Virtual Sports Stock Exchange

Chi-Chih Chen

VIRTUAL SPORTS STOCK EXCHANGE

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Chi-Chih Chen

June 2005

VIRTUAL SPORTS STOCK EXCHANGE

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Chi-Chih Chen

June 2005

Approved by:

[redacted]

George M. Georgiou, Chair, Computer Science — 06/07/05 Date

[redacted]

Kerstin Voigt

[redacted]

Owen Murphy

ABSTRACT

The goal of this project is to provide a learning environment for the people who want to experience the stock exchange market and to benefit from the use of this web based application. It is also an opportunity to experience the development of the economic model within the development of a real world application. This application will allow the users to experiment different economic models and to practice the stock exchange.

The database used in this project is Oracle 9i databases standard edition and the web server is Apache Tomcat Web Server Version 5.5. Virtual Sports Stock Exchange uses the JDBC driver provided by Oracle to establish the connection between the frontend web application and the backend database. Through the user interface built in Standard HTML and Java Server Page (JSP), it allows the users to interact with the database simultaneously. Virtual Sports Stock Exchange (VSSX) uses HTML and Java Server Page to generate the output and calculations and it uses Java Servlet to interact with the Oracle database.

VSSX is in its early stages and much enhancement may be applied later on. For example, to provide more detailed

information on each player stock with allied web sites which will make the rating system more efficient or more complex AI system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

SOFTWARE REQUIREMENT SPECIFICATION

Introduction

Purpose

The basic concept for our economic system is the balance of supply and demand. [3] When the demand is higher than the supply, the price for the demanding goods goes up; when the supply is higher than demand, the price for the goods goes down. This concept also applies to the stock market. Virtual Sports Stock Exchange Modeling System is a web based application designed to simulate the online stock exchange based on this concept.

The goal of this project is to provide a client-server learning environment for user to practice their investment skills and fulfill the Master's Degree requirement for Mr. Chi-Chih Chen.

Scope

The Virtual Sports Stock Exchange is an application that incorporates the stock market trading environment into an easy to use Web-based application. Like other online trading system, there is no need for brokers as in the real stock market does. The primary purpose of this model is to allow end user to buy and sell their virtual stocks, to track their orders, and to review their Portfolios performance as what they would do in the real stock exchange. The Virtual Sports Stock Exchange also introduces entertainment factors by using the sports players into the stock.

This project, Virtual Sports Stock Exchange, will build an open web application to support different platforms. Because this project is built with Java technology, it is client dependent. It can be run on Microsoft Windows machines, UNIX machines, Mac OS machines and more.

Definition, Acronyms, and Abbreviations

This section defines terms, acronyms, and abbreviations used in this SRS.

Apache Web Server. A open source web server that supports wide range of features and programming languages such as HTML, PHP, Java, and ASP (With Chili Soft Plug-In).

Browser. A graphical display application used to display world wide web pages. Basic functions include navigation and printing.

DBMS. A DBMS (database management system), sometimes just called a database manager, is a program that lets one or more computer users create, manage and access data in a database.

ER diagrams. Diagrams that use Entity-Relation model to design or describe database.

GUI - Graphical User Interface. Interface system that relies on graphics as well as text to display and convey information to the user.

HTML v. 3.2 - Hypertext Markup Language. HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser. The markup tells the Web browser how to display a Web page's words and images for the user. HTML v. 3.2 is the first version of HTML to support tables.

Hyperlink. A highlighted portion of text that causes a browser to take some action such as displaying another page or graphic when clicked.

Java. Java is a programming language expressly designed for use in the distributed environment of the Internet. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces a completely object-oriented view of programming. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build small application modules or applets for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.

Java Servlet. Java servlets are a key component of serve-side Java development. A servlet is a small, pluggable extension to a server that enhances the server's functionality.

Java Server Page. Java Server Pages are similar technology as the Application Server Page made by Microsoft as part of the components for serve-side development. It incorporates Java syntax into the standard HTML web page. The server takes the responsibility for compiling them into the bytes codes when the first request by the web server is initiated.

JDBC. Java Database Bridge Connectivity is a Java Application Interface that provides Java programmers with a uniform interface for accessing and manipulating a wide range of relational databases.

Oracle 9i Database. An Object database system built by Oracle which support powerful features as SQL/PLUS, Packages, Triggers and DBMS Scheduling [10]. There are not many similar database systems in the market that has come closed to this powerful system.

Document Overview

Section 2 includes product perspective, product functions, user characteristics, constraints and assumptions, and dependencies. It provides a background and requirements for the whole VSSX project.

In Section 3, the specific requirements of the external interfaces, functions, performance requirements, logical database requirements, design constraints and software system attributes will be discussed.

Overall Description

Virtual Sports Stock Exchange Overview

The virtual sports stock exchange system simulates market trading based on the world of sports. Someone who registers will be allocated a certain amount of play dollars. Players accumulate their winning amounts. The system will have the functionality of buying and selling stocks. A model will be developed to price the stock in such a way so that few participants will not be able to manipulate the price of the market. Now there are several online sports exchange systems available online but lack of user friendliness and difficult to understand interfaces are the major drawbacks of their websites, for example, Marcopoly [9] and Sports Team Stock Market [13] fall into this category.

Virtual Sports Stock Exchange relies on HTML V.3.2 compliant browsers to provide navigation, display, text and image rendering, and basic print capabilities. The client side browser will handle some parts of the form validation requirements for VSSX and the server will handle the rest. VSSX will run on any operating system platform containing browsers that support HTML V.3.2 and JavaScript.

The server side portion of the VSSX requires a Java web server and a database. The current VSSX is hosted on the Apache Web Server with JServ engine.

VSSX provides the following basic functionality:

- a user creates one's own account in the database.
- a user buys or trades stocks with the simulation model.
- a user interacts with one's orders such as cancel order, view order status.
- a user competes with other users online.

- navigation between various VSSX and and the other Sports website when researching players' information.
- Short-Trading as the real world stock exchange.

VSSX is primarily designed for traders that are not ready to enter the market or do not wish to trade with the real money. These users are likely to be non-technical persons but familiar with browser functionality. No special consideration is given to documenting the use of browser technology in VSSX. In addition, it is assumed that the VSSX user is familiar with browsing the Internet, clicking on the links, filling forms, and other controls common in web interfaces. No special consideration is given to documenting browser navigation.

VSSX interfaces are based on HTML V.3.2 and client side JavaScript supported by the client browser. Because of this, VSSX does not require any functionality that relies on client-side executable code outside of the browser environment.

VSSX relies on tables, and will only run on browser that is HTML V.3.2 compliant.

The VSSX server relies on Java Server Page, Java Servlet and Oracle Database. While many companies offer these functionalities, VSSX is currently developed on the Apache Web Server with JServ Engine. There is no much adjustment if one would like to move VSSX to another platform.

While increasingly unlikely, some VSSX users may connect to the server via modem, consideration has been given to page sizes, image complexity, and other factors that effect download speed.

Product Perspective

User Interface. The user interface for this project is the web front. It is used for the users to navigate the entire application. VSSX is composed of five major components: HTML pages and forms, JSP Scripts, Java Servlet, JDBC and Oracle Database. HTML pages and forms, JSP Scripts, and partial of Java Servlet creates the User Interface for VSSX.

Hardware Interfaces. The client-side browser and operating system manage hardware interface issues. There is no known hardware related hardware interface issues.

Software Interfaces. VSSX relies on HTML 3.2 compliant browsers for the client-side to provide navigation, display, text and image rendering, and basic print capability. The client-side browsers will handle all GUI requirements for

VSSX. Popular browsers like Internet Explorer, Netscape, Mozallia, and Safari will work with VSSX. On the server side, VSSX requires a Java web server that supports JSP 1.0+ and JDK 1.3+. The current iteration of VSSX is designed to run on the Apache Tomcat Web Server 5.5.

Communication Interfaces. VSSX relies on the client browser and operating system, and Java web server, JSP, Java servlet, JDBC, DBMS and operating system, to manage communication issues.

Memory and Hardware Constrains. There is no known VSSX client-side memory constraint issue but as more users sign up with VSSX, we may need more memory on the server.

Additionally, the server requires the following components for optimal content generation in a multi-user environment.

- 256 MB additional memory on top of the minimum memory requirement of the operating system (For both database and Java web server).
- Fast architecture machine, something along the lines of a PII 450 or greater.

Adaptation Requirements. This project has no known site adaptation at this time.

Product Functions

VSSX provides the following basic functionality:

- A user creates one's own account in the database.
- A user buys or trades stocks with the simulation model.
- A user interacts with one's orders such as cancel order, view order status.
- A user to compete with other users online.
- Navigation between various VSSX and and the other Sports website when researching players' information.
- Short-Trading as the real world stock exchange.

User Characteristics

VSSX is primarily designed for end users. These users are likely to have experience on browsing the Internet with mouse and keyboard. No special consideration is given to documenting the use of browser.

Constraints

VSSX relies on HTML v.3.2 compliant browser on the client-side and it does not require any client-side executable code. Therefore VSSX does not require any functionality that relies on client-side executable code.

Assumptions and Dependencies

This proposal makes the following assumptions:

- Assume that all virtual stocks are owned by one computer user who is characterized as the central bank in this model.
- Additionally, Current VSSX has the following product dependencies:
    - It is assumed that the web server supports Java with the following components installed and working properly:
    - Apache Tomcat 5.5 or other web server system
    - JServ Module for Apache 1.3.12 for .jsp redirection
    - Oracle 9i Database.
    - JDK 1.3

Specific Requirements

External Interfaces

VSSX allows users to navigate through web pages via the following browser related actions:

- Clicking on hyperlinks.
- Using the forward and back buttons on the browsers' navigation toolbar.

- Any additional operations supported by the browser for general HTML navigation.

VSSX has navigation tabs on the upper portion of each page.

Functions

VSSX shall perform all standard navigational functions supported by HTML and web browsers. All pages will be accessible via HTML hyperlinks.

Home

The main access to VSSX is through the Home Page, which will provide login field for already registered users or an URL to signup for VSSX. The Home page introduces the default rules for playing VSSX. On the upper portion of the page, there are different tabs for the navigation. Users will be allowed to navigate to their portfolios, trading stocks and view current available stocks.

My Portfolios

Once a user login to VSSX, he/she will be directed to the portfolios page which contains his/her stock portfolios. On the top of the page, it shows the total worth of the virtual stocks. Below it are the basic information needed to play VSSX such as current available cash, current ranking, stock performance and interests earned.

Order Status

There are two sub-links in the portfolios page. One is order status which contains current pending orders. The order will remain pending if there is no buyer/seller that is willing to trade with your offer.

Order History

The other sub-link in the portfolios page is order history which contains the trading record completed with the current user account.

Trading

The main buy/sell function in VSSX is located at trading page. Users can buy or sell their virtual stocks in this page. If the user does not have the virtual stock in their portfolios, they can still buy or sell that stock. This is equivalent to buy short or sell short in the real stock market. Currently there is no rule of limitation on buy short or sell short the virtual stocks but it can be applied for more realistic simulation.

Stocks

The stocks page contains listing of available stocks in VSSX. There is a "Quick Trade" link to trading page for each virtual stock. Under each stock name, it links to the players' information which are hosted at external source such as NBA.com.

<u>User Management</u>

VSSX has three main administrator pages, user management, stock management and transaction management. The user management will allow admin user to create, modify and delete users. This includes creating virtual users in the system.

<u>Stock Management</u>

The stock management page will allow admin user to create, modify and delete stocks. Administrator can temporary inactive a stock from here.

