr>
<td>1/25/2007</td>
<td>501.00</td>
<td>488.09</td>
<td>504.50</td>
<td>485.66</td>
<td>6348500</td>
<td>493.09</td>
<td>483.80</td>
<td>490.08</td>
<td>481.47</td>
<td>46.56</td>
<td>0.00</td>
<td>49.61</td>
</tr>
<tr>
<td>1/26/2007</td>
<td>490.93</td>
<td>495.84</td>
<td>497.90</td>
<td>487.03</td>
<td>5940500</td>
<td>492.70</td>
<td>485.72</td>
<td>491.73</td>
<td>482.81</td>
<td>58.34</td>
<td>-7.93</td>
<td>48.65</td>
</tr>
<tr>
<td>1/29/2007</td>
<td>498.00</td>
<td>492.47</td>
<td>498.75</td>
<td>490.50</td>
<td>4777500</td>
<td>491.49</td>
<td>496.94</td>
<td>491.37</td>
<td>463.75</td>
<td>52.71</td>
<td>-7.52</td>
<td>48.81</td>
</tr>
<tr>
<td>1/30/2007</td>
<td>494.00</td>
<td>494.32</td>
<td>498.00</td>
<td>491.22</td>
<td>4180500</td>
<td>490.46</td>
<td>493.53</td>
<td>491.91</td>
<td>494.75</td>
<td>55.68</td>
<td>-7.20</td>
<td>48.41</td>
</tr>
<tr>
<td>1/31/2007</td>
<td>496.49</td>
<td>501.50</td>
<td>501.00</td>
<td>495.51</td>
<td>5226100</td>
<td>490.88</td>
<td>492.58</td>
<td>493.65</td>
<td>466.34</td>
<td>59.49</td>
<td>-7.51</td>
<td>52.01</td>
</tr>
<tr>
<td>2/1/2007</td>
<td>506.60</td>
<td>501.75</td>
<td>506.01</td>
<td>501.55</td>
<td>5555100</td>
<td>492.77</td>
<td>491.26</td>
<td>491.96</td>
<td>485.91</td>
<td>54.59</td>
<td>-5.14</td>
<td>39.99</td>
</tr>
<tr>
<td>2/2/2007</td>
<td>492.60</td>
<td>491.60</td>
<td>486.00</td>
<td>477.81</td>
<td>6285500</td>
<td>494.44</td>
<td>491.50</td>
<td>485.67</td>
<td>465.49</td>
<td>38.25</td>
<td>-4.98</td>
<td>24.98</td>
</tr>
<tr>
<td>2/3/2007</td>
<td>477.50</td>
<td>467.16</td>
<td>478.00</td>
<td>468.19</td>
<td>7228900</td>
<td>498.06</td>
<td>490.26</td>
<td>491.56</td>
<td>463.75</td>
<td>29.78</td>
<td>-9.93</td>
<td>34.71</td>
</tr>
<tr>
<td>2/4/2007</td>
<td>468.10</td>
<td>471.48</td>
<td>473.30</td>
<td>467.26</td>
<td>5321900</td>
<td>497.32</td>
<td>495.59</td>
<td>492.02</td>
<td>462.58</td>
<td>40.93</td>
<td>-1.61</td>
<td>40.77</td>
</tr>
<tr>
<td>2/7/2007</td>
<td>477.80</td>
<td>470.01</td>
<td>474.35</td>
<td>466.78</td>
<td>4139800</td>
<td>494.41</td>
<td>489.92</td>
<td>493.55</td>
<td>491.39</td>
<td>40.80</td>
<td>-0.45</td>
<td>41.85</td>
</tr>
<tr>
<td>2/8/2007</td>
<td>468.85</td>
<td>471.03</td>
<td>473.75</td>
<td>465.15</td>
<td>4077800</td>
<td>482.71</td>
<td>476.90</td>
<td>478.00</td>
<td>480.40</td>
<td>41.54</td>
<td>0.39</td>
<td>40.09</td>
</tr>
<tr>
<td>2/9/2007</td>
<td>471.85</td>
<td>461.89</td>
<td>467.66</td>
<td>461.50</td>
<td>4989600</td>
<td>479.31</td>
<td>486.01</td>
<td>478.91</td>
<td>478.64</td>
<td>37.27</td>
<td>1.74</td>
<td>41.25</td>
</tr>
<tr>
<td>2/12/2007</td>
<td>460.68</td>
<td>458.29</td>
<td>462.29</td>
<td>455.02</td>
<td>5784500</td>
<td>475.89</td>
<td>463.67</td>
<td>472.62</td>
<td>476.71</td>
<td>38.69</td>
<td>3.05</td>
<td>46.91</td>
</tr>
<tr>
<td>2/13/2007</td>
<td>459.15</td>
<td>459.10</td>
<td>462.76</td>
<td>457.36</td>
<td>4052600</td>
<td>472.37</td>
<td>461.41</td>
<td>470.17</td>
<td>475.04</td>
<td>37.46</td>
<td>1.90</td>
<td>39.48</td>
</tr>
<tr>
<td>2/14/2007</td>
<td>460.60</td>
<td>465.93</td>
<td>469.13</td>
<td>459.22</td>
<td>5698000</td>
<td>468.61</td>
<td>479.85</td>
<td>469.40</td>
<td>474.17</td>
<td>41.99</td>
<td>-4.16</td>
<td>45.99</td>
</tr>
<tr>
<td>2/15/2007</td>
<td>466.00</td>
<td>461.67</td>
<td>466.33</td>
<td>462.72</td>
<td>4048800</td>
<td>466.70</td>
<td>478.53</td>
<td>461.95</td>
<td>472.97</td>
<td>28.46</td>
<td>4.58</td>
<td>40.45</td>
</tr>
<tr>
<td>2/16/2007</td>
<td>462.80</td>
<td>469.94</td>
<td>470.15</td>
<td>462.06</td>
<td>6177000</td>
<td>465.63</td>
<td>477.54</td>
<td>468.32</td>
<td>472.66</td>
<td>42.76</td>
<td>4.23</td>
<td>46.77</td>
</tr>
</tbody>
</table>

Figure 48. Statistical Analyses for GOOG Daily View Chart for GOOG.

For example, if MACD is below 0, as in the example in Figure 48 (-8.00), it would be beneficial to sell during the corresponding time period.

Neural Network Prediction is a powerful feature of Stock Portal. With this tool users are able to use the historical performance data to predict the growth (or reduction) of the stock value over the period of the next three days (weeks). For example, if one is interested if the GOOG stock will move up or down for the next three days, she/he would click on the Neural Network Prediction menu item. The resulting prediction is shown and can be used to
determine if it is beneficial to sell or buy over the specified period of three days. For our example, it is favorable to buy more stocks of GOOG during the 2/17/07-trading day.

![Stock Portal](image)

**Figure 49.** Neural Networks Prediction for GOOG.

It is highly recommended that users practice clear and common judgment while making financial decisions. The best method would be to use as many technical indicators and analyses as possible. Even then, users need to be cautious when deciding which stocks to buy or sell.
CHAPTER 9 CONCLUSIONS

9.1 Contributions

In this project we have developed a user-friendly, reliable and interactive Web Stock Market Portal with analytical and visualization capabilities. The strength of our portal is that it provides a multitude of chart and analytical tools valuable to today’s investors without a charge.

The portal uses real-time stock data to generate a number of visual charts including: Line Charts, Bar Charts, and Candlestick Charts with different time frame. The charts provide users with an educated insight into the stock data that can be used to direct the future financial investments.

The portal provides several analytical tools including: Technical Indicators Calculation, Trend Line Analysis, and Neural Networks Predictions. The obtained statistical measures provide information about the stocks behaviors and can help users in determining their investment risks.

Users can benefit for various prediction opportunities, ranging from visual help, such as charts, to analytical measures and scientific intelligent predictions using artificial neurons.

One the most important feature of our system is the database provided services. User can benefit from the historical stock data saved within the database, and filter it and personalize the

85
data views based on their saved portfolios. Since the portal is hosted on a computer with a web server, the portal and the personalized data can be reached from any Internet enabled computer in the world.

Table 3 lists the comparison of our Stock Web portal application against other available similar applications on the Internet.

Table 3

*Internet Applications Comparison*

<table>
<thead>
<tr>
<th>Tools</th>
<th>Prerequisites</th>
<th>Charts</th>
<th>Technical Analysis</th>
<th>Historical Data</th>
<th>RealTime Quote</th>
<th>Prediction Tool</th>
<th>Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yahoo! Finance</td>
<td>N/A</td>
<td>Bar Chart, Line Chart, Candlestick</td>
<td>Indicator</td>
<td>Yes</td>
<td>Yes</td>
<td>Human Expert</td>
<td>Free</td>
</tr>
<tr>
<td>Google Finance</td>
<td>Flash</td>
<td>Line Chart</td>
<td>Indicator</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
<td>Free</td>
</tr>
<tr>
<td>Bloomberg</td>
<td>Flash</td>
<td>Bar Chart, Line Chart, Candlestick</td>
<td>Indicator</td>
<td>Yes</td>
<td>Yes</td>
<td>Statistical Monitors</td>
<td>Free</td>
</tr>
<tr>
<td>StockCharts</td>
<td>Java</td>
<td>Bar Chart, Line Chart, Candlestick</td>
<td>Indicator Annotation</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Free/Pay</td>
</tr>
<tr>
<td>Easychart</td>
<td>N/A</td>
<td>Bar Chart, Line Chart, Candlestick</td>
<td>Indicator</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Free</td>
</tr>
<tr>
<td>Stock Web Portal</td>
<td>N/A</td>
<td>Bar Chart, Line Chart, Candlestick</td>
<td>Indicator Annotation</td>
<td>Yes</td>
<td>Yes</td>
<td>Neural Network Prediction</td>
<td>Free</td>
</tr>
</tbody>
</table>

9.2. Lessons Learned

After extensive coding and testing of our portal, a number of conclusions can be drawn:

- The three-tier architecture provides more control over the complete system: flexible, scalable, and reliability. Code modifications are easily implemented, the security of the tiers is more strenuous and controllable, and the fault tolerance of the system is improved.
- The Visual Studio integrated development environment (IDE) is highly configurable and programmer-friendly. The code management and the services control can be done using a single interface that makes any needed project/code modifications a snap.
- The embedded visualization, analytical and prediction tools combined in a single portal provide users with more information than any free publicly available stock portal. The number of tools can be easily extended if needed.

9.3. Future Improvements

Given the amount of scientific literature devoted to stock market behavior, our project can benefit from adding additional capabilities such as:

- Additional Pattern Recognition algorithms
- Two and three dimensional visualization analysis
- Additional Artificial Intelligence approaches and techniques
- Additional data collections such as financial, social and political indicators.

The technical part of the portal can be improved in the direction of providing more services to users, especially ones with mobile devices.
APPENDIX
This Appendix provides majority of source code written in C#. The complete project source is not presented due to the length requirements. The author will gladly provide the complete project with the source code by request.

