

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2001

Virtual online stock trading system

Tao Zhu

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Zhu, Tao, "Virtual online stock trading system" (2001). *Theses Digitization Project*. 1893.
<https://scholarworks.lib.csusb.edu/etd-project/1893>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

VIRTUAL ONLINE STOCK TRADING SYSTEM

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
Tao Zhu
December 2001

VIRTUAL ONLINE STOCK TRADING SYSTEM

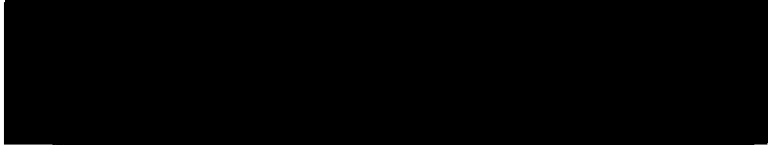
A Project
Presented to the
Faculty of
California State University,
San Bernardino

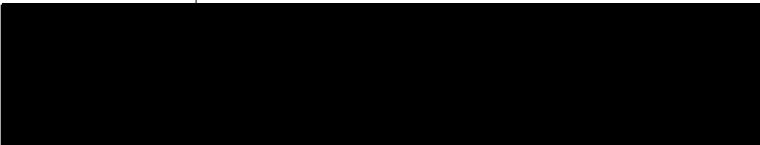
by
Tao Zhu
December 2001

Approved by:


Dr. Owen J. Murphy, Chair

11/12/01
Date


Dr. Arturo I. Concepcion


Dr. Kerstin Voigt

ABSTRACT

In general, stock trading provides a relatively low-risk and hassle-free channel for people to invest, when compared with other methods. However, it is unfortunate that, for those other people who may be equally interested and financially able in stock trading, they have yet to experience the simplicity of, and the wealth accumulated from, stock investments. Reasons for their hesitation to be involved in stock trading vary; but primarily it is due to their lack of experience in trading, insufficient financial resources for major investments, and the unwillingness to get into the highly volatile stock market. For these people, the author has developed a powerful stock simulation tool, VCOSTS (Virtual Customer-to-Customer Online Stock Trading System), to train individuals on how to trade stocks.

The VCOSTS project is motivated by the fact that novice investors need training tools that will enable them to make sound investment decisions that are deep-rooted in the acquired financial knowledge and skill set from these training tools. VCOSTS is developed as a Web-based stock trading simulation with PHP4, MySQL database, Apache Web server on Red Hat Linux operating system. The PHP4 session

handling is adopted for effectively tracking visitors on the site and managing their information. The interactive and informative VCOSTS site provides users with insight into the dynamics of the stock market by catching real-time stock data in the market. The object-oriented concepts and adequate security measures are both implemented in the project design stage.

ACKNOWLEDGEMENTS

I would like to express my gratitude and thanks to my advisor, Dr. Owen J. Murphy, for his thoughtful guidance, timely encouragement, and for his willingness to serve as my thesis committee chairman. His guidance and assistance have helped me to complete this project in a timely fashion. I am also thankful for my other committee members Drs. Arturo I. Concepcion and Kerstin Voigt, whose additional insight and suggestions that have enhanced this project work in major ways.

I am grateful to the Department of Computer Science for its years of support and for its academic rigor that I have enjoyed. Indeed I have been privileged to expand my horizon for studying in such a leading research institution.

My special thank is also extended to my husband, Yaojun Shi, for his love, understanding and encouragement.

All of these I will keep in my mind forever.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGEMENTS	v
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER ONE: STOCK TRADING MARKET RESEARCH	
Traditional Brokers	1
Online Brokers	4
Psychology of Stock Trading	6
Comparison	6
CHAPTER TWO: INTRODUCTION OF SYSTEM DESIGN	
Scope and Potential Use	8
Definitions and Abbreviations	9
Project Overview	10
CHAPTER THREE: SYSTEM CONFIGURATION AND DEVELOPMENT TOOLS	
Tool for Database Management	12
Apache Web Server	14
Hypertext Preprocessor	15
CHAPTER FOUR: PROJECT DESIGN	
Database Design	18
User Phase Design	24
Administrator Phase Design	30

Trading System Design	32
User Session Control	33
Object-oriented Design	35
Security	36

CHAPTER FIVE: PROJECT IMPLEMENTATION

Obtain the Real-time Stock Data	38
Register a New User Account	39
Confirmation Login	40
Login	41
Logout	43
Submit Buy Stock Request	44
Submit Sell Stock Request	45
Functions for Account Transaction	45
Functions for Account View	47
Stock Transaction	49
Administrator Functionality	52
Backend Cron Jobs	53

CHAPTER SIX: TESTING

System Validation	55
Unit Testing	56
Integration Testing	57
System Testing	59

CHAPTER SEVEN: USER MANUAL

Manual Overview	66
Registration	66
Confirm Login	68
Login	69
Logout	69
Buy Stock	69
Sell Stock	71
Option - Change Password	72
Forget Password	72
Account Summary	73
Stock Summary	73
Administration Tools	73
Utility Pages	74
Clock and Calculator	75

CHAPTER EIGHT: CONCLUSIONS AND FUTURE DEVELOPMENT

77

APPENDIX A: PROJECT FIGURES	80
---------------------------------------	----

APPENDIX B: PROJECT DATABASE SCHEMA	110
---	-----

APPENDIX C: PROJECT SOURCE CODES	114
--	-----

BIBLIOGRAPHY	176
------------------------	-----

LIST OF TABLES

Table 1. Comparison of Traditional Brokers, Online Brokers and This Project	6
Table 2. Project Cron Jobs	54
Table 3. Test Case Reports	60

LIST OF FIGURES

Figure 1. Project User Case Diagram	26
Figure 2. Project Class Diagram	35
Figure 3. Login Diagram	42
Figure 4. Logout Diagram	43

CHAPTER ONE

STOCK TRADING MARKET RESEARCH

Traditional Brokers

Stock trading has always been a prevalent means of investment for most people in the US. According to a recent survey, nearly half of the US families own stock or mutual funds in some ways as part of their investment portfolio. Normally, investors who want to buy or sell stocks will place orders through brokers (Klein, 1998). The brokers earn commission in exchange for routing their customers' orders to a stock exchange.

In general, the traditional stock trading brokers include the following categories (Farley, 2000; Labier, 2000; Edwards, 1997; Bruner, 1998):

- Full-service broker

They provide local branch offices, personal account representative (trade stocks instead of customers), and other miscellaneous services such as market research, and newsletters.

- Half-service broker

They provide basic services such as advice, newsletters, and literature. In addition, they execute

stock trading for customers based on customers' requests to buy or sell stocks. They do exactly as customers requested.

- Minimum-service broker

They execute stock trades only. This is similar to online stock trading broker in that customers pretty much charge the whole stock trading processes. People who do not require or desire much advice tend to choose minimum-service brokers.

In a stock exchange, the broker routes all the orders relating to a particular stock to a market maker. Market makers are people assigned to the particular security. Their job is to facilitate the trading of stocks. Market makers attempt to match the purchase orders routed to them by other brokers with sales orders. If no exact match is available, the market maker will trade the incoming order for or against his or her own stock holdings by acting as a principal in the stock transaction.

Market makers charge commissions from price difference between the bidding in an offer to buy and the asking price. They trade their own stock holdings against other brokers' orders. During such trading processes,

price changes are recorded instantly when an order is traded. This means that market makers will execute an investor's buy or sell orders in a trade immediately upon receiving the order.

The above section described the basic stock trading processes. Looking into the processes more closely, we soon find out that, for a typical broker, his profit sharing in a market maker's profits, in general, is significantly more than the commissions he earns from investors. One potential problem associated with such trading practice is that a broker whom investors have entrusted with their money does not necessarily seek the best interests for his investors. He may actually seek first his share of profits from the market maker, because such profits are generally better than the commissions he could have earned from his investors.

Another problem in such stock-trading practice is that a market maker may manipulate the number of shares in each trade to maximize his profits. For example, a market maker may not fully fill an investor's order at the level that the investor desired at the posted price in a rising market. Market makers are now allowed to quote as few as one hundred shares on a stock per trade. For a day trader,

he is facing unfair competition from market makers who assess a \$15 to \$25 per trade fee for their services. For a day trader, it is extremely difficult to be profitable without sufficient liquidity that allows him to trade in 500 to 1000 shares per trade at a time. Market makers, on the other hand, have an unfair advantage over day traders and other small investors. Commission rate of \$15 to \$25 per trade adds up quickly. Assuming a market maker who trades only 20 rounds per day, each with a \$20 fee, he will generate \$2,000 per week in commissions. That is over \$100,000 per year!

Online Brokers

The Internet has generated many new opportunities for people. Online brokerage services, for example, have provided great convenience to many small stock investors. With an online brokerage service, an investor needs to pay only a nominal service charge to trade stocks. They can trade stocks in their own time and by their terms with a small per trade charge.

However, the biggest problem in the current stock market is the limitation of the open market schedule. Individual investors are restricted to trading stocks only

during the open market hours. This is frustrating to individual investors, because stock prices can rise or fall sharply in the next trading day. Moreover, most investors are not full-time traders - they need to work full-time during the stock market trading time, which makes it nearly impossible for them to trade during the markets' regular business hours.

The NASDAQ stock market's regular trading hours are 9:30 a.m. to 4:00 p.m. EST for every business day. The market is closed during holidays. The following eight dates are holidays when NASDAQ closes: January 17 - Martin Luther King Jr.'s Memorial, February 21 - Presidents' Day, April 21 - Good Friday, May 29 - Memorial Day, July 4 - Independence Day, September 4 - Labor Day, November 23 - Thanksgiving Day, and December 25 - Christmas Day.

Taking away the above eight holidays and weekends, the total number of trading days in the year 2000 is only 252 days, which converts to a total of 1,638 trading hours out of the total available hours of 8,760 in a year. Therefore, during those 7,122 (i.e. 8,760 minus 1,638) non-trading hours, investors can complete no trades, no matter how profitable it could be possible for them.

Psychology of Stock Trading

High risk in the stock market and insufficient prior experience are often quoted as the two major reasons that prevent many from actively engaging in stock trading. The third most often quoted reason for inactivity in stock market is small investors' lack of funds while desiring to engage in this thrilling investing game.

Comparison

To resolve the problems addressed in above, VCOSTS - Virtual Customer-to-Customer (C2C) Online Stock Trading System was developed. VCOSTS connects stock buyers and sellers via Internet directly. The system offers a year-round, continuous 24-hour trading day, 7 days a week (including holidays). Not only the system can be used as a training tool for those novice investors, but it can also be extended to the real world stock trading.

Table 1 shows the difference among traditional brokers, online brokers and VCOSTS project.

Table 1. Comparison of Traditional Brokers, Online Brokers and This Project

Category	Characteristics
Traditional Brokers	People help customers trading stock in market and conduct trading only in business hours.

E*Trade	It is an online stock trading with limited extended trading hours. Use expensive operating system and software. Connect to NASDAQ network and credit union network.
This project	It is an online stock trading with 24/7 uninterrupted trading hours, which is developed by open-source applications. It is also a good investment training tools. Two different types of account: regular and premium. No physical connection to NASDAQ network and credit Union Network. Trade as market price during business hours; auction within VCOSTS during off-business hours.

CHAPTER TWO

INTRODUCTION OF SYSTEM DESIGN

Scope and Potential Use

The stock trading simulation software is named "Virtual Customer-to-Customer Online Stock Trading System", or VCOSTS. The project has completed both the system design and the implementation phases in its development. The end product of the project, a stand-alone software program VCOSTS, can be used by a wide spectrum of users. For instance, the NASDAQ firm members performing sophisticated on-line trading can use VCOSTS as an intelligent advising device. A variety of other beginning traders or students conducting training simulations are able to acquire necessary knowledge and skills in stock trading through VCOSTS.

One unique feature of VCOSTS is its extension of trading time to go beyond the current stock trading practice - presently the stock markets only allows a trader to deal during open business hours. VCOSTS has automated an auction utility that allows its users to trade stocks even after the stock markets have been closed. The automation of the trading in VCOSTS enables

its users to trade stocks twenty-four hours a day and seven days a week, which allows them to access their account balances and history, to change passwords, and to trade stocks at any point of time at their own discretion. Upon its full implementation, we project VOCOSTS to be used in a wide range of applications by on-line traders, financial and educational institutional users, as well as other e-business owners.

Definitions and Abbreviations

VOCOSTS, the Online C2C stock trading system, is configured as follows:

Operating System:	Red Hat Linux 7.0
Web server:	Apache Web Server 1.3.12
Database:	MySQL 3.23.32
Development tools:	PHP4.0.1

The following provides definitions and terms of each component in the systems:

C2C:	The abbreviation for customer-to-customer
NASDAQ:	The abbreviation for the National Association of Securities Dealers Automated Quotation system. It is also commonly called the OTC market. The NASDAQ market is an inter-dealer

market represented by over 600 securities dealers trading more than 15,000 different issues.

RDBMS: The abbreviation for relational database management system.

SQL: The abbreviation for Structured Query Language.

Project Overview

The project will develop this online C2C stock trading system. It functions during both normal business hours stock trading system and as an after hours customer-to-customer stock auction system. In particular this project will deal with the time limitation of current stock market by supplying the auction service during the closing quotation time. The automation of the trading would allow customers to trade their stocks 24 hours a day, 7 days a week. In the remaining sections, the following contents and organization are explained:

The remaining chapters in this report describe general factors considers during the design and implementation stages as well as the resulting project

requirements and test results. The chapters are organized in the following fashion:

Chapter 3 gives a brief description of system configuration and development tools such as setting up database, Apache, PHP.

Chapter 4 is a detailed description of project design, which includes user, administrator phase design, trading system design, and database design.

Chapter 5 introduces the project implementation. It is consisted of the following introduction: obtaining the real-time stock data online, registering new user account, and online stock trading.

Chapter 6 emphasizes on user session control, object-oriented design and security issues.

Chapter 7 provides detailed information on system validation, unit testing, and integration testing.

Chapter 8 is a VOCOST system's Users' Manual.

And finally, Chapter 9 summarizes contributions of this project, and suggests directions for other future developments.

CHAPTER THREE

SYSTEM CONFIGURATION AND DEVELOPMENT TOOLS

Tool for Database Management

VCOSTS adopts a relational database management system (RDBMS), MySQL, in its design process. An RDBMS is an essential database tool in many environments that range from traditional users in business, research and education contexts to newer applications such as powerful search engines on the Internet.

VCOSTS uses the MySQL RDBMS for its database management. During the system design stage, VCONSTS took into account factors such as system performance, support, system features such as SQL conformance and extensions, licensing terms and conditions, and prices to reach the conclusion in its RDBMS selection. In addition, other unique features in MySQL documented in the book (Dubois, 2000) also help the adoption of MySQL as the database management system in VCONSTS.

In adopting MySQL in VCONSTS, the following factors reported in the book (Dubois, 2000) were considered:

1. Speed: The developers contend that MySQL is high-performance database engine that VCOSTS can possibly get within its budget.
2. Cost: Even though MySQL is not exactly an Open source product therefore its use is not free. Nonetheless, it allows a free use for general functions in RDBMS. Under Linux and other non-Windows platforms, MySQL's client programs and its client-programming library can be used free of charge on condition that the resulting software is not sold as a commercial software in open market. For the database management needs in the VCOSTS project, we view the \$200 usage fee as a real bargain for a professional RDBMS such as MySQL. The many modules that are available in MySQL have facilitated an effective database management implementation in VCOSTS.
3. Query language support: MySQL understands SQL, which is the language of choice for nearly all modern RDBMSs.
4. Ease of use: MySQL is a high-performance but relatively simple to use. It is easier to set up and maintain than many RDBMSs.

5. Portability: MySQL runs on varieties of Unix, Linux and other non-Unix systems such as Windows and OS/2.

Apache Web Server

According to a recent survey on web server uses completed by Netcraft, over 60 percent of the web sites all over the world are using Apache Web server. The same survey reports that the other competing products are far behind at 19.6 percent for Microsoft IIS and at 7 percent for Netscape/iPlanet, respectively. It also has many advantages over other competing products.

The Apache web server is an open-source HTTP server that can be used in various desktops and server operating systems such as UNIX, Linux and Windows NT/2000 (Harlow, 2000) with free of use. It is a secure and efficient web server that provides HTTP services that are in sync with current HTTP standards.

Apache was designed to correctly implement the HTTP standards. It is solid, reliable, and portable when used in any operating systems. In addition, it is highly configurable and extensible with third-party modules. It accommodates most of the function areas a web server is

expected to do. Nearly all the full source codes for the Apache web server are freely available for developers who want to make customized changes by writing 'modules' using the Apache module API.

Hypertext Preprocessor

Today's dynamic web sites have accelerated the development of server-side scripting technologies. In the early days of web development, any kind of dynamic interaction requiring CGI scripts was costly in implementation and difficult to maintain. For example, Perl is still a popular language for CGI scripts. Although the mod_perl module in Apache server can reduce load and execution times for scripts, it still requires significant maintenance to keep separate HTML files and perl scripts. According to Maxfield(2000), many design elements are hard-coded into Perl scripts, which make them difficult to maintain and to update for developers.

According to a survey on PHP use done by NetCraft and Security Space, over a third of all Apache servers use PHP. PHP is an HTML embedded server-side scripting language. The "HTML embedded" feature means that the code is inserted directly into the HTML document between

special tags thus can be interpreted by the server immediately each time the web page is requested. This feature ensures speedy access and processing under PHP.

In addition to features such as variable manipulation, flow control and function constructs that are available in other typical general-purpose programming languages, PHP has incorporated other features. For example, it is tuned for Web development with automatic conversion of form variables to PHP variables. It also includes built-in functions to handle cookies and session management. Furthermore, it also accommodates many third-party modules to seamlessly integrate with PHP to extend its functionality - modules that allow a developer to access other Internet services such as e-mail (Meloni, 2000), operation systems and several database packages.

In addition to the aforementioned factors, the VCOSTS project has also considered the following features in adopting PHP as its processor:

1. PHP is a full-featured debugger that comes with breakpoints and step-through execution.
2. PHS can access other COM objects.

3. PHP contains an output buffer that allows the abortion of a page in the middle of the script and the redirection of header.
4. PHP allows automatic resource reallocation.

CHAPTER FOUR

PROJECT DESIGN

The VCOSTS project underwent the following two phases in its system design: the User phase and the Administrator phase. During the User phase, the focus of the design was on various user functions in VCOSTS. Since it involves most user functions in the project, hence it is more important in the system design. The Administrator phase involves functions for administrator to monitor users' accounts and balances, transactions, and other trading-related activities. It is not as important as the User phase.

Database Design

The project uses MySQL to design and implement C2Costs database. First, data tables are constructed to record stock online C2C trading and auction system. This C2Costs database consists of the following six tables: trader, transaction, prem_trader, account, account_stock, and stock.

VCOSTS database application is used to illustrate the Entity-Relationship (ER) model concepts and their use in schema design (Elmasri & Navathe 2000; Blaha & Premerlani

1998). Figures 5-15 in Appendix A show how the schema for this database application is displayed by ER model diagrams and object model diagram. The database keeps track of VCOSTS's trader, account, stock, and transaction. The following description of VCOSTS is represented in the database:

1. VCOSTS have many registered traders. Each trader has a unique number. A trader is either regular or premium trader. One trader has an account, regular account or premium one.

2. When each trader confirms his registration, he is assigned to a unique account number corresponding to his registered email number. We keep track of the trader's account information. Many accounts trade many different stocks. And one account could have many transactions to deal with.

3. A stock uses unique symbol, and keeps track of the stock price in the market. One stock could be dealt with in many different transactions.

4. A transaction records every trader's buy or sell processing, each of which has a unique transaction number. We keep track of the start time, transaction time, end time, and trader price.

As far as the Object model and the ER model are concerned, for instance, Figure 6 shows TRADER entity using object model. This entity has the following attributes: name (lname, fname, middle_initial), address (addr1, addr2, city, state), phone (home_phone, work_phone, cell_phone), fax, email, password, confirm_hash, active and trader_id, which is the primary key (PK). Figure 8 is using ER model to describe TRADER data as entity, relationship, and attributes.

The trader table contains information about traders. Attributes in the table and their definitions are listed below:

trader_id-	a trader ID that is auto-generated by MySQL
fname-	The trader's first name
middle_initial-	The trader's initial name
lname-	The trader's last name
addr1-	The trader's address 1
addr2-	The trader's address 2
city-	The trader's residence city
state-	The trader's residence state
home_phone-	The trader's home telephone number
work_phone-	The trader's work telephone number

cell_phone-	The trader's cell phone number
fax-	The trader's fax number
email-	The trader's email address
password-	The password for the trader to login VCOSTS site
confirm_hash-	The confirmed hash for security
active-	account status (active or not)

The statement to create this table is shown in

Appendix B.

The transaction table contains the following attributes:

transaction_id-	transaction ID that is automatically generated by MySQL
account_id-	account ID
transaction_type-	stock transaction type
symbol-	the stock symbol
stock_quantity-	the quantity of stock
trader_price-	the current trader asking price
start_time-	the start time of stock transaction
end_time-	the end time of stock transaction
transaction_time-	the complete time of stock transaction

status- the transaction status
trans_time_type- the transaction time type

The trader_perm table contains the following
attributes:

trader_id- the trader ID
SSN- the trader social security number
birthdate- the trader birth date

The account table contains the following attributes:

account_id- the account ID
trader_id- the trader ID
account_type- the account type for trader
cash_balance- the cash balance in the account
original_value- the original visual cash in the
account
reward_pct- the percentage of reward for
premium account

The account_stock table contains following
attributes:

account_id- the account ID
stock_symbol- the stock symbol
stock_quantity- the quantity of stock

Finally, the stock table contains the following
attributes:

symbol-	stock symbol
company_name-	the company name
description-	the description of the company
last_trade_price-	the last trading stock price
day_change-	the daily change percentage of stock
day_high-	the daily highest price for stock
day_low-	the daily lowest price for stock

This database is fully utilized to trade or auction stocks in VCOSTS site. The C2Costs database and its application will need to perform the following tasks:

1. The registration when a trader signs in to use VCOSTS services.
2. The login into and logoff from the VCOSTS system.
3. Viewing trader's account summary, such as stock value, cash balance in the account.
4. Buying stock either during normal business hours or extended hours by using the auction function.
5. Selling stock either during normal business hours or extending-hour by using the auction function.
6. Account administrator function such as viewing all the transaction, stopping certain transaction record, and changing the password.