<u>Transaction Management</u>

The transaction management page will allow admin user to create, modify and delete user transactions. This will include orders that are pending as well.

<u>Virtual Users</u>

Virtual Sports Stock Exchange is driven by two forces. One is by the registered VSSX users and the other is by the virtual users. The virtual users are computer generated users that simulate the trading of virtual stocks. These virtual users are assigned with the same parameters as the regular users which include the same amount of playing money and the option to buy and sell stocks at anytime. In addition, each virtual user is assigned with a percentage number as its behavior parameter and a maximum quantity parameter. The behavior parameter determines whether a virtual user will purchase or sell the stock at the asked price. This design makes VSSX more interactive, and less prone to stock manipulation by individuals or group of individuals, especially during the initial launch when the numbers of registered VSSX users are limited.

Virtual users can be created in the Administrator Panel. The administrator can create as many virtual users as he or she wants to keep the market balanced. Each created virtual users will begin in the "sleep" state. During the sleep state, the virtual users are not allowed to trade any virtual stocks. At a random time, randomly selected virtual users will be waken by the system and the state will be changed to "active". During the active state, the virtual users will look at the transaction pool which contains all pending transactions and compare it with their portfolios to decide whether to buy or to sell. First activated virtual users will try to purchase the stocks that are available in the transaction pool but not in their portfolios. If the given price is different from the market price, virtual users will decide whether the price is within the behavior parameters. If the given price is considered "reasonable" to the virtual user, then it will decide to buy or sell. For example, if a virtual user is given a 30% behavior parameter, the current price for a particular stock is $10.00, and the given price in the transaction pool is $13.00. Then the virtual user will issue an order of $13.00 and submit it to the transaction pool. The amount of stocks that will be purchased is determined by the maximum quantity parameter. The number of quantity purchased or sold can not be more than the maximum quantity parameter. For example, if a virtual user is given 5000 shares of maximum quantity parameter, it can not submit an order of purchase or sell at more than 5000 shares.

Other than the configurable behavior parameter and the maximum quantity parameter, the numbers of virtual users trading in the system also create randomness to the system. When fewer virtual users are trading in the system, the market will be more likely to move toward regular users' preference; when more virtual users are trading in the system, the market will be more likely to move toward virtual users' preference. In a scenario of one virtual user trading in the system and five regular users registered in the system. The virtual user is given a 10% percentage of behavior parameter. The price for one particular stock is $10.00. Virtual user will acquire its initial ownership of this stock and wait for its 10% profit or loss to begin selling the stock. The virtual user's influence to the market becomes minimum especially when regular users are trading in a larger quantity than the virtual user's maximum quantity parameter. In another scenario of five virtual users trading in the system and one regular user registered in the system. The virtual users are given a range of 10% to 50% percentage of behavior parameter. The price for one particular stock is $10.00. The virtual users will acquire its initial ownership of this stock and wait for its return. During its initial acquisition, the stock price already increases and the virtual users have began trading within virtual users regardless of the regular user's action. Because of different behavior parameter, the price for this stock has been going up and down by trading within virtual users only. Therefore the numbers of virtual users trading in the system and the parameters given to the virtual users are a challenge to the administrator to keep the market balanced.

This virtual user schema is designed from my experience of stock trading. By listening to some of the brokers' advice, a conservative trader will sell a stock when the profit reaches 30% of its purchased value but not more. A conservative trader will also sell a stock when the loss reaches 30% of its purchased value but not more. This will lower the risk factor of the whole portfolios. By controlling these two parameters and limiting the virtual users to trade under these rules, the market will not go extremely high or low just by trading between virtual users themselves. There are many other factors that affects the users' decision of choosing their stocks, accumulate more stocks or sell short of stocks. There can be even more complicated artificial intelligent schema installed in this model such as whether the virtual users decide to accumulate more stocks or sell short of stocks after analyzing the market. It requires a larger knowledge base which allows each virtual user to analyze each stock and the market in general.

Performance Requirements

The performance of VSSX depends on the bandwidth of the connection, web server performance and database performance. Depends on the size of users that will be using VSSX, the speed of CPU, size of system memory, speed of the hard drive may affect the performance of VSSX. Another measurement would be to separate the database server and the web server to achieve better transaction throughput.

Logical Database Requirements

VSSX stores all data in a DBMS, such as Oracle Database. The class diagram and ER diagram of the database system for VSSX is shown as follows:

Design Constraints

There are no design constraints at this time.

Software System Attributes

Security and Reliability. JDBC support safe programming practices on a number of levels. Because they are written in Java, JDBCs inherit the strong type safety of the Java language.

Portability and Availability. Because Java is platform dependence and it is conformed to a well-defined and widely used language, they are highly portable across operating systems and across server implementations. You can use them on a Windows machine running any type of Java Web Server and later deploy it effortlessly on a high-end Unix server running different Java Web Server.

Maintainability. The majority of VSSX is implemented as script-based .jsp pages and they are easily maintained using a web development tools such as Frontpage or Dreamweaver. They can also be edited with any text editor utility. Java Servlet can also be edited with any text editor utility but they will have to be compiled with JDK 1.3 or above.

CHAPTER TWO

DESIGN

Database Design

VSSX database uses Oracle as its database management system (DBMS), which will store all the data needed for VSSX. VSSX database accepts the connection from the JDBC class, which executes SQL statements via the JDBC driver. VSSX contains four major parts: HTML forms, JSP Script Pages, Java Servlet based classes, and Oracle Database. This section concentrates on the architecture representation that leads to the VSSX Oracle database class diagram.

In order to store and process the information of VSSX, there are five tables, two PL/SQL packages and two triggers used in this project. The following E-R diagram represents the relationships among these five tables:

Table 1. The Properties of Members Table

Field Name	Field Type
MEMBER_ID	NUMBER(5)
EMAIL	VARCHAR2(50) UNIQUE
TYPE	CHAR(1)
PASSWORD	VARCHAR2(15)
FIRST_NAME	VARCHAR2(25)
LAST_NAME	VARCHAR2(25)
MAX_PERCENTAGE	NUMBER(11,2)
MAX_QUANTITY	NUMBER(11,2)
BALANCE	NUMBER(11,2)
STATUS	VARCHAR2(6)

Stocks Table

The stocks table contains stock information. It stores information of virtual stocks, which contains symbol, virtual stock name, last, high, low, close, shares, stock issue date, sports exchange, url to player information, and status. The primary key is SYMBOL.

Table 2. The Properties of Stocks Table

Field Name	Field Type
SYMBOL	VARCHAR2(5)
NAME	VARCHAR2(50)
LAST	NUMBER(6,2)
HIGH	NUMBER(6,2)
LOW	NUMBER(6,2)
CLOSE	NUMBER(6,2)
SHARES	NUMBER(9)
ISSUE_DATE	DATE
EXCHANGE	VARCHAR2(5)
URL	VARCHAR2(500)
STATUS	VARCHAR2(6)

Transaction Table

The transaction table contains all users' virtual stock transaction history. Each transaction contains transaction id, member id, stocks symbol, buy or sell, price of purchase or sell, quantity of buy or sell, total amount, completed transaction date, and status. The primary key is TRANS_ID. The foreign keys are MEMBER_ID of members table and SYMBOL of stock table.

Table 3. The Properties of Transaction Table

Field Name	Field Type
TRANS_ID	NUMBER(5)
MEMBER_ID	NUMBER(5)
SYMBOL	VARCHAR2(5)
ACTION	VARCHAR2(6)
PRICE	NUMBER(6,2)
QUANTITY	NUMBER(9)
TOTAL	NUMBER(11,2)
TRANS_DATE	DATE
STATUS	VARCHAR2(8)

Pending Order Table

The pending order table contains all users' virtual stock pending orders. Each record contains transaction id, member id, virtual stocks symbol, buy or sell, price of purchase or sell, quantity of buy or sell, total amount, and order date. This table is used as a temporary storage for all users' order placed. When a user places an order, either buy or sell, the information will be store in this table. When this transaction is completed, the data from this table will be moved to the transaction table. The foreign keys are TRANS_ID of transaction table, MEMBER_ID of members table, SYMBOl of stock table.

Table 4. The Properties of Pending Order Table

Field Name	Field Type
TRANS_ID	NUMBER(5)
ORDER_DATE	DATE
MEMBER_ID	NUMBER(5)
SYMBOL	VARCHAR2(5)
ACTION	VARCHAR2(6)
PRICE	NUMBER(6,2)
QUANTITY	NUMBER(9)

Share Holders Table

The share holders table contains the ownership information of the virtual stocks. Each record contains member_id, symbol, price, quantity, and total amount fields. When an user buys a virtual stock, he or she will be the owner of that stock. When a user purchases the same stock more than once, the record for the same stock will be combined into one. The primary key is MEMBER_ID and SYMBOL of members table and stock table.

Table 5. The Properties of Share Holders Table

Field Name	Field Type
MEMBER ID	NUMBER(5)
SYMBOL	VARCHAR2(5)
PRICE	NUMBER(6,2)
QUANTITY	NUMBER(9)
TOTAL	NUMBER(11,2)

Software Architecture

The user interface of VSSX is developed using Java programming language and the connection bridge is developed by using JDBC class, which is a member of Java programming language family. The basic network diagram for the VSSX is shown as following:

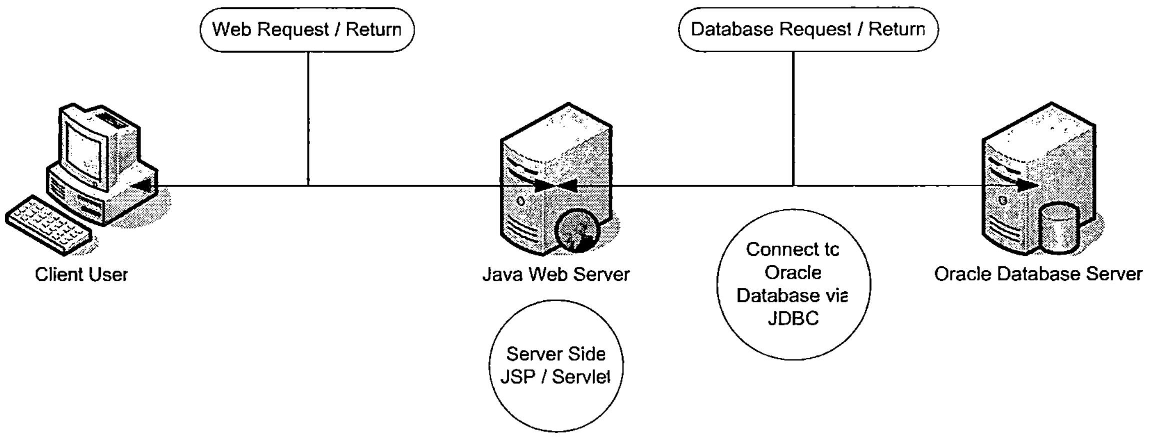

Web Request / Return
Database Request / Return
Client User
Java Web Server
Connect to Oracle Database via JDBC
Oracle Database Server
Server Side JSP / Servlet

Figure 2. Structure Network Diagram

As shown in the diagram above, VSSX relies on JDBC to communicate with the database server. This is also called a

3-tier system which is widely used in the business world nowadays. When the numbers of client users grow, database clustering may be implemented to avoid bottleneck on retrieving data from the database and load balancing on multiple web servers can also be placed as consideration.

The JDBC driver that is used comes with the Oracle Database Server. We are using Oracle JDBC type 4 thin driver to access the Oracle database. The alternative way to connect to the Oracle Database is to use Oracle OCI Driver.