File CatalogAccess.cs

```csharp
using System;
using System.Data;
using System.Data.Common;
using System.Collections;
/// <summary>
/// Product catalog business tier component
/// </summary>
public static class CatalogAccess
{
    static CatalogAccess()
    {
    }

    // Retrieve the Symbol_Category
    public static DataTable GetSymbols()
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetSymbols";
        // execute the stored procedure and return the results
        return GenericDataAccess.ExecuteSelectCommand(comm);
    }

    // Retrieve the symbol list
    public static ArrayList GetSymbolList()
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "GetSymbolList";
        // execute the stored procedure and return the results
        return GenericDataAccess.ExecuteSelectCommand(comm);
    }

    // Update symbol category details
    public static bool UpSymbolCategory(String ticker, Stock myStock)
    {
        // get a configured DbCommand object
        DbCommand comm = GenericDataAccess.CreateCommand();
        // set the stored procedure name
        comm.CommandText = "UpSymbolCategory";
        // create a new parameter
        DbParameter param = comm.CreateParameter();
        param.ParameterName = "@mySymbol";
        param.Value = ticker;
        param.DbType = DbType.String;
        param.Size = 10;
        comm.Parameters.Add(param);

        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@myDate";
        param.Value = myStock.day;
        param.DbType = DbType.Date;
        comm.Parameters.Add(param);

        // create a new parameter
        param = comm.CreateParameter();
        param.ParameterName = "@myOpen";
        param.Value = myStock.open;
        param.DbType = DbType.Decimal;
        comm.Parameters.Add(param);

        // create a new parameter
```
param = comm.CreateParameter();
param.ParameterName = "myHigh";
param.Value = myStock.high;
param.DbType = DbType.Decimal;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateParameter();
param.ParameterName = "myLow";
param.Value = myStock.low;
param.DbType = DbType.Decimal;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateParameter();
param.ParameterName = "myClose";
param.Value = myStock.close;
param.DbType = DbType.Decimal;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateParameter();
param.ParameterName = "myVolume";
param.Value = myStock.volume;
param.DbType = DbType.Int32;
comm.Parameters.Add(param);

// result will represent the number of changed rows
int result = -1;
try {
    // execute the stored procedure
    result = GenericDataAccess.ExecuteNonQuery(comm);
}

catch {
    // any errors are logged in GenericDataAccess, we ignore them here
}

// result will be 1 in case of success
if (result != -1) {
    return true;
}

// Add a new symbol
public static bool AddSymbol(string symbol, string name, string exchange) {
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "AddSymbol";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "$symbol";
    param.Value = symbol;
    param.DbType = DbType.String;
    param.Size = 10;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "$name";
    param.Value = name;
    param.DbType = DbType.String;
    param.Size = 50;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "$exchange";
    param.Value = exchange;
    param.DbType = DbType.String;
    param.Size = 10;
    comm.Parameters.Add(param);

    // result will represent the number of changed rows
    int result = -1;

    // execute the stored procedure
    result = GenericDataAccess.ExecuteNonQuery(comm);
}

// Add a new symbol
public static bool AddSymbol(string symbol, string name, string exchange) {
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "AddSymbol";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "$symbol";
    param.Value = symbol;
    param.DbType = DbType.String;
    param.Size = 10;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "$name";
    param.Value = name;
    param.DbType = DbType.String;
    param.Size = 50;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "$exchange";
    param.Value = exchange;
    param.DbType = DbType.String;
    param.Size = 10;
    comm.Parameters.Add(param);

    // result will represent the number of changed rows
    int result = -1;

    // execute the stored procedure
    result = GenericDataAccess.ExecuteNonQuery(comm);

    // result will be 1 in case of success
    return result != -1;
}
try
{
    // execute the stored procedure
    result = GenericDataAccess.ExecuteNonQuery(comm);
}
catch
{
    // any errors are logged in GenericDataAccess, we ignore them here
    // result will be 1 in case of success
    return (result != -1);
}

// Update Symbol
public static bool UpdateSymbol(string symbol, string name, string exchange)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "UpdateSymbol";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@symbol";
    param.Value = symbol;
    param.DbType = DbType.String;
    param.Size = 10;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@name";
    param.Value = name;
    param.DbType = DbType.String;
    param.Size = 50;
    comm.Parameters.Add(param);
    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@exchange";
    param.Value = exchange;
    param.DbType = DbType.String;
    param.Size = 10;
    comm.Parameters.Add(param);
    // result will represent the number of changed rows
    try
    {
        // execute the stored procedure
        result = GenericDataAccess.ExecuteNonQuery(comm);
    }
    catch
    {
        // any errors are logged in GenericDataAccess, we ignore them here
        // result will be 1 in case of success
        return (result != -1);
    }
}

// Delete Symbol
public static bool DeleteSymbol(string symbol)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "DeleteSymbol";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@symbol";
    param.Value = symbol;
    param.DbType = DbType.String;
    param.Size = 10;
    comm.Parameters.Add(param);
    // execute the stored procedure; an error will be thrown by the
    // database in case the department has related categories, in which case
    // it is not deleted

91

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
int result = -1;
try {
    result = GenericDataAccess.ExecuteNonQuery(comm);
} 
catch {
    // any errors are logged in GenericDataAccess, we ignore them here
}
// result will be 1 in case of success
return (result != -1);

public static bool CreateNewTable(String tableName) {
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "CreateNewTable";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@TableName";
    param.Value = tableName;
    param.DbType = DbType.String;
    param.Size = 20;
    comm.Parameters.Add(param);
    // result will represent the number of changed rows
    bool result = false;
    try {
        // execute the stored procedure
        result = GenericDataAccess.ExecuteNonQuery(comm);
    } 
catch {
        // any errors are logged in GenericDataAccess
    }
    // result will be 1 in case of success
    return (result);
}

public static bool InsertStockData(String tableName, Stock myStock) {
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "InsertStockData";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@TableName";
    param.Value = tableName;
    param.DbType = DbType.String;
    param.Size = 20;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@myDate";
    param.Value = myStock.day;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@myOpen";
    param.Value = myStock.open;
    param.DbType = DbType.Decimal;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@myHigh";
    param.Value = myStock.high;
param.DbType = DbType.Decimal;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateParameter();
param.ParameterName = "@myLow";
param.Value = myStock.low;
param.DbType = DbType.Decimal;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateParameter();
param.ParameterName = "@myClose";
param.Value = myStock.close;
param.DbType = DbType.Decimal;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateParameter();
param.ParameterName = "@myVolume";
param.Value = myStock.volume;
param.DbType = DbType.Int32;
comm.Parameters.Add(param);

// result will represent the number of changed rows
int result = -1;
try
{
    // execute the stored procedure
    result = GenericDataAccess.ExecuteNonQuery(comm);
}
catch
{
    // any errors are logged in GenericDataAccess
}
// result will be 1 in case of success
return (result != -1);

public static DataTable GetStockData(String tableName, DateTime startTime, DateTime endTime)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "SelectStockData";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@tableName";
    param.Value = tableName;
    param.DbType = DbType.String;
    param.Size = 20;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@startTime";
    param.Value = startTime;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "@endTime";
    param.Value = endTime;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // execute the stored procedure and return the results
    return GenericDataAccess.ExecuteSelectCommand(comm);
}
public static DataTable GetStockPrice(String tableName, DateTime startTime, DateTime endTime)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "SelectClosePrice";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "$tableName";
    param.Value = tableName;
    param.DbType = DbType.String;
    param.Size = 20;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "$startTime";
    param.Value = startTime;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "$endTime";
    param.Value = endTime;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // execute the stored procedure and return the results
    return GenericDataAccess.ExecuteScalarCommand(comm);
}

public static decimal GetMaxHighPrice(String tableName, DateTime startTime, DateTime endTime)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "GetMaxHighPrice";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "$tableName";
    param.Value = tableName;
    param.DbType = DbType.String;
    param.Size = 20;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "$startTime";
    param.Value = startTime;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "$endTime";
    param.Value = endTime;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // execute the stored procedure and return the results
    return GenericDataAccess.ExecuteScalar_decimal(comm);
}

public static decimal GetMinLowPrice(String tableName, DateTime startTime, DateTime endTime)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "GetMinLowPrice";
// create a new parameter
DbParameter param = comm.CreateCommandParameter();
param.ParameterName = "@tableName";
param.Value = tableName;
param.DbType = DbType.String;
param.Size = 20;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateCommandParameter();
param.ParameterName = "@startTime";
param.Value = startTime;
param.DbType = DbType.Date;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateCommandParameter();
param.ParameterName = "@endTime";
param.Value = endTime;
param.DbType = DbType.Date;
comm.Parameters.Add(param);

// execute the stored procedure and return the results
return GenericDataAccess.ExecuteScalar_decimal(comm);

public static int GetMaxVolume(String tableName, DateTime startTime, DateTime endTime)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "GetMaxVolume";
    // create a new parameter
    DbParameter param = comm.CreateCommandParameter();
    param.ParameterName = "@tableName";
    param.Value = tableName;
    paramDbType = DbType.String;
    param.Size = 20;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateCommandParameter();
    param.ParameterName = "@startTime";
    param.Value = startTime;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateCommandParameter();
    param.ParameterName = "@endTime";
    param.Value = endTime;
    param.DbType = DbType.Date;
    comm.Parameters.Add(param);

    // execute the stored procedure and return the results
    return GenericDataAccess.ExecuteScalar_decimal(comm);
}

public static int GetMinVolume(String tableName, DateTime startTime, DateTime endTime)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "GetMinVolume";
    // create a new parameter
    DbParameter param = comm.CreateCommandParameter();
    param.ParameterName = "@tableName";
    param.Value = tableName;
    paramDbType = DbType.String;
    param.Size = 20;
}
param.Size = 20;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateParameter();
param.ParameterName = "startTime";
param.Value = startTme;
param.DbType = DbType.Date;
comm.Parameters.Add(param);

// create a new parameter
param = comm.CreateParameter();
param.ParameterName = "endTime";
param.Value = endTime;
param.DbType = DbType.Date;
comm.Parameters.Add(param);

// execute the stored procedure and return the results
return GenericDataAccess.ExecuteScalar(comm);
}

public static DataTable GetStockDataByRows(String tableName, int rowNum)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "GetStockDataByRows";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "tableName";
    param.Value = tableName;
    param.DbType = DbType.String;
    param.Size = 20;
    comm.Parameters.Add(param);

    // create a new parameter
    param = comm.CreateParameter();
    param.ParameterName = "rows";
    param.Value = rowNum;
    param.DbType = DbType.Int16;
    comm.Parameters.Add(param);

    // execute the stored procedure and return the results
    return GenericDataAccess.ExecuteScalarCommand(comm);
}

public static DataTable SearchSymbol(String words)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "SearchSymbol";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "words";
    param.Value = words;
    param.DbType = DbType.String;
    param.Size = 100;
    comm.Parameters.Add(param);

    // execute the stored procedure and return the results
    return GenericDataAccess.ExecuteScalarCommand(comm);
}

public static DataTable GetGlossary(string index)
{
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "GlossaryIndex";
    // create a new parameter
    DbParameter param = comm.CreateParameter();