7. Account password change.
8. Removal of pending trading or auction records.
9. Sending email to VCOSTS support representative.

User Phase Design

Online stock trading for user involves several basic functions:

- Register new account.
- Login and logout account.
- Buy stocks.
- Sell stocks.
- Change password.
- Get stock quote.
- View account summary.

The project user case diagram is shown in Figure 1. Figure 16 in Appendix A is a diagram depicting the stock trading process of a mock user, Mr. Danny Taylor.

VCOSTS contains the following eight link pages. Each of them can be directly accessed from the main web page. The eight pages are login, register, introduction, rule, manual, FAQ, education, and contact information.

The following sections provide a detailed illustration of the login and the register pages. Item

numbers 1 and 2 in below describe the introduction and the rule pages, followed by the manual page, which is a users' guide detailing all functions a user can use upon signing in.

The FAQ page allows a user to find answers to their frequently asked questions, while the Contact page provides information concerning the VCOSTS company. The education corner recommends good books on stock investments and supplies twenty golden strategies for stock traders.

The following describes a mock transaction process by Mr. Taylor's, the mock trader's, trading script while using VCOSTS.

1. The introduction page provides a quick preview for VCOSTS works and briefly describes user requirements for running the VCOSTS system. It also forwards a user to either the registration page for new user signing in, or the login page for registered users.
2. For a perspective new user to system, VCOSTS will suggest that he go over the rule page, which describes the terms while login in VCOSTS, in order to familiarize them with various terms and conditions

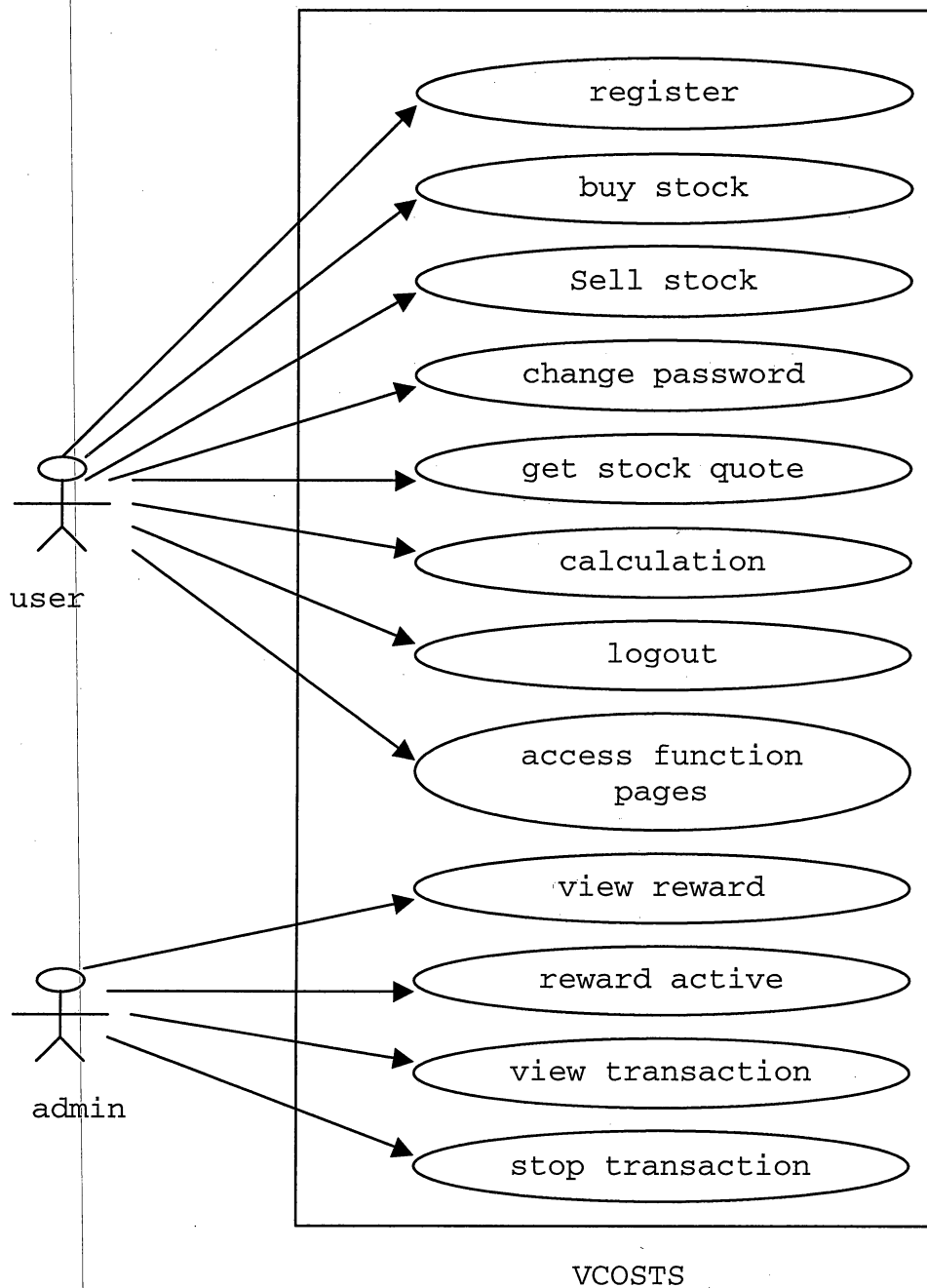


Figure 1. Project User Case Diagram

in using the system. After such brief introduction, the user enters into the login page to sign up with their user information in order to create a user account. Once the account is created, the user can start using the VCOSTS for stock trading simulation.

3. The mock user, Mr. Danny Taylor, now visits the VCOSTS website and accesses the registration page.

4. He chooses the regular account registration, and fills out the registration form with the following user information:

email: 'tao1990@hotmail.com'

password: 'test'

5. The VCOSTS server checks the email format to see if it is valid. VCOSTS will send an error message if the typed email is not in a valid format. After VCOSTS verified the email format, it begins to search in its user name database to see if a typed email address is already included in the database previously. If VCOSTS finds out duplication in Mr. Taylor's email registration, it will return him back to the registration page to repeat the signing in process.

6. After Mr. Danny Taylor submits the correct information, the server will send an instant message along with confirmation code, which the system assigns to all valid users, to his verified email address. By accessing the URL, which passes confirmation code, Mr. Taylor logs in VCOSTS via his email account of 'tao1990@hotmail.com' whose password is 'test'.
7. After the above verification, Mr. Taylor's new account should be updated to "active one", which activates his account, and leads him to the main page to begin the trade simulation. The purpose of the above confirmation process is to ensure the proper use from genuine users while at the same time prevent people from abusive use of VCOSTS either intentionally or unintentionally.
8. Once the first successful login in the confirmation page is established, Mr. Taylor, as well as any valid users, can access his account directly from the regular login page, where the user's registered email address is used as his user name and his registered password as the key to access VCOSTS.

9. By clicking the buy stock button in the main page, Mr. Taylor obtains a stock-buy page, from which he can submit the buying request by specifying stock symbol, quantity, price and expire time. The VCOSTS server verifies the expire time, then tries to find an optimal match of Mr. Taylor's bid price with either the market price or other VCOSTS members' selling price. This process continues, until VCOSTS finds a match and Mr. Taylor's accounts updated based on the completed transactions.
10. Likewise, VCOSTS contains a sell page that functions in similar ways to those of the buy stock page. One additional step in the sell page is that the VCOSTS server will verify the user's account balance to see if he has sufficient shares of the designated stock on hand before the system proceeds to process the requested sell transactions.
11. Mr. Taylor can change his password anytime he has established his account. This is done via the password replacement procedure, which requires him to type the old password once for verification, and type the replaced password twice to safeguard the valid use of users' accounts.

12. The get quote button on the same page allows Mr. Taylor to obtain real-time stock data from stock markets at anytime.
13. After completing all his transactions in VCOSTS, Mr. Taylor is able to logout from the system. A valid logout procedure is required for the user to exit the system, which will lead him back to the Home page.

Administrator Phase Design

Only VCOSTS system administrators have access to the Administration page. Regular users cannot access the page. The Administration page provides system tools for an administrator to monitor all users' transactions and accounts. There are the following four basic functions that a system administrator may perform in the course of a VCOST login session: view rewards, reward activation, view transactions, and stop transactions. The following provides further details for each of the four functions.

View Reward

This screen shows each corresponding account's information such as trader ID, social security number, date of birth, and reward percentage, for a particular premium account user. Among the shown items on this

screen, the reward percentage column merits more attention. For newly registered premium account users who have not yet paid their membership dues, their account is set at 0 to begin with, which indicates that it is impossible for the user to begin trading or to receive rewards until their account has been verified and activated.

Reward Active

Upon receiving a premium account user's membership fees, the administration can use the Reward Activation button to activate the user's account in order for the user to begin earning trading rewards for up to 0.2 percent of his earnings.

View Transaction

VCOSTS administrator can monitor all user transactions, including user account management and other transactions such as pending, trading and aborting stock transactions from this screen.

Stop Transaction

The administrator can access this page to change pending transactions into obsolete ones. From this page, the administrator manages all the pending transactions in the system.

Trading System Design

Market Trading Design

During regular business hours, VCOSTS users trade stocks according to real-time market price. Based on the real-time stock data, the VCOSTS' server searches through its database to find the optimal matching between the trader's offering price and the real-time market prices. If the trader's buy price is equal or higher than the market price, VCOSTS will identify the match and allows the buying transaction to proceed. On the other hand, if the trader's sell price is equal or lower than that of the market, VCOSTS identifies the match, and allow the selling transaction to proceed.

Internal Auction Design

During after-business hours such as on weekends and holidays when regular stock markets are closed and stock trading is halted, VCOSTS designs an internal auction function that allows its users to continue trading even when the stock markets are closed. This function compares the offering prices from traders who are bidding and selling prices from traders who are buying. When an offering price matches with a selling price, the stocks change hands and the transaction is recorded. VCOSTS, in

turn, will record the transaction both for the buyer and for the seller of the stocks.

User Session Control

The following sections describe additional features in the VCOSTS. These features include user session control, object-oriented programming, and system security. The VCOSTS uses primarily HTTP session functions to keep track of user session. For example, the first time a trader logs in the VCOSTS, a session with an identifier is created. The PHP scripts then use the identifier for the same user to his future sessions whenever he logs in. Through such a mechanism, VCOSTS maintains and updates the user's transaction information such as buying a stock via an online order using the user's identifier. Every request the system's browser makes to the server will include this session-identifier parameter, which facilitates identification of the user and all transactions associated with his request.

In order for a user to generate a unique session ID for himself, he may use the unique function. With this command, the resulting session ID will be unique as long as no identical IDs are generated at the same microsecond.

As an option to the user, the VCOSTS allows the user to use the unique command as a built-in function that can be passed to the user's IP address to preclude having two users getting the same session identifier simultaneously.

Every time when a user logs in successfully, the VCOSTS starts to track the session information. The VCOSTS will track the session_register() for the following three fields: s_account_id, s_first_name, and s_last_name. These three global variables will remain active during the whole session until the user logs out and exits the system. After the user has logged out from the VCOSTS, the above three variables will be nullified by using the function of session_unregister().

Every time when users login successfully, the VCOSTS function starts to hold the session information. First it handles session_register() for s_account_id, s_first_name, and s_last_name. Thus those three global variables will keep being active during the whole session until users logout. On logout, the three variables will be released by the function of session_unregister().

Object-oriented Design

The VCOSTS adopts the objected-oriented programming (OOP) paradigm in its system implementation. OOP, in general, involves reusability, encapsulation, inheritance, and polymorphism. The conceptual objects in VCOSTS are modeled in the program codes. Encapsulation keeps an object's data and methods that use the data together as part of the object. Inheritance makes it possible to take

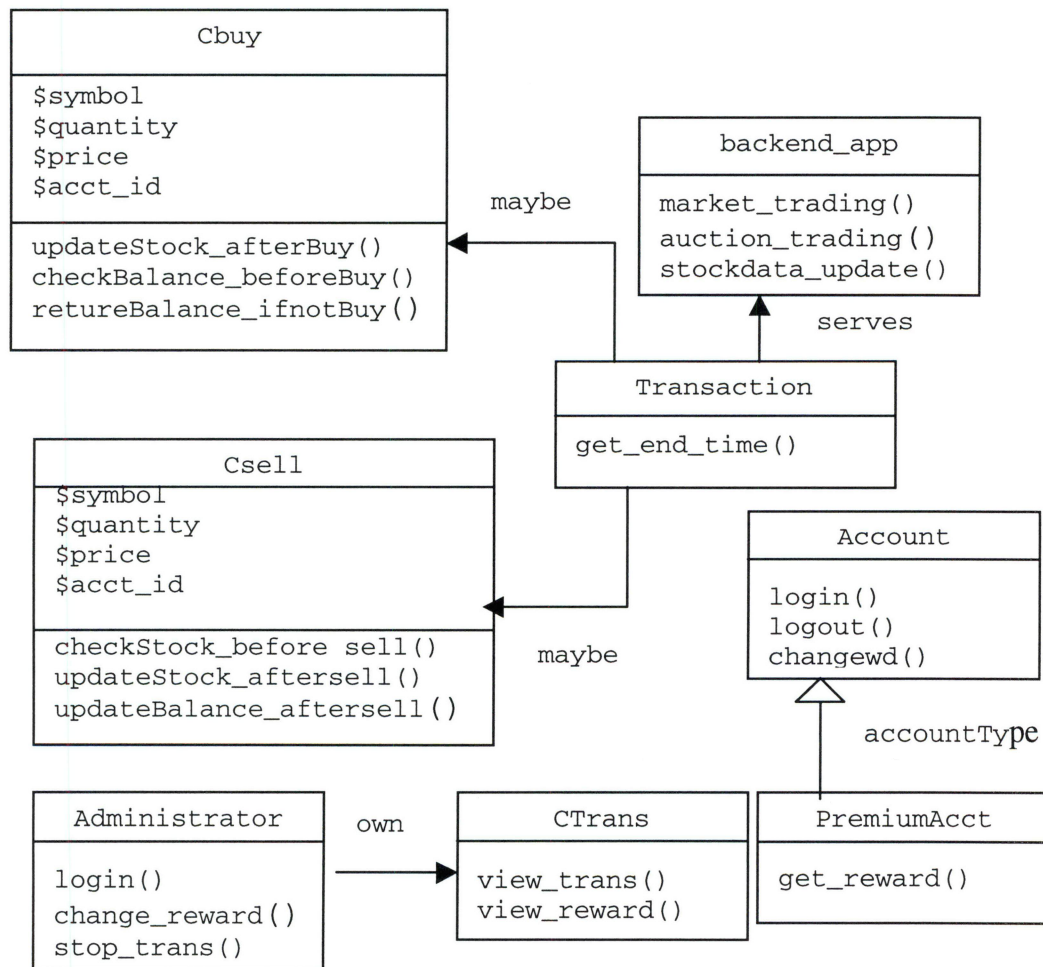


Figure 2. Project Class Diagram

existing object-oriented codes and add capabilities to the existing codes without changing the existing codes.

Polymorphism typically applies to objects that are part of the same inheritance hierarchy; all of the objects within a hierarchy can potentially respond to requests that objects earlier in the hierarchy could respond to. The VCOSTS project design also adopts the OOP concept, such as in the classes of CBuy, CSell, CTrans(See Figure 2 for VCOSTS class diagram). Class diagram is the expression of the object model. It is very efficient to abstract many individual instance diagrams to document data structure thoroughly. For example, Cbuy class is denoted by a box with the class name in the top portion of the box. A second portion of the box lists attributes \$symbol, \$quantity, \$price, and \$acct_id. The third portion of the box lists operations updateStock_afterBuy(), checkBalance_beforeBuy(), and retureBalance_ifnotBuy(). Cbuy class also has the relationship to the other classes of objects like transaction.

Security

The VCOSTS project utilizes the md5() function, a message-digest algorithm, to calculate the md5 hash of a

string. For example, the project uses this function to encrypt the password that the user inputs. When a user gives a password during registration, the function will encrypt it as `md5($password)` before it is stored into the VCOSTS database. Every time when a user logs in, VCOSTS will encrypt the password that the user entered, then compare it with the one in the VCOSTS database.

The MD5 message-digest algorithm takes a message of arbitrary length as input, and generates a 128-bit "fingerprint" or "message digest" of the input as its output. Although it is not computationally impossible to produce two messages having the same message digest, it is not computationally feasible to produce any message having a given pre-specified target message digest. The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables - the algorithm can be coded quite compactly.

CHAPTER FIVE

PROJECT IMPLEMENTATION

Obtain the Real-time Stock Data

VCOSTS has its own database, which maintains market stock information in stock table. To provide real-time stock information for traders, the stock table is updated frequently by real-time market data using Yahoo's nifty.csv download option. This is Yahoo's free service that provides "Stock Quotes" in Comma Separated Values (CSV) format. The process for downloading stock data is as follows:

1. First the VCOSTS activates the PHP function of `fopen()` to open the financial URL of Yahoo page, `http://finance.yahoo.com/d/quotes.csv?s=%s&f=s11d1t1c1ohgv`, which contains the source price information of stocks.
2. Then the VCOSTS uses the function of `fgetcsv()` parses the line it reads for fields in CSV format, and returns an array containing the fields read.
3. Finally, all these online information of stocks in the array will be used for updating stock table in VCOSTS database timely.

Register a New User Account

Figure 17 in Appendix A shows how the VCOSTS registers a new user. After a user submits his registration form online, functions in the register.php file will handle normal account registration while functions in the premRegister.php file will handle premium account user's registrations.

The flow chart in Figure 17 of Appendix A shows that the VCOSTS server completes the registration in the following procedure:

1. First, the VCOSTS server checks the following required fields - password 1, password 2, and email field - to make sure if these fields are filled out properly.
2. Once the verification of the above fields is completed, VCOSTS proceeds to check if the given email has the validated format. After that, the function will go through the database to see if the email is duplicated. If the VCOSTS finds a duplicated email address, it will send an error message, and requires the user to redo the registration process with another email address that is not duplicated with any existing accounts in the current database. Otherwise, VCOSTS will insert the entered data into its trader table and account table based on the

registration form that the user just entered. In storing passwords information, VOCSTS uses a hashed password instead of a real password that the user typed. In addition, it uses a coded confirm_hash in the database tables.

3. After VOCSTS stores the password information, it send an email confirmation with a conformation code to the new user, while at the same time, the VOCSTS server passes the screen information back to the main page and show a success message to indicate the validation of the user registration.

Confirmation Login

Figure 18 in Appendix A shows the login confirmation process in VOCSTS. Once a user receives a confirmation email message from the VOCSTS server, he may either click on the Confirmation Button or type the URL that comes with his confirmation code to access to the VOCSTS' Login Confirmation page. The following is a description of the process encompassing the Login Confirmation process:

1. First the function in the confirm.php page will check if the user's email and password fields are valid. If these fields are invalid, the VOCSTS server shows an

error message to send the user back to the Main page to restart the login process, otherwise it passes the valid confirm_hash to locate the corresponding email and password of the user from its database.

2. VCOSTS will then proceed to compare the email and password that the new user entered against the ones recorded in VCOSTS' database to validate the user data. Once the user input is validated, the VCOSTS system updates the trader table by setting the attribute of activation as "Yes" while at the same time replacing the current content in confirm_hash, since it is now validated and updated.

3. After the confirmation of the user data, VCOSTS continues to retrieve user information by triggering the get_user_data function. Once this process is completed, VCOSTS returns the user back to either the Main page or the Premium Account Main page where the user logged in to VCOSTS originally.

Login

Figure 3 shows the Login process in the VCOSTS. The Login.php page handles a user's login process in the following way:

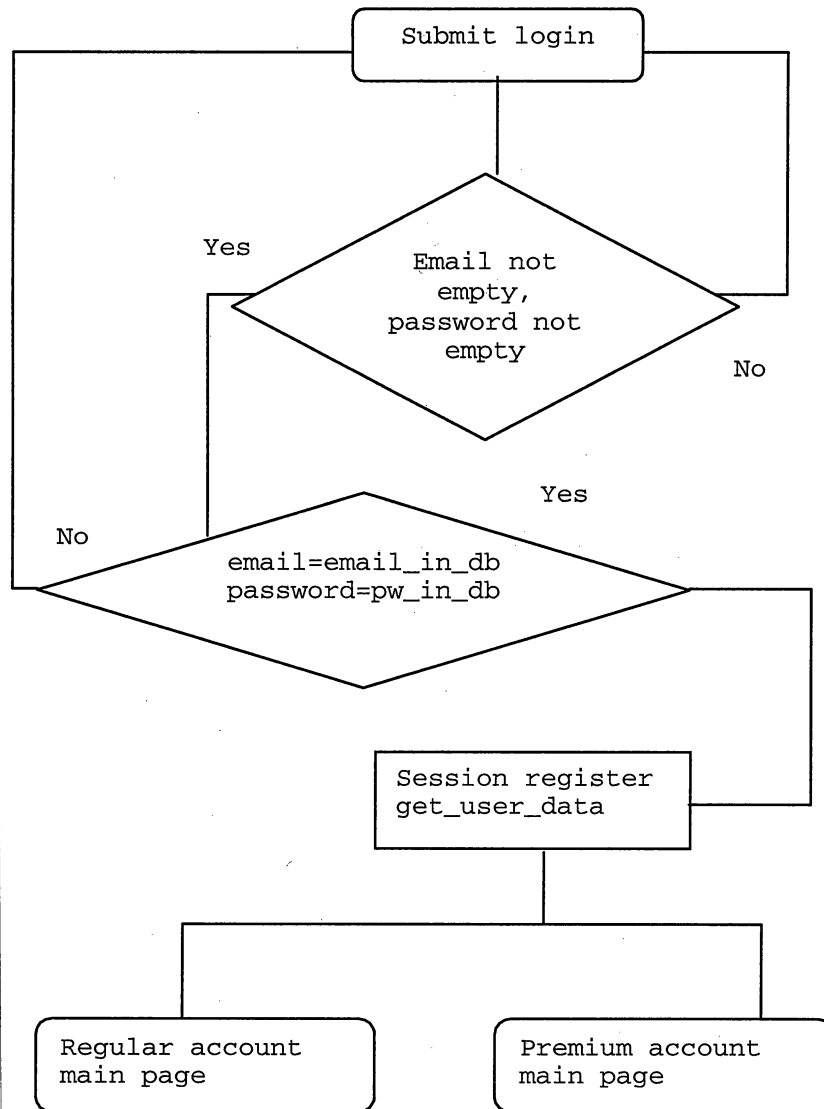


Figure 3. Login Diagram

1. First Login.php checks to see if the required fields of email and password are properly entered, and then it finds the corresponding user's account information by the email that the user input when login.