Connection pool is used with Oracle JDBC thin driver. Every request for the database must use the connection pool. In other words, each request for database does not establish its own connection. This way it can reduce the number of connections established to the database and make the system more efficient. The max number of connections can be opened at the same time is ten as default. Once the eleventh request comes in, it has to wait until one of those three connections has handed back to the connection pool.

Each JSP or servlet will call the JDBC class, manipulate the received data and represented to the web client. This method has been widely used in the web application architecture.

Detailed Design

This section will concentrate with VSSX's main pages. At the end of the each sub-session, it describes the functionality for each page. The following diagram shows the main pages for VSSX:

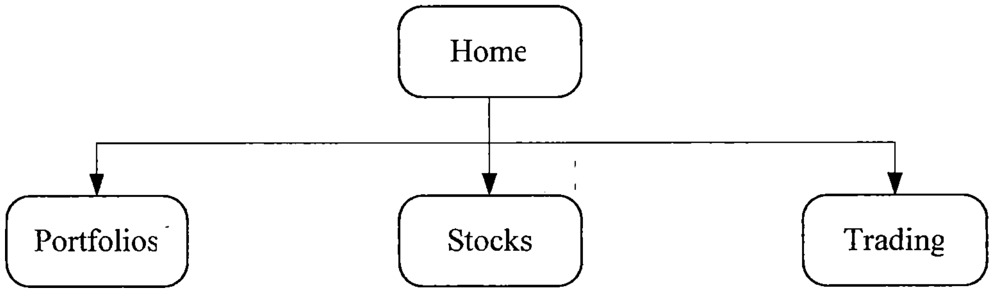
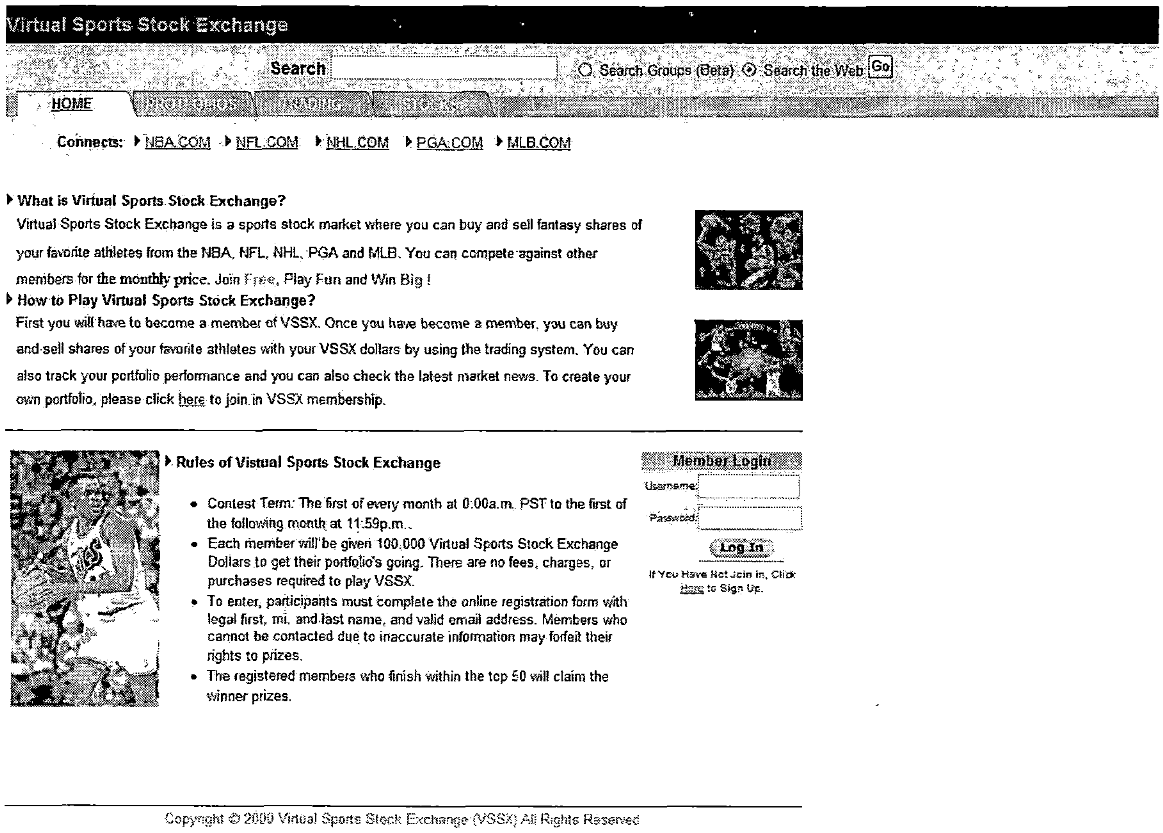


Figure 3. Home Navigation Diagram

Home

The home page contains the navigation bars to go to Portfolios, stocks, and trading pages. It also contains the links to the popular sports website such as nba.com, nfl.com, nhl.com, pga.com, and mlb.com. On the home page, it briefly describes how the simulator works and the rules for VSSX. The following screenshot is the home page for VSSX:



What is Virtual Sports Stock Exchange?

Virtual Sports Stock Exchange is a sports stock market where you can buy and sell fantasy shares of your favorite athletes from the NBA, NFL, NHL, PGA and MLB. You can compete against other members for the monthly price. Join Free, Play Fun and Win Big !

How to Play Virtual Sports Stock Exchange?

First you will have to become a member of VSSX. Once you have become a member, you can buy and sell shares of your favorite athletes with your VSSX dollars by using the trading system. You can also track your portfolio performance and you can also check the latest market news. To create your own portfolio, please click here to join in VSSX membership.

Rules of Vistual Sports Stock Exchange

- Contest Term: The first of every month at 0:00a.m. PST to the first of the following month at 11:59p.m..
- Each member will be given 100,000 Virtual Sports Stock Exchange Dollars to get their portfolio's going. There are no fees, charges, or purchases required to play VSSX.
- To enter, participants must complete the online registration form with legal first, mi. and last name, and valid email address. Members who cannot be contacted due to inaccurate information may forfeit their rights to prizes.
- The registered members who finish within the top 50 will claim the winner prizes.

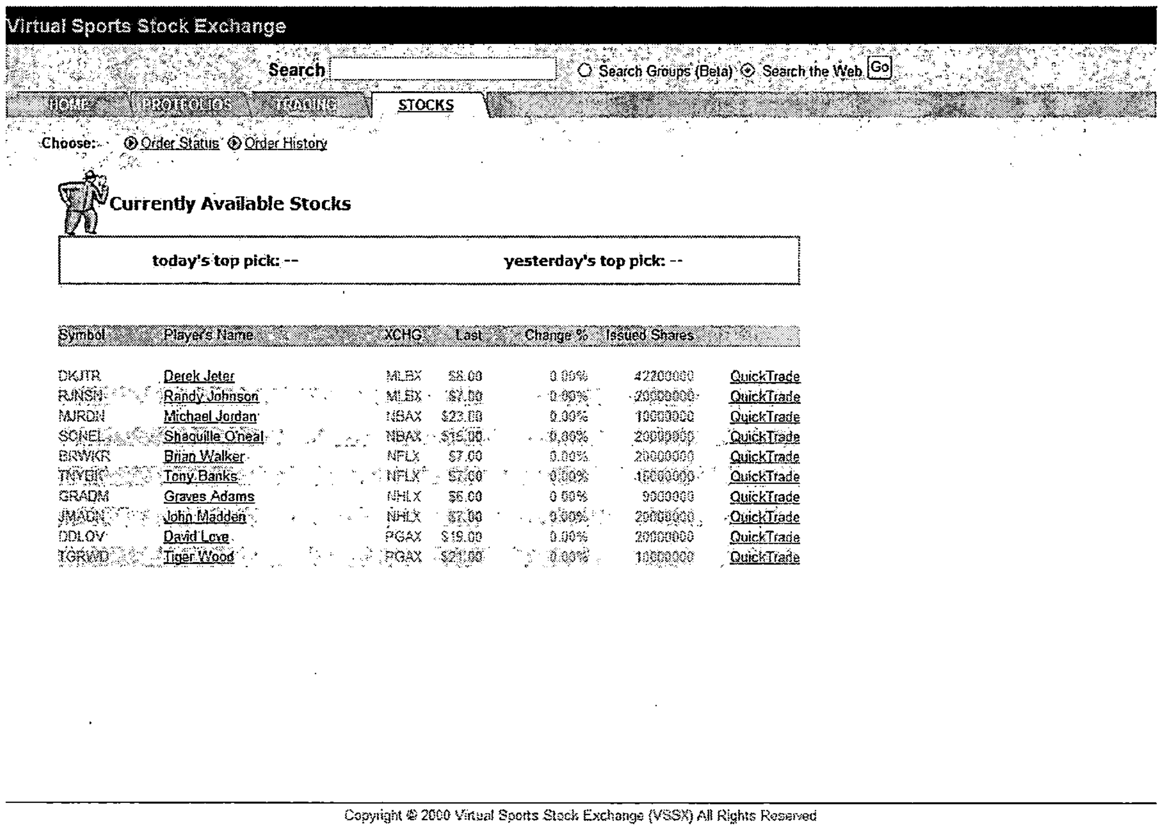
If You Have Not Join in, Click Here to Sign Up.

Copyright © 2000 Virtual Sports Stock Exchange (VSSX) All Rights Reserved

Figure 4. Screen Shot of Home Page

Portfolios

The portfolios page is the core page for VSSX. It contains information of the virtual stocks and the virtual money that users have. This page is divided into different type of sports such as MLB, NBA, NFL, NHL, and PGA. The reason is that different sports have different scoring methods. The following is the screenshot for the portfolios page:



Currently Available Stocks

today's top pick: -- yesterday's top pick: --

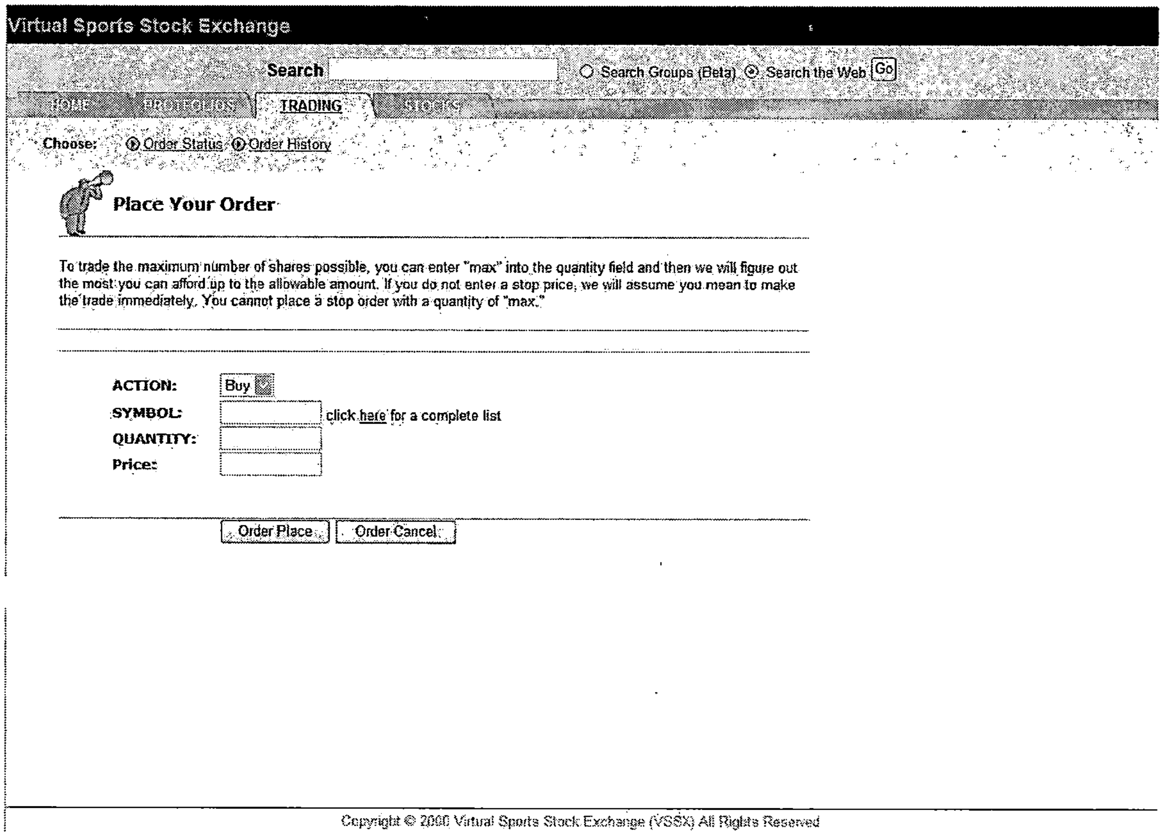
Symbol	Player's Name	XCHG	Last	Change %	Issued Shares	
DKJTR	Derek Jeter	MLBX	$8.00	0.00%	42200000	QuickTrade
RJNSN	Randy Johnson	MLBX	$7.00	0.00%	20000000	QuickTrade
MJRDN	Michael Jordan	NBAX	$23.00	0.00%	10000000	QuickTrade
SONEL	Shaquille O'neal	NBAX	$15.00	0.00%	20000000	QuickTrade
BRWKR	Brian Walker	NFLX	$7.00	0.00%	20000000	QuickTrade
TNYBK	Tony Banks	NFLX	$7.00	0.00%	10000000	QuickTrade
GRADM	Graves Adams	NHLX	$6.00	0.00%	9000000	QuickTrade
JMADN	John Madden	NHLX	$7.00	0.00%	20000000	QuickTrade
DDLOV	David Love	PGAX	$19.00	0.00%	20000000	QuickTrade
TGRWD	Tiger Wood	PGAX	$21.00	0.00%	10000000	QuickTrade

Copyright © 2000 Virtual Sports Stock Exchange (VSSX) All Rights Reserved

Figure 6. Screen Shot of Stocks Listing Page

Trading

Trading page is for used for users to place orders. Users can buy, sell, or short the virtual stocks. At this time, the system allows users to short the stocks but it can be taken out to avoid manipulation in the market. The trading page also links to order status, and order history for the user. The following screen shot is the trading page:



To trade the maximum number of shares possible, you can enter "max" into the quantity field and then we will figure out the most you can afford up to the allowable amount. If you do not enter a stop price, we will assume you mean to make the trade immediately. You cannot place a stop order with a quantity of "max."

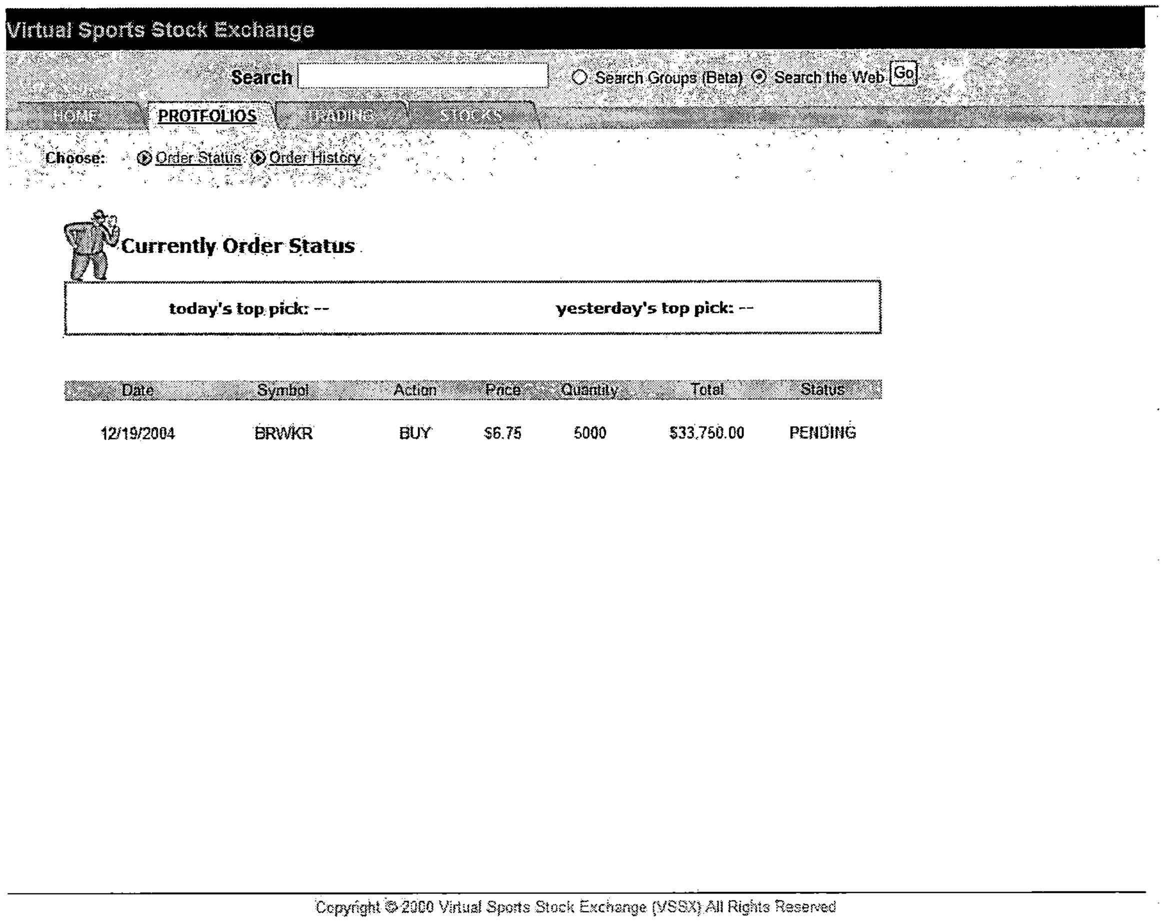
ACTION:	Buy
SYMBOL:	click here for a complete list
QUANTITY:	
Price:	

Copyright © 2000 Virtual Sports Stock Exchange (VSSX) All Rights Reserved

Figure 7. Screen Shot of Trading Page

<u>Order Status</u>

The order status page shows an placed order's status. If a user placed an order but the transaction is not completed, the status will show "PENDING" just as shown in the screenshot below. In the other case, it will show "COMPLETED". The following is the screenshot for the Order Status page:



Currently Order Status

today's top pick: -- yesterday's top pick: --

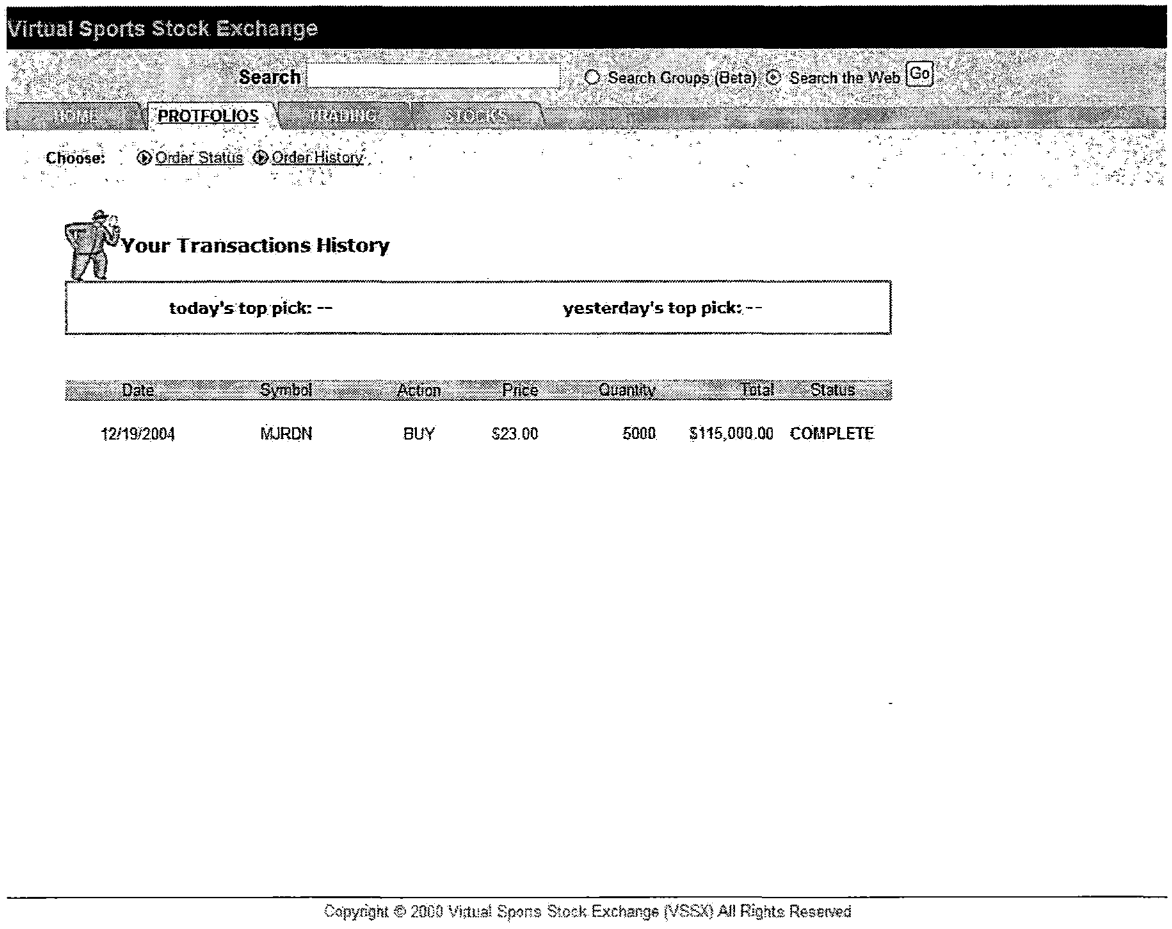
Date	Symbol	Action	Price	Quantity	Total	Status
12/19/2004	BRWKR	BUY	$6.75	5000	$33,750.00	PENDING

Copyright © 2000 Virtual Sports Stock Exchange (VSSX) All Rights Reserved

Figure 8. Screen Shot of Order Status Page

Order History

The order history page shows all completed transaction ever made by the user. The following is the screenshot of Order History page:



Your Transactions History

today's top pick: --   yesterday's top pick: --

Date	Symbol	Action	Price	Quantity	Total	Status
12/19/2004	MJRDN	BUY	$23.00	5000	$115,000.00	COMPLETE

Copyright © 2000 Virtual Sports Stock Exchange (VSSX) All Rights Reserved

Figure 9. Screen Shot of Transactions History Page

Admin Pages

The following diagram shows the main navigation flow of the admin pages for VSSX:

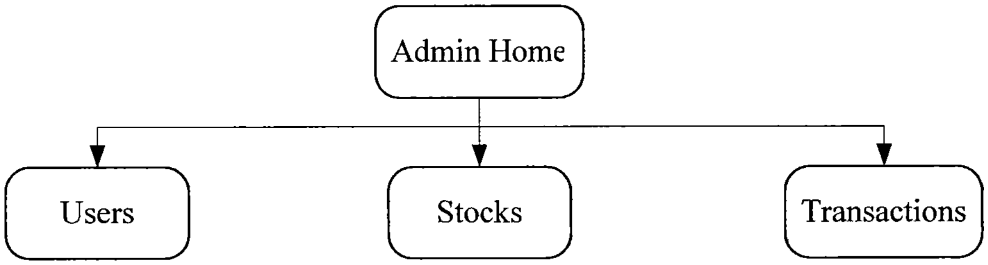
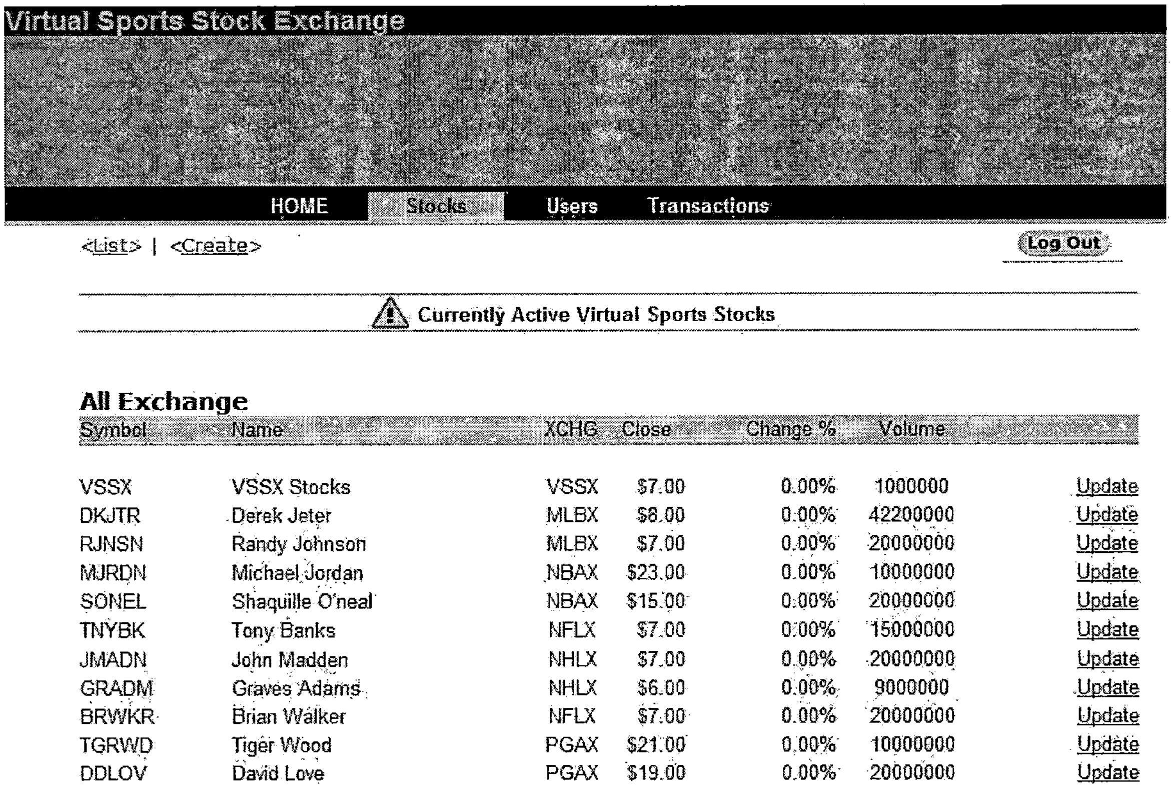


Figure 10. Admin Navigation Diagram

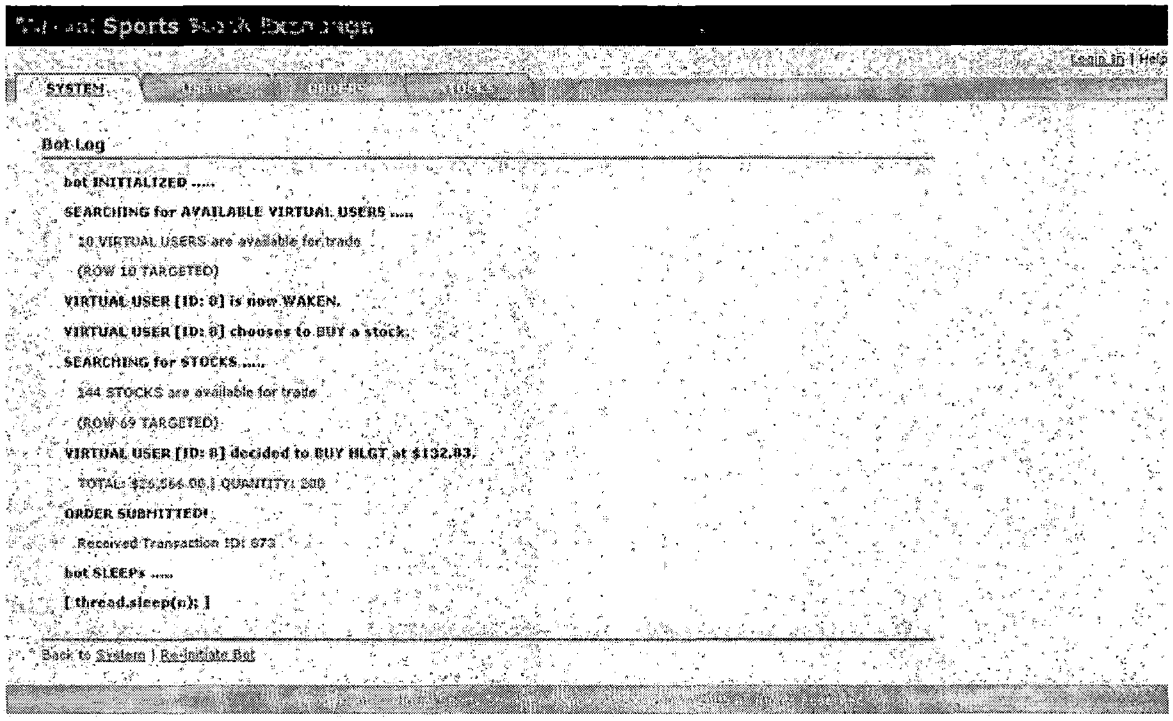
The admin page provides "Add", "Delete", and "Modify" for the following subpages: USERS, STOCKS, and TRANSACTIONS. The following screenshots is the admin page for stocks:


Virtual Sports Stock Exchange
HOME
Stocks
Users
Transactions
<List> | <Create>
Log Out
Currently Active Virtual Sports Stocks

All Exchange

Symbol	Name	XCHG	Close	Change %	Volume	
VSSX	VSSX Stocks	VSSX	$7.00	0.00%	1000000	Update
DKJTR	Derek Jeter	MLBX	$8.00	0.00%	42200000	Update
RJNSN	Randy Johnson	MLBX	$7.00	0.00%	20000000	Update
MJRDN	Michael Jordan	NBAX	$23.00	0.00%	10000000	Update
SONEL	Shaquille O'neal	NBAX	$15.00	0.00%	20000000	Update
TNYBK	Tony Banks	NFLX	$7.00	0.00%	15000000	Update
JMADN	John Madden	NHLX	$7.00	0.00%	20000000	Update
GRADM	Graves Adams	NHLX	$6.00	0.00%	9000000	Update
BRWKR	Brian Walker	NFLX	$7.00	0.00%	20000000	Update
TGRWD	Tiger Wood	PGAX	$21.00	0.00%	10000000	Update
DDLOV	David Love	PGAX	$19.00	0.00%	20000000	Update

Figure 11. Screen Shot of Admin Stocks Page



Bot Log

```
bot INITIALIZED .....
SEARCHING for AVAILABLE VIRTUAL USERS .....
  10 VIRTUAL USERS are available for trade
  (ROW 10 TARGETED)
VIRTUAL USER [ID: 8] is now WAKEN.
VIRTUAL USER [ID: 8] chooses to BUY a stock.
SEARCHING for STOCKS .....
  144 STOCKS are available for trade
  (ROW 69 TARGETED)
VIRTUAL USER [ID: 8] decided to BUY HLGT at $132.83.
  TOTAL: $26,566.00 | QUANTITY: 200
ORDER SUBMITTED!
  Received Transaction ID: 673
bot SLEEPs .....
[ thread.sleep(n); ]
```

Figure 12. Screen Shot of Virtual Users Log

CHAPTER THREE

SOFTWARE QUALITY ASSURANCE

Create Virtual Stocks

The test is done by creating virtual stocks in the administrator panel.

Table 6. Test Result of Creating Virtual Stocks

Test Case	Actual Result	Expected Result
Create DKJTR Virtual Stocks	Return no error	Return no error
Create RJNSN Virtual Stocks	Return no error	Return no error
Create MJRDN Virtual Stocks	Return no error	Return no error
Create SONEL Virtual Stocks	Return no error	Return no error
Create BRWKR Virtual Stocks	Return no error	Return no error
Create TNYBK Virtual Stocks	Return no error	Return no error
Create GRADM Virtual Stocks	Return no error	Return no error
Create JMADN Virtual Stocks	Return no error	Return no error
Create TGRWD Virtual Stocks	Return no error	Return no error

Register Users

The test is done by using the user signup page to create a user profile.

Table 7. Test Result of Register User

Test Case	Actual Result	Expected Result
Regsiter User cchen	Return no error	Return no error

Browse Stock Listings

The test is done by verifying with the database of the numbers of entries that has been input.

Table 8. Test Result of Stock Listings

Test Case	Actual Result	Expected Result
Verify Stocks Listings with SQL Statement: "SELECT COUNT(*) FROM STOCKS"	Return 10	Return 10

Buying a Stock

The test is done by triggering the buy sequence of virtual stocks and verifying with the database result.

Table 9. Test Result of Buying a Stock

Test Case	Actual Result	Expected Result
Buy MJRDN 5000 Shares. Verify with SQL Statement: "SELECT * FROM PENDING_ORDERS WHERE USER_ID = '2' AND SYMBOL = 'MJRDN'"	Return 12/19/2004 MJRDN BUY 23.00 5000 115000	Return 12/19/2004 MJRDN BUY 23.00 5000 115000

Verify Scheduled Virtual User Tasks

The test is done by verifying when a transaction is completed.