    96

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
param.ParameterName = "$index";
param.Value = index;
param.DbType = DbType.String;
param.Size = 1;
comm.Parameters.Add(param);
// execute the stored procedure and return the results
return GenericDataAccess.ExecuteSelectCommand(comm);
}

public static DateTime GetLastDate(String tableName) {
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "SelectTop";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@TableName";
    param.Value = tableName;
    param.DbType = DbType.String;
    param.Size = 20;
    comm.Parameters.Add(param);
    // execute the stored procedure and return the results
    return GenericDataAccess.ExecuteSelectCommand_Top(comm);
}

public static int CheckSymbol(String symbol) {
    // get a configured DbCommand object
    DbCommand comm = GenericDataAccess.CreateCommand();
    // set the stored procedure name
    comm.CommandText = "GetMatchSymbol";
    // create a new parameter
    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@Symbol";
    param.Value = symbol;
    param.DbType = DbType.String;
    param.Size = 10;
    comm.Parameters.Add(param);
    return (int)GenericDataAccess.ExecuteScalar(comm);
}

File ChartBuild.cs

using System;
using System.Data;
using System.Data.Common;
using System.ComponentModel;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Configuration;
using System.Collections;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.IO;
    /// <summary>
    /// Summary description for ChartBuild
    /// </summary>
    public class ChartBuild
{
    #region Enums
    public enum ChartTypes
    {
        /// <summary>
/// Candlestick Chart
/// </summary>
[Description("Candl")]
candl = 0,
/// </summary>
/// Open High Low Close Bar Chart
/// </summary>
[Description("OHLC")]
ohlc = 1,
/// </summary>
/// High Low Close Bar Chart
/// </summary>
[Description("HLC")]
hlc = 2,
/// </summary>
/// Line Chart
/// </summary>
[Description("Line")]
line

/// </summary>
/// Chart Scheme
/// </summary>
public enum ChartScheme
{
    defaultSet = 0,
    blackSet = 1
}

#region
determine the label area
private const int sideLable = 60;
private const int headLable = 36;
private const int footLable = 15;
private const int dateLable = 15;
private const int IndicatorH = 80;
#region members
private int _width;
private int _heigh;
private string _ticker;
private ChartData _mydata;
private ChartTypes _mytype;
private ChartScheme _myscheme;
private int _type;
private bool IsDrawS;
private int chartheigh;
private int chartWidth;
private Graphics g;
private Page p;
public Bitmap b;
#endregion
private SolidBrush textBrush;
private SolidBrush dwBrush;
private SolidBrush upBrush;
private SolidBrush bgBrush;
private SolidBrush grayBrush;
private Pen bluePen;
private Pen greenPen;
private Pen fuchsiaPen;
private Pen bluePen2;
private Pen greenPen2;
private Pen fuchsiaPen2;
private Pen markPen;
private Pen dash_pen;
private Pen grayPen;
private Pen upPen; //Pen for draw up price
private Pen dwPen; //Pen for draw down price
#endregion
#region Font Set
private Font head_font;
private Font head_font1;
private Font head_font2;
private Font small_font;
private Font smaller_font;
private Font datefont;
#endregion

private Color bg;
#endregion

#region Accessors
public ChartData mydata
{
    get { return _mydata; }
    set { _mydata = value; }
}

public ChartTypes mytype
{
    get { return _mytype; }
    set { _mytype = value; }
}

public ChartScheme Myscheme
{
    get { return _myscheme; }
    set { _myscheme = value; }
}
#endregion

#region constructor
public ChartBuild()
{
}

public ChartBuild(ChartData data, int width, int heigh, int inType, Page pPage)
{
    this._mydata = data;
    this._ticker = data.Symbol;
    this.inType = inType;
    if (inType == 0)
    {
        this._width = width;
        this._heigh = heigh;
    }
    else
    {
        this._width = width;
        heigh = heigh + IndicatorH;
        this._heigh = heigh;
    }

    b = new Bitmap(width, heigh);
    g = Graphics.FromImage(b);
    p = pPage;
}

public ChartBuild(string ticker, int width, int heigh, int inType, Page pPage)
{
    this._ticker = ticker;
    this.inType = inType;
    if (inType == 0)
    {
        this._width = width;
        this._heigh = heigh;
    }
    else
    {
        this._width = width;

99

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
heigh = heigh + IndicatorH;
    this._heigh = heigh;
}

this.inType = inType;
b = new Bitmap(width, heigh);
g = Graphics.FromImage(b);
p = pPage;
}
#endif constructor

private void SetFont()
{
    head_font = new Font("Arial", 12, FontStyle.Bold);
    head_font1 = new Font("Arial", 10, FontStyle.Bold);
    head_font2 = new Font("Arial", 10);
    small_font = new Font("Arial", 8);
    smaller_font = new Font("Arial", 7);
    datefont = new Font("Arial", 8);
}

private void SetPen()
{
    bluePen = new Pen(Color.Blue, 1);
    greenPen = new Pen(Color.Lime, 1);
    fuchsiaPen = new Pen(Color.Fuchsia, 1);
    bluePen2 = new Pen(Color.Blue, 2);
    greenPen2 = new Pen(Color.Lime, 2);
    fuchsiaPen2 = new Pen(Color.Fuchsia, 2);
    markPen = new Pen(Color.Gray);
    dash_pen = new Pen(Color.Gray);
    grayPen = new Pen(Color.LightGray);
    dash_pen.DashStyle = DashStyle.Dot;
}

private void SetScheme(int myscheme)
{
    switch (myscheme)
    {
    case 0:
    {
        //create back color for chart
        bg = Color.FromArgb(245, 245, 245);
        //create brushes and pens
        textBrush = new SolidBrush(Color.Black);
        dwBrush = new SolidBrush(Color.Red);
        upBrush = new SolidBrush(Color.Black);
        bkBrush = new SolidBrush(bg);
        grayBrush = new SolidBrush(Color.LightGray);
        upPen = new Pen(Color.Black);
        dwPen = new Pen(Color.Red);
    }
    break;
    case 1:
    {
        //create back color for chart
        bg = Color.FromArgb(0, 0, 0);
        //create brushes and pens
        textBrush = new SolidBrush(Color.Black);
        dwBrush = new SolidBrush(Color.Red);
        upBrush = new SolidBrush(Color.Green);
        bkBrush = new SolidBrush(bg);
        grayBrush = new SolidBrush(Color.LightGray);
        upPen = new Pen(Color.Black);
        dwPen = new Pen(Color.Red);
    }
    break;
    }
}
private int GetHowmanyUnits()
{
    return _width < 800 ? 8 : 10;
}

private void SetChartSize()
{
    chartWidth = _width - sideLabel;
    if (inType != 0)
    {
        chartheigh = _heigh - (headLable + footLable) - IndicatorH;
    }
    else
    {
        chartheigh = _heigh - (headLable + footLable);
    }
}

public void draw(ChartData mydata, int myscheme, int type, int inType)
{
    //initialize the chart image
    g.FillRectangle(new SolidBrush(Color.White), 0, 0, _width, _heigh);
    g.SmoothingMode = SmoothingMode.AntiAlias;

    //create fonts
    SetScheme(myscheme);
    SetPen();
    SetFont();
    //chart area
    SetChartSize();
    int priceAreaH = chartheigh * 2 / 3;
    int priceAreaW = chartWidth;
    int volumeAreaH = chartheigh - (priceAreaH + dateLable * 2);
    int volumeAreaW = chartWidth;

    //draw head title
    g.DrawString(_ticker.ToUpper(), head_font, textBrush, 1, 1);
    Stock lastcd = new Stock();
    lastcd = (Stock)mydata.dataList[mydata.dataList.Count - 1];
    DateTime lastdate = Convert.ToDateTime(lastcd.day);
    g.DrawString(lastdate.ToString("ddmmdd yyyy"), head_font2, textBrush, (_width - 180), 1);
    g.DrawString("Copyright Dongxiao", small_font, grayBruh, (_width - 120), _heigh - 15);

    //g.DrawString("MA(20) MA(10)", head_font1, textBrush, 30, _heigh - 15);
    g.DrawString("Open High Low Close Volume", head_font1, textBrush, 1, 18);
    //g.DrawLine(bluePen2, 4, _heigh - 8, 20, _heigh - 8);
    //g.DrawLine(greenPen2, 80, _heigh - 8, 96, _heigh - 8);

    string o = Convert.ToString(lastcd.open);
    string h = Convert.ToString(lastcd.high);
    string l = Convert.ToString(lastcd.low);
    string c = Convert.ToString(lastcd.close);
    int v = lastcd.volume / 1000;
    g.DrawString(o, head_font2, textBrush, 40, 18);
    g.DrawString(h, head_font2, textBrush, 115, 18);
    g.DrawString(l, head_font2, textBrush, 190, 18);
    g.DrawString(c, head_font2, textBrush, 270, 18);
    g.DrawString(v + "K", head_font2, textBrush, 380, 18);

    //draw and fill work area
    //g.DrawRectangle(new Pen(Color.Gray, 1), 0, headLable, width - 1, priceAreaH);
    //g.DrawRectangle(new Pen(Color.Blue, 1), 0, priceAreaH + dateLable + headLable, width - 1, volumeAreaH + dateLable);
g.FillRectangle(bgBrush, 0, headLabel, priceAreaW, priceAreaH);
g.FillRectangle(bgBrush, 0, headLabel + priceAreaH + 2 * dateLabel, volumeAreW, volumeAreH);

if (inType != 0)
{
g.FillRectangle(bgBrush, 0, headLabel + priceAreaH + 2 * dateLabel + volumeAreW, volumeAreW, IndicatorH);
g.DrawRectangle(markPen, 0, headLabel + priceAreaH + 2 * dateLabel + volumeAreW, volumeAreW, IndicatorH);
}
g.DrawRectangle(markPen, 0, headLabel, priceAreaW, priceAreaH);
g.DrawRectangle(markPen, 0, headLabel + priceAreaH + dateLabel, volumeAreW, volumeAreH);

//g.DrawLine(thinPen, 0, headLabel, sideLabel, priceAreaH + headLabel + dateLabel + volumeAreaH + dateLabel); //15 date label
//g.DrawLine(thinPen, 0, priceAreaH + headLabel, width - 2, priceAreaH + headLabel);

int y, i, low_pt, high_pt, open_pt, close_pt, pre_close, avg_pt, preavg_pt, avg_pt1, preavg_pt1, eav_pt, preeav_pt, vol_pt, p_ticks, v_ticks, y_mag, v_mag;
decimal p_range, p_high, p_low, p_unit, p_mag, p_label, v_volume, v_unit;

determine the price magnitude and units
p_range = (mydata.PriceMax - mydata.PriceMin); p_high = mydata.PriceMax + decimal.Multiply(p_range, 0.05m);

p_low = mydata.PriceMin - decimal.Multiply(p_range, 0.05m);

p_range = p_high - p_low;
p_unit = (p_high - p_low) / priceAreaH;
p_mag = Mag.GetPriceMag(p_range);
p_ticks = (int)(p_range / p_mag);
p_label = (int)(p_high - (p_high % p_mag));

y = headLabel + 1;
RectF rf = new RectF(priceAreaW - 40, -5, 80, 25);
StringFormat sf = new StringFormat();
sf.Alignment = StringAlignment.Far;
sf.LineAlignment = StringAlignment.Far;