2. Once the user's password is verified, the VCOSTS server proceeds with the registration session by performing the `get_user_data` function. This will lead a user to his main page for either the regular account or the premium account user.

Logout

Figure 4 shows VCOSTS' logout procedure as follows: The `Logout.php` page, which includes the logout functions in VCOSTS, handles user activities such as `session_unregister` of `s_account_id`, `s_first_name`, and `s_last_name`. Once the above logout activities are completed, the VCOSTS server leads the user back to the main page in order to logout from the system.

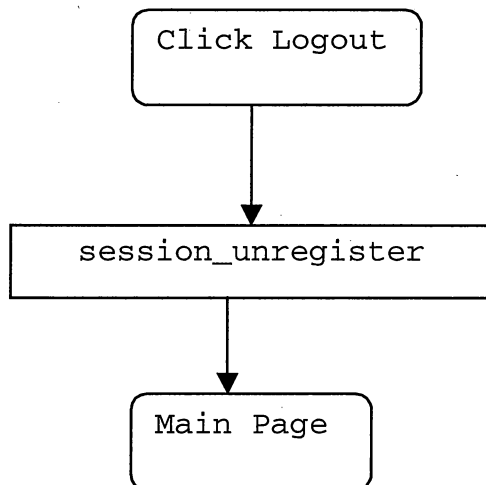


Figure 4. Logout Diagram

Submit Buy Stock Request

Figure 19 in Appendix A shows the process in the VCOSTS to complete a Buy Stock request once such request is submitted.

Once a user submits a buy Stock request, the function in the `buy_page.php` page will first check to see if all input fields of buy, symbol, quantity and price are valid. In addition, the system checks to see if there is enough balance in the user's account in order for him to place the order. Upon input verification, the function proceeds by storing the purchase request in the transaction table according to different expire times in one of the following five categories: (1) until market close today, (2) 1 (day + Ext.Hours), (3) 2 (day + Ext.Hours), and (4) 3 (day + Ext.Hours), (5) 1 week. For the Buy requests whose expiration time are "until market close today", the VCOSTS server will first check to see if the request was submitted during regular market opening time before the order information is stored into the VCOSTS database. If the market is already closed when the user submitted his buy request, and the user chooses the expire time of "until market close today," then VCOSTS will disqualify

the buy request, and send an error message that will require the user to re-enter a valid buy request.

Submit Sell Stock Request

Figure 20 in Appendix A shows the process used to sell stocks, which is similar to the Buy Request process described in above. The `sell_page.pgh` page verifies the validity of the buy, symbol, quantity and price fields, and then checks to see if there is enough stock balance in the trader's account in order for the sell stock request to take place. Once the input data are verified, the VCOSTS server enters the input data to sell stocks into the transaction table. Like those in the buy stock function, there are the same five expiration times for a trader to choose when he placed a sell stock order.

Functions for Account Transaction

This section describes other miscellaneous functions in the VCOSTS that control other transactions.

The function `checkBalance_beforeBuy($quantity, $price, $acct_id)` invalidates those people who do not have enough account balances when they place an order to buy stocks. For traders whose account balance come short of trading needs, the VCOSTS will trigger this function to

terminate an ongoing trading request and issue a warning message to the user for his invalid transaction request.

After the request to buy stock is completed, the `updateStock_afterBuy($symbol, $quantity, $acct_id)` function will update the stocks in the buyer's account by changing the amount of the stock if the buyer had the same type of stock before the transaction, or adding the new stock that he just bought into the VCOSTS database.

After a user submits his buy stock requests, VCOSTS checks for matching price to fill the request. Occasionally VCOSTS will not find a matching price due to the market price being higher than the offering price or there is no corresponding match price in the VCOSTS database before the user's specified expiration time, the VCOSTS' `returnBalance_ifnotBuy ($quantity, $price, $acct_id)` function will trigger the refund of the user's input data back to his account.

The `checkStock_beforeSell($symbol,$quantity,$acct_id)` function is designed to preclude a user who submits a sell stock request without valid number of stocks in his account. The function verifies if there are a sufficient number of stocks in the user's account balance before it

validates the transaction request to insert the sell stock request into the transaction table.

Once the sell stock request is validated, the `updateStock_afterSell($symbol,$quantity,$acct_id)` function updates the seller's stock in his account by changing his current holding in the VCOSTS database.

Similar to the `updateStock_afterSell` function, the `updateBalance_afterSell($quantity, $price,$acct_id)` function increases the user's account balance for the amount he receives from selling his stock holdings for any of his valid sell stock transactions.

Functions for Account View

VCOSTS also include the functions that allow a user to view his account balances. The command `get_stock_value($acct_id)` returns the real-time market value of a particular stock that is in a user's account.

The `get_account_value($acct_id)` function returns a user's holding value, which equals to the sum of cash balances and the total value of stocks in the user's account.

The `get_reward($acct_id)` command is only applicable for premium account users. It returns the current reward

based on the reward percentage the user has earned in stock transactions per his account.

The `get_stock_summary($acct_id)` function lists the information of the stocks that the user owns in his account, including symbol, original quantity, last price, day change, day high, day low, and value. This function lists the information of the stocks that the user owns in his/her account, which includes symbol, original quantity, last price, day change, day high, day low, and value.

The `get_regl_acc_summary($acct_id)` function provides an account summary for a regular user, which includes available cash, stock value, and account value.

Similar to the `get_regl_acc_summary` command in above, the `get_pre_acc_summary($acct_id)` function lists the account summary for a premium user. In addition to those accounts in a regular account such as available cash, stock value, and account value, the command also generates an additional reward percentage for the premium user.

The `changepasswd.php` page validates input fields in old password and new password, then it retrieves the old password from the VCOSTS database according to the `account_id`, which is the same as the registered session of `$s_account_id`. After the above password verification

process finishes correctly, VCOSTS updates the new password in the trader table.

Stock Transaction

Stock trading is implemented by timely updates in the VCOSTS' transaction table. Depending on trading time, VCOSTS implements the stock trading in either business-hour trading or extended-hour auction.

Trading during Market Time

Figure 21 in Appendix A shows an example of trading when it takes place during the regular stock market opening time. In such cases, all trading transactions are processed in the market_trading.php page as follows:

1. First, the VCOSTS function marks all transactions to be obsolete for all pending transactions in the VCOSTS database whose transaction time has expired (i.e. the end_time field is equal to or earlier than the current_time field.)

2. Second, for all remaining pending transactions in the VCOSTS database, the system looks for valid transactions to proceed. In these cases, all pending transactions whose buy_price field is equal to or higher than their market_price field, or whose sell_price field

is equal to or lower than their `market_price` field will be validated to complete the transactions.

3. When VCOSTS validates a transaction, it will update the information in the transaction table by setting the `transaction_time` as the `current_time` and by setting the transaction status as completed. Furthermore, VCOSTS updates such transaction information in the account table for each user whose transactions have been completed.

4. Finally, for all completed transactions, the VCOSTS server sends out an email message to notify the traders of the completion of the stock trading, as well as to provide relevant confirmation for the users to keep the transaction records.

Trading during Off-Market Time

Figure 22 in Appendix A shows the process in VCOSTS to handle off-market time stock trading. In VCOSTS, all off-market stock trading is done via an auction process among all system users as follows:

First, VCOSTS updates all expired transactions by setting their status as obsolete.

Secondly, the VCOSTS server creates a temporary `hold_buy` table and a `hold_sell` table to keep off-hour

trading requests in ascending order based on offered trading prices.

Then the VCOSTS server identifies the matched prices from the hold_buy and hold_sell tables, and obtains the earliest trading time for those matched trading prices. The price match can occur in one of the following three possible situations:

In the first case, if the quantity in the sell request equals that in the buy request, then VCOSTS updates the transaction table in the buy deal and the sell deal fields. It then updates the account_stock table for the buyer and for the seller, followed by an update of the cash_balance in the seller's account table. Finally the system sends out an email confirmation to the traders concerning the completion of the auction. Secondly, if the quantity in the buy request is higher than that in the sell request, then VCOSTS server first updates the buyer's transaction by setting up the attributes as `transaction_time >= NOW(); status = 'completed' and stock_quantity = '$sell_quantity'`.

In addition to the above settings, VCOSTS further updates the following data in various data tables: it first updates the transaction table with the transactions

to buy and to sell; secondly, it updates the account_stock table for the buyer and the seller to reflect on the stock trade. Thirdly, it updates the cash_balance in the account table for the seller. Finally, it sends out email confirmations to both the buyer and the seller for the completion of the trade.

If the quantity in the buy request is higher than that in the sell request, then the VCOSTS updates the transaction table for the sell request by setting the stock_quantity as the buy_quantity and inserts in the transaction table the rest of the sell trade. Similarly, it updates the transaction table for the buy deal, updates the account_stock table both for the buyer and the seller, updates the cash_balance in the account table for the seller, and finally emails the confirmation message to the trader.

Administrator Functionality

VCOSTS contains functions that allow system administrators to manage transactions and premium users' accounts online. These functions include view transactions, stop transactions, view premium users

rewards, and activate premium user account upon receiving the paid premium account membership fees.

Backend Cron Jobs

Under the backend of "c2costs.csci.csusb.edu", online trading and auction commands are run automatically using cron daemon at the specified date and time.

Regularly scheduled commands can be specified according to instructions contained in the crontab files. The cron daemon runs commands according to the crontab file entries.

The cron daemon examines crontab files only when the cron daemon is initialized. When the changes are made to the crontab file using the crontab command, a message indicating the change is sent to the cron daemon. A crontab file contains entries for each cron job. Newline characters separate entries. Each crontab file entry contains six fields separated by spaces or tabs in the following form:

minute hour day_of_month month weekday command

Eight cron jobs are defined at the backend to implement C2C online stock trading and auction system.

Table 2. Project Cron Jobs

```
0,20,40 9-15 * * 1-5 php
/home/ztao/back_app/market_trading.php
30,45 8 * * 1-5 php
/home/ztao/back_app/market_trading.php 0,20,40 * * *
6,0 php /home/ztao/back_app/auction_trading.php
0,20,40 16-23 * * 1-5 php
/home/ztao/back_app/auction_trading.php
0,20 0-8 * * 1-5 php
/home/ztao/back_app/auction_trading.php
40 0-7 * * 1-5 php
/home/ztao/back_app/auction_trading.php
5,25,45 9-15 * * 1-5 php /home/ztao/syupdate.php
45 8 * * 1-5 php /home/ztao/syupdate.php
```

To run the stock auction script at every 0,20,40 minutes, every Saturday, and Sunday, enter:

```
0,20,40 * * * 6,0 php
/home/ztao/back_app/auction_trading.php
```

CHAPTER SIX

TESTING

In order to implement VCOSTS functionality completely, many tests are performed in order to ensure VCOSTS works properly with normal business hour trading and extended hour auction. Refer to Table 3 for the reports of the test cases.

System Validation

The objective of system validation is to enable a smooth and quick integration of MySQL, PHP, and Linux-based system. The VCOSTS design that is part of the validation process include:

- Red Hat Linux and Apache web server setup
- PHP and MySQL Modules
- VCOSTS Principles

Validation is performed on Red Hat Linux using a few samples only. The validation does, however, provides a high level of confidence that a particular password generator, stock business-hour trading and extended-hour auction, cron jobs, etc. meets the specification and therefore can be successfully implemented in a system.

The validation process uses standardized procedures and methodologies to verify that the system adheres to the specification. VCOSTS initial development involves a lot of system validation work.

Unit Testing

A unit test is a test that the "unit", usually a single program, works properly. A unit test is very different from a system or functional test; these latter types of test are oriented to application or overall system testing. The system testing can be performed effectively until individual scripts or programs behave as expected.

During the VCOSTS development, the author completes a lot of unit testing and has a correspondingly high level of confidence in the whole system. The tests are the means to improve the quality of the codes.

Initially, some simple unit tests are performed based on VCOSTS internal requirements such as trader registration and login modules. Then, an approach of unit testing is developed for business-hour trading and extended-hour auction. It is very important to have a better understanding about what these test needs to be

implemented. On the other hand, the data input and the expected results from running the programs should be determined, while a various of values under different unit testing conditions are very helpful to build the separate test cases for VCOSTS programs.

It is very important to understand what the program is supposed to do and how it generates accurate results. It is also important to isolate the tests to address difficult functions, i.e., buy and sell stock, stock trading and auction modules. The tests written run automatically and provide clear indicators of success or failure. When a bug is detected, a test case is written to verify the bug. After then, analyze the code, identify the reason and fix it.

Integration Test

Integration testing is one of the important steps to ensure that VCOSTS is able to function properly. What it needs to perform is hardware, operating system, web server and development tools.

As far as VCOSTS is concerned, many functions and modules must work together and interact seamlessly. This requires not only that the functions and modules work as

they were designed, but also they interact in a consistent and correct way. A robust series of tests and procedures have been designed to ensure that the system functions correctly, and handles user errors in a helpful and understandable manner.

In order to proceed in a logical manner in the integration, each of the modules has been integrated and tested. Then, the higher-level modules are performed.

Testing is performed by re-doing all of the unit tests for all of the functions and modules included in the VCOSTS integrated functions and modules.

Once all of the modules in both normal-hour trading and extended-hour auction systems have been tested, we need to integrate both of the systems with their user interfaces. All of the interfaces have been thoroughly tested to ensure that they work correctly and independently of the system back-end. The integration consists of running as many of the unit tests as possible to perform all of the interface tests to ensure that there is no loss in interface functionality.

System Testing

The total system between the business-hour trading and extended-hour auction systems is tested to ensure that the system as a whole will work as expected. Because there are a number of parameters between the systems and the functions, which produce the expected output and process the input, have been tested that the VCOSTS is working well. This will require several specific tests detailed below.

- Automatic run of business-hour trading script for investors.
- Automatic run of extended-hour auction script between investors.
- Automatic update of stock price during business hours.
- Automatic processing of sell and buy transaction if they meet the requirements.

Administrator tests are performed in the following items:

- Change the password for the trader in VCOSTS system.
- Navigate the transaction, and remove unnecessary transaction records.

Trader tests are completed by the following functions:

- Register as a new trader.
- Change the trader's password.
- View the trading history and balance.

Table 3. Test Case Reports

Test Name/Number	Online Stock Data 1
Test Objective	To test the functionality of catching online stock data
Test Description	Define stock symbols. Then compare output with the expected results.
Test Condition	X=ORCL, Y=SUNW, Z=RHAT
Expected Results	last_trade_price(X)=15.42 day_high(X)=16.78 day_low(X)=15.36 last_trade_price(Y)=10.53 day_high(Y)=10.69 day_low(Y)=10.25 last_trade_price(Z)=6.42 day_high(Z)=6.89 day_low(Z)=6.37
Actual Results	last_trade_price(X)=15.42 day_high(X)=16.78 day_low(X)=15.36 last_trade_price(Y)=10.53 day_high(Y)=10.69 day_low(Y)=10.25 last_trade_price(Z)=6.42 day_high(Z)=6.89 day_low(Z)=6.37
Test Name/Number	Online Stock Data 2
Test Objective	To test the update of stock table at C2Costs database using online stock data
Test Description	Define stock symbols. Then compare output with the expected results.

Test Condition	X=ORCL, Y=SUNW
Expected Results	last_trade_price(X)=15.43 day_high(X)=16.77 day_low(X)=15.21 day_change=4.89% last_trade_price(Y)=10.68 day_high(Y)=10.69 day_low(Y)=10.25 day_change=-2.14% last_trade_price(Z)=6.52 day_high(Z)=6.89 day_low(Z)=6.27 day_change=0.76%
Actual Results	last_trade_price(X)=15.43 day_high(X)=16.77 day_low(X)=15.21 day_change=4.89% last_trade_price(Y)=10.68 day_high(Y)=10.69 day_low(Y)=10.25 day_change=-2.14% last_trade_price(Z)=6.52 day_high(Z)=6.89 day_low(Z)=6.27 day_change=0.76%
Test Name/Number	Online Stock Data 3
Test Objective	To test the automatic update of stock table at C2Costs database using online stock data with backend cron jobs, which are running every twenty minutes during the business trading hours.
Test Description	Define stock symbols and cron job. Observe two subset results. Then compare output with the expected results.
Test Condition	X=ORCL, Y=SUNW
Expected Results	last_trade_price(X,20min)=15.39 day_high(X,20min)=16.77 day_low(X,20min)=15.20 day_change(X,20min)=3.89% last_trade_price(Y,20min)=10.58 day_high(Y,20min)=10.71 day_low(Y,20min)=10.23

		day_change(Y,20min)=-1.14% last_trade_price(X,40min)=16.00 day_high(X,40min)=16.77 day_low(X,40min)=15.20 day_change(X,40min)=7.89% last_trade_price(Y,40min)=10.48 day_high(Y,40min)=10.71 day_low(Y,40min)=10.23 day_change(Y,40min)=-3.14%
Actual Results		last_trade_price(X,20min)=15.39 day_high(X,20min)=16.77 day_low(X,20min)=15.20 day_change(X,20min)=3.89% last_trade_price(Y,20min)=10.58 day_high(Y,20min)=10.71 day_low(Y,20min)=10.23 day_change(Y,20min)=-1.14% last_trade_price(X,40min)=16.00 day_high(X,40min)=16.77 day_low(X,40min)=15.20 day_change(X,40min)=7.89% last_trade_price(Y,40min)=10.48 day_high(Y,40min)=10.71 day_low(Y,40min)=10.23 day_change(Y,40min)=-3.14%
Actual Results		last_trade_price(X)=15.43 day_high(X)=16.77 day_low(X)=15.21 day_change=4.89% last_trade_price(Y)=10.68 day_high(Y)=10.69 day_low(Y)=10.25 day_change=-2.14% last_trade_price(Z)=6.52 day_high(Z)=6.89 day_low(Z)=6.27 day_change=0.76%
Test Name/Number		Normal trading 1
Test Objective		To test the functionality of business-hour stock buying
Test Description		Login as tao1990@hotmail.com and submit buy stock request. Then compare output with the expected results.

Test Condition	Symbol: X=ORCL Quatity:Y=200 shares Buy price=16.00 Time:Until market close today
Expected Results	Since current stock price is \$15.42, the buy request should be successful.
Actual Results	The user succeeds in this transaction. The 200 shares of ORCL are added into user account, and \$3,200.00 is withdrawn from user account.
Test Name/Number	Normal trading 2
Test Objective	To test the functionality of business-hour stock selling
Test Description	Login as tao1990@hotmail.com and submit sell stock request. Then compare output with the expected results.
Test Condition	Symbol: X=SUNW Quatity:Y=500 shares Buy price=7.00 Time:Until market close today
Expected Results	Since current stock price is \$7.21, the sell request should be successful.
Actual Results	The user succeeds in this transaction. The 500 shares of SUNW are removed from user account, and \$3,500.00 is deposited into user account.
Test Name/Number	Normal trading 3
Test Objective	To test the functionality of business-hour stock selling
Test Description	Login as tao1990@hotmail.com and submit sell stock request. Then compare output with the expected results.
Test Condition	Symbol: X=ORCL Quatity:Y=200 shares Sell price=20.00 Time:Until market close today
Expected Results	Since the stock price is never hit \$20 after the user submit this sell

	request, this transaction does not occur at all.
Actual Results	The user still has 200 shares of ORCL in his account until market closes.
Test Name/Number	Extended trading 1
Test Objective	To test the functionality of extended hour stock buying
Test Description	Login as tao1990@hotmail.com and submit buy stock request. Then compare output with the expected results.
Test Condition	Symbol: X=CSCO Quatity:Y=400 shares Buy price=13.25 Start Time=6 AM on Saturday Time:2 days+Ext hours
Expected Results	It will depend whether there is other VCOSTS user want to sell the price not more than \$13.25. On Sunday, the other user submits the sell request for 400 shares of CSCO at \$13.25. The buy request should be successful.
Actual Results	The user succeeds in this transaction once there is a match in the sell request. The 400 shares of CSCO are added into user account, and \$5,300 is withdrawn from user account.
Test Name/Number	Extended trading 2
Test Objective	To test the functionality of extended-hour stock selling
Test Description	Login as tao1990@hotmail.com and submit sell stock request. Then compare output with the expected results.
Test Condition	Symbol: X=CSCO Quatity:Y=400 shares Sell price=15.00 Start Time=8 AM on Tuesday Time:2 days+Ext hours
Expected Results	Since there is no last trade price of the normal hour trading and buy

		price during the extended hour auction are not less than \$15, this sell request can not be successful.
Actual Results		The user still holds 400 shares of CSCO in his account. No transaction occurs.

CHAPTER SEVEN

USER MANUAL

Manual Overview

VCOSTS offers a most realistic simulation of the stock market on the Internet. It is an online investment simulation game for trading stocks of NASDAQ. It won't cost you a penny to play with real-time virtual stock with your regular account, and you could win one of our excellent cash prizes if you are the premium member. Share prices are updated every 20 minutes so that you can watch the value of your portfolio change throughout the day. Please see Figures 23-38 in Appendix A for detail.

Registration

Users have to register for an account before starting to trade stock. After entering register page, you can choose regular or premium account according your interest and financial ability. Regular account registration only requires some general information of users, and doesn't ask for social security number and date of birth, but premium member's registration does strictly due to the real money involved, such as annual fee and prize.

Note that the email address that you supply to register for either regular or premium membership is a valid one, to which the system will send an URL including confirmation code. Before new users first login, they have to access those specific URLs, which will pass confirmation codes to system, to login into their accounts. After that, the system will update the database of users' accounts so that the status of accounts will become active. Thus users can login from login page freely in the future. This feature is designed to avoid those people who are unserious about applying for accounts and those who will never be interested in entering their accounts with our system.

Please be sure to fill out email address, password, password re-enter during registration for both types, and social security number and date of birth for premium application. If you miss one of them, the system will show you an error message to notify you to finish those required fields, and won't let you register until you supply them all. On the other hand, if you try to use an old email address, which is already in our system when you applied last time, the system will also prevent you from doing so and show an error message. Remember that the

email address must be unique in our system, which is used as login name.

VCOSTS has the function to check the input format of email address field and if the email that you just input is an existing one. Please be careful to enter correct information, otherwise it will notify you by showing error message.

Note that the premium account will be available for trading only when the VCOSTS administrator receives the membership fee after your online registration.