Table 10. Test Result of Verifying Completed Transaction

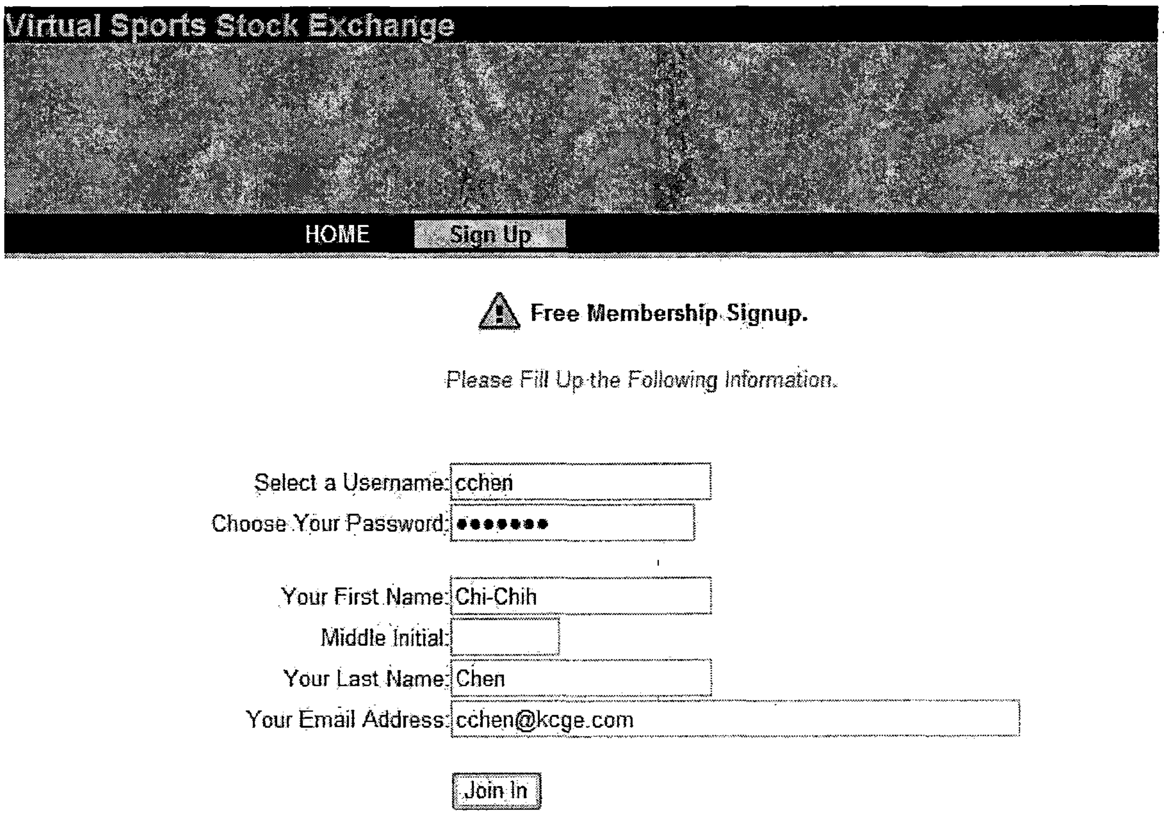
Test Case	Actual Result	Expected Result
Verify with SQL Statement: "SELECT STATUS FROM TRANSACTION WHERE USER_ID = '2' AND SYMBOL = 'MJRDN'"	Return COMPLETED	Return COMPLETED

CHAPTER FOUR

EXAMPLE SESSIONS

This project uses JSP pages to build the user interface and use the JDBC classes to access the Oracle Database. This following example will show how the test result was performed above.

On the home page, there is a "click here to sign up" link. This is the initial step for the registration of new users. Users will be required to enter their username, password, names and email address in order to complete the registration. By clicking on the link, it will take you to the user registeration page shown below:

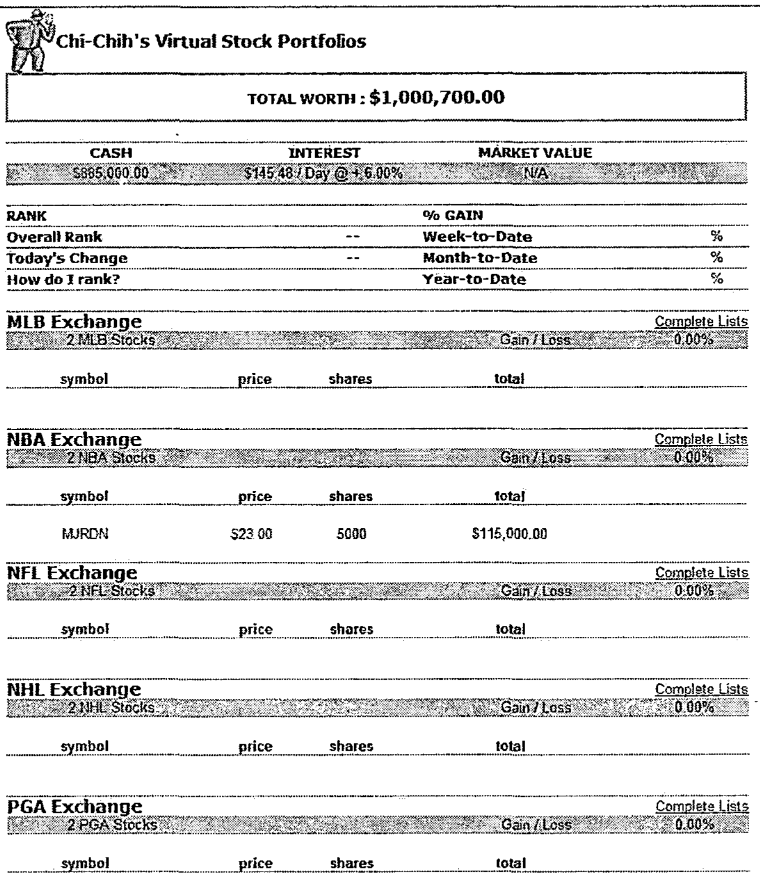

Virtual Sports Stock Exchange
HOME
Sign Up
Free Membership Signup.
Please Fill Up the Following Information.
Join In

Select a Username:	cchen
Choose Your Password:	●●●●●●●
Your First Name:	Chi-Chih
Middle Initial:	
Your Last Name:	Chen
Your Email Address:	cchen@kcge.com

Figure 13. Screen Shot of User SignUp Page

After fill in the information and click on "Join in", we will be redirected back to the homepage where we can now sign in.

After signing in, VSSX redirects us to the Portfolios page shown below. This pages shows that the stock that was purchased during the test session is listed under NBA Exchange.



Chi-Chih's Virtual Stock Portfolios

TOTAL WORTH : $1,000,700.00

CASH	INTEREST	MARKET VALUE
$885,000.00	$145.48 / Day @ + 6.00%	N/A

RANK		% GAIN	
Overall Rank	--	Week-to-Date	%
Today's Change	--	Month-to-Date	%
How do I rank?		Year-to-Date	%

MLB Exchange — Complete Lists

2 MLB Stocks — Gain / Loss 0.00%

symbol	price	shares	total

NBA Exchange — Complete Lists

2 NBA Stocks — Gain / Loss 0.00%

symbol	price	shares	total
MJRDN	$23.00	5000	$115,000.00

NFL Exchange — Complete Lists

2 NFL Stocks — Gain / Loss 0.00%

symbol	price	shares	total

NHL Exchange — Complete Lists

2 NHL Stocks — Gain / Loss 0.00%

symbol	price	shares	total

PGA Exchange — Complete Lists

2 PGA Stocks — Gain / Loss 0.00%

symbol	price	shares	total

Figure 14. Screen Shot of User Portfolios Page

CHAPTER FIVE

MAINTENANCE MANUAL

Source Code

In this Virtual Sports Stock Exchange project, there are six HTML pages, forteen JSP pages, six Java servlet and six SQL files. All the source files are stored in the attached CD. The /WEBAPPS directory contains all JSP pages and JAVA classes. The sql statement files are stored in /SQL directory. All JAVA files are put in directory /JAVA. The directory /DOCUMENTS stores all word files.

Here is the list of the source files.

- Scripts/index.jsp
- Scripts/INC_top.jsp
- Scripts/INC_bottom.jsp
- Scripts/INC_styles.css
- Scripts/login.jsp
- Scripts/logout.jsp
- Scripts/Portfolios.jsp
- Scripts/signup.jsp
- Scripts/stocks.jsp
- Scripts/order.jsp
- Scripts/confirm.jsp

- Scripts/history.jsp
- Admin/index.jsp
- Admin/bot.jsp
- Admin/INC_top.jsp
- Admin/INC_bottom.jsp
- Admin/INC_styles.css
- Admin/login.jsp
- Admin/logout.jsp
- Admin/users.jsp
- Admin/usrUpd.jsp
- Admin/usrCtrl.jsp
- Admin/stocks.jsp
- Admin/stkUpd.jsp
- Admin/stkCtrl.jsp
- dbConnection.Oracle.class (.java)
- member.login.class (.java)
- member.signup.class (.java)
- member.update.class (.java)
- stock.ipo.class (.java)
- stock.trade.class (.java)
- stock.update.class (.java)
- cleanup.sql
- tables.sql

- sequences.sql
- triggers.sql
- packages.sql
- data.sql

Re-Compile

This session shows how to recompile Java Servlet when there are changes to the .java source files.

First, specify the classpath where the command "javac" is found. Overrides the default path or the "CLASSPATH" environment variable if it is set. If there is more than one directory, using semicolon to separate the directory names. It is often useful for the directory containing the source files to be on the class path. Moverover, users should always include the system classes and oracle JDBC driver file at the end of the path. For example:

```
javac -classpath .;C:\jdk_1.4.2\classes;
C:\oracle\jdbc\lib
```

Then change the directory to the one that holds the class or classes needed to re-compile. After changing the directory path, key in the following command:

javac *filename*.java -include classes12.zip

Installation Process

There four major steps in the installation process. The first step is to store all the HTML and JSP files under the root directory of the Java web server from the same directory structure listed under /WEB in the CD. Second, creating an oracle database instance in the database with username vssx and password vssx. Then start the sql script. Connect to Oracle by using sql/plus console, run the following SQL files in orders: tables.sql, sequences.sql, triggers.sql, packages.sql, and data.sql. Placed the compiled .class Java Servlet files in WEB-INF directory (WEB-INF directory is used in Apache Tomcat Java Web Server). The finally step is to copy Oracle's JDBC thin driver to WEB-INF/lib under the directory had defineded in the class path of the web server.

User Manual

The VSSX uses Java Server Pages and Java Servlet to interacte with the database process. Buying and selling virtual stocks, and virtual users tractions are done in the database PL/SQL triggers and packages.

Before Using Virtual Sports Stock Exchange

Users have to be registered with VSSX in order to continue. By using "Click Here to Signup", an user can signup with the VSSX for immediately access.

Overall Descriptioin

Users can navigate VSSX through the upper portion of the page which has the navigation bar. Below the navigation bar is the actual contents display.

Using Virtual Sports Stock Exchange

A registered user should start with their portfolios page. In the portfolios page, there are stocks available to be purchased. To view the available stocks, simply click on "Complete Lists" at each exchange market. If a user has decided which stock to buy, he/she can click on Quick Trade to start the purchasing process. After the process is completed, the purchased stock will be listed in the user's portfolios. The same process can be applied to sell the stocks.

CHAPTER SIX

CONCLUSION AND FUTURE DIRECTIONS

Conclusion

The purpose of VSSX project is to simulate the online broker services with the supply and demand concept in the business world by using popular sports players as the virtual stocks.

It is a complete system that allows students to register and start trading with virtual stocks. It is educational in that users can learn the principle of stock trading, just like the real stock market.

VSSX is composed with 5 major components: HTML forms, Java Server Pages, the Java Servlet classes, the Oracle Type-4 thin driver, and Oracle database. 3 major languages are used in the VSSX project: HTML, Java programming language, and PL/SQL. Without these components, we will not be able to simulate a 3-tier system.

Before I started with this project, some of these components were new to me. For HTML forms, I have learned some advanced tags along with Java Server Pages scriptlets.

For Java Server Pages and Java Servlet, I have learned how to administrate and configure Apache Tomcat web server. Also to access the database via the Oracle JDBC thin driver.

For JDBC, I have learned the single pass recordset and multi-pass recordset which allows the record to go from the beginning to the end and as well as from the end to the beginning. The main difference is to declare the flag and to keep track of the bookmarks.

For Oracle Database, it is a good experience to put what I have learned from the database classes into work. Tasks like database design, triggers and packages programming, and database performance tuning.

I have also learned the basis of economic in the real world and the stock trading system. By implementing simple artificial intelligent virtual users, I have learned some of the factors that make the market changes, how users choose their stocks and when users decide to buy or sell a stock.