//draw price label
y += (int)((p_high % p_mag) / p_unit);
rf.Y = y - 20;
for (i = 0; i < p_ticks; i++)
{
g.DrawString(p_label.ToString("#.00"), small_font, textBrush, rf, sf);
g.DrawLine(dash_pen, 0, y, priceAreaW, y);
g.DrawLine(markPen, priceAreaW, y, priceAreaW + 3, y);
p_label -= p_mag;
y += (int)(p_mag / p_unit);
rf.Y = y - 20;
}

y = priceAreaH + headLabel + dateLabel;
rf.Y = y - 20;

determine the volume magnitude and units
v_volume = Convert.ToDecimal(mydata.VolumeMax * 1.1);
v_unit = v_volume / volumeAreH;
v_mag = Mag.GetVolumeMag((int)v_volume);
v_ticks = (int)(v_volume / v_mag);
v_label = (int)(v_volume - (v_volume % v_mag));

//draw volume label
y += (int)((v_volume % v_mag) / v_unit) + 2;
rf.Y = y - 20;
for (i = 0; i < v_ticks; i++)

102

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
int v_lable = v_label / 10000;
g.DrawString(v_lable.Tostring("", "", "", ""), small_font, textBrush, rf, sf);
g.DrawLine(dashPen, 0, y, volumeAreW, y);
g.DrawLine(markPen, volumeAreW, y, volumeAreW + 3, y);
v_label = v_mag;
y += (int)(v_mag / v_unit);
rf.Y = y - 20;
}
g.DrawString("x10K", small_font, textBrush, rf, sf);

//determine the date magnitude and units
//String date = null;
int iDate = GetHowmanyUnits();
int count = mydata.dataList.Count;
float iSpace = count / iDate;
float xN = (float)(chartWidth - 2) / count;

int y1 = priceAreaH + headLabel;
int y2 = volumeAreH + priceAreaH + headLabel + dateLable + 2;
int y3 = _heigh - footLabel;

//draw the chart
int iNumPts = count - 19;
float xS = (float)chartWidth / iNumPts;
float ox = (float)(2.0);
preavg_pt = -1;
preavg_pt1 = -1;
preavgPt = -1;

pre_close = -1;
Stock d = new Stock();
DateTime myday = new DateTime();
DateTime last_day = new DateTime();

decimal sum_10 = 0.0m;
decimal sum_20 = 0.0m;

// Calculate moving average for first 10 day
for (i = 10; i < mydata.dataList.Count; i++)
{
    d = (Stock)mydata.dataList[i];
    sum_10 += d.close;
}
decimal pre_EMA10 = sum_10 / 10;

// Calculate moving average for first 20 day
for (i = 0; i < mydata.dataList.Count; i++)
{
    d = (Stock)mydata.dataList[i];
    sum_20 += d.close;
}
decimal pre_EMA20 = sum_20 / 20;

//loop through data and print bars for price and volume
for (i = 20; i < mydata.dataList.Count; i++)
{
    d = (Stock)mydata.dataList[i];

    if (i % iSpace == 0)
    {
        myday = Convert.ToDateTime(d.day);
        if (myday.Year != last_day.Year)
        {
            g.DrawString(myday.ToString("MM dd, yyyy"), datefont, textBrush, ox + (xS - 3) / 2, y1);
            g.DrawString(myday.ToString("MM dd, yyyy"), datefont, textBrush, ox + (xS - 3) / 2, y2);
        }
        else
        {
            g.DrawLine(dashPen, ox + (xS - 3) / 2, y1, ox + (xS - 3) / 2, y2);
        }
    }
    else
    {
```csharp
{ g.DrawString(myday.ToString("MMMM dd"), datefont, textBrush, ox + (xS - 3) / 2, y1);
g.DrawString(myday.ToString("MMMM dd"), datefont, textBrush, ox + (xS - 3) / 2, y2);
}
g.DrawLine(dash_pen, ox + (xS - 3) / 2, headLable, ox + (xS - 3) / 2, y3);
}

low_pt = Convert.ToInt32((p_high - Convert.ToDecimal(d.low)) / p_unit) + headLable;
high_pt = Convert.ToInt32((p_high - Convert.ToDecimal(d.high)) / p_unit) +
headLable;
close_pt = Convert.ToInt32((p_high - Convert.ToDecimal(d.close)) / p_unit) +
headLable;
open_pt = Convert.ToInt32((p_high - Convert.ToDecimal(d.open)) / p_unit) +
headLable;
vol_pt = Convert.ToInt32(y - (Convert.ToDecimal(d.volume) / v_unit));

//determine direction of price and use appropriate brush
if (Convert.ToDecimal(d.close) > Convert.ToDecimal(d.open))
{
if (type == 0)//candle
{
g.FillRectangle(upBrush, ox + (xS - 3) / 2, high_pt, 1, low_pt - high_pt);
g.FillRectangle(upBrush, ox, close_pt, xS - 3, open_pt - close_pt);
g.DrawRectangle(upPen, ox, close_pt, xS - 3, open_pt - close_pt);
}
else if (type == 1)
{
g.DrawLine(upPen, ox + (xS - 3) / 2, high_pt, ox + (xS - 3) / 2, low_pt);
g.DrawLine(upPen, ox - 1, open_pt, ox + (xS - 3) / 2, open_pt);
g.DrawLine(upPen, ox + (xS - 3) / 2, close_pt, ox + (xS - 3) / 2 + [xS - 3] / 2 + 1, close_pt);
}
else if (type == 2)
{
g.DrawLine(upPen, ox + (xS - 3) / 2, high_pt, ox + (xS - 3) / 2, low_pt);
g.DrawLine(upPen, ox + (xS - 3) / 2, close_pt, ox + (xS - 3) / 2 + [xS - 3] / 2 + 1, close_pt);
}
else
{
g.DrawRectangle(upPen, ox, vol_pt, xS, y - vol_pt);
}
else
{
if (type == 0)//candle
{
g.FillRectangle(dwBrush, ox + (xS - 3) / 2, high_pt, 1, low_pt - high_pt);
g.FillRectangle(dwBrush, ox, open_pt, xS - 3, close_pt - open_pt);
g.DrawRectangle(dwPen, ox, open_pt, xS - 3, close_pt - open_pt);
}
else if (type == 1)
{
g.DrawLine(dwPen, ox + (xS - 3) / 2, high_pt, ox + (xS - 3) / 2, low_pt);
g.DrawLine(dwPen, ox - 1, open_pt, ox + (xS - 3) / 2, open_pt);
g.DrawLine(dwPen, ox + (xS - 3) / 2, close_pt, ox + (xS - 3) / 2 + [xS - 3] / 2 + 1, close_pt);
}
else if (type == 2)
{
g.DrawLine(dwPen, ox + (xS - 3) / 2, high_pt, ox + (xS - 3) / 2, low_pt);
g.DrawLine(dwPen, ox + (xS - 3) / 2, close_pt, ox + (xS - 3) / 2 + [xS - 3] / 2 + 1, close_pt);
}
else
{
g.DrawRectangle(dwPen, ox, vol_pt, xS, y - vol_pt);
}
}
```
//draw line chart
if (type == 3)
{
    if (pre_close != -1)
    {
        if (pre_close < close_pt)
        {
            g.DrawLine(dwPen, ox + (xS - 3) / 2, pre_close, ox + (xS - 3) / 2 + xS, close_pt);
        }
        else
        {
            g.DrawLine(upPen, ox + (xS - 3) / 2, pre_close, ox + (xS - 3) / 2 + xS, close_pt);
        }
        pre_close = close_pt;
    }
    //Calculate last 10 days sum for Moveaverage
    decimal sum_1 = 0.0m;
    for (int n = 0; n < 10; n++)
    {
        Stock pd_1 = (Stock)mydata.dataList[i + n - 10];
        sum_1 += pd_1.close;
    }
    decimal avgClose_1 = sum_1 / 10;
    //Calculate last 20 days sum for Moveaverage
    decimal sum_2 = 0.0m;
    for (int j = 0; j < 20; j++)
    {
        Stock pd = (Stock)mydata.dataList[i + j - 20];
        sum += pd.close;
    }
    decimal avgClose = sum / 20;

    last_day = myday;
    avg_pt = Convert.ToInt32((p_high - avgClose) / p_unit) + headLabel;
    avg_pt1 = Convert.ToInt32((p_high - avgClose_1) / p_unit) + headLabel;

    //draw simple MA
    if (preavg_pt1 != -1)
    {
        g.DrawLine(greenPen, ox + (xS - 3) / 2, preavg_pt1, ox + (xS - 3) / 2 + xS, avg_pt1);
    }
    preavg_pt1 = avg_pt1;

    if (preavg_pt != -1)
    {
        g.DrawLine(bluePen, ox + (xS - 3) / 2, preavg_pt, ox + (xS - 3) / 2 + xS, avg_pt);
    }

    //Calculate EMA
    decimal preEMA10 = (d.close - pre_EMA10) * 0.1818m + pre_EMA10;
    pre_EMA10 = preEMA10;
    eav_pt = Convert.ToInt32((p_high - preEMA10) / p_unit) + headLabel;

    if (preeav_pt != -1)
    {
        g.DrawLine(fuchsiaPen, ox + (xS - 3) / 2, preeav_pt, ox + (xS - 3) / 2 + xS, eav_pt);
    }
    preeav_pt = avg_pt;
    preeav_pt = eav_pt;
    ox += xS;
public void drawfromtable(StockTable mytable, decimal priceMax, decimal priceMin, int maxVolume, int intScheme, int intType, int inType) {
    //initialize the chart image
    g.FillRectangle(new SolidBrush(Color.White), 0, 0, _width, _heigh);
    g.SmoothingMode = SmoothingMode.AntiAlias;
    //create fonts
    SetScheme(intScheme);
    SetPen();
    //chart area
    SetChartSize();
    int priceAreaH = chartheigh * 2 / 3;
    int priceAreaW = chartWidth;
    int volumeAreaH = chartheigh - (priceAreaH + datelable * 2);
    int volumeAreaW = chartWidth;

    //draw head title
    g.DrawString(_ticker.ToUpper(), head_font, textBrush, 1, 1);
    // Stock lastcd = new Stock();
    // List lastcd = (Stock) mydata.dataList[mydata.dataList.Count - 1];
    DateTime lastdate = Convert.ToDateTime(dr["Date"]);
    g.DrawString(lastdate.ToString("dddd, dd-MMM-yyyy"), head_font2, textBrush, (_width - 180), 1);
    g.DrawString("Copyright Dongxiao", small_font, grayBrush, (_width - 120), _heigh = 15);

    g.DrawString("MA(20) MA(10) EMA_10", head_font1, textBrush, 30, _heigh - 15);
    g.DrawString("Open High Low Close Volume", head_font1, textBrush, 1, 18);
    g.DrawLine(bluePen2, 4, _heigh - 8, 20, _heigh - 8);
    g.DrawLine(greenPen2, 90, _heigh - 8, 96, _heigh - 8);
    g.DrawLine(fuchsiaPen2, 165, _heigh - 8, 176, _heigh - 8);

    string o = Convert.ToString(dr["Open"]);
    string h = Convert.ToString(dr["High"]);
    string l = Convert.ToString(dr["Low"]);
    string c = Convert.ToString(dr["Close"]);
    int v = Convert.ToInt32(dr["Volume"]) / 1000;

    g.DrawString(o, head_font2, textBrush, 40, 18);
    g.DrawString(h, head_font2, textBrush, 115, 10);
    g.DrawString(l, head_font2, textBrush, 130, 18);
    g.DrawString(c, head_font2, textBrush, 270, 18);
    g.DrawString(v + "K", head_font2, textBrush, 380, 18);