Confirm Login

After you receive the confirmation E-mail sent by the VCOSTS system after registration, you have to access this page by click or input the URL in the email, which includes the confirmation md5 code generated by the system. There are two text fields, which let the user input the E-mail address and password, and a submit button for sending the login information. When the user successfully submits the E-mail address and password, the confirmation code will be passed too. After being verified, the user will obtain the first page of his/her account. The user only needs to do the confirmation login

at the first time. After that, registered VCOSTS users can enter their user name and password at any time through the normal login page to access their personal accounts.

Login

There is a login page, which has text fields for E-mail and password, login button, and reset button. There is also a link, "If you have forgotten your password, you can request that and we'll E-mail to you", for users who have login problem, like having forgotten their passwords. By typing the correct E-mail and password for login, the user can enter the main page of his/her account.

Logout

This function is only available when you are already in your account. By clicking the logout button in the right of the header, you can logout of your account at any time. In this case, you will have your session unregistered by VCOSTS automatically, and you will return to the main page of VCOSTS.

Buy Stock

By clicking the "Buy Stock" button, you will see a buy page for you to send buy stock request. On this page,

there is a stock summary to view the stocks that you own, four text fields of symbol, quantity, price and expiration time, submit button, and reset button.

There are five choices for expiration time: until market close today, one day, two days, three days, and a week. Note that you should submit the request during market time if you choose the expired time of "until market close today", otherwise the system will give you a warning and won't let you trade.

In order to buy the stock that you want, you have to be sure that the balance in your account is sufficient. When you submit your order, the VCOSTS system will check if there is money for this order, if not, you will get a warning message and you can resubmit it when you have sufficient funds. After you successfully submit the request, you will get a confirm action notice and return to the main page of your account.

When the system accepts your buy request, it will hold the balance of your account for the buy order, and wait on the buy order as long as the status is pending in the VCOSTS database. The system keeps checking if the market price is equal to or lower than the one that you

provide when the market opens, or waits for a matched price of a sell order during off-market time.

If your buy request matches with the market price during business hours, VCOSTS will make the deal for you, and update your account with the stock amount and appropriate balance. During off-business hours, the system treats all deals as auction, and tries to find the earliest matched price for each deal. If the price of buy order is more than that of the earliest sell order, the system will successfully make this deal, and update both buyer and seller's balance and stock in their accounts. Even with no sufficient sell order, the system still remains the additional buy order, and waits for the next match until this transaction's ending time expires. It is analogous when the price of sell order is more than that of buy order.

Sell Stock

Click the "Sell Stock" button in your account main page, and you will be led to the stock sell page. It is similar with the buy stock procedure. However, the difference between buy and sell request is that the system

will verify if you have enough stocks in the account for sale when you submit the sell request.

Option - Change Password

After you have already logged into your own account, you will find an option button, which leads you to the page to change your password.

When accessing this page, you must input the old password and new password to process the password change. Be sure that those three fields are not empty, and the two fields of new password input and re-input have the same contents. The system will give you warning messages when you type old password wrong or two new passwords different.

Forget Password

If you happen to forget your password, there is a link on login page for those who can't remember their passwords. You just need to click it, and input your email address, which you use when you registered. The VCOSTS system will send you an instant E-mail, which contains your password.

Account Summary

You will visit the main page of your account immediately after login. There is an account summary on the main page. By viewing the account summary, you can get information about your account, such as available cash, stock value, and account value. For premium users, there are also earned rewards shown in their account summaries.

Stock Summary

On the user main page, there is a stock summary under account's summary. The stock summary includes all the necessary information about the stocks that the user owns, such as symbol, original quantity, last trade price, day change, day high, day low, and current value.

Administration Tools

The administration utility can let the administrator view and manage premium user's reward and all transactions. The tool includes four functions:

Reward Active

After receiving a premium trader's membership fee, change his account information to allow him earn the reward.

View Reward

Find the information of all the traders who registered premium accounts including the people whose membership fee is not ready yet.

View Transaction

View all the transactions that the system handles including obsolete, completed, and pending ones.

Stop Transaction

Change the current pending transaction to the status of obsolete if needed.

Utility Pages

Introduction Page

It is a quick introduction for VCOSTS, including general information like who can play, when do you play, and how do you get started.

Rules Page

It contains the general rules for VCOSTS users, regarding start balance, stock type, security regulation, and etc.

Manual Page

The page of user manual provides the detailed guide on how to utilize the VCOSTS system. The guide covers all

the functions which consist of overview, registration, confirm login, normal login, logout, buy stock, sell stock, change password after login, what to do with forgotten passwords, account summary, stock summary, utility pages, administration tools, and etc.

Education Page

This page tutors traders by supplying several of the hottest books on the market and twenty golden rules for stock trading. These strategies will be sure to benefit traders in their investment life.

FAQ Page

The page of frequently asked questions is addressed from four aspects, System Requirements, Playing the Game, Dealing, and Entry Requirements.

Contact Information

This page provides the detailed information on how to get in touch with VCOSTS.

Clock and Calculator

Using JavaScript, VCOSTS also supplies clock and calculator functions on the web page. By viewing the clock on the upper-right of each page, user obtains the current time so that he/she can figure out if it is in market

hours, which is always useful for you to choose the expiration time. For example, if you select "until market close today" as the expiration time when you don't notice that it is already at 5:00 pm on Monday, the system will not accept the request. The calculator can be used only after users login. It is a useful tool for users to manage their accounts by calculating their stocks and balance.

CHAPTER EIGHT

CONCLUSIONS AND FUTURE DEVELOPMENT

This project focuses on the stock simulation tool, VCOSTS (Virtual C2C Online Stock Trading System), which is developed by open-source applications such as MySQL database, Apache web server, and PHP language.

While potential investors can always risk trading stocks without much prior exposure and with "life ammunition" (i.e. real money) that cause them hefty financial losses, VCOSTS allows them a viable alternative to practice trading in a realistic setting that will not endanger their financial resources before they gain a solid understanding in stock trading. Using VCOSTS, a user can get real-time stock prices and trade stocks according to timely market prices. In addition to regular trading hours, VCOSTS has extended-hour services, which allows people to auction their stocks during off-business hours, weekends, and holidays.

The basic operation involved in VCOSTS is quite straightforward. There are two types of accounts for a user to choose from - a regular accounts or a premium

account. Both accounts start with the same \$100,000 virtual cash that the system assigns to a registered user. For regular account users, the registered trader simply trades for fun, and there is no monetary arrangement involved. On the other hand, a premium account user needs to pay an annual membership fee of \$100 via an on-line secured registration process. After registration, the premium account user also trades stock in the market, but gets to keep 0.2 percent of the profits generated from stock trading.

PHP4 session handling is adopted for effectively tracking visitors on the site and managing their information. Object-oriented concepts and adequate security measures such as MD5 message-digest algorithm are both adopted during the project design stage. The complex software development process for VCOSTS is tested completely, and provides to be very useful tool for the stock investors. For instance, md5 is used here as a function to encrypt the customer password and saved in c2costs database. Every time a trader wants to access his/her account with user name and password, the system will authenticate him by comparing his/her password's md5 hash with the encrypted password in the database.

In addition to using VCOSTS as a training tool, with its high performance, VCOSTS could also be extended to handle real stock transactions in the future. It may also be used to supplement off-business hours stock transactions.

This project has achieved the set goal, which utilizes the open-source applications to develop VCOSTS site. Based on the currently complete VCOSTS development, there are several considerations for the future development:

1. Because of the physical limitation, the real credit card payment for stocks sell/buy behaviors could not be implemented in VCOSTS now. This feature will be very helpful for the VCOSTS production systems with venture capital funding.
2. The tax calculation for trader's gain in VCOSTS stock sell will be considered based on the user's state and county requirements.

APPENDIX A:
PROJECT FIGURES

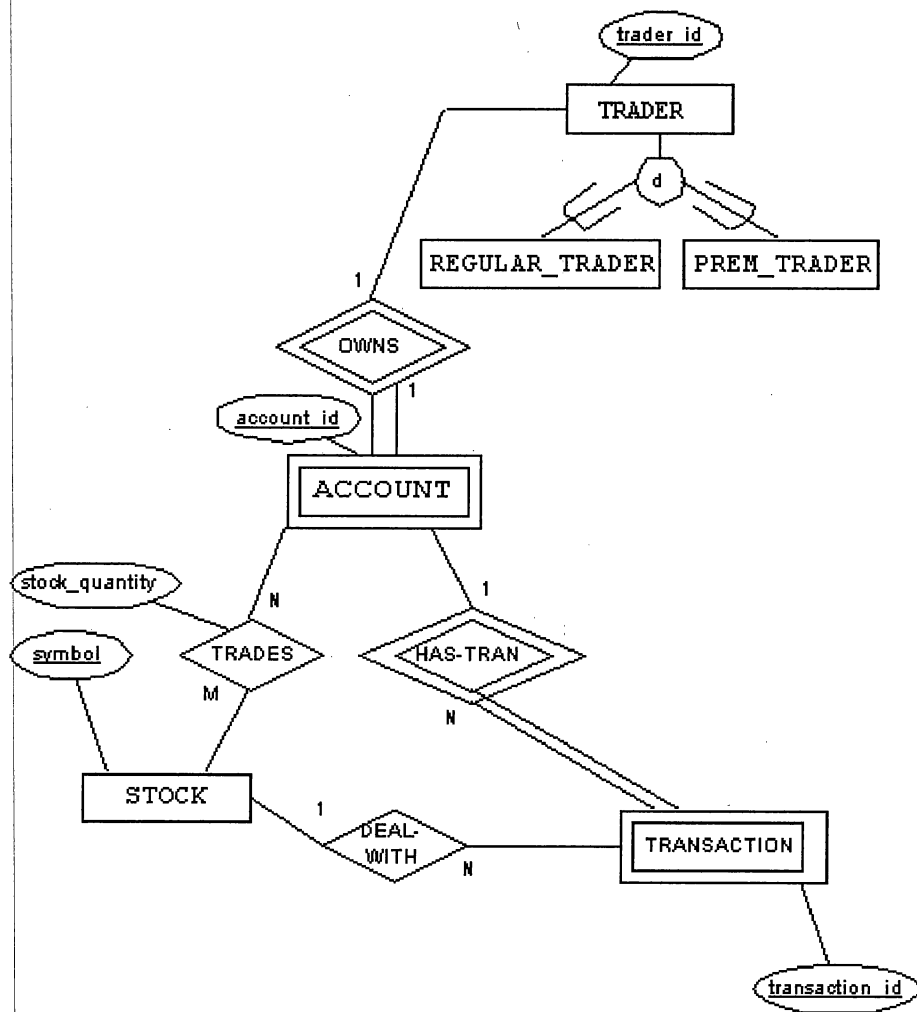


Figure 5. Database Entity-Relationship Diagram

TRADER
trader_id (PK)
name
fname
middle_initial
lname
address
addr1
addr2
city
state
phone
home_phone
work_phone
cell_phone
fax
email
password
confirm_hash
active

Figure 6. The TRADER Entity Type Using Object Model

PREM_TRADER
SSN (PK)
birthdate

Figure 7. The PREM_TRADER Entity Type
Using Object Model

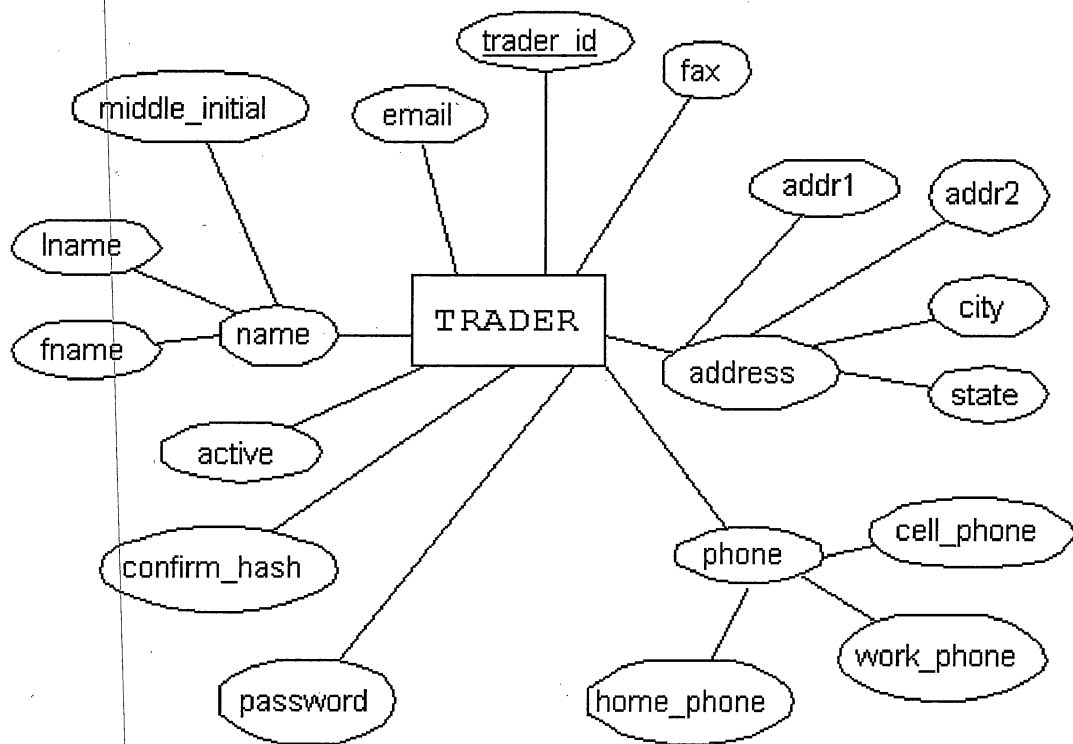


Figure 8. The TRADER Entity Type Using ER Model

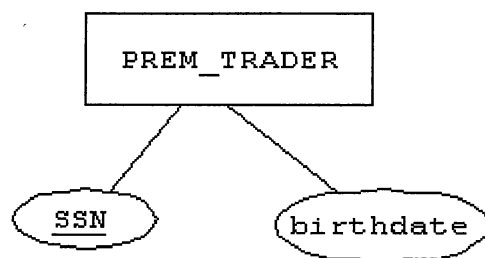


Figure 9. The PREM_TRADER Entity Type Using ER Model

TRANSACTION
transaction_id(PK)
transaction_type{buy,sell}
stock_quantity
trader_price
start_time
end_time
transaction_time
status{pending,completed,obsolete}
trans_time_type{day,extended hour}

Figure 10. The TRANSACTION Entity Type
Using Object Model

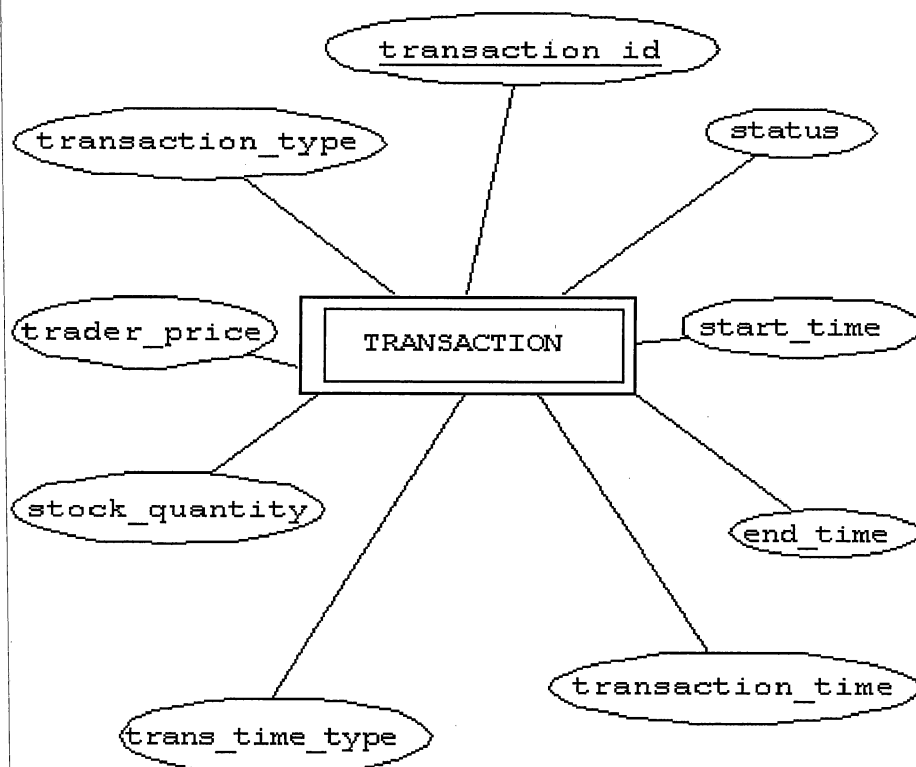


Figure 11. The TRANSACTION Entity Type
Using ER Model

ACCOUNT
account_id(PK)
account_type{regular,premium}
original_value
reward_pct
/cash_balance