Overall, this project involves using many powerful tools available nowadays such as Java Programming Language, and Oracle SQL/PLSQL Database. I have to learn how to plan, design, coding, testing, and administrating in order to fullfill the requirements.

Future Directions

There are several improvements that can be done to the VSSX. The virtual stocks information is limited. By connecting to the well established players database, VSSX can provide more complicated information on the virtual stocks. It can use this enhancement to create live information on trading players or live game information. More complicated knowledge base for virtual users can be implemented; such knowledge base will allow virtual users to behavior differently when it comes to choosing and trading stocks. By adding different factors to each behavior, Virtual Sports Stock Exchange will become more and more interesting.

APPENDIX A

ORACLE TRIGGERS AND PACKAGES

```
-- 1 PACKAGE Share

CREATE OR REPLACE PACKAGE Shares IS


PROCEDURE Verify (

     pTrans_ID IN NUMBER

 );

END Shares;

/


CREATE OR REPLACE PACKAGE BODY Shares IS


PROCEDURE Verify (

     pTrans_ID IN NUMBER

 ) IS


CURSOR cTransaction IS

  SELECT *

  FROM Transactions

  WHERE Trans_ID = pTrans_ID

  FOR UPDATE;


  vTransaction Transactions%ROWTYPE;
```

```
    CURSOR cShare_Holders IS
      SELECT 1
      FROM Share_Holders
      WHERE Member_ID = vTransaction.Member_ID
      AND   Symbol    = vTransaction.Symbol;

      vExist NUMBER;

     BEGIN
       OPEN cTransaction;
         FETCH cTransaction INTO vTransaction;
         OPEN cShare_Holders;
         FETCH cShare_Holders INTO vExist;

         IF cShare_Holders%FOUND THEN
           IF vTransaction.Action = 'BUY' THEN

             DBMS_OUTPUT.PUT_LINE( 'BUY' );
             DBMS_OUTPUT.PUT_LINE( 'vTransaction.Trans_ID'
|| vTransaction.Trans_ID );

             UPDATE Share_Holders
```

```
              SET Price = ( ( Price * ABS ( Quantity ) ) +
( vTransaction.Price * ABS ( vTransaction.Quantity ) ) ) /
( ABS( Quantity ) + ABS( vTransaction.Quantity ) )
                 ,Quantity = Quantity +
vTransaction.Quantity
                 ,Total = ( ( Price * Quantity ) +
( vTransaction.Price * vTransaction.Quantity ) ) /
( ABS( Quantity ) + ABS( vTransaction.Quantity ) ) *
( Quantity + vTransaction.Quantity )
              WHERE Member_ID = vTransaction.Member_ID
                AND Symbol = vTransaction.Symbol;

            ELSE
              DBMS_OUTPUT.PUT_LINE( 'SELL' );

              UPDATE Share_Holders
              SET Quantity = Quantity -
vTransaction.Quantity
                 ,Total = Price * ( Quantity -
vTransaction.Quantity )
              WHERE Member_ID = vTransaction.Member_ID
                AND Symbol = vTransaction.Symbol;
            END IF;

          ELSE

            DBMS_OUTPUT.PUT_LINE( 'NotFound' );

            INSERT INTO SHARE_HOLDERS VALUES

( vTransaction.Member_ID, vTransaction.Symbol,

vTransaction.Price, DECODE( vTransaction.action, 'BUY', 1, -

1 ) * vTransaction.Quantity, vTransaction.Price *

vTransaction.Quantity);

          END IF;


          -- Update balance:

          UPDATE Members

          SET Balance = Balance +

( DECODE( vTransaction.Action, 'BUY', -1, 'SELL', 1, 0 ) *

                        ( vTransaction.Price *

vTransaction.Quantity) )

          WHERE Member_ID = vTransaction.Member_ID;


          -- Update status of transactions:

          UPDATE Transactions

          SET Status = 'COMPLETE'

          WHERE CURRENT OF cTransaction;
```

```
        -- Update last of stocks:
        UPDATE Stocks
        SET Last = vTransaction.Price
        WHERE SYMBOL = vTransaction.Symbol;

       CLOSE cShare_Holders;
     CLOSE cTransaction;

EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE( SQLERRM );
END Verify;
END Shares;
/

-- 2 PACKAGE Trading
CREATE OR REPLACE PACKAGE Trading IS

PROCEDURE Verify (
 pTrans_ID IN Number,
 pMember_ID IN NUMBER,
 pSymbol IN VARCHAR2,
 pAction IN VARCHAR2,
 pPrice IN NUMBER,
 pQuantity IN NUMBER
 );
END Trading;
/

CREATE OR REPLACE PACKAGE BODY Trading IS

PROCEDURE Verify (
 pTrans_ID IN Number,
 pMember_ID IN NUMBER,
 pSymbol IN VARCHAR2,
 pAction IN VARCHAR2,
 pPrice IN NUMBER,
 pQuantity IN NUMBER
 ) IS
CURSOR cPending_Orders IS
  SELECT Trans_ID,
 Member_ID,
 Symbol,
 Action,
 Price,
 Quantity
 FROM Pending_Orders
 WHERE Action != pAction
 AND Price = pPrice
 AND Symbol = pSymbol
 ORDER BY Trans_ID;

vPending_Orders cPending_Orders%ROWTYPE;
vQuantity Number := pQuantity;

BEGIN
  OPEN cPending_Orders;

    LOOP

      FETCH cPending_Orders INTO vPending_Orders;

      IF cPending_Orders%FOUND THEN
        IF vQuantity > vPending_Orders.Quantity THEN

          DBMS_OUTPUT.PUT_LINE( '>' );
          DELETE Pending_Orders
          WHERE Trans_ID = vPending_Orders.Trans_ID;
```

```
                DBMS_OUTPUT.PUT_LINE( 'vQuantity : ' ||
vQuantity );

DBMS_OUTPUT.PUT_LINE( 'vPending_Orders.Quantity : ' ||
vPending_Orders.Quantity );

                UPDATE Pending_Orders
                SET Quantity = vQuantity -
vPending_Orders.Quantity
                WHERE Trans_ID = pTrans_ID;

                Shares.Verify(vPending_Orders.Trans_ID );

DBMS_OUTPUT.PUT_LINE( 'vPending_Orders.Trans_ID : ' ||
vPending_Orders.Trans_ID );

              ELSIF vQuantity = vPending_Orders.Quantity
THEN

                DBMS_OUTPUT.PUT_LINE( '=' );

                DELETE Pending_Orders
                WHERE Trans_ID = vPending_Orders.Trans_ID;
                Shares.Verify(vPending_Orders.Trans_ID );

DBMS_OUTPUT.PUT_LINE( 'vPending_Orders.Trans_ID : ' ||
vPending_Orders.Trans_ID );

                DELETE Pending_Orders
                WHERE Trans_ID = pTrans_ID;

                Shares.Verify(pTrans_ID );
                DBMS_OUTPUT.PUT_LINE( 'pTrans_ID : ' ||
pTrans_ID );

              ELSE

                DBMS_OUTPUT.PUT_LINE( '<' );

                UPDATE Pending_Orders
                SET Quantity = Quantity - vQuantity
                WHERE Trans_ID = vPending_Orders.Trans_ID;

                DELETE Pending_Orders
                WHERE Trans_ID = pTrans_ID;
```

```
                  Shares.Verify(pTrans_ID );

                  DBMS_OUTPUT.PUT_LINE( 'pTrans_ID : ' ||
pTrans_ID );

                END IF;

                vQuantity := vQuantity -
vPending_Orders.Quantity;

              END IF;

            EXIT WHEN cPending_Orders%NOTFOUND;
          END LOOP;
          DBMS_OUTPUT.PUT_LINE( 'cPending_Orders%ROWCOUNT : '
|| cPending_Orders%ROWCOUNT );
          CLOSE cPending_Orders;

       EXCEPTION
           WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE( SQLERRM );
       END Verify;
       END Trading;
```

```
/

-- 1 Insert Pending_Orders Trigger

CREATE OR REPLACE TRIGGER Order_Pending AFTER INSERT ON
Transactions
REFERENCES OLD AS OLD NEW AS NEW FOR EACH ROW
DECLARE
    vExist NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO vExist
    FROM Pending_Orders
    WHERE Trans_ID = :new.Trans_ID;

    IF vExist = 0 THEN
      INSERT INTO Pending_Orders VALUES (
        :new.Trans_ID,
        :new.Member_ID,
        :new.Symbol,
        :new.Action,
        :new.Price,
        :new.Quantity
        );
      ELSE
        UPDATE Pending_Orders
        SET Quantity = Quantity + :new.Quantity
        WHERE Trans_ID = :new.Trans_ID;
      END IF;

    END;
    /

    -- 2 Update High-Low Price Trigger

    CREATE OR REPLACE TRIGGER Price_History BEFORE UPDATE
ON Stocks
    REFERENCES OLD AS OLD NEW AS NEW FOR EACH ROW
    BEGIN

          IF (:new.Last > :old.High) THEN
            :new.High := :new.Last;

          ELSIF (:new.Last < :old.Low) THEN
            :new.Low := :new.Last;
        END IF;
    END;
    /

    -- 3 Create Default VSSX Shares for New Members

    CREATE OR REPLACE TRIGGER Default_Shares AFTER INSERT
ON Members
    REFERENCES OLD AS OLD NEW AS NEW FOR EACH ROW
    BEGIN

        INSERT INTO SHARE_HOLDERS VALUES (
        :new.member_id, 'VSSX', 7, 100, 700);

    END;
    /

    -- 4 Let All Shares Owned by Administrator

    CREATE OR REPLACE TRIGGER Default_holders AFTER INSERT
ON STOCKS
    REFERENCES OLD AS OLD NEW AS NEW FOR EACH ROW
    BEGIN
```

```
          INSERT INTO SHARE_HOLDERS VALUES (
            1, :new.symbol, :new.last, :new.shares, :new.last
* :new.shares
          );
      END;
      /
```