    //draw and fill work area
    g.FillRectangle(bgBrush, 0, headLable, priceAreaW, priceAreaH);
    g.FillRectangle(bgBrush, 0, headLable + priceAreaH + datelable, volumeAreaW, volumeAreaH);

    if (inType != 0) {
    g.FillRectangle(bgBrush, 0, headLable + priceAreaH + 2 * datelable + volumeAreaH, volumeAreaW, IndicatorH);
    g.DrawRectangle(markPen, 0, headLable + priceAreaH + 2 * datelable + volumeAreaH, volumeAreaW, IndicatorH);
    }
    g.DrawRectangle(markPen, 0, headLable, priceAreaW, priceAreaH);
}
g.DrawRectangle(markPen, 0, headLabel + priceAreaH + dateLabel, volumeAreaW, volumeAreaH);

int y, i, low_pt, high_pt, open_pt, close_pt, pre_close, avg_pt, preavg_pt, avg_ptl, preavg_ptl, eav_pt, preeav_pt, vol_pt, p_ticks, v_ticks, v_mag, v_label;
decimal p_range, p_high, p_low, p_unit, p_mag, p_label, v_volume, v_unit;

//determine the price magnitude and units
p_range = priceMax - priceMin;
p_high = priceMax + decimal.Multiply(p_range, 0.05m);
p_low = priceMin - decimal.Multiply(p_range, 0.05m);
p_range = p_high - p_low;
p_unit = (p_high - p_low) / priceAreaH;
p_mag = Mag.GetPriceMag(p_range);
p_ticks = (int)(p_range / p_mag);
p_label = (p_high - (p_high % p_mag));
y = headLabel + 1;

Rectangle rf = new RectangleF(priceAreaW - 40, -5, 80, 25);
StringFormat sf = new StringFormat();
sf.Alignment = StringAlignment.Far;
sf.LineAlignment = StringAlignment.Far;

//draw price label
y += (int)((p_high % p_mag) / p_unit);
rf.Y = y - 20;
for (i = 0; i < p_ticks; i++)
{
    g.DrawString(p_label.ToString("#.00"), small_font, textBrush, rf, sf);
g.DrawLine(dash_pen, 0, y, priceAreaW, y);
g.DrawLine(markPen, priceAreaW, y, priceAreaW + 3, y);
p_label -= p_mag;
y += (int)(p_mag / p_unit);
rf.Y = y - 20;
}
y = priceAreaH + headLabel + dateLabel;
rf.Y = y - 20;

//determine the volume magnitude and units
v_volume = Convert.ToDecimal(maxVolume * 1.1);
v_unit = v_volume / volumeAreaH;
v_mag = Mag.GetVolumeMag((int)v_volume);
v_ticks = (int)(v_volume / v_mag);
v_label = (int)((v_volume - (v_volume % v_mag)) / v_unit);

//draw volume label
y += (int)((v_volume % v_mag) / v_unit) + 2;
rty = y - 20;
for (i = 0; i < v_ticks; i++)
{
    int v_label = v_label / 10000;
g.DrawString(v_label.ToString("#.###"), small_font, textBrush, rf, sf);
g.DrawLine(dash_pen, 0, y, volumeAreaW, y);
g.DrawLine(markPen, volumeAreaW, y, volumeAreaW + 3, y);
v_label -= v_mag;
y += (int)(v_mag / v_unit);
rty = y - 20;
}
g.DrawString("x10K", small_font, textBrush, rf, sf);

//determine the date magnitude and units
//String date = null;
int iDate = GetHowManyUnits();
int count = mytable.dataTab.Rows.Count - 1;
float iSpace = count / iDate;
float xn = (float)(chartWidth - 2) / count;

int y1 = priceAreaH + headLabel;
int y2 = volumeAreaH + priceAreaH + headLabel + dateLabel + 2;
int y3 = _height - footLabel;

//draw the chart
int iNumPts = count;
float xS = (float)chartWidth / iNumPts;
float ox = (float)(2.0);
preavg_pt = -1;
preavg_pt1 = -1;
preeav_pt = -1;
float rsi_pt;
float preRis_pt = -1;

pre_close = -1;
Stock d = new Stock();
DateTime myday = new DateTime(); ...
DateTime last_day = new DateTime();

float r_start = (float)_height - footLabel;
float r_unit = (float)IndicatorH / 100;

rf.Y = r_start - 30;
rf.X = rf.X - 15;
if (inType != 0)
{
    for (int iy = 0; iy < 5; iy++)
    {
        int rLabel = iy * 20 + 10;
        g.DrawString(rLabel.ToString(), small_font, textBrush, rf, sf);
        float uy = r_start - (float)((iy * 20 + 10) * r_unit);
        g.DrawLine(dash_pen, 0, uy, volumeAreaW + 3, uy);
        rf.Y -= 20 * r_unit;
    }
    g.DrawString("RIS", head_font1, textBrush, 15, _height - 30);
}

//loop through data and print bars for price and volume
for (i = 0; i < count; i++)
{
    DataRow drow = mytable.dataTable.Rows[i];
    if (i % iSpace == 0)
    {
        myday = Convert.ToDateTime(drow["Date"]);
        if (myday.Year != last_day.Year)
        {
            g.DrawString(myday.ToString("MM/dd/yyyy"), datefont, textBrush, ox + (xS - 3) / 2, y1);
            g.DrawString(myday.ToString("MM/dd/yyyy"), datefont, textBrush, ox + (xS - 3) / 2, y2);
        }
        else
        {
            g.DrawString(myday.ToString("MM/dd"), datefont, textBrush, ox + (xS - 3) / 2, y1);
            g.DrawString(myday.ToString("MM/dd"), datefont, textBrush, ox + (xS - 3) / 2, y2);
        }
    }
    g.DrawLine(dash_pen, ox + (xS - 3) / 2, headLabel, ox + (xS - 3) / 2, y3);
    low_pt = Convert.ToInt32((p_high - Convert.ToDecimal(drow["Low"])) / p_unit) + headLabel;
    high_pt = Convert.ToInt32((p_high - Convert.ToDecimal(drow["High"])) / p_unit) + headLabel;
}

108
close_pt = Convert.ToInt32((p_high - Convert.ToDecimal(drow["Close"])) / p_unit) + headLine;
open_pt = Convert.ToInt32((p_high - Convert.ToDecimal(drow["Open"])) / p_unit) + headLine;
vol_pt = Convert.ToInt32(y - (Convert.ToDecimal(drow["Volume"]) / v_unit));

// determine direction of price and use appropriate brush
if (Convert.ToDecimal(drow["Close"]) > Convert.ToDecimal(drow["Open"]))
{
    if (type == 0) // candle
    {
        g.FillRectangle(upBrush, ox + (xS - 3) / 2, high_pt, 1, low_pt - high_pt);
        g.FillRectangle(upBrush, ox, close_pt, xS - 3, open_pt - close_pt);
        g.DrawLine(upPen, ox, close_pt, xS - 3, open_pt - close_pt);
    }
    else if (type == 1)
    {
        g.DrawLine(upPen, ox + (xS - 3) / 2, high_pt, ox + (xS - 3) / 2, low_pt);
        g.DrawLine(upPen, ox - 1, open_pt, ox + (xS - 3) / 2, open_pt);
        g.DrawLine(upPen, ox + (xS - 3) / 2, close_pt, ox + (xS - 3) / 2 + (xS - 3) / 2 + 1, close_pt);
    }
    else if (type == 2)
    {
        g.DrawLine(upPen, ox + (xS - 3) / 2, high_pt, ox + (xS - 3) / 2, low_pt);
        g.DrawLine(upPen, ox + (xS - 3) / 2, close_pt, ox + (xS - 3) / 2 + (xS - 3) / 2 + 1, close_pt);
    }
    else
    {
        g.DrawRectangle(upPen, ox, vol_pt, xS, y - vol_pt);
    }
}
else
{
    if (type == 0) // candle
    {
        g.FillRectangle(dwBrush, ox + (xS - 3) / 2, high_pt, 1, low_pt - high_pt);
        g.FillRectangle(dwBrush, ox, open_pt, xS - 3, close_pt - open_pt);
        g.DrawLine(dwPen, ox, open_pt, xS - 3, close_pt - open_pt);
    }
    else if (type == 1)
    {
        g.DrawLine(dwPen, ox + (xS - 3) / 2, high_pt, ox + (xS - 3) / 2, low_pt);
        g.DrawLine(dwPen, ox - 1, open_pt, ox + (xS - 3) / 2, open_pt);
        g.DrawLine(dwPen, ox + (xS - 3) / 2, close_pt, ox + (xS - 3) / 2 + (xS - 3) / 2 + 1, close_pt);
    }
    else if (type == 2)
    {
        g.DrawLine(dwPen, ox + (xS - 3) / 2, high_pt, ox + (xS - 3) / 2, low_pt);
        g.DrawLine(dwPen, ox + (xS - 3) / 2, close_pt, ox + (xS - 3) / 2 + (xS - 3) / 2 + 1, close_pt);
    }
    else
    {
        g.DrawRectangle(dwPen, ox, vol_pt, xS, y - vol_pt);
    }
}

// draw line chart
if (type == 3)
{
    if (pre_close != -1)
    {
        if (pre_close < close_pt)
        {
            g.DrawLine(dwPen, ox + (xS - 3) / 2, pre_close, ox + (xS - 3) / 2 + xS, close_pt);
        }
        else
        {
            g.DrawLine(upPen, ox + (xS - 3) / 2, pre_close, ox + (xS - 3) / 2 + xS, close_pt);
        }
    }
}

109

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
pre_close = close_pt;

last_day = myday;
avg_pt = Convert.ToInt32((p_high - Convert.ToDecimal(drow("MA_10"))) / p_unit) +
headLabel;
avg_ptl = Convert.ToInt32((p_high - Convert.ToDecimal(drow("MA_20"))) / p_unit) +
headLabel;
float ux = ox - xS + (xS - 3) / 2;
float ux1 = ox + (xS - 3) / 2;

//draw simple MA
if (preavg_ptl != -1)
{
    g.DrawLine(greenPen, ux, preavg_ptl, ux1, avg_ptl);
} preavg_ptl = avg_ptl;

if (preavg_pt != -1)
{
    g.DrawLine(bluePen, ux, preavg_pt, ux1, avg_pt);
}

//Calculate EMA
eav_pt = Convert.ToInt32((p_high - Convert.ToDecimal(drow("EMA_10"))) / p_unit) +
headLabel;
if (preeav_pt != -1)
{
    g.DrawLine(fuchsiaPen, ux, preeav_pt, ux1, eav_pt);
}

if (inType != 0)
{
    rsi_pt = r_start - ((float)Convert.ToDecimal(drow(StockTable.COL_RSI))) *
    r_unit;
    if (preRis_pt != -1)
    {
        g.DrawLine(markPen, ux, preRis_pt, ux1, rsi_pt);
    }
    preRis_pt = rsi_pt;
}

preavg_pt = avg_pt;
preavg_pt = eav_pt;
ox += xS;