Figure 12. The ACCOUNT Entity Type Using Object Model

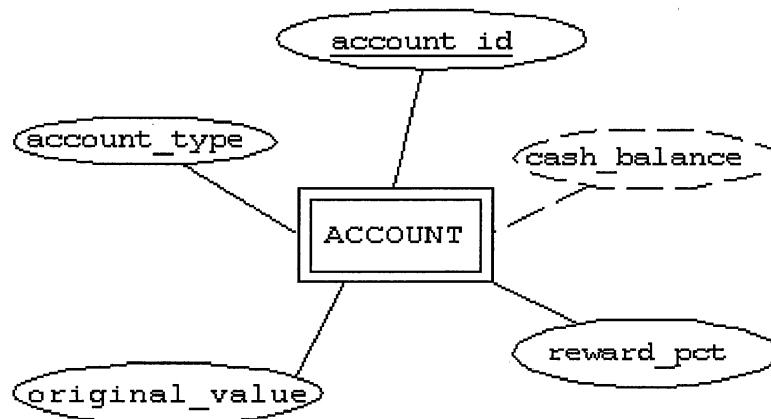


Figure 13. The ACCOUNT Entity Type Using ER Model

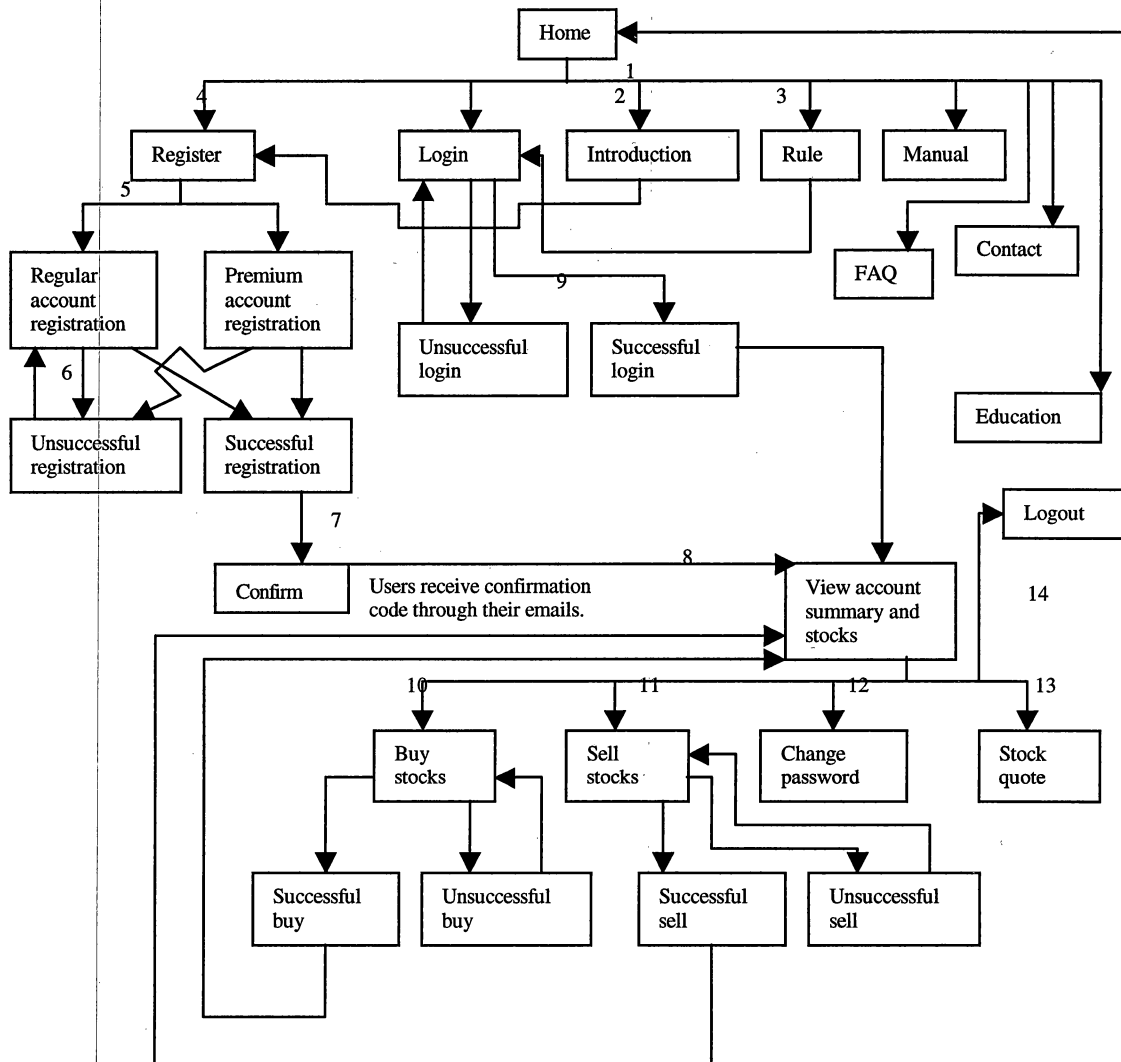


Figure 16. Project Work Flow Diagram

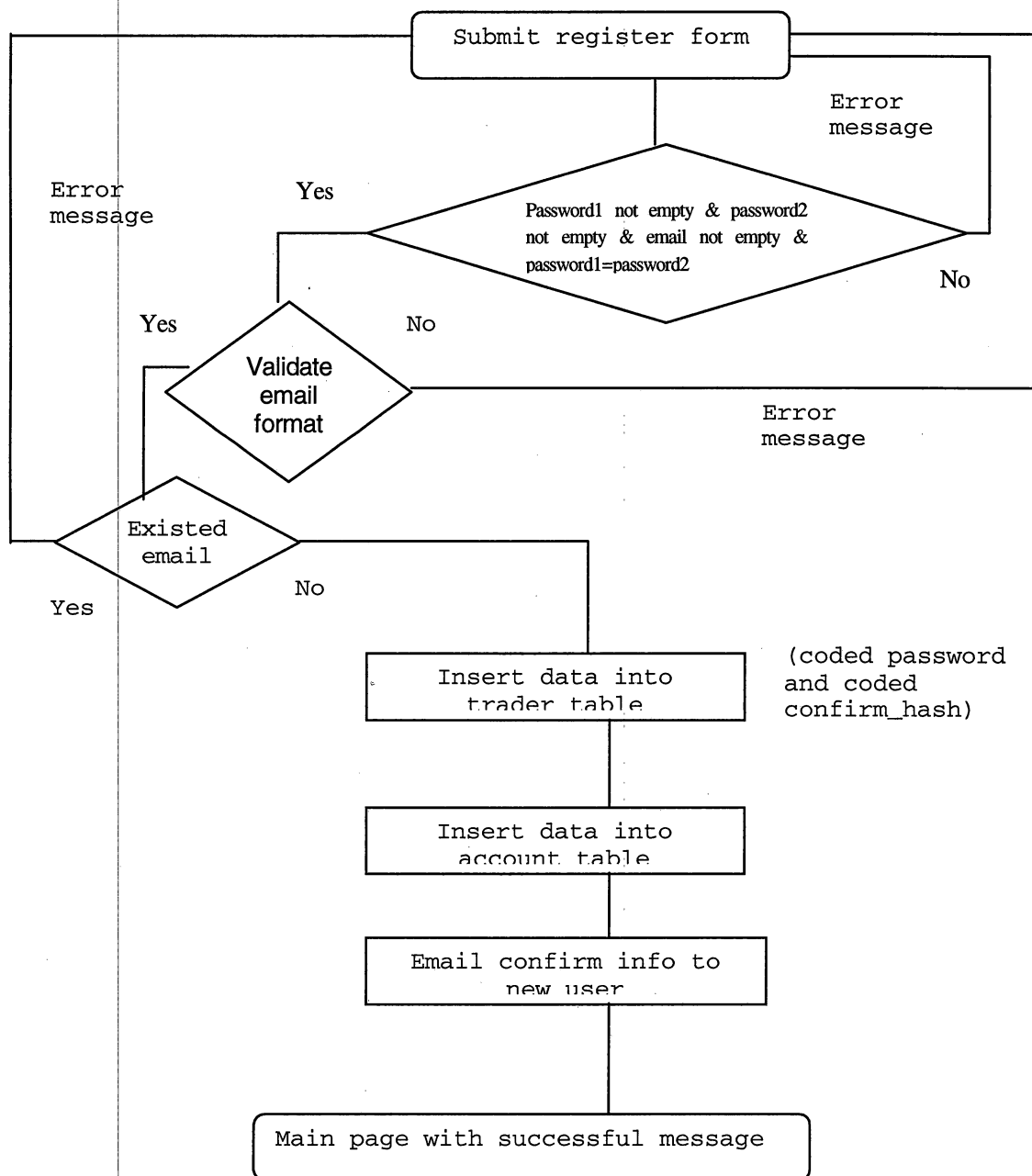


Figure 17. Register Diagram

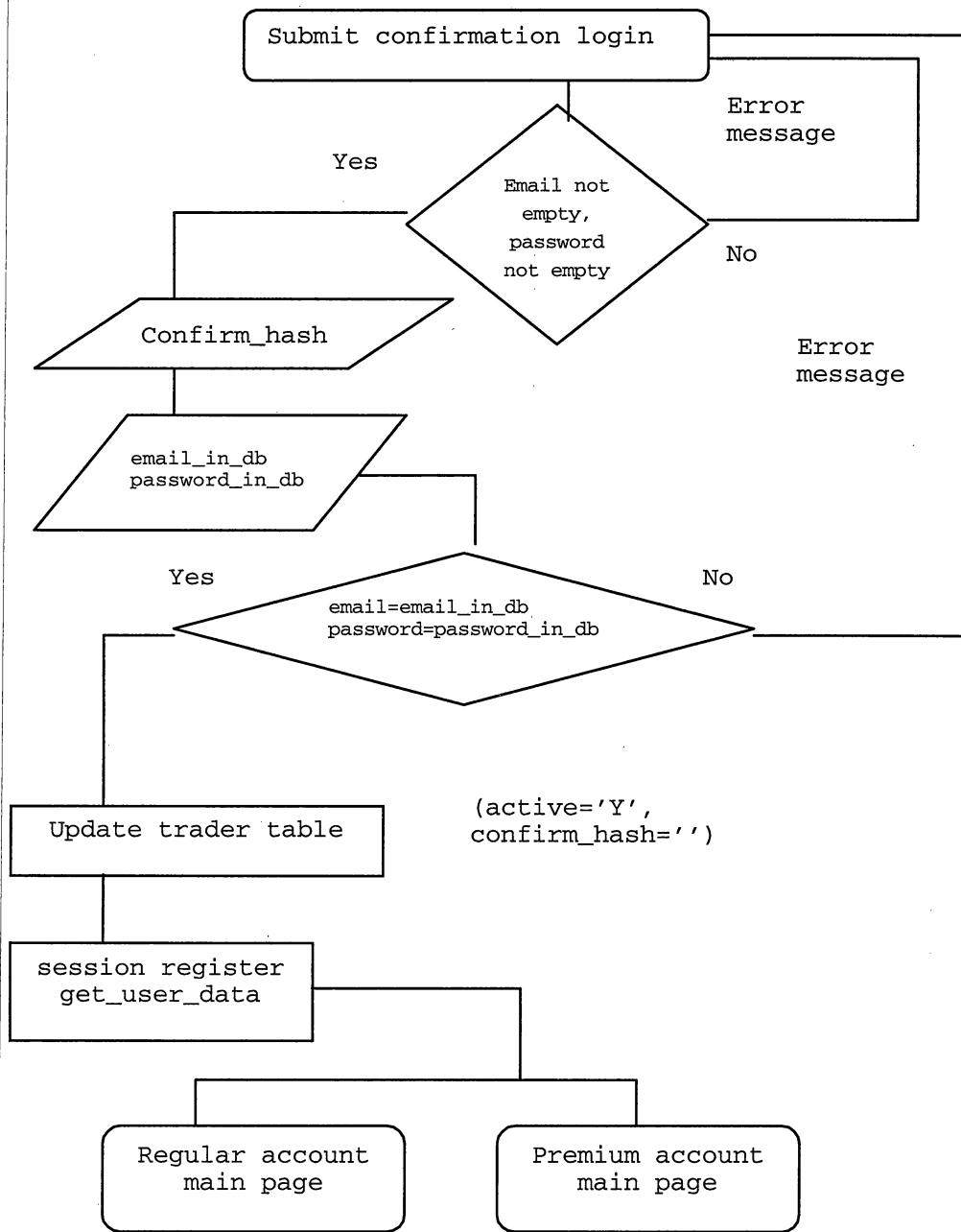


Figure 18. Confirm Diagram

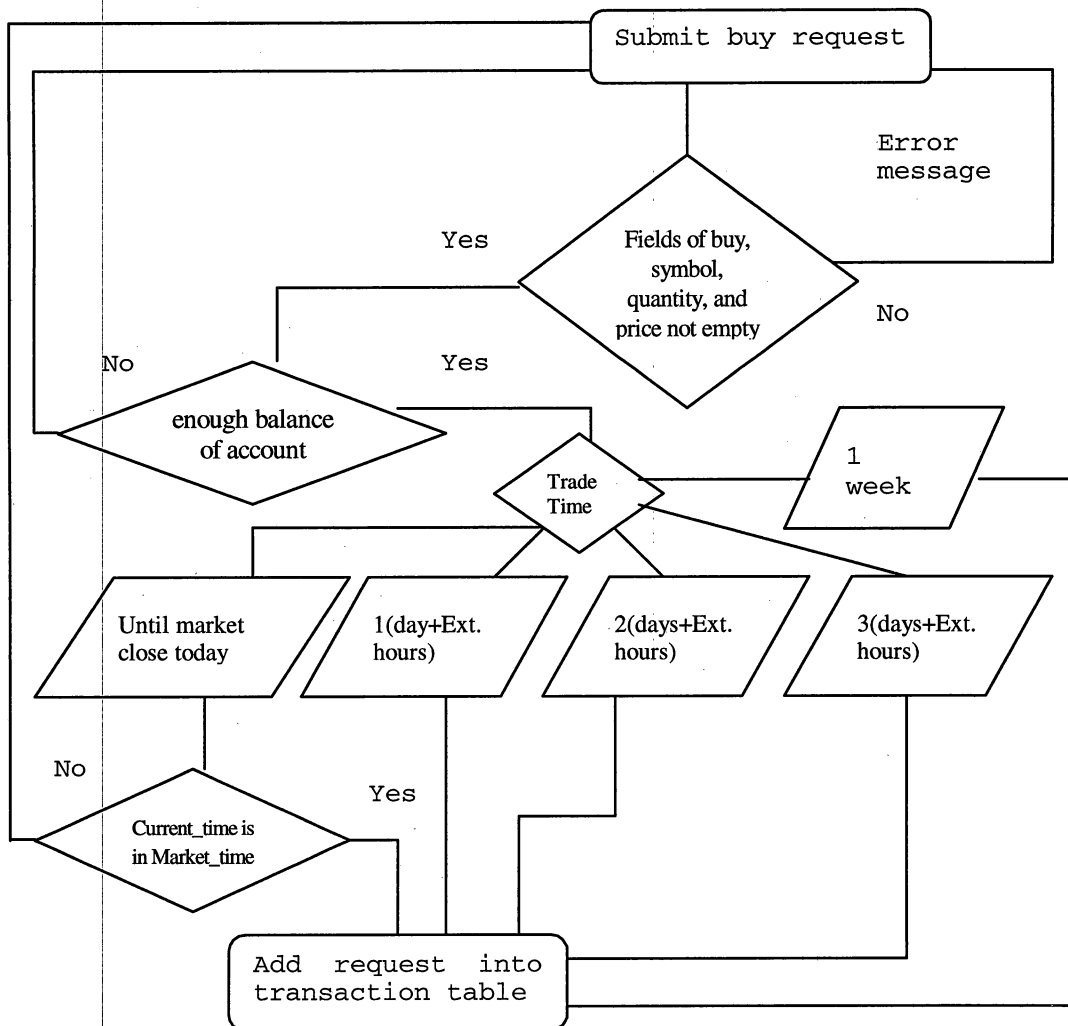


Figure 19. Buy Stock Diagram

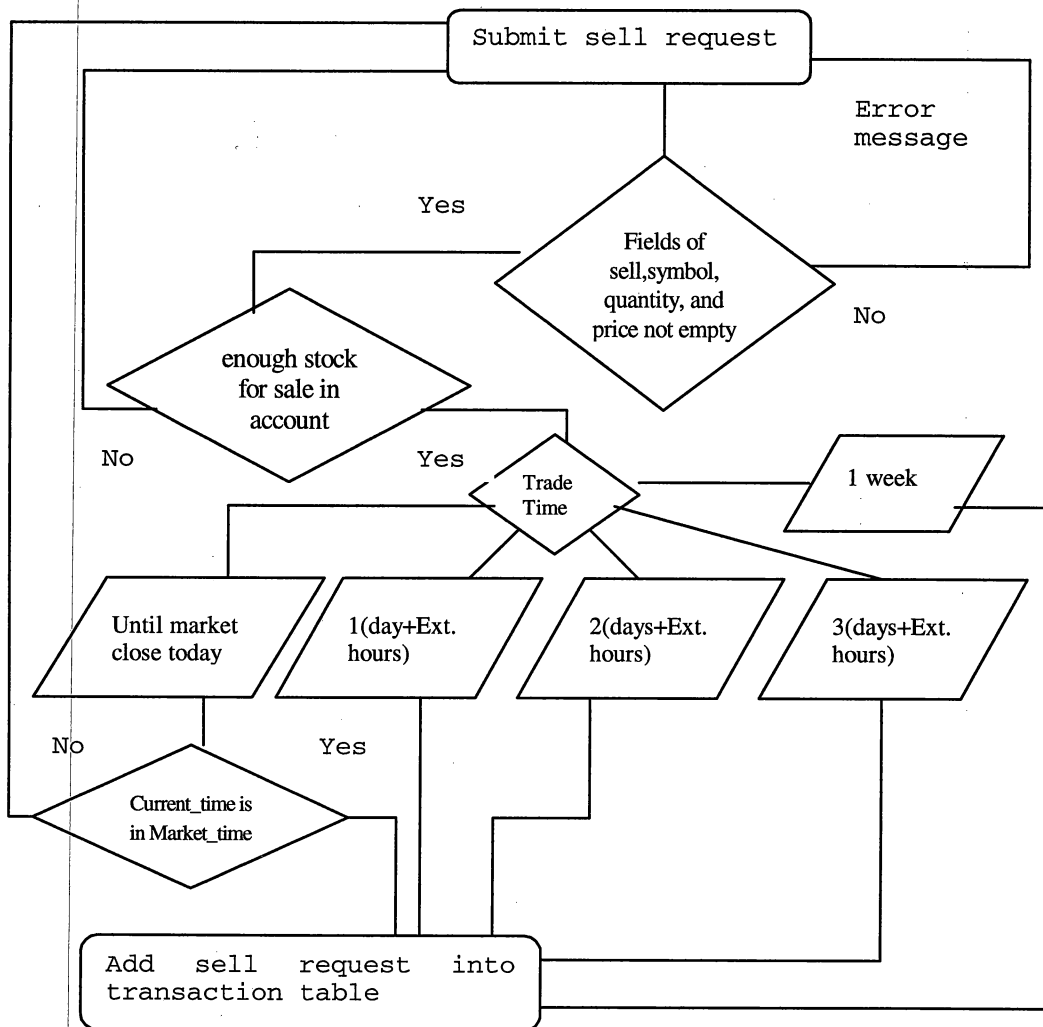


Figure 20. Sell Stock Diagram

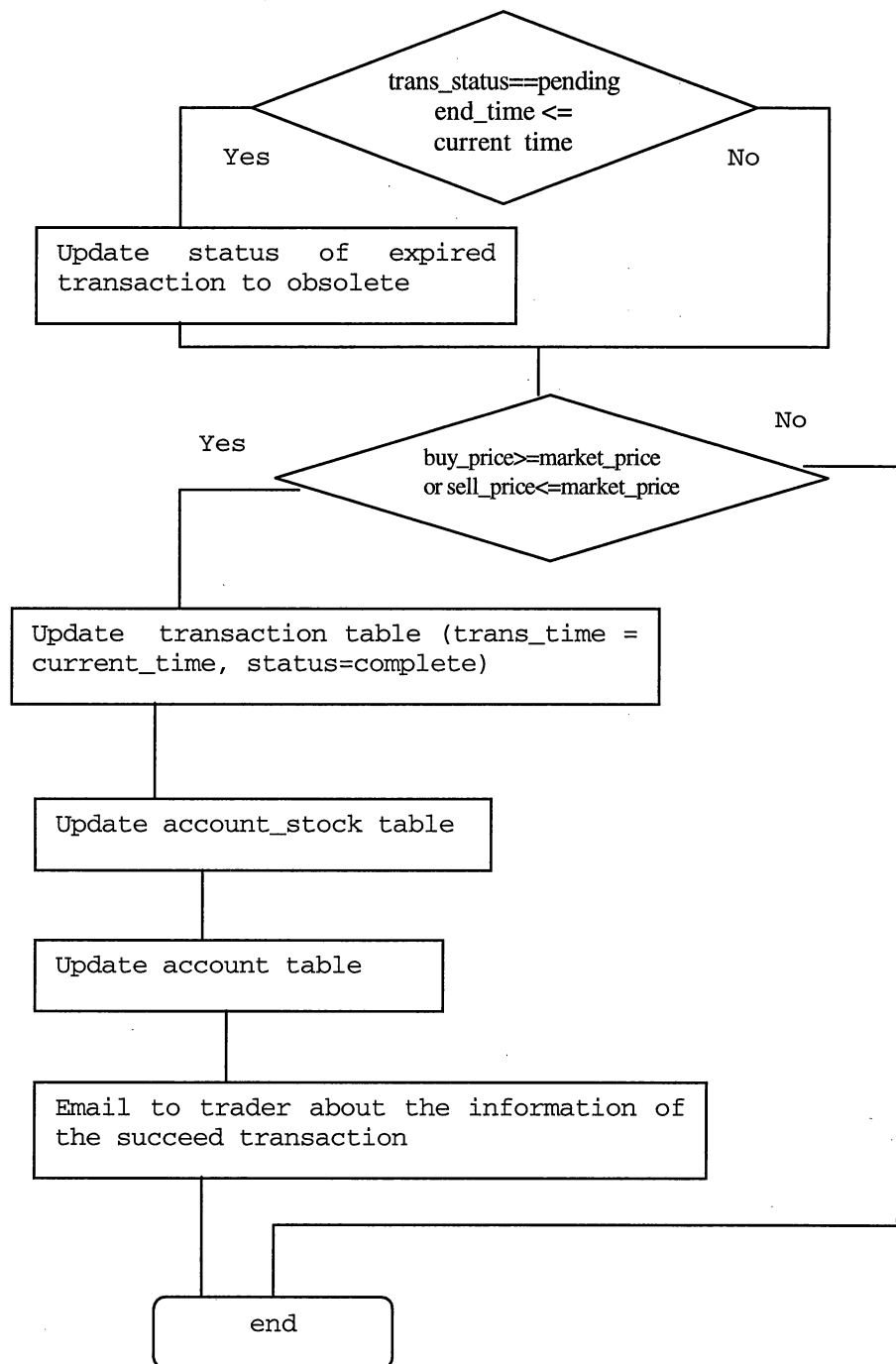


Figure 21. Trading during Market Time

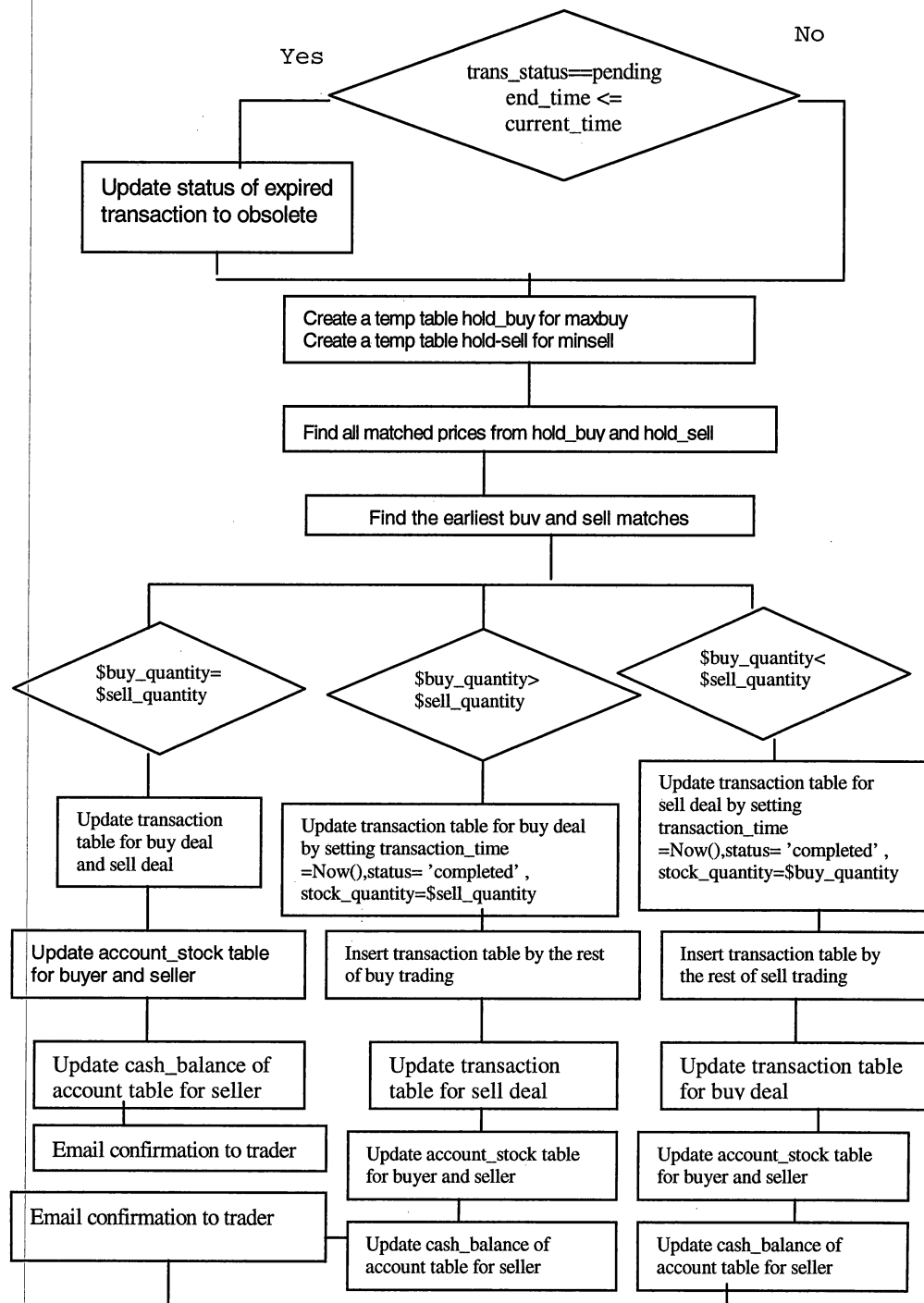


Figure 22. Trading during Off-market Time

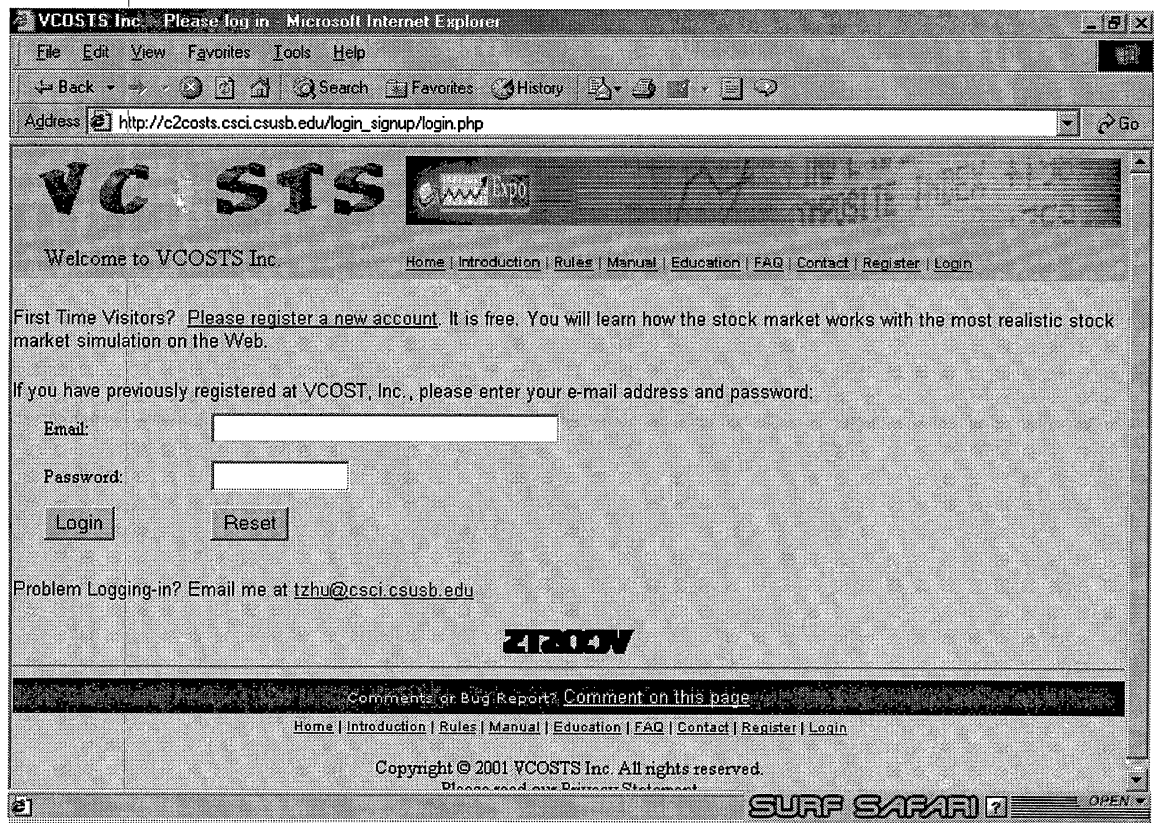


Figure 23. Login Page

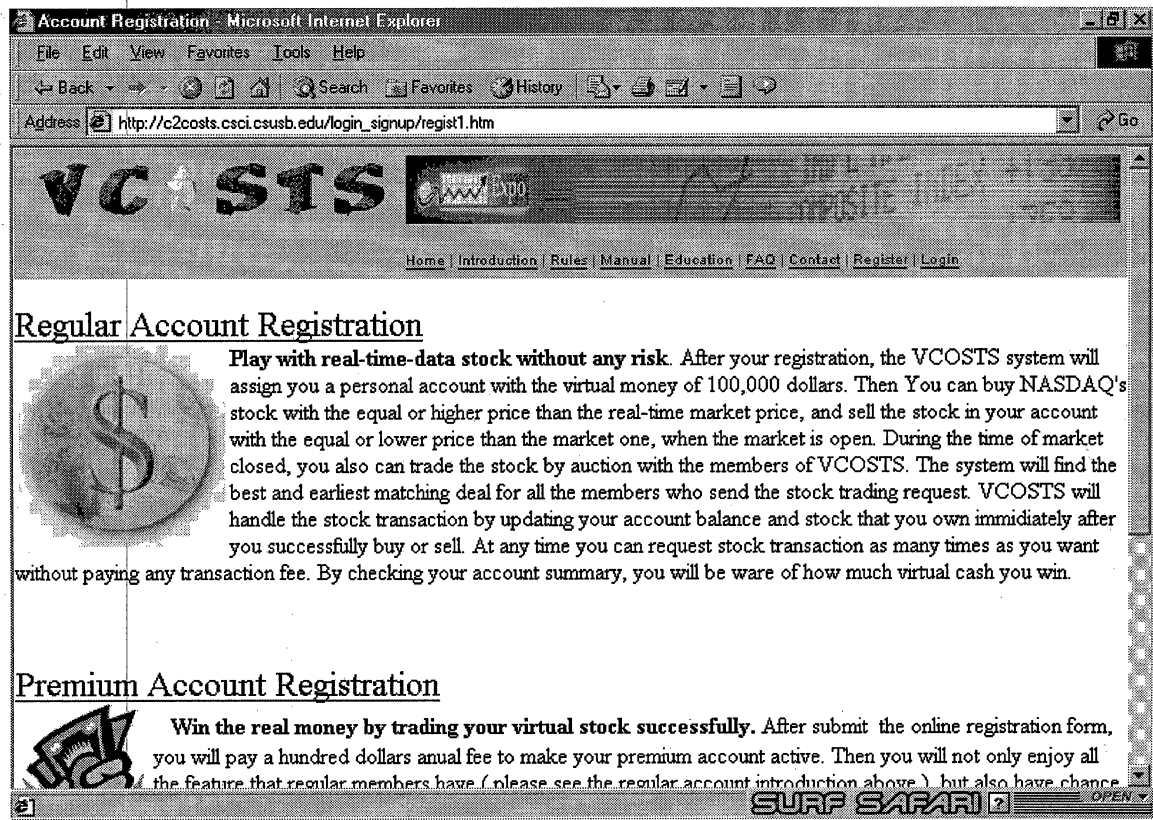


Figure 24. Register Page

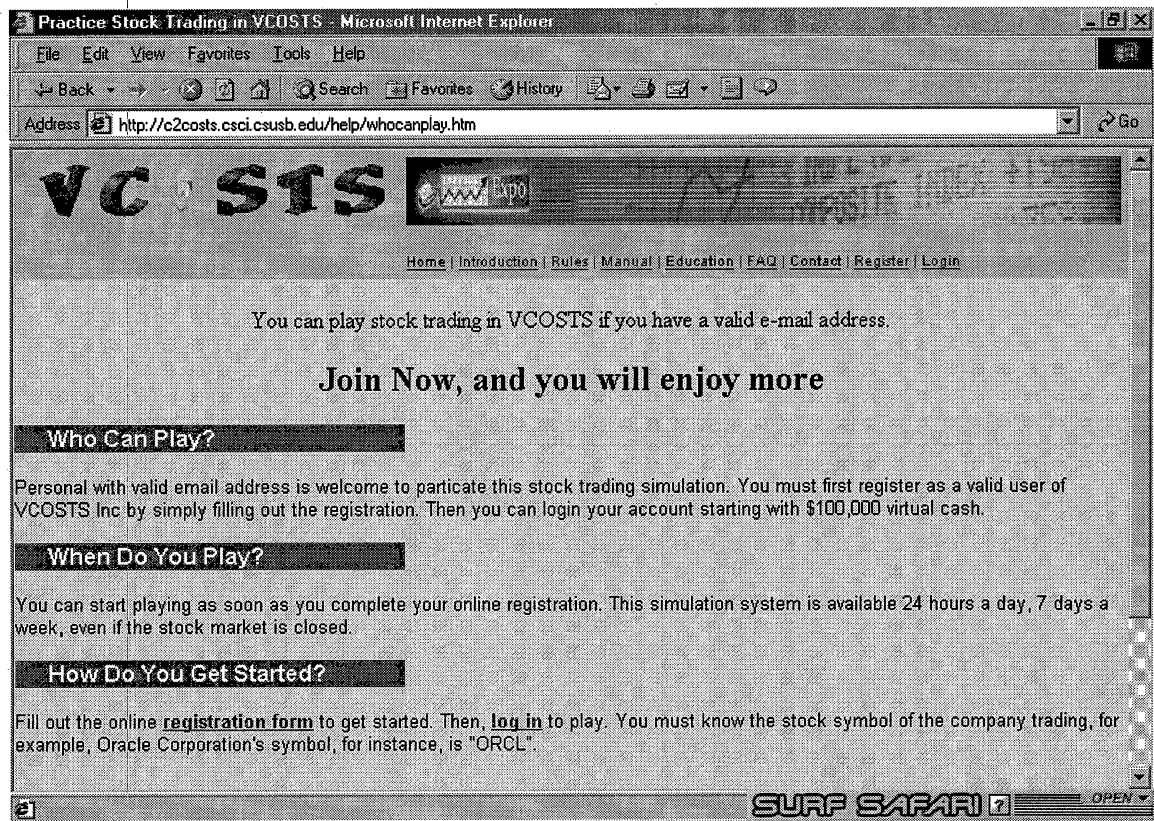


Figure 25. Who Can Play Page

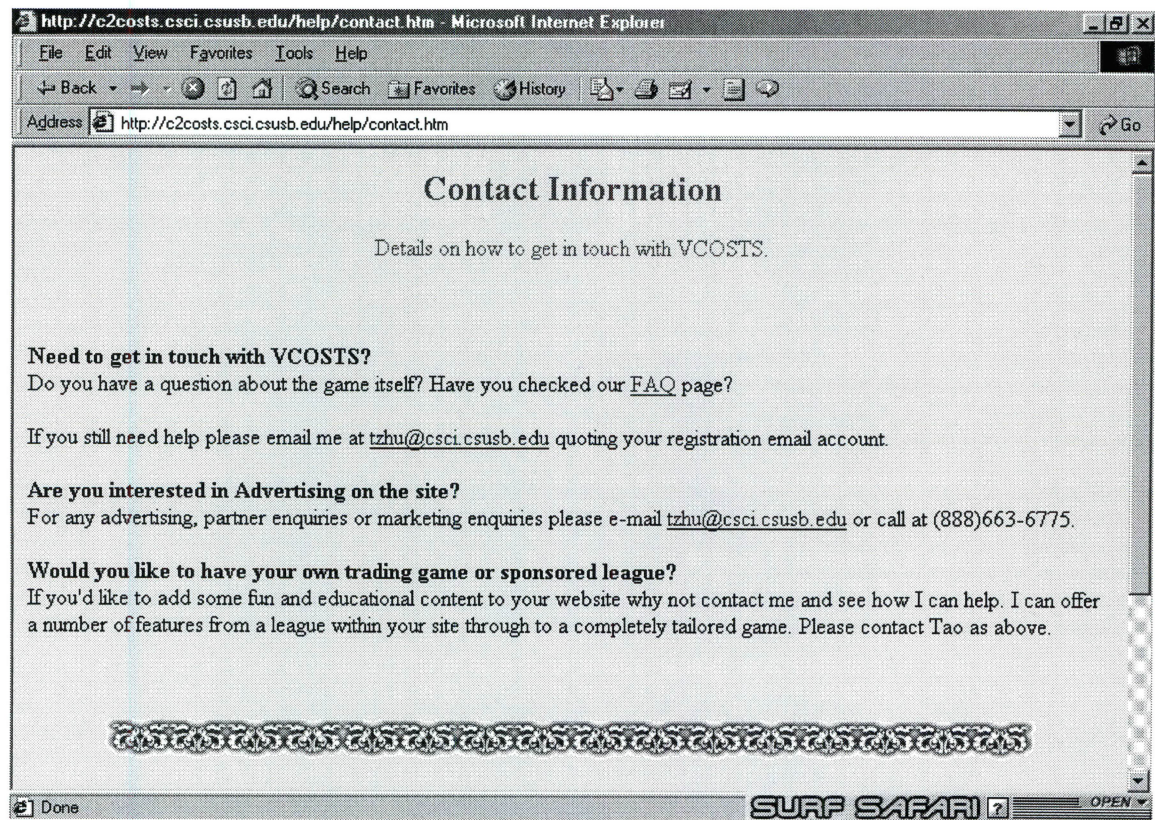


Figure 26. Contact Page

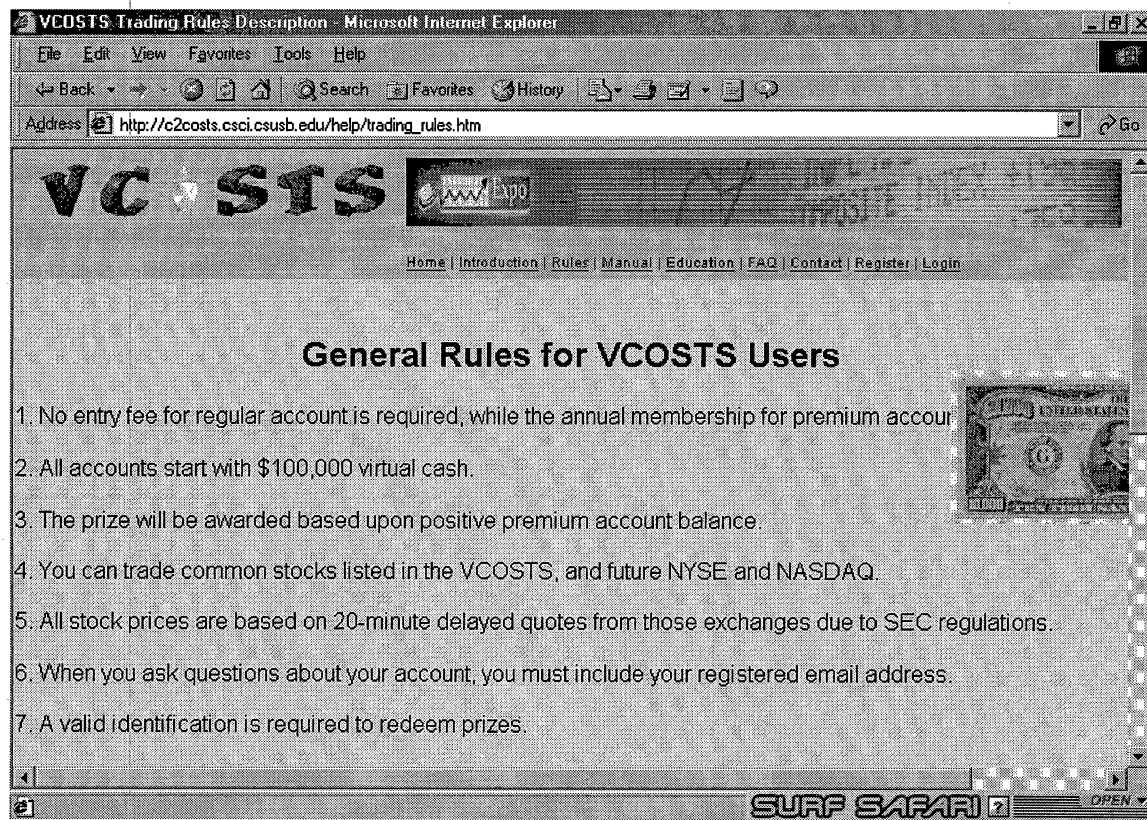


Figure 27. Trading Rules Page

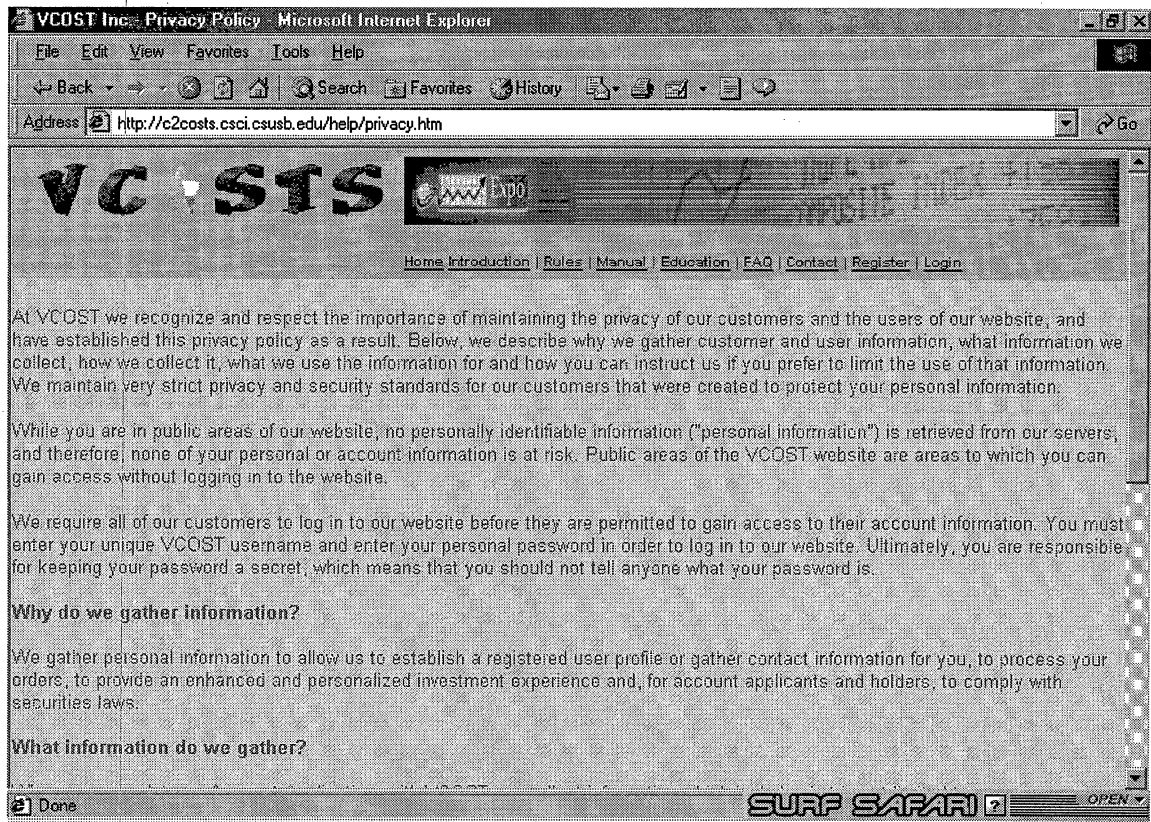


Figure 28. Privacy Page

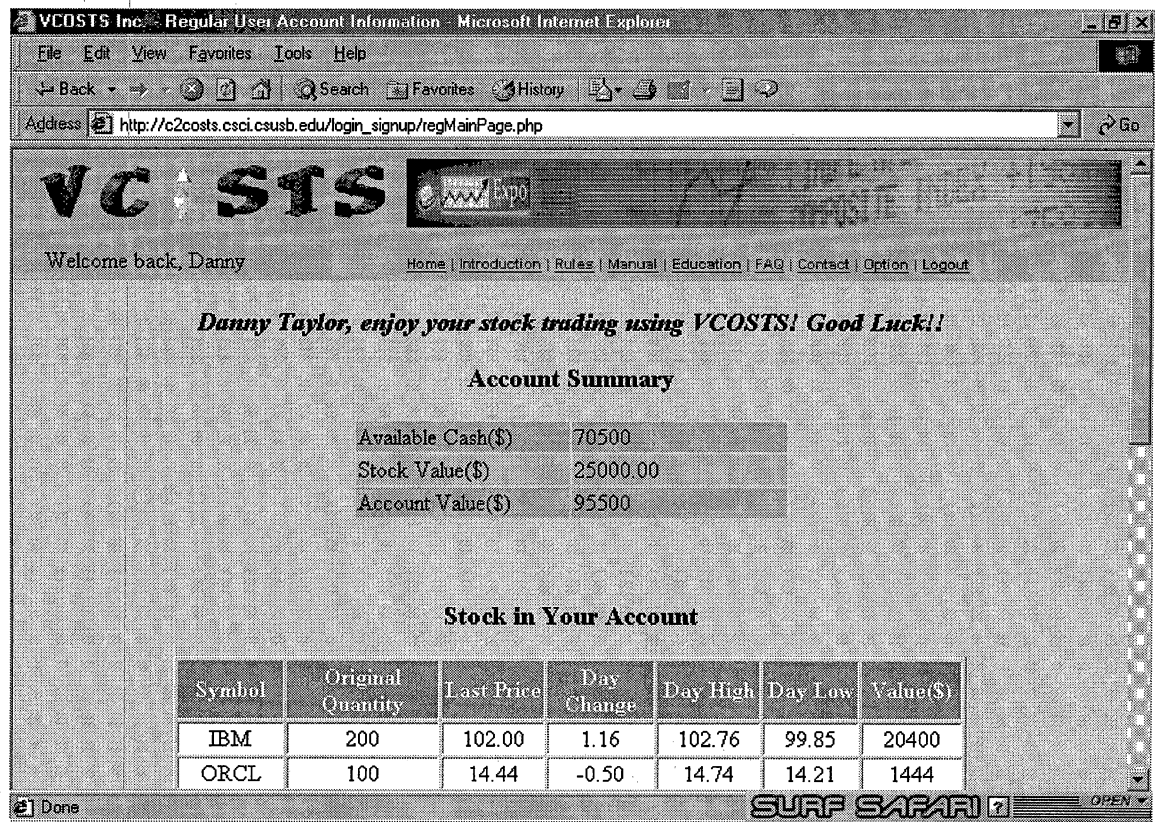