APPENDIX B

JAVA SERVLET SOURCE CODE

```
package member;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class login extends HttpServlet {
  /**Initialize global variables*/
  public void init(ServletConfig config) throws
ServletException {
    super.init(config);
      try{
       Class.forName("oracle.jdbc.driver.OracleDriver");
    }
    catch (ClassNotFoundException e) {
       System.err.println(e.getMessage());
    }
  }

  //Process the HTTP Post request
  public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    try {
      String USERNAME = request.getParameter("username");
      String PASSWORD = request.getParameter("password");

      // Generate Session
      HttpSession session = request.getSession(true);

      // Create Connection Object
      Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@bishop:1521:o
racle","vssx","vssx");
      Statement stmt = conn.createStatement();
      ResultSet rset = stmt.executeQuery("SELECT MEMBER_ID,
PASSWORD, FIRST_NAME, STATUS FROM MEMBERS WHERE
UPPER(USERNAME) = '" + USERNAME.toUpperCase() + "'");

      if(rset.next()) {
        if (PASSWORD.equals(rset.getString("PASSWORD")) &&
(rset.getString("STATUS").equals("UNLOCK"))) {
          session.putValue("access","granted");
          session.putValue("member_id",
rset.getString("MEMBER_ID"));
          session.putValue("username",USERNAME);
          session.putValue("first_name",
rset.getString("FIRST_NAME"));
          if (USERNAME.equals("admin")) {
            response.sendRedirect("/VSSX/admin/index.jsp");
          } else {
            response.sendRedirect("/VSSX/portfolios.jsp");
          }
        } else {
            response.sendRedirect("/VSSX/login_error.html");
        }
      } else {
        response.sendRedirect("/VSSX/login_error.html");
      }
      rset.close();
      stmt.close();
      conn.close();
    } catch (SQLException e) {
      System.err.println(e.getMessage());
    }
  }
```

```
}

package member;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class signup extends HttpServlet {
  /**Initialize global variables*/
  public void init(ServletConfig config) throws
ServletException {
    super.init(config);
      try{
       Class.forName("oracle.jdbc.driver.OracleDriver");
    }
    catch (ClassNotFoundException e) {
       System.err.println(e.getMessage());
    }
  }
  //Process the HTTP Post request
  public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    try {
      String USERNAME = request.getParameter("username");
      String PASSWORD = request.getParameter("password");
      String FIRST_NAME = request.getParameter("first_name");
      String MI = request.getParameter("mi");
      String LAST_NAME = request.getParameter("last_name");
      String EMAIL = request.getParameter("email");

      // Create Connection Object
      Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@bishop:1521:o
racle","vssx","vssx");
      Statement stmt = conn.createStatement();

      stmt.executeUpdate("INSERT INTO MEMBERS
VALUES(MEMBER_SEQ.NEXTVAL, '" + USERNAME + "', '" + PASSWORD
+ "', '" + FIRST_NAME + "', '" + MI + "', '" + LAST_NAME +
"', '" + EMAIL.toLowerCase() + "', 1000000, 'UNLOCK')");
```

```
        // Close Connections
        stmt.close();
        conn.close();

        // Redirect Back to User Manager
        response.sendRedirect("/VSSX/");

      } catch (SQLException e) {
        System.err.println(e.getMessage());
      }
  }
}

package member;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class update extends HttpServlet {
  /**Initialize global variables*/
  public void init(ServletConfig config) throws
ServletException {
    super.init(config);
      try{
       Class.forName("oracle.jdbc.driver.OracleDriver");
    }
    catch (ClassNotFoundException e) {
       System.err.println(e.getMessage());
    }
  }

  //Process the HTTP Post request
  public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    try {
      String MEMBER_ID = request.getParameter("member_id");
      String USERNAME = request.getParameter("username");
      String PASSWORD = request.getParameter("password");
      String FIRST_NAME = request.getParameter("first_name");
      String MI = request.getParameter("mi");
      String LAST_NAME = request.getParameter("last_name");
      String EMAIL = request.getParameter("email");
      String BALANCE = request.getParameter("balance");
      String STATUS = request.getParameter("status");

      // Create Connection Object
      Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@bishop:1521:o
racle","vssx","vssx");
      Statement stmt = conn.createStatement();
      stmt.executeUpdate("UPDATE MEMBERS SET USERNAME='" +
USERNAME + "', PASSWORD='" + PASSWORD + "', FIRST_NAME='" +
FIRST_NAME + "', MI='" + MI + "', LAST_NAME='" + LAST_NAME +
"', EMAIL='" + EMAIL.toLowerCase() + "', BALANCE=" + BALANCE
+ ", STATUS='" + STATUS + "' WHERE MEMBER_ID=" + MEMBER_ID);

      // Close Connections
      stmt.close();
      conn.close();

      // Redirect Back to User Manager
      response.sendRedirect("/VSSX/admin/users.jsp");
    } catch (SQLException e) {
      System.err.println(e.getMessage());
    }
```

```
  }
}

package stock;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class ipo extends HttpServlet {
  /**Initialize global variables*/
  public void init(ServletConfig config) throws
ServletException {
    super.init(config);
      try{
       Class.forName("oracle.jdbc.driver.OracleDriver");
    }
    catch (ClassNotFoundException e) {
       System.err.println(e.getMessage());
    }
  }
  //Process the HTTP Post request
  public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    try {
      String SYMBOL = request.getParameter("symbol");
      String NAME = request.getParameter("name");
      String LAST = request.getParameter("last");
      String HIGH = request.getParameter("high");
      String LOW = request.getParameter("low");
      String CLOSE = request.getParameter("close");
      String SHARES = request.getParameter("shares");
      String EXCHANGE = request.getParameter("exchange");
      String STATUS = request.getParameter("status");

      // Create Connection Object
      Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@bishop:1521:o
racle","vssx","vssx");
      Statement stmt = conn.createStatement();
      stmt.executeUpdate("INSERT INTO STOCKS VALUES('" +
SYMBOL + "', '" + NAME + "', " + LAST + ", " + HIGH + ", " +
LOW + ", " + CLOSE + ", " + SHARES + ", SYSDATE, '" +
EXCHANGE + "', '" + STATUS + "')");

       // Close Connections
       stmt.close();
       conn.close();

       // Redirect Back to User Manager
       response.sendRedirect("/VSSX/admin/stocks.jsp");
     } catch (SQLException e) {
       System.err.println(e.getMessage());
     }
  }
}

package stock;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
```

```
public class trade extends HttpServlet {
  /**Initialize global variables*/
  public void init(ServletConfig config) throws
ServletException {
    super.init(config);
      try{
       Class.forName("oracle.jdbc.driver.OracleDriver");
    }
    catch (ClassNotFoundException e) {
       System.err.println(e.getMessage());
    }
  }

  //Process the HTTP Post request
  public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    try {
      String MEMBER_ID = request.getParameter("member_id");
      String SYMBOL = request.getParameter("symbol");
      String ACTION =
request.getParameter("action").toUpperCase();
      String PRICE = request.getParameter("price");
      String QUANTITY = request.getParameter("quantity");
      String TRANS_NUM = "0";

      // Create Connection Object
      Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@bishop:1521:o
racle","vssx","vssx");

      // Insert into Transactions
      Statement stmt = conn.createStatement();
      stmt.executeUpdate("INSERT INTO TRANSACTIONS
VALUES(TRANS_SEQ.NEXTVAL, " + MEMBER_ID + ", '" + SYMBOL +
"', '" + ACTION + "', " + PRICE + ", " + QUANTITY + ", " +
Double.valueOf(PRICE).doubleValue()*Double.valueOf(QUANTITY)
.doubleValue() + ", SYSDATE, 'PENDING')");

      // Get Current Sequence Count
      Statement stmt1 = conn.createStatement();
      ResultSet rset = stmt1.executeQuery("SELECT
TRANS_SEQ.CURRVAL AS TRANS_NUM FROM DUAL");
      while (rset.next()) {
        TRANS_NUM = rset.getString(1);
      }
      // Call Oracle Stored Procedure
      CallableStatement proc = conn.prepareCall("{ call
TRADING.VERIFY(?, ?, ?, ?, ?, ?) }");
      proc.setString(1, TRANS_NUM);
      proc.setString(2, MEMBER_ID);
      proc.setString(3, SYMBOL);
      proc.setString(4, ACTION);
      proc.setString(5, PRICE);
      proc.setString(6, QUANTITY);
      proc.execute();

      // Close Connections
      stmt.close();
      conn.close();

      // Redirect Back to User Manager
      response.sendRedirect("/VSSX/portfolios.jsp");
    } catch (SQLException e) {
      System.err.println(e.getMessage());
    }
  }
}
```

```
package stock;


import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;


public class update extends HttpServlet {
  /**Initialize global variables*/
  public void init(ServletConfig config) throws
ServletException {
    super.init(config);
      try{
       Class.forName("oracle.jdbc.driver.OracleDriver");
    }
    catch (ClassNotFoundException e) {
       System.err.println(e.getMessage());
    }
  }


  //Process the HTTP Post request
  public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    try {
      String SYMBOL = request.getParameter("symbol");
      String NAME = request.getParameter("name");
      String LAST = request.getParameter("last");
      String HIGH = request.getParameter("high");
      String LOW = request.getParameter("low");
      String CLOSE = request.getParameter("close");
      String SHARES = request.getParameter("shares");
      String EXCHANGE = request.getParameter("exchange");
      String STATUS = request.getParameter("status");

      // Create Connection Object
      Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@bishop:1521:o
racle","vssx","vssx");
      Statement stmt = conn.createStatement();
      stmt.executeUpdate("UPDATE STOCKS SET NAME='" + NAME +
"', LAST=" + LAST + ", HIGH=" + HIGH + ", LOW=" + LOW + ",
CLOSE=" + CLOSE + ", SHARES=" + SHARES + ", EXCHANGE='" +
EXCHANGE + "', STATUS='" + STATUS + "' WHERE SYMBOL='" +
SYMBOL + "'");

      // Close Connections
      stmt.close();
      conn.close();

      // Redirect Back to User Manager
      response.sendRedirect("/VSSX/admin/stocks.jsp");
    } catch (SQLException e) {
      System.err.println(e.getMessage());
    }
  }
}
```

APPENDIX C

JAVA SERVER PAGE SOURCE FILES

The JSP Source Files are listed in the table below:

(You can find the source codes of these files in the attached CD-ROM)

Scripts/index.jsp	Admin/index.jsp
Scripts/INC_top.jsp	Admin/INC_top.jsp
Scripts/INC_bottom.jsp	Admin/INC_bottom.jsp
Scripts/INC_styles.css	Admin/INC_styles.css
Scripts/login.jsp	Admin/login.jsp
Scripts/logout.jsp	Admin/logout.jsp
Scripts/portfolios.jsp	Admin/bot.jsp
Scripts/signup.jsp	Admin/usrUpd.jsp
Scripts/order.jsp	Admin/usrCtrl.jsp
Scripts/confirm.jsp	Admin/stocks.jsp
Scripts/history.jsp	Admin/stkUpd.jsp
	Admin/stkCtrl.jsp

APPENDIX D

HTML SOURCE FILES

The HTML Source Files are listed in the table below:

(You can find the source codes of these files in the attached CD-ROM)

Index.html	Signup.html
Login_Error.html	Faq.html
Portfolios.html	Stocks.html
Trading.html	

REFERENCES

This section contains a list of all books, articles and documents cited in this SRS.

[1] *Core Java 2: Volumn I - Fundamentals,* by Cay S. Horstmann and Gary Cornell, Sun Microsystem, Inc. ISBN 0-13-081933-6.

[2] "Developer.com is the leading develop resource for IT professionals" (http://www.develop.com).

[3] "Economic equilibrium" (http://en.wikipediä.org/wiki/Economic equilibrium)

[4] "Hollywood Stock Exchange" (http://www.hsx.com).

[5] Homer Alex, Sussman David, Francis Brian, Reilly George, Esposito, Esposito Dino, Chiarelli Andrea, Kropog Bill, McQueen Craig, Godfrey Nolan, Robinson Simon, Schenken John, Tegel Kent, Professional Active Server Pages 3.0, Wrox Press Inc, September, 1999.

[6] Horstmann S. Cay, Cornel Gary, Core Java2, Volume 1: Fundamentals, Prentice Hall, December 15, 1998.

[7] *JDBC API Tutorial and Reference, 2nd Edition,* by Seth White, Maydene Fisher, Rick Cattell, Graham Hamilton, and Mark Hapner, Sun Microsystem, Inc. ISBN 0-201-43328-1.

[8] Koch George, Loney Kelvin, Oracle8: The Complete Reference, Oracle Pr., September,1997.

[9] "Marcopoly" (http://www.marcopoly.scorpioweb.com).

[10] *Oracle 9i Programming: A Primer,* by Rajshekhar Sunderraman, Addison Wesley Longman, Inc. ISBN 0-321-19498-5.

[11] Serge Abiteboul, Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufmann Publishers, NY 1998.

[12] Tulloch Mitch, Santry Patrick, Administering IIS 5, Computing Mc-Graw-Hill, March 20, 2000.

[13] "Sports Team Stock Market" (http://www.fantasy-ballstreet.com/).

[14] "Virtual Stock Exchange" (http://www.virtualstockexchange.com/).

[15] "Yahoo! Finance" (http://finance.yahoo.com/)