//Render the image to output stream
p.Response.ContentType = "image/Png";
MemoryStream mem = new MemoryStream();
b.Save(mem, System.Drawing.Imaging.ImageFormat.Png);

//Write the MemoryStream data to the output stream.
mem.WriteTo(p.Response.OutputStream);
}

} ChartBuild()
{
    if (g != null)
    {
        g.Dispose();
    }
    if (b != null)
    {
        b.Dispose();
    }
}
using System;
using System.Data;
using System.Text;
using System.Collections;
using System.ComponentModel;
using System.Configuration;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Net;
using System.IO;

/// <summary>
/// The data for build chart
/// </summary>
public class ChartData
{
    /// <summary>
    /// List of Data columns name
    /// </summary>
    public enum ColumnsName
    {
        /// <summary>
        /// Date
        /// </summary>
        [Description("Date")]
        Date,
        /// <summary>
        /// Open Price
        /// </summary>
        [Description("OpenPrice")]
        OpenPrice,
        /// <summary>
        /// Highest Price
        /// </summary>
        [Description("High")]
        High,
        /// <summary>
        /// Lowest Price
        /// </summary>
        [Description("Low")]
        Low,
        /// <summary>
        /// Close Price
        /// </summary>
        [Description("ClosePrice")]
        ClosePrice,
        /// <summary>
        /// Volume
        /// </summary>
        [Description("Volume")]
        Volume
    }

    // region constant
    public const string DTH = "_DTH";
    public const string WTH = "_WTH";
    public const string MTH = "_MTH";
    //endregion

    // region members
    /// <summary>
    /// the symbol string
    /// </summary>
}
/// <summary>
private string symbol;
/// </summary>

/// <summary>
/// The start time
/// </summary>
private DateTime start;
/// </summary>

/// <summary>
/// The end time
/// </summary>
private DateTime end;
/// </summary>

/// <summary>
/// Frequence range
/// </summary>
private char fq;
/// </summary>

/// <summary>
/// The flag for loading table
/// </summary>
private bool success = true;
/// </summary>

/// <summary>
/// Max highest price
/// </summary>
private decimal priceMax = decimal.MinValue;
/// </summary>

/// <summary>
/// Min Lowest price
/// </summary>
private decimal priceMin = decimal.MaxValue;
/// </summary>

/// <summary>
/// The max volume
/// </summary>
private int volumeMax = int.MinValue;
/// </summary>

/// <summary>
/// The table for holding trade history
/// </summary>
private DataTable tradeHistory = null;
/// </summary>

private ArrayList _dataList;
private Stock mydata;

#region accessors

tpublic decimal PriceMax
{
    get { return priceMax; }
}

tpublic decimal PriceMin
{
    get { return priceMin; }
}

tpublic int VolumeMax
{
    get { return volumeMax; }
}

tpublic bool Success
{
    get { return success; }
}
/// </summary>
/// Gets or sets the cache policy for TradeHistory Table
/// </summary>
public DataTable TradeHistory
{
get { return tradeHistory; }
set { tradeHistory = value; }
}

public String Symbol
{
    get { return symbol; }
}

public ArrayList dataList
{
    get { return _dataList; }
}

#endregion

#region Constructor
/// <summary>
/// Default Constructor
/// </summary>
public ChartData()
{
    this.priceMax = 0.0m;
    this.priceMin = 0.0m;
    this.volumeMax = 0;

    _dataList = new ArrayList();
}

/// <summary>
/// Constructor
/// </summary>
/// <param name="symbol"></param>
/// <param name="start"></param>
/// <param name="end"></param>
public ChartData(string symbol, DateTime start, DateTime end, char fq)
{
    this.symbol = symbol;
    this.start = start;
    this.end = end;
    this.fq = fq;
}

#endregion

/// <summary>
/// The method for loading data from database
/// </summary>
public void LoadDataFromDatabase()
{
    string tablename;
    //Set the table name
    if (fq == 'd')
    {
        tablename = symbol + DTH;
    }
    else if (fq == 'w')
    {
        tablename = symbol + WTH;
    }
    else
    {
        tablename = symbol + MTH;
    }
    //Get the data from database
    try
    {
        tradeHistory = CatalogAccess.GetStockData(tablename, start, end);
        priceMax = CatalogAccess.GetMaxHighPrice(tablename, start, end);
        priceMin = CatalogAccess.GetMinLowPrice(tablename, start, end);
        volumeMax = CatalogAccess.GetMaxVolume(tablename, start, end);
    }
    catch (}
public void LoadTableFromYahoo()
{
    InitialTable();
    string strData;
    GetYahooTradeDataServer ss = new GetYahooTradeDataServer();
    strData = ss.GetYahooData(symbol, start, end, fq);
    if (strData != "error")
    {
        string[] strDataArr = new string[] { },
        strDataArr = strData.Split(new Char[] { '
' });
        Array.Reverse(strDataArr);
        string[] stringLineArr = new String[strDataArr.Length];
        // last line is Yahoo column names, don't want!
        int iMarker = strDataArr.Length - 1;
    }
}

public void InitialTable()
{
    tradeHistory = new DataTable();
    DataColumn dc = new DataColumn(Convert.ToString(ColumnsName.Date), typeof(String));
    tradeHistory.Columns.Add(dc);
    dc = new DataColumn(Convert.ToString(ColumnsName.OpenPrice), typeof(Decimal));
    tradeHistory.Columns.Add(dc);
    dc = new DataColumn(Convert.ToString(ColumnsName.High), typeof(Decimal));
    tradeHistory.Columns.Add(dc);
    dc = new DataColumn(Convert.ToString(ColumnsName.Low), typeof(Decimal));
    tradeHistory.Columns.Add(dc);
    dc = new DataColumn(Convert.ToString(ColumnsName.ClosePrice), typeof(Decimal));
    tradeHistory.Columns.Add(dc);
    dc = new DataColumn(Convert.ToString(ColumnsName.Volume), typeof(Decimal));
    tradeHistory.Columns.Add(dc);
}

public string LoadTable(string tablename, DateTime start, DateTime end)
{
    Boolean init = false;
    try
    {
        DataTable myTable = CatalogAccess.GetStockData(tablename, start, end);
        for (int i = myTable.Rows.Count - 1; i >= 0; i--)
        {
            if (!init)
            {
                priceMax = Convert.ToDecimal(myTable.Rows[i]["High"]);
                priceMin = Convert.ToDecimal(myTable.Rows[i]["Low"]);
                volumeMax = Convert.ToInt32(myTable.Rows[i]["Volume"]);
                init = true;
            }
        }
    }
    //AddDataList
    mydata = new Stock();
    mydata.day = Convert.ToDateTime(myTable.Rows[i]["Date"]);
    mydata.open = Convert.ToDecimal(myTable.Rows[i]["OpenPrice"]);
    mydata.high = Convert.ToDecimal(myTable.Rows[i]["High"]);
    mydata.low = Convert.ToDecimal(myTable.Rows[i]["Low"]);
    mydata.close = Convert.ToDecimal(myTable.Rows[i]["ClosePrice"]);
    mydata.volume = Convert.ToInt32(myTable.Rows[i]["Volume"]);
    _dataList.Add(mydata);
    if (priceMax < Convert.ToDecimal(myTable.Rows[i]["High"]))
    {
        priceMax = Convert.ToDecimal(myTable.Rows[i]["High"]);
    }
    if (priceMin > Convert.ToDecimal(myTable.Rows[i]["Low"]))
    {

114

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
priceMin = Convert.ToDecimal(myTable.Rows[i]"Low");

if (volumeMax < Convert.ToInt32(myTable.Rows[i]"Volume"))
{
    volumeMax = Convert.ToInt32(myTable.Rows[i]"Volume");
}

} catch (Exception ex)
{
    return ex.Message + ": " + ex.StackTrace;
}

return "OK";
}

public string LoadData(string ticker, DateTime s, DateTime e, char f)
{
    string strData = " ";

    try
    {
        strData = GetYahooData.GetYahooString(ticker, s, e, f);

        if (strData != "error")
        {
            string[] strDataArr = new string[] { };
            strDataArr = strData.Split(new Char[] \{ \'\n\});
            Array.Reverse(strDataArr);
            string[] stringLineArr = new String[strDataArr.Length];

            int iMarker = strDataArr.Length - 1; // last line is Yahoo column names, don't want!
            Boolean init = false;
            for (int i = 1; i < iMarker; i++)
            {
                stringLineArr = strDataArr[i].Split(new Char[] \{ \',\});

                if (!init)
                {
                    priceMax = Convert.ToDecimal(stringLineArr[2]);
                    priceMin = Convert.ToDecimal(stringLineArr[3]);
                    volumeMax = Convert.ToInt32(stringLineArr[5]);

                    init = true;
                } // if
                DateTime c0 = Convert.ToDateTime(stringLineArr[0]);
                decimal c1 = Convert.ToDecimal(stringLineArr[1]);
                decimal c2 = Convert.ToDecimal(stringLineArr[2]);
                decimal c3 = Convert.ToDecimal(stringLineArr[3]);
                decimal c4 = Convert.ToDecimal(stringLineArr[4]);
                int c5 = Convert.ToInt32(stringLineArr[5]);

                AddDataList(c0, c1, c2, c3, c4, c5);
            }
        }
    }

    mydata = new Stock();
    mydata.day = c0;
    mydata.open = c1;
    mydata.high = c2;
    mydata.low = c3;
    mydata.close = c4;
    mydata.volume = c5;
    _dataList.Add(mydata);

    if (priceMax < Convert.ToDecimal(stringLineArr[2]))
    {
        priceMax = Convert.ToDecimal(stringLineArr[2]);
    }
    if (priceMin > Convert.ToDecimal(stringLineArr[3]))
    {
        priceMin = Convert.ToDecimal(stringLineArr[3]);
    }

    return strData;
private string GetUriString(string lv_tick, int freq)
{
    StringBuilder url = new StringBuilder();
    DateTime endTime = DateTime.Now;
    TimeSpan myTimespan = new TimeSpan(200, 0, 0, 0);
    DateTime startTime = endTime.Subtract(myTimespan);

    url.Append(lv_tick);
    url.Append("&s=");