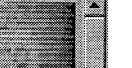
Figure 29. User Account Information

VCOSTS Inc. Buy Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites History

Address http://c2costs.csci.csusb.edu/transaction/buy_page.php Go

VCOSTS     

Welcome back, Danny

[Home](#) | [Introduction](#) | [Rules](#) | [Manual](#) | [Education](#) | [FAQ](#) | [Contact](#) | [Option](#) | [Logout](#)

Please fill out this form to BUY stock

Symbol Quantity Price Expire

Stock in Your Account

Symbol	Original Quantity	Last Price	Day Change	Day High	Day Low	Value(\$)
IBM	200	102.00	1.16	102.76	99.85	20400
ORCL	100	14.44	-0.50	14.74	14.21	1444

Done **SURF SAFARI** ? OPEN

Figure 30. Buy Stock Page

VCOSTS Inc. - Sell Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print Mail News RSS Go

Address http://c2costs.csci.csusb.edu/transaction/sell_page.php

Please fill out this form to SELL the stock in your account

Symbol Quantity Price Expire

Stock in Your Account

Symbol	Original Quantity	Last Price	Day Change	Day High	Day Low	Value(\$)
IBM	200	102.00	1.16	102.76	99.85	20400
ORCL	100	14.44	-0.50	14.74	14.21	1444
SPLS	200	15.78	0.15	16.03	15.42	3156

Done SURF SAFARI ? OPEN

Figure 31. Sell Stock Page

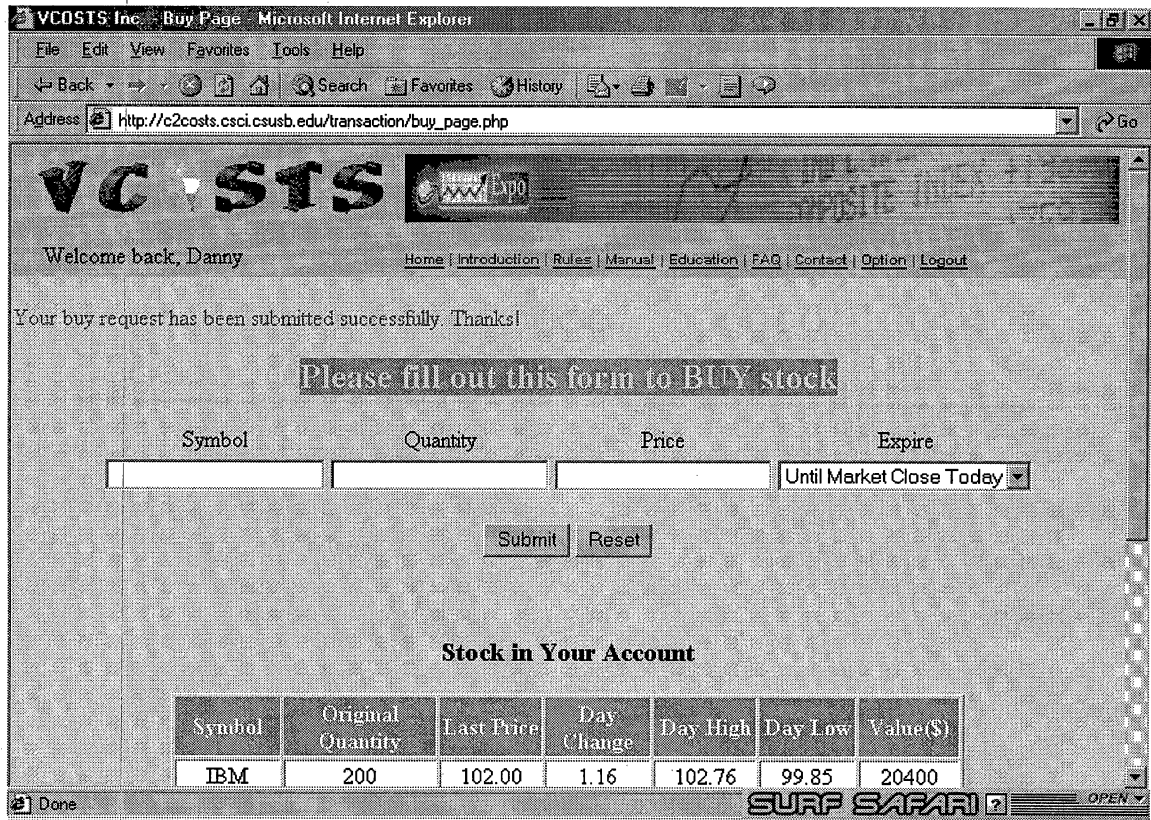


Figure 32. Buy Request Success Page

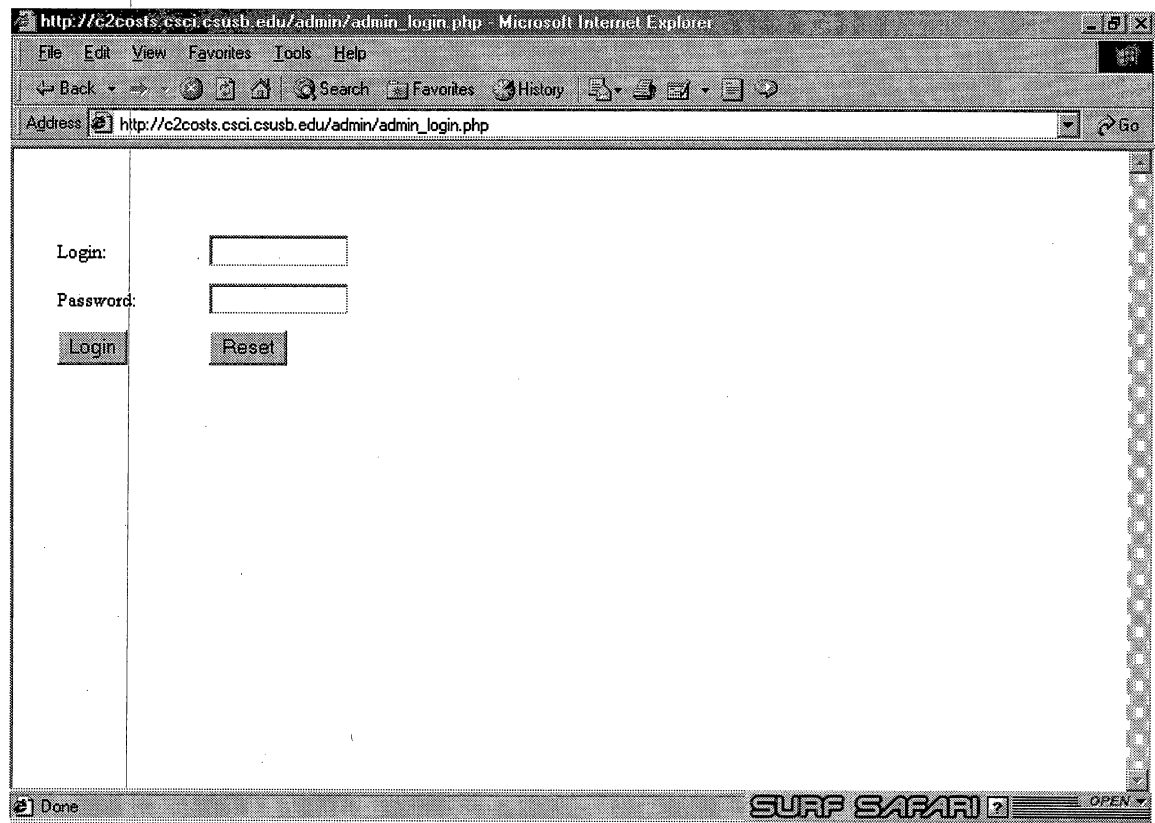


Figure 33. Admin Login Page

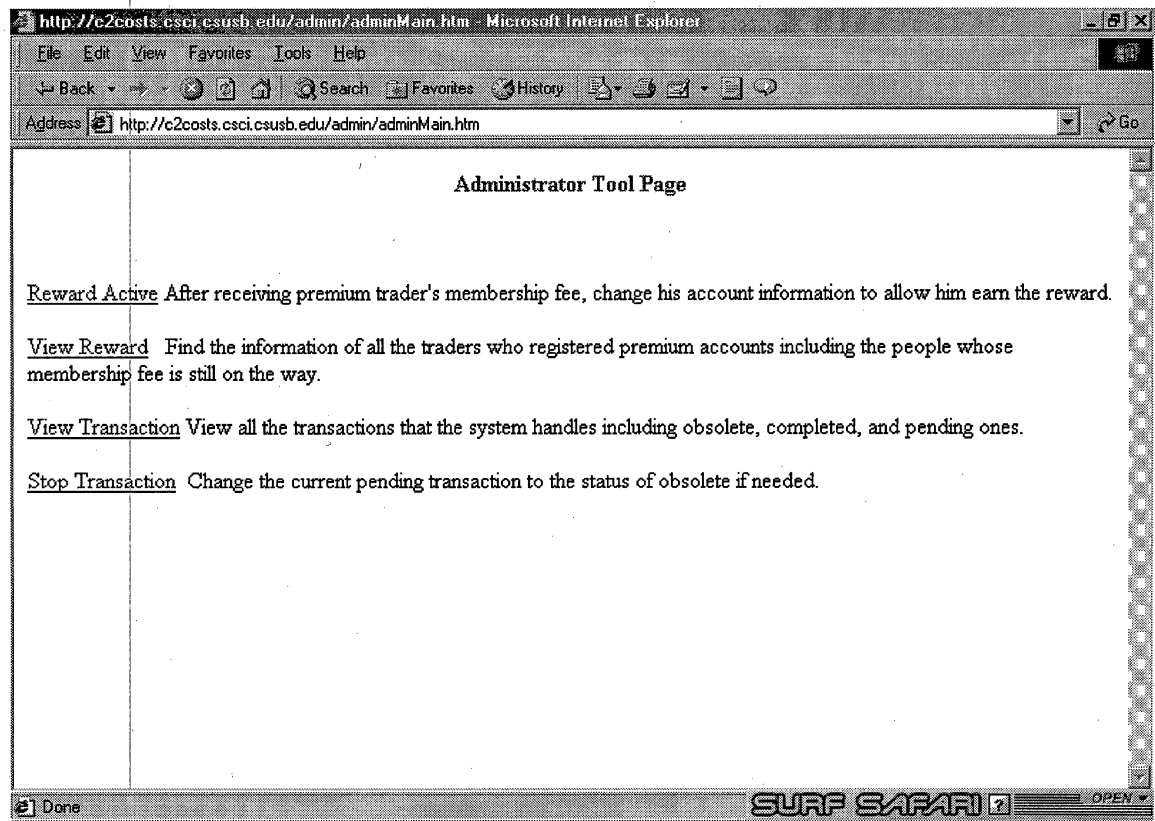


Figure 34. Administrator Tools Page

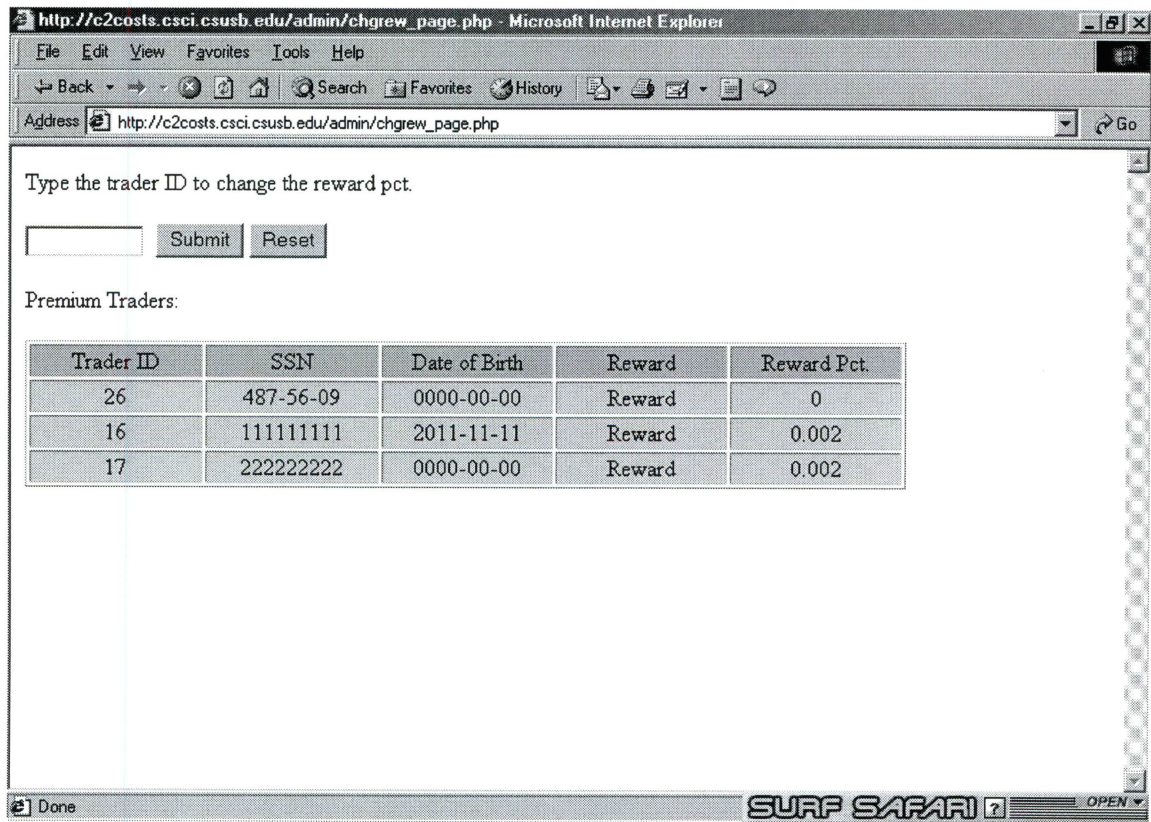


Figure 35. Change Reward Percent Page

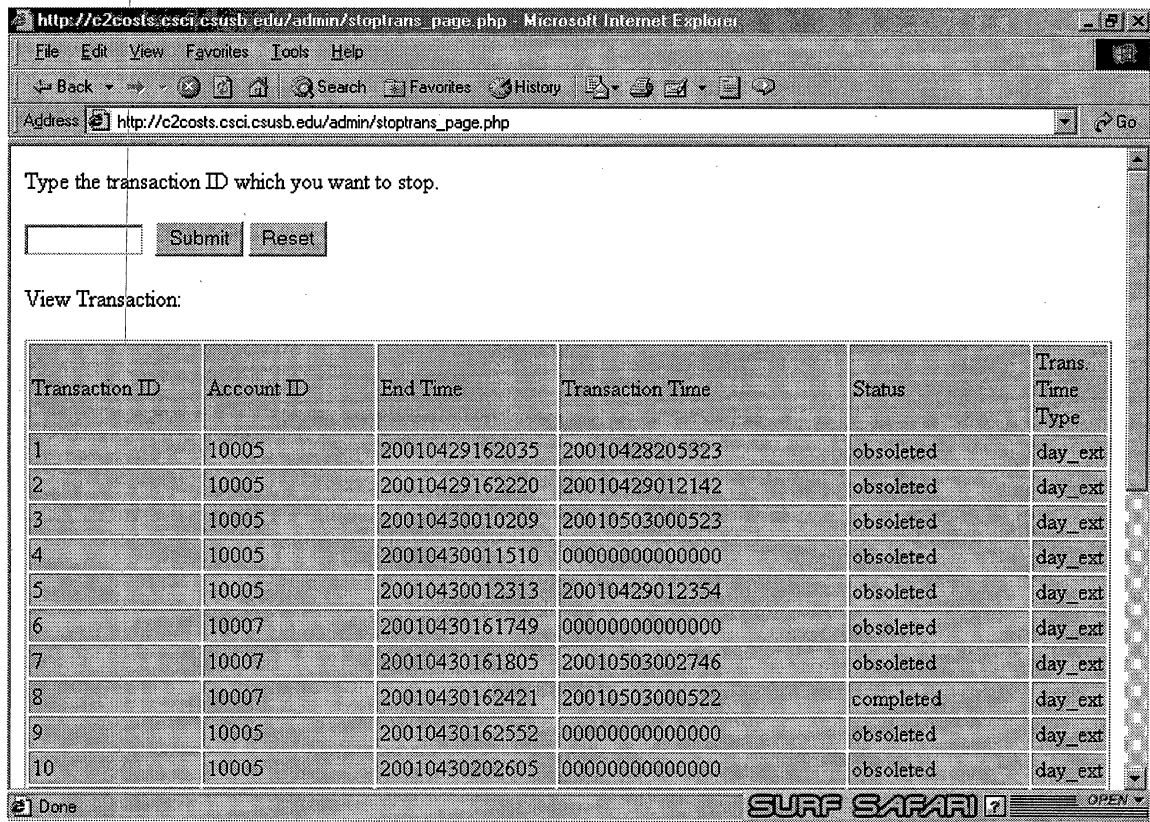


Figure 36. Stop Transaction Page

http://c2costs.csci.csusb.edu/admin/viewrew_page.php - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites History Print Mail Print Preview

Address http://c2costs.csci.csusb.edu/admin/viewrew_page.php Go

Premium Traders:

Trader ID	SSN	Date of Birth	Reward	Reward Pct.
26	487-56-09	0000-00-00	Reward	0
16	111111111	2011-11-11	Reward	0.002
17	222222222	0000-00-00	Reward	0.002

Done SURF SAFARI OPEN

Figure 37. Premium Trader

http://c2costs.csci.csusb.edu/admin/viewtrans_page.php - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites History

Address http://c2costs.csci.csusb.edu/admin/viewtrans_page.php Go

View Transaction:

Transaction ID	Account ID	End Time	Transaction Time	Status	Trans Time Type
1	10005	20010429162035	20010428205323	obsoleted	day_ext
2	10005	20010429162220	20010429012142	obsoleted	day_ext
3	10005	20010430010209	20010503000523	obsoleted	day_ext
4	10005	20010430011510	00000000000000	obsoleted	day_ext
5	10005	20010430012313	20010429012354	obsoleted	day_ext
6	10007	20010430161749	00000000000000	obsoleted	day_ext
7	10007	20010430161805	20010503002746	obsoleted	day_ext
8	10007	20010430162421	20010503000522	completed	day_ext
9	10005	20010430162552	00000000000000	obsoleted	day_ext
10	10005	20010430202605	00000000000000	obsoleted	day_ext
11	10005	20010430010209	20010503002746	completed	day_ext
12	10007	20010430161805	00000000000000	obsoleted	day_ext
13	10005	20010523165914	00000000000000	obsoleted	day_ext

Done SURF SAFARI OPEN

Figure 38. View Transaction Page

APPENDIX B:
PROJECT DATABASE SCHEMA

```

# Host: c2costs.csci.csusb.edu      Database: c2costs
#-----
# Server version      3.23.32

#
# Table structure for table 'account'
#

CREATE TABLE account (
  account_id int(11) NOT NULL auto_increment,
  trader_id int(11) NOT NULL default '0',
  account_type varchar(10) default NULL,
  cash_balance float(10,2) default NULL,
  original_value float(10,2) NOT NULL default '100000.00',
  reward_pct float default '0',
  PRIMARY KEY (account_id)
);

#
# Table structure for table 'account_stock'
#

CREATE TABLE account_stock (
  account_id int(11) NOT NULL default '0',
  stock_symbol varchar(8) NOT NULL default '',
  stock_quantity int(11) default NULL,
  PRIMARY KEY (account_id,stock_symbol)
);

#
# Table structure for table 'stock'
#

CREATE TABLE stock (
  symbol varchar(8) NOT NULL default '',
  company_name varchar(60) default NULL,
  description text,
  last_trade_price float(4,2) default NULL,
  day_change float(4,2) default NULL,
  day_high float(4,2) default NULL,
  day_low float(4,2) default NULL,
  PRIMARY KEY (symbol)
);

#

```

```
# Table structure for table 'trader'
#
```

```
CREATE TABLE trader (
  trader_id int(11) NOT NULL auto_increment,
  fname varchar(30) default NULL,
  middle_initial varchar(5) default NULL,
  lname varchar(30) default NULL,
  addr1 varchar(100) default NULL,
  addr2 varchar(100) default NULL,
  city varchar(40) default NULL,
  state char(2) default NULL,
  zipcode varchar(16) default NULL,
  home_phone varchar(30) default NULL,
  work_phone varchar(30) default NULL,
  cell_phone varchar(30) default NULL,
  fax varchar(30) default NULL,
  email varchar(60) NOT NULL default '',
  password varchar(32) default NULL,
  confirm_hash varchar(32) default NULL,
  active enum('Y','N') NOT NULL default 'N',
  PRIMARY KEY (trader_id)
);
```

```
#
# Table structure for table 'prem_trader'
#
```

```
CREATE TABLE prem_trader (
  trader_id int(11) default NULL,
  SSN varchar(9) NOT NULL default '',
  birthdate date NOT NULL default '0000-00-00'
);
```

```
#
# Table structure for table 'transaction'
#
```

```
CREATE TABLE transaction (
  transaction_id int(11) NOT NULL auto_increment,
  account_id int(11) default NULL,
  transaction_type varchar(8) default NULL,
  symbol varchar(8) default NULL,
  stock_quantity int(11) default NULL,
  trader_price float(8,2) default NULL,
```



```
start_time timestamp(14) NOT NULL,  
end_time timestamp(14) NOT NULL,  
transaction_time timestamp(14) NOT NULL,  
status varchar(10) default NULL,  
trans_time_type varchar(10) default NULL,  
PRIMARY KEY (transaction_id)  
);  
  
CREATE INDEX trans_in ON  
transaction(transaction_type,symbol,trader_price);  
  
GRANT ALL ON c2costs.* TO ztao@localhost IDENTIFIED BY  
"c20sts";
```

APPENDIX C:
PROJECT SOURCE CODES

```

//*****
// File name: admin_login.php
//*****
<?php

$a_name = "admin";
$a_password = "admin";

if (($action == "login") && ($admin_name != "") &&
    ($admin_pw != ""))
{
    if ( ($admin_name == $a_name) && ($admin_pw ==
        $a_password) )
    {
        header("Location: adminMain.htm");
    }
    else
        echo "Sorry, the information you entered is not
            valid. Please try again.";
}

if ($action == "login") {
    if ($admin_name == "")
        { echo "Your login name cannot be left blank!\n"; }
    if ($admin_pw == "")
        { echo "You must enter a password!\n"; }
}

?>

<table border=0 cellpadding=3 cellspacing=4 width="300">
<font size=-1>
<FORM method="post" action="admin_login.php">
    <tr>
        <td><font size=-1>Login:</font></td><td nowrap>
            <INPUT type="text" name="admin_name" value="<? print
            $admin_name; ?>" size="12" maxlength="25">
        </td>
    </tr>
    <tr>
        <td><font size=-1>Password:</font></td>
        <td nowrap>
            <INPUT type="password" name="admin_pw" value="<?
            print $admin_pw; ?>" size="12" maxlength="25">
        </td>
    </tr>
</table>

```

```

        </td>
    </tr>
    <tr>
    <td>
        <INPUT type="hidden" name="action" value="login">
        <INPUT type="submit" value="Login"></td>
        <td><INPUT type="reset"></center></td>
    </tr>
</FORM>
</table>

//*****
// File name: viewtrans_page.php
//*****

<?php
include "../admin/trans.php";
$trans = new CTrans;
$trans->view_trans();
?>

//*****
// File name: viewrew_page.php
//*****

<?php

include "../admin/reward.php";

$reward = new CReward;
$reward->view_rew();

?>

//*****
// File name: trans.php
//*****

<?php

include "../include/database.php";
include "../util/validate_funcs.php";

class CTrans
{