    switch (freq)
    {
    case 1: // for daily view
        url.Append(startTime.Month - 1);
        url.Append("&b=");
        url.Append(startTime.Day);
        url.Append("&c=");
        url.Append(startTime.Year);
        url.Append("&d=");
        url.Append(endTime.Month - 1);
        url.Append("&e=");
        url.Append(endTime.Day);
        url.Append("&f=");
        url.Append(endTime.Year);
        url.Append("&g=d");
        break;
    case 2:
        url.Append(endTime.Month - 1);
        url.Append("&b=");
        url.Append(endTime.Day);
        url.Append("&c=");
        url.Append(endTime.Year - 2);
        url.Append("&d=");
        url.Append(endTime.Month - 1);
        url.Append("&e=");
        url.Append(endTime.Day);
        url.Append("&f=");
        url.Append(endTime.Year);
        url.Append("&g=w");
        break;
    default:
        url.Append(endTime.Month - 1);
        url.Append("&b=");
        url.Append(endTime.Day);
        url.Append("&c=");
        url.Append(endTime.Year - 5);
        url.Append("&d=");
        url.Append(endTime.Month - 1);
        url.Append("&e=");
        url.Append(endTime.Day);
        url.Append("&f=");
        url.Append(endTime.Year);
        url.Append("&g=m");
        break;
    }
    return url.ToString();
}
```csharp
return url.ToString();

public string LoadDataFromYahoo(string ticker, int f)
{
    Uri myuri = new Uri(GetUriString(ticker, f));
    try {
        WebRequest req = WebRequest.Create(myuri);
        WebResponse resp = req.GetResponse();
        StreamReader content = new StreamReader(resp.GetResponseStream(), Encoding.ASCII);
        string strData = content.ReadToEnd();
        resp.Close();

        string[] strDataArr = new string[] { ''};
        strDataArr = strData.Split(new Char[] { '\n' });
        Array.Reverse(strDataArr);
        string[] stringLineArr = new String[strDataArr.Length];

        int iMarker = strDataArr.Length - 1; // last line is Yahoo column names, don't want!
        Boolean init = false;
        for (int i = 1; i < iMarker; i++)
        {
            stringLineArr = strDataArr[i].Split(new Char[] { ',', '' });
            try {
                if (!init)
                {
                    priceMax = Convert.ToDecimal(stringLineArr[2]);
                    priceMin = Convert.ToDecimal(stringLineArr[3]);
                    volumeMax = Convert.ToInt32(stringLineArr[5]);
                }
                init = true;
            } //if
            DateTime c0 = Convert.ToDateTime(stringLineArr[0]);
            decimal c1 = Convert.ToDecimal(stringLineArr[1]);
            decimal c2 = Convert.ToDecimal(stringLineArr[2]);
            decimal c3 = Convert.ToDecimal(stringLineArr[3]);
            decimal c4 = Convert.ToDecimal(stringLineArr[4]);
            int c5 = Convert.ToInt32(stringLineArr[5]);
            //AddDataList(c0, c1, c2, c3, c4, c5);
            mydata = new Stock();
            mydata.day = c0;
            mydata.open = c1;
            mydata.high = c2;
            mydata.low = c3;
            mydata.close = c4;
            mydata.volume = c5;
            _dataList.Add(mydata);
            if (priceMax < Convert.ToDecimal(stringLineArr[2]))
            {
                priceMax = Convert.ToDecimal(stringLineArr[2]);
            }
            if (priceMin > Convert.ToDecimal(stringLineArr[3]))
            {
                priceMin = Convert.ToDecimal(stringLineArr[3]);
            }
            if (volumeMax < Convert.ToInt32(stringLineArr[5]))
            {
                volumeMax = Convert.ToInt32(stringLineArr[5]);
            }
            catch (Exception ex)
        }
    }
```


{ return ex.Message + ": " + ex.StackTrace; }
}

return "OK";
}
catch (Exception ex)
{
    return ex.Message + ": " + ex.StackTrace;
}
}

File DongxiaoConfiguration.cs
using System;
using System.Configuration;

/// <summary>
/// Summary description for DWApplicationConfiguration
/// </summary>
public class DongxiaoConfiguration
{
    // Caches the connection string
    private static string dbConnectionString;

    // Caches the data provider name
    private static string dbProviderName;

    static DongxiaoConfiguration()
    {
        dbConnectionString = ConfigurationManager.ConnectionStrings["StockDatabaseConnection"].ConnectionString;
        dbProviderName = ConfigurationManager.ConnectionStrings["StockDatabaseConnection"].ProviderName;
    }

    // Return the connection string for the StockDatabase
    public static string DbConnectionString
    {
        get
        {
            return dbConnectionString;
        }
    }

    // Returns the data provider name
    public static string DbProviderName
    {
        get
        {
            return dbProviderName;
        }
    }

    // Returns the address of the mail server
    public static string MailServer
    {
        get
        {
            return ConfigurationManager.AppSettings["MailServer"];
        }
    }

    // Send error log emails?
    public static bool EnableErrorLogEmail
```
{  
    get  
    {  
        return bool.Parse(ConfigurationManager.AppSettings["EnableErrorLogEmail"]);  
    }  
}

// Returns the email address where to send error reports  
public static string ErrorLogEmail  
{  
    get  
    {  
        return ConfigurationManager.AppSettings["ErrorLogEmail"];  
    }  
}

File GenericDataAccess.cs

using System;
using System.Data;
using System.Data.Common;
using System.Collections;
using System.Configuration;

///<summary>/
/// Class contains generic data access functionality to be accessed from
/// the business tier
///</summary>
public static class GenericDataAccess
{

    // static constructor
    static GenericDataAccess()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    // execute a command and returns the results as a DataTable object  
    public static DataTable ExecuteSelectCommand(DbCommand command)
    {
        // The DataTable to be returned  
        DataTable table = new DataTable();  
        // Execute the command making sure the connection gets closed in the end
        try
        {
            // Open the data connection
            command.Connection.Open();  
            // Execute the command and save the results in a DataTable
            DbDataReader reader = command.ExecuteReader();  
            table = new DataTable();  
            table.Load(reader);  
            // Close the reader
            reader.Close();
        }
        catch (Exception ex)
        {
            Utilities.SendErrorLogEmail(ex);
            throw ex;
        }
        finally
        {
            // Close the connection
            command.Connection.Close();
        }
        return table;
    }

119

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
// execute a command and returns the results as a ArrayList
public static ArrayList ExecuteSelectCommand_List( DbCommand command )
{
    // The ArrayList to be returned
    ArrayList myArrayList = new ArrayList();
    // Execute the command making sure the connection gets closed in the end
    try
    {
        // Open the data connection
        command.Connection.Open();
        // Execute the command and save the results in a DataTable
        DbDataReader reader = command.ExecuteReader();
        // Load date to arraylist
        while ( reader.Read() == true )
        {
            myArrayList.Add(reader["Symbol"]);
        }
        // Close the reader
        reader.Close();
    }
    catch ( Exception ex )
    {
        Utilities.LogError(ex);
        throw ex;
    }
    finally
    {
        // Close the connection
        command.Connection.Close();
    }
    return myArrayList;
}

// execute a command and returns the results as a string
public static DateTime ExecuteSelectCommand_Top( DbCommand command )
{
    // The ArrayList to be returned
    DateTime myDate = new DateTime();
    // Execute the command making sure the connection gets closed in the end
    try
    {
        // Open the data connection
        command.Connection.Open();
        // Execute the command and save the results in a DataTable
        DbDataReader reader = command.ExecuteReader();
        // Load date to arraylist
        while ( reader.Read() == true )
        {
            myDate = Convert.ToDateTime( reader["Date"]);
        }
        // Close the reader
        reader.Close();
    }
    catch ( Exception ex )
    {
        Utilities.SendErrorLogEmail(ex);
        throw ex;
    }
    finally
    {
        // Close the connection
        command.Connection.Close();
    }
    return myDate;
}

// execute an update, delete, or insert command
// and return the number of affected rows
public static int ExecuteNonQuery( DbCommand command )
{
    // The number of affected rows
    int affectedRows = -1;
    // Execute the command making sure the connection gets closed in the end
    try
    {
        // Open the connection of the command
        command.Connection.Open();
        // Execute the command and get the number of affected rows
        affectedRows = command.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        // Log eventual errors and rethrow them
        Utilities.LogError(ex);
        throw ex;
    }
    finally
    {
        // Close the connection
        command.Connection.Close();
    }
    // return the number of affected rows
    return affectedRows;

    // execute an create command
    // and return the result
    public static bool ExecuteCreate(DbCommand command)
    {
        // The return value
        string value = null;
        // Execute the command making sure the connection gets closed in the end
        try
        {
            // Open the connection of the command
            command.Connection.Open();
            // Execute the command and get the number of affected rows
            value = Convert.ToString(command.ExecuteNonQuery());
        }
        catch (Exception ex)
        {
            // Log eventual errors and rethrow them
            Utilities.LogError(ex);
            throw ex;
        }
        finally
        {
            // Close the connection
            command.Connection.Close();
        }
        // return the number of affected rows
        return (value != null);
    }

    // execute a select command and return a single result as a string
    public static string ExecuteScalar(DbCommand command)
    {
        // The value to be returned
        int value=-1 ;
        // Execute the command making sure the connection gets closed in the end
        try
        {
            // Open the connection of the command
            command.Connection.Open();
            // Execute the command and get the number of affected rows
            value = (int) command.ExecuteScalar();
        }
        catch
        {
            // Log eventual errors and rethrow them
            Utilities.LogError(ex);
            throw ex;
        }
    }
public static decimal ExecuteScalar_decimal(DbCommand command) {
    // The value to be returned
    decimal value = -1.0m;
    // Execute the command making sure the connection gets closed in the end
    try {
        // Open the connection of the command
        command.Connection.Open();
        // Execute the command and get the number of affected rows
        value = (decimal)command.ExecuteScalar();
    } catch (Exception ex) {
        // Log eventual errors and rethrow them
        Utilities.LogError(ex);
    }
    finally {
        // Close the connection
        command.Connection.Close();
    }
    // return the result
    return value;
}

// creates and prepares a new DbCommand object on a new connection
public static DbCommand CreateCommand() {
    // Obtain the database provider name
    string dataProviderName = DongxiaoConfiguration.DbProviderName;
    // Obtain the database connection string
    string connectionString = DongxiaoConfiguration.ConnectionString;
    // Create a new data provider factory
    DbProviderFactory factory = DbProviderFactories.GetFactory(dataProviderName);
    // Obtain a database specific connection object
    DbConnection conn = factory.CreateConnection();
    // Set the connection string
    conn.ConnectionString = connectionString;
    // Create a database specific command object
    DbCommand cmd = conn.CreateCommand();
    // Set the command type to stored procedure
    cmd.CommandType = CommandType.StoredProcedure;
    // Return the initialised command object
    return cmd;
}

File GetYahooQuoteService.cs

using System;
using System.Web;
using System.Text;
using System.Collections;
using System.Web.Services;

122
using System.Web.UI.HtmlControls;
using System.Net;
using System.IO;

/// <summary>
/// Summary description for GetYahooQuoteService
/// </summary>
[WebService(Namespace = "http://tempuri.org")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
{
    public GetYahooQuoteService()
    {

        // Uncomment the following line if using designed components
        InitializeComponent();
    }

    private void InitializeComponent()
    {
    }

    [WebMethod(Description = "Using stock symbol Quote information from Yahoo.",
 ENABLE_SESSION = false)]
    public string GetQuote(string ticker)
    {
        string url; // stores url of yahoo quote engine
        string buffer = "error";

        url = "http://quote.yahoo.com/d/quotes.csv?s=" + ticker + "&t=slid1tclobahvijpp2wern";

        try
        {
            WebRequest req = HttpWebRequest.Create(url);
           WebResponse resp = req.GetResponse();
            StreamReader sr = new StreamReader(resp.GetResponseStream(), Encoding.ASCII);
            buffer = sr.ReadToEnd();
            sr.Close();
        }
        catch
        {
            buffer = "error";
        }

        return buffer;
    }
}

File GetYahooQuoteService.cs

using System;
using System.Web;
using System.Text;
using System.Collections;
using System.Web.Services;
using System.Web.UI.HtmlControls;
using System.Net;
using System.IO;

/// <summary>
/// Summary description for GetYahooQuoteService
/// </summary>
[WebService(Namespace = "http://tempuri.org")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
{
public GetYahooQuoteService()
{
    // Uncomment the following line if using designed components
    InitializeComponent();
}

private void InitializeComponent()
{

    [WebMethod(Description = "Using stock symbol Quote information from Yahoo.",
        EnableSession = false)]
    public string GetQuote(string ticker)
    {
        string url; // stores url of yahoo quote engine
        string buffer = "error";
        url = "http://quote.yahoo.com/d/quotes.csv?s=" + ticker + "&sd=t&f=s1d1t1clobahgvyjpp2wern";
        try
        {
            WebRequest req = HttpWebRequest.Create(url);
            WebResponse resp = req.GetResponse();
            StreamReader sr = new StreamReader(resp.GetResponseStream(), Encoding.ASCII);
            buffer = sr.ReadToEnd();
            sr.Close();
        }
        catch
        {
            buffer = "error";
        }
        return buffer;
    }
}

File NeuralNet.cs

using System;
using System.Data;

/// <summary>
/// Summary description for NeuralNet
/// </summary>
public class NeuralNet
{
    #region constants
    /// <summary>
    /// three layers: Input+ #of Hidden + Output
    /// </summary>
    public const int Layers = 3;
    private const double alpha = 0.0;
    private const double eta = 0.001;
    private const double errEps = 0.001;
    private const double testEps = 0.5;
    private const double outEps = 0.99;
    private const int MaxEpochs = 100;
    private const double activeA = 1.7159;
    private const double activeB = 1.66666666666666667;
    private const int whistory = 2;
    private const double SQR_3 = 1.73205080756887729352;
    public const int TrainSize = 100;
}
#endregion constants

// region variables
public int[] Neurons = {14, 8, 10};
public int[] Neurons = {15, 8, 3};

private double[,][][] w;
private double[,][] bias;

private double[][] Inputs;
private double[][] Outputs;
private double[] totalError;
private double[] y;
private double[] delta;
private double[] error;

Random rand;
private double[] finalOutput;
private double[] finalInput;

public double[] FinalOutputs
{
    get { return finalOutput; }
}

public NeuralNet(double[] inDataArr, double[] outDataArr, double[] finalInputData)
{
    int rows, cols;
    rand = new Random();

    //Initialize Inputs and Outputs
    init();

    //Read input data as input layer and convert two demention array
    for (rows = 0; rows < TrainSize; rows++)
    {
        for (cols = 0; cols < Neurons[0]; cols++)
        {
            Inputs[rows][cols] = inDataArr[rows * Neurons[0] + cols]; //row by row
            Inputs[rows][cols] = inDataArr[rows + cols]; //one by one
        }
    }

    //Read input data as output layer and convert two demention array
    for (rows = 0; rows < TrainSize; rows++)
    {
        for (cols = 0; cols < Neurons[Layers - 1]; cols++)
        {
            //Outputs[rows][cols] = outDataArr[rows * Neurons[Layers - 1] + cols];
            //Outputs[rows][cols] = outDataArr[rows + cols]; //one by one
            finalInput = new double[Neurons[0]];
            //Read final input array
            for (int i = 0; i < Neurons[0]; i++)
            {
                finalInput[i] = finalInputData[i];
            }
        }
    }

}
private void init() {
    Inputs = new double[TrainSize][];
    Outputs = new double[TrainSize][];
    for (int i = 0; i < TrainSize; i++) {
        Inputs[i] = new double[Neurons[0]];
        Outputs[i] = new double[Neurons[Layers - 1]];
    }
}

public double[] ExecutTrain() {
    int l, k, m, i;
    int count = 0;
    double[,,] bias;
    double[] y;
    double[][] delta;
    double[] error;
    double Error;
    double totalError = Double.MaxValue;
    double PrevError;
    adjustInputs();
    delta = new double[Layers - 1][];
    y = new double[Layers][];
    bias = new double[whistory, Layers - 1][];
    w = new double[whistory, Layers - 1][];
    for (l = 1; l < Layers; l++) {
        delta[l - 1] = new double[Neurons[l]];
        y[l] = new double[Neurons[l]];
        for (k = 0; k < whistory; k++) {
            bias[k, l - 1] = new double[Neurons[l]];
            w[k, l - 1] = new double[Neurons[l]];[];
            for (i = 0; i < Neurons[l]; i++)
                w[k, l - 1][i] = new double[Neurons[l - 1]];
        }
    }
    error = new double[Neurons[Layers - 1]];
    initWeights(w, bias);

    // /Training
    //
    do {
        count++;
        PrevError = totalError;
        totalError = 0;
        for (m = 0; m < TrainSize; m++) {
            y[0] = Inputs[m];
            ForwardPass(w, bias, y);
            calcError(y, Outputs[m], error, out Error);
            BackwardPass(w, bias, y, error, delta);
            totalError += Error;
        }
    }

    //
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
} while (!Stopping(w, bias, y, Inputs, Outputs, error));

// //Prodict Test //

double[] yy;
y[0] = finalInput;
ForwardPass(w, bias, y);
finalOutPut = GetFinalOutput(y[Layers-1]);
return finalOutPut;
}

private void initWeights(double[][],[] w, double[][],[] bias)
{
    int i, j, k, l;
double s;
    for (i = 0; i < wHistory; i++)
        for (j = l; j < Layers; j++)
            { s = 1.0 / Math.Sqrt(Neurons[j - 1] + 1.0); // 1/sqrt(m)
                for (k = 0; k < Neurons[j]; k++)
                    { for (l = 0; l < Neurons[j - 1]; l++)
                        w[i, j - 1][k][l] = Uniform(0, s);
                        bias[i, j - 1][k] = Uniform(0, s);
                    }
            }
}

private double Uniform(double mean, double sigma)
{
    // Random rand = new Random();
    return ((SQR_3 * sigma * (2.0 * rand.NextDouble() - 1.0));
}

private double Activate(double v)
{
    return (activeA * Math.Tanh((activeA * v)));
}

private double Derivative(double y)
{
    return (activeB * (activeA + y) * (activeA - y) / activeA);
}

// <summary>
///<summary>
public void adjustInputs()
{
    double[] x = new double[Neurons[0]]; double[] xx = new double[Neurons[0]];
    int i, j;
    for (j = 0; j < Neurons[0]; j++)
        { x[j] = xx[j] = 0;
            for (i = 0; i < TrainSize; i++)
                { x[j] += Inputs[i][j];
                    xx[j] = Inputs[i][j] * Inputs[i][j];
                }
            xx[j] = (xx[j] - x[j] * x[j]) / (double)TrainSize / (double)(TrainSize - 1); //Var
            x[j] = x[j] / (double)TrainSize; // mean
        }

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
for (i = 0; i < TrainSize; i++)
    for (j = 0; j < Neurons[0]; j++)
        Inputs[i][j] = (Inputs[i][j] - x[j]) / xx[j];
//Delete x and xx?

} // ForwardPass

public void ForwardPass(double[][][] w, double[][][] bias, double[] y)
{
    double v;
    int i, j, l;

    for (l = 1; l < Layers; l++)
    {
        for (j = 0; j < Neurons[l]; j++)
        {
            v = 0;
            for (i = 0; i < Neurons[l - 1]; i++)
                v += w[0, l - 1][i] * y[0, l - 1][i];
            v += bias[0, l - 1][j];
            y[l][j] = Activate(v);
        }
    }
}

public void BackwardPass(double[][][] w, double[][][] bias, double[] y, double[] error, double[][][] delta)
{
    int i, j, k, l;
    double temp;

    for (j = 0; j < Neurons[Layers - 1]; j++)
        delta[Layers - 2][j] = error[j] * Derivative(y[Layers - 1][j]);
    for (l = Layers - 2; l > 0; l--)
        for (j = 0; j < Neurons[l]; j++)
        {
            temp = 0;
            for (k = 0; k < Neurons[l + 1]; k++)
                delta[l][k] = w[0, l][k][j];
            temp += Derivative(y[l][j]) * temp;
        }

    for (l = 0; l < Layers - 1; l++)
        for (j = 0; j < Neurons[l + 1]; j++)
        {
            for (i = 0; i < Neurons[l]; i++)
            {
                temp = w[0, l][j][i] + alpha * w[0, l][j][i] + eta * delta[l][j] * y[l][i];
                for (k = wHistory - 1; k > 0; k--)
                    w[k, l][j][i] = w[k - 1, l][j][i]; // History Shift
                w[0, l][j][i] = temp;
            }
            temp = bias[0, l][j] + alpha * bias[l, l][j] + eta * delta[l][j];
            for (k = wHistory - 1; k > 0; k--)
                bias[k, l][j] = bias[k - 1, l][j]; // History Shift
            bias[0, l][j] = temp;
        }

    public void calcError(double[][] y, double[] Out, double[] err, out double Err)
    {
        Err = 0;
        for (int j = 0; j < Neurons[Layers - 1]; j++)
        {
            err[j] = Out[j] - y[Layers - 1][j];
            Err += err[j] * err[j];
        }
        Err /= (double)Neurons[Layers - 1];

        Err = 0;
        for (int j = 0; j < Neurons[Layers - 1]; j++)
        {
            err[j] = Out[j] - y[Layers - 1][j];
            Err += err[j] * err[j];
        }
        Err /= (double)Neurons[Layers - 1];

        return Err;
    }
} // BackwardPass
public bool Stopping(double[,][][] w, double[,][] bias, double[,][] y, double[,][] In, double[,][] Out, double[] err)
{
    int m, j;
    int total = 0, HowMany;
    double Err;
    
    for (m = 0; m < TrainSize; m++)
    {
        y[0] = In[m];
        ForwardPass(w, bias, y);
        calcError(y, Out[m], err, out Err);
        HowMany = 0;
        for (j = 0; j < Neurons[Layers - 1]; j++)
            if (Math.Abs(err[j]) < testEps) HowMany++;
        if (HowMany == Neurons[Layers - 1])
            total++;
    }
    return (total == TrainSize);
}

public double[] GetFinalOutput(double[] y)
{
    double[] finalOut = new double[Neurons[Layers - 1]];
    for (int i = 0; i < Neurons[Layers - 1]; i++)
    {
        if (y[i] > 1 - outEps)
            finalOut[i] = 1;
        else if (y[i] < -1 + outEps)
            finalOut[i] = -1;
        else
            finalOut[i] = 0;
    }
    
    return finalOut;
}
REFERENCES

http://www.bloomberg.com/analysis/index.html

*Mathematics of Control, Signals and Systems*, 2, 303-314.


http://finance.easychart.net/

Magee ; New York : AMACOM.

Microsoft Corporation (2006a) Introduction to ASP.NET web pages. Retrieved on

Microsoft Corporation (2006b) SQL server overview. Retrieved on March 2006 from

Microsoft Corporation (2006c) GDI+. Retrieved on March 2006 from

Microsoft Corporation (2006d) ASP.NET security architecture. Retrieved on March

130