```

```

function view_trans()
{
    $query = "SELECT transaction_id, account_id,
                end_time, transaction_time, status,
                trans_time_type
            FROM transaction
            ORDER BY transaction_id";
    $result = mysql_query($query) or die ("Query
        failed");
    echo "<p>View Transaction:</p>
    <table border='1' width='100%'>
    <tr>
    <td width='6%' bgcolor='#COCOCO'>Transaction ID
    </td>
    <td width='16%' bgcolor='#COCOCO'>Account ID</td>
    <td width='17%' bgcolor='#COCOCO'>End Time</td>
    <td width='27%' bgcolor='#COCOCO'>Transaction Time
    </td>
    <td width='17%' bgcolor='#COCOCO'>Status</td>
    <td width='17%' bgcolor='#COCOCO'>Trans. Time Type
    </td>
    </tr>";

    for ($i=0; $i<mysql_num_rows($result); $i++)
    {
        $row = mysql_fetch_array($result);
        $transaction_id = $row['transaction_id'];
        $account_id = $row['account_id'];
        $end_time = $row['end_time'];
        $transaction_time = $row['transaction_time'];
        $status = $row['status'];
        $trans_time_type = $row['trans_time_type'];
        echo "<tr>
        <td width='16%' bgcolor='#CCCCFF'>$transaction_id
        </td>
        <td width='16%' bgcolor='#CCCCFF'>$account_id
        </td>
        <td width='17%' bgcolor='#CCCCFF'>$end_time</td>
        <td width='17%' bgcolor='#CCCCFF'>
        $transaction_time</td>
        <td width='17%' bgcolor='#CCCCFF'>$status</td>
        <td width='17%' bgcolor='#CCCCFF'>
        $trans_time_type</td>
        </tr>";
    }
}

```

```

        echo "</table>";
    } // end of function view_trans
} // end of CTrans class
?>

//*****
// File name: stoptrans_page.php
//*****

<?php

include "../admin/trans.php";

echo "<p>Type the transaction ID which you want to
stop.</p>
<form method='POST' action='stoptrans_page.php'>
<p>
    <input type='text' name='text1' value='$text1'
        size='10'>
    <input type='hidden' name='action' value='stop_trans'>
    <input type='submit' value='Submit' name='B1'>
    <input type='reset' value='Reset' name='B2'>
</p>
</form>";

if (($action == "stop_trans") && ($text1 != ""))
{
    if ( validate_numbers($text1) )
    {
        $query = "UPDATE transaction
                SET status = 'obsoleted'
                WHERE transaction_id = '$text1'";
        $result = mysql_query($query) or die ("Update
        Query failed");
        $message = "You stopped the transaction".$text1."
        successfully!";
        print '<FONT color="#FF0000">'.$message.'
        </FONT><BR>';
    }
    else
    {
        $message = "Sorry!! Please type the correct
        transaction ID!";
        print '<FONT color="#FF0000">'.$message.'
        </FONT><BR>';
    }
}

```

```

    }
}

$trans = new CTrans;
$trans->view_trans();

?>

//*****
// File name: reward.php
//*****

<?php

include "../include/database.php";

class CReward
{
function view_rew()
{
    $query = "SELECT trader_id, SSN, birthdate
              FROM prem_trader";
    $result = mysql_query($query) or die ("Query failed");
    echo "<p>Premium Traders:</p>";
    <table border='1' width='81%'>
    <tr>
    <td width='20%' align='center' bgcolor='#COCOCO'>Trader
    ID</td>
    <td width='20%' align='center' bgcolor='#COCOCO'>SSN</td>
    <td width='20%' align='center' bgcolor='#COCOCO'>Date of
    Birth</td>
    <td width='20%' align='center' bgcolor='#COCOCO'>Reward
    </td>
    <td width='20%' align='center' bgcolor='#COCOCO'>Reward
    Pct.</td>
    </tr>";

for ($i=0; $i<mysql_num_rows($result); $i++)
{
    $row = mysql_fetch_array($result);
    $trader_id = $row['trader_id'];
    $ssn = $row['SSN'];
    $birth = $row['birthdate'];
    $query1 = "SELECT reward_pct
              FROM account

```

```

        WHERE trader_id = '$trader_id' ";
$result1 = mysql_query($query1) or die ("Query
        failed");
$pct = mysql_result($result1, 0, "reward_pct");
echo "
<tr>
<td width='20%' align='center' bgcolor='#FFCCCC'>
$trader_id</td>
<td width='20%' align='center' bgcolor='#FFCCCC'>$ssn</td>
<td width='20%' align='center' bgcolor='#FFCCCC'>$birth
</td>
<td width='20%' align='center' bgcolor='#FFCCCC'>Reward
</td>
<td width='20%' align='center' bgcolor='#FFCCCC'>$pct</td>
</tr>";
    }//end of for

echo "</table>";
} //end of function view_rew

} //end of class CReward
?>

//*****
// File name: chgrew_page.php
//*****

<?php

include "../admin/reward.php";
include "../util/validate_funcs.php";

echo "<p>Type the trader ID to change the reward pct.</p>
<form method='POST' action='chg_rew.php'>
<p><input type='text' name='text1' value='$text1'
size='10'>
<input type='hidden' name='action' value='chg_reward'>
<input type='submit' value='Submit' name='B1'>
<input type='reset' value='Reset' name='B2'>
</p>
</form>";

if (($action == "chg_reward") && ($text1 != ""))
{
    if ( validate_numbers($text1) )

```



```

    {
        $query = "UPDATE account
                SET reward_pct = '0.002'
                WHERE trader_id = '$text1'";
        $result = mysql_query($query) or die ("Update111
                Query failed");
        $message = "You changed the reward pct for
                trader".$text1." successfully!";
        print '<FONT color="#FF0000">'.$message.'
        </FONT><BR>';
    }
    else
    {
        $message = "Sorry!! Please type the correct
                trader ID!";
        print '<FONT color="#FF0000">'.$message.'
        </FONT><BR>';
    }
} //end if
$reward = new CReward;
$reward->view_rew();

?>

//*****
// File name: auction_trading.php
//*****

<?php

include "../include/database.php";
include "../back_app/funcs.php";
include "../util/ans_funcs.php";

updateStatus_ifExpire();

//create a temp table for maxbuy
mysql_query("create temporary table hold_buy TYPE=heap
        select symbol, max(trader_price) as maxbuy
        from transaction
        where transaction_type='buy'
        and status='pending'
        and trans_time_type='day_ext'
        group by symbol") or die ("Query failed");

```

```

//create a temp table for minsell
mysql_query("create temporary table hold_sell TYPE=heap
            select symbol, min(trader_price) as minsell
            from transaction
            where transaction_type='sell'
            and status='pending'
            and trans_time_type='day_ext'
            group by symbol") or die ("Query failed");

//select from these two temp tables (a.maxbuy >=
b.minsell)
$result1 = mysql_query("SELECT a.symbol, a.maxbuy,
                        b.minsell
                        FROM hold_buy a, hold_sell b
                        WHERE a.maxbuy >= b.minsell
                        AND a.symbol = b.symbol") or die
("Query1 failed");

for ($i=0; $i<mysql_num_rows($result1); $i++)
{
    $row1 = mysql_fetch_array($result1);
    $symbol = $row1['symbol'];
    $maxbuy = $row1['maxbuy'];
    $minsell = $row1['minsell'];

    ////////////////////////////////// for buy //////////////////////////////////
    //get the earliest start_time for all buy price matches
    $result2 = mysql_query("SELECT min(start_time) as
                            earliestMaxbuy
                            FROM transaction
                            WHERE trader_price='$maxbuy'
                            AND transaction_type = 'buy'
                            AND symbol = '$symbol'
                            GROUP BY trader_price") or
    die("Query2 failed");

    $row2 = mysql_fetch_array($result2);
    $earliestMaxbuy = $row2['earliestMaxbuy'];

    //get information of the highest buy price and the
    earliest start_time
    $result3 = mysql_query("SELECT stock_quantity,
                                transaction_id, account_id,
                                end_time
                                FROM transaction
                                WHERE trader_price = '$maxbuy'

```

```

AND transaction_type = 'buy'
AND symbol = '$symbol'
AND start_time =
'$earliestMaxbuy' ") or die
("Query3 failed");
$row3 = mysql_fetch_array($result3);
$buy_quantity = $row3['stock_quantity'];
$buy_trans_id = $row3['transaction_id'];
$buy_acct_id = $row3['account_id'];
$buy_end_time = $row3['end_time'];

////////// for sell //////////////////////////////////////
//get the earliest start_time for all sell price matches
$result4 = mysql_query("SELECT min(start_time) as
earliestMinsell
FROM transaction
WHERE trader_price =
'$minsell'
AND transaction_type = 'sell'
AND symbol = '$symbol'
GROUP BY trader_price") or
die("Query4 failed");
$row4 = mysql_fetch_array($result4);
$earliestMinsell = $row4['earliestMinsell'];

//get information of the lowest sell price and the
earliest start_time
$result5 = mysql_query("SELECT stock_quantity,
transaction_id,
account_id, end_time
FROM transaction
WHERE trader_price =
'$minsell'
AND transaction_type = 'sell'
AND symbol = '$symbol'
AND start_time =
'$earliestMinsell' ") or
die ("Query5 failed");
$row5 = mysql_fetch_array($result5);
$sell_quantity = $row5['stock_quantity'];
$sell_trans_id = $row5['transaction_id'];
$sell_acct_id = $row5['account_id'];
$sell_end_time = $row5['end_time'];

////////// compare the buy and sell price //////////

```

```

if ( $buy_quantity == $sell_quantity )
{
    //update transaction table( transaction_time and status )
    mysql_query("UPDATE transaction
                SET transaction_time = NOW(), status =
                  'completed'
                WHERE transaction_id='$buy_trans_id'")
    or
    die ("Query failed");

    //update account_stock table (stock and quantity)
    $buy = new CBuy;
    $buy->updateStock_afterBuy($symbol, $buy_quantity,
                              $buy_acct_id);

    //email the trading success information to the user
    //email_buySuccess($buy_acct_id, $buy_quantity, $symbol);

    //update transaction table( transaction_time and status )
    mysql_query("UPDATE transaction
                SET transaction_time = NOW(), status =
                  'completed'
                WHERE transaction_id='$sell_trans_id'")
    or die ("Query failed");

    //update account_stock table (stock's quantity)
    $sell = new CSell;
    $sell->updateStock_afterSell($symbol, $sell_quantity,
                                $sell_acct_id);

    //update account table (cash_balance)
    $sell->updateBalance_afterSell($sell_quantity,
                                   $minsell, $sell_acct_id);

    //email the trading success information to the user
    //email_sellSuccess($sell_acct_id, $sell_quantity,
    $symbol, $minsell);

    } //end of if ( $buy_quantity == $sell_quantity )

if ( $buy_quantity > $sell_quantity )
{
    //update transaction table( transaction_time and status )
    mysql_query("UPDATE transaction
                SET transaction_time = NOW(), status =
                  'completed'
                WHERE transaction_id='$buy_trans_id'")
    or
    die ("Query failed");

    //update account_stock table (stock and quantity)
    $buy = new CBuy;
    $buy->updateStock_afterBuy($symbol, $buy_quantity,
                              $buy_acct_id);

    //email the trading success information to the user
    //email_buySuccess($buy_acct_id, $buy_quantity, $symbol);

    //update transaction table( transaction_time and status )
    mysql_query("UPDATE transaction
                SET transaction_time = NOW(), status =
                  'completed'
                WHERE transaction_id='$sell_trans_id'")
    or die ("Query failed");

    //update account_stock table (stock's quantity)
    $sell = new CSell;
    $sell->updateStock_afterSell($symbol, $sell_quantity,
                                $sell_acct_id);

    //update account table (cash_balance)
    $sell->updateBalance_afterSell($sell_quantity,
                                   $minsell, $sell_acct_id);

    //email the trading success information to the user
    //email_sellSuccess($sell_acct_id, $sell_quantity,
    $symbol, $minsell);

    } //end of if ( $buy_quantity > $sell_quantity )
}

```

```

// (1)update transaction table: transaction_time, status,
quantity
mysql_query("UPDATE transaction
            SET transaction_time=NOW(),
            status='completed',
            stock_quantity='$sell_quantity'
            WHERE transaction_id='$buy_trans_id'")
            or die ("Query failed");

// (2)insert transaction table: the rest of buy trading
$therest = $buy_quantity - $sell_quantity;
mysql_query("INSERT INTO transaction (account_id,
            transaction_type, symbol,
            stock_quantity,
            trader_price, start_time, end_time,
            status, trans_time_type )
            VALUES ('".$buy_acct_id."', 'buy',
            '".$symbol."', '".$therest."',
            '".$maxbuy."', '".$earliestMaxbuy."',
            '".$buy_end_time."', 'pending',
            'day_ext')") or die ("Query failed");

//update account_stock table (stock and quantity)
$buy = new CBuy;
$buy->updateStock_afterBuy($symbol, $sell_quantity,
                        $buy_acct_id);

//email the trading success information to the user
//email_buySuccess($buy_acct_id, $sell_quantity, $symbol);

////////// update for sell trading//////////

//update transaction table( transaction_time and status )
mysql_query("UPDATE transaction
            SET transaction_time = NOW(), status =
            'completed'
            WHERE transaction_id='$sell_trans_id'")
            or die ("Query failed");
//update account_stock table (stock's quantity)
$sell = new CSell;
$sell->updateStock_afterSell($symbol,
                        $sell_quantity,$sell_acct_id);

//update account table (cash_balance)
$sell->updateBalance_afterSell($sell_quantity,

```

```

        $minsell,$sell_acct_id);

//email the trading success information to the user
//email_sellSuccess($sell_acct_id, $sell_quantity,
$symbol, $minsell);

    } // end of if ( $buy_quantity > $sell_quantity )

        if ( $buy_quantity < $sell_quantity )
        {
////////// update for buy trading//////////
//update transaction table( transaction_time and status )
mysql_query("UPDATE transaction
            SET transaction_time = NOW(), status =
            'completed'
            WHERE transaction_id='$buy_trans_id'")
            or
            die ("Query failed");

//update account_stock table (stock and quantity)
    $buy = new CBuy;
    $buy->updateStock_afterBuy($symbol,
        $buy_quantity, $buy_acct_id);

//email the trading success information to the user
//email_buySuccess($buy_acct_id, $buy_quantity, $symbol);
////////// update for sell trading //////////
// (1)update transaction table: transaction_time, status,
quantity
mysql_query("UPDATE transaction
            SET transaction_time=NOW(),
            status='completed',
            stock_quantity='$buy_quantity'
            WHERE transaction_id='$sell_trans_id'")
            or die ("Query failed");

// (2)insert transaction table: the rest of buy trading
    $therest = $sell_quantity - $buy_quantity;
mysql_query("INSERT INTO transaction
            (account_id,transaction_type,
            symbol, stock_quantity,
            trader_price, start_time,
            end_time, status, trans_time_type
)
            VALUES ('".$sell_acct_id."',

```

```

        'sell',
        '". $symbol."', '". $therest."',
        '". $minsell."',
        '". $earliestMaxbuy."',
        '". $sell_end_time."',
        'pending', 'day_ext')") or die
        ("Query failed");

//update account_stock table (stock's quantity)
$sell = new CSell;
$sell->updateStock_afterSell($symbol, $buy_quantity,
                             $sell_acct_id);
//update account table (cash_balance)
$sell->updateBalance_afterSell($buy_quantity,
                              $minsell, $sell_acct_id);

//email the trading success information to the user
//email_sellSuccess($sell_acct_id, $buy_quantity, $symbol,
$minsell);

    }// end of if ( $buy_quantity < $sell_quantity )

    }// end of for ($i=0; $i<mysql_num_rows($result1); $i++)
in line 33
mysql_query("drop table hold_buy") or die ("Drop table
failed");
mysql_query("drop table hold_sell") or die ("Drop table
failed");

?>

//*****
// File name: funcs.php
//*****

<?php

include "../include/database.php";

function updateStatus_ifExpire()
{
    $query = "SELECT trader_price, stock_quantity,
                  transaction_type, account_id,
                  transaction_id
              FROM transaction

```

```

        WHERE status = 'pending'
        AND end_time <= NOW()";
$result = mysql_query($query) or die ("Query
        failed");

for ($i=0; $i<mysql_num_rows($result); $i++)
{
    $row = mysql_fetch_array($result);
    $price = $row['trader_price'];
    $quantity = $row['stock_quantity'];
    $type = $row['transaction_type'];
    $acct_id = $row['account_id'];
    $tran_id = $row['transaction_id'];

    //update status to obsoleted of transaction
    $query = "UPDATE transaction
        SET status = 'obsoleted'
        WHERE transaction_id='$tran_id'";

    $result = mysql_query($query) or die ("Query
        failed");

    //return the held money to account when buy request is
    expired
        if ( $type="buy" )
        {
            //require("../util/ans_funcs.php");
            $buy = new CBuy;
            $buy->returnBalance_ifnotBuy ($quantity, $price,
                $acct_id);
        } //end of if
    } //end of for

} //end of function updateStatus()

function email_buySuccess($acct_id, $quantity, $symbol)
{
    $query = "SELECT a.fname, a.lname, a.email
        FROM trader a, account b
        WHERE a.trader_id=b.trader_id
        AND b.account_id = '$acct_id' ";
    $result = mysql_query($query) or die ("Query
        failed");

    while ($row = mysql_fetch_array($result))

```



```

        {
            $firstname = $row['fname'];
            $lastname = $row['lname'];
            $email = $row['email'];
        }

$mail_text = "\n"
            ."Dear ".$firstname." ".$lastname.":\n\n"
            ."Thank you for your order.\n\n"
            ."You bought ".$quantity." shares of
            ".$symbol.". ". "\nThank you for using VCOSTS
            online service.\n\n\n"
            ."Sincerely,\n\nVCOSTS Support";

mail($email,"Execution Report from VCOSTS",
    $mail_text);
}

function email_sellSuccess($acct_id, $quantity, $symbol,
    $price)
{
    $query = "SELECT a.fname, a.lname, a.email
        FROM trader a, account b
        WHERE a.trader_id=b.trader_id
        AND b.account_id = '$acct_id' ";
    $result = mysql_query($query) or die ("Query
        failed");

    while ($row = mysql_fetch_array($result))
    {
        $firstname = $row['fname'];
        $lastname = $row['lname'];
        $email = $row['email'];
    }

    $mail_text = "\n"
        ."Dear ".$firstname." ".$lastname.":\n\n"
        ."Thank you for your order.\n\n"
        ."You sold ".$quantity." shares of
        ".$symbol." at \$. $price."."
        ." \nThank you for using VCOSTS online
        service.\n\n\n"
        ."Sincerely,\n\nVCOSTS Support";

    mail($email,"Execution Report from VCOSTS",

```

```

        $mail_text);
    }
    ?>
    /*******
    // File name: market_trading.php
    /*******

<?php

include "../include/database.php";
include "../back_app/funcs.php";
include "../util/ans_funcs.php";

updateStatus_ifExpire();

//if buy price satisfied with market price
$query = "SELECT a.transaction_id, a.account_id, a.symbol,
              a.stock_quantity
          FROM transaction a, stock b
          WHERE status = 'pending'
          AND transaction_type = 'buy'
          AND a.trader_price >= b.last_trade_price
          AND a.symbol = b.symbol";
$result = mysql_query($query) or die ("Query failed");

for ($i=0; $i<mysql_num_rows($result); $i++)
{
    $row = mysql_fetch_array($result);
    $tran_id = $row['transaction_id'];
    $acct_id = $row['account_id'];
    $symbol = $row['symbol'];
    $quantity = $row['stock_quantity'];

    //update transaction table( transaction_time and status )
    $query1 = "UPDATE transaction
              SET transaction_time = NOW(), status =
                'completed'
              WHERE transaction_id='$tran_id'";
    $result1 = mysql_query($query1) or die ("Query
              failed");
    //update account_stock table (stock and quantity)
    $buy = new CBuy;
    $buy->updateStock_afterBuy($symbol, $quantity,
                              $acct_id);
    //email the trading success information to the user

```

```

//email_buySuccess($acct_id, $quantity, $symbol);
//if sell price satisfied with market price
$query = "SELECT a.transaction_id, a.account_id,
               a.symbol,
               a.stock_quantity, a.trader_price
FROM transaction a, stock b
WHERE status = 'pending'
AND transaction_type = 'sell'
AND a.trader_price <= b.last_trade_price
AND a.symbol = b.symbol";
$result = mysql_query($query) or die ("Query
failed");

for ($i=0; $i<mysql_num_rows($result); $i++)
{
    $row = mysql_fetch_array($result);
    $tran_id = $row['transaction_id'];
    $acct_id = $row['account_id'];
    $symbol = $row['symbol'];
    $quantity = $row['stock_quantity'];
    $price = $row['trader_price'];

//update transaction table( transaction_time and status )
$query1 = "UPDATE transaction
            SET transaction_time = NOW(), status
              = 'completed'
            WHERE transaction_id='$tran_id'";
$result1 = mysql_query($query1) or die ("Query
failed");

//update account_stock table (stock's quantity)
$sell = new CSell;
$sell->updateStock_afterSell($symbol, $quantity,
    $acct_id);

//update account table (cash_balance)
$sell->updateBalance_afterSell($quantity,
    $price, $acct_id);
//email the trading success information to the user
//email_sellSuccess($acct_id, $quantity, $symbol, $price);
}
?>

```

```

//*****
// File name: database.php
//*****

<?php

$db_hostname = "c2costs.csci.csusb.edu";
// database server name
$db_name = "c2costs";           // database name
$db_user = "tzhu";             // database user
$db_pass = "c2c0n11ne";        // database user password

mysql_connect($db_hostname, $db_user, $db_pass) or die
("Unable to connect to database");
mysql_select_db($db_name) or die ("Unable to select
database");

?>

//*****
// File name: site.php
//*****

<?php

include "../include/udfunctions.php";

/*
function color() {
    global $s_color;
    if (logged_in() && ($s_color != ""))
        { return $s_color; }
    else { return "006699"; }
}
*/

function site_header($params) {
    global $s_first_name;
    print '<HTML><HEAD><TITLE>VCOSTS Inc.';
    if ($params['title'])
        { print " - " . $params['title']; }
    print '</TITLE><LINK rel="stylesheet"
        href="vcost.css" type="text/css"></HEAD>'

```



```

        . '<a href="../../../login_signup/premMainPage.php"
        class="page_title">Premium Acct View</a> | '
        . '<a href="../../../transaction/buy_page.php"
        class="page_title">Buy Stock</a> | '
        . '<a href="../../../transaction/sell_page.php"
        class="page_title">Sell Stock</a> | '
        . '<a href="../../../login_signup/logout.php"
        class="page_title">Logout</a></Font>  ';
    }
    else {
        print '<FONT face="arial, helvetica" size="-2"
              color="#336699" class="textlink">
              <b><a href="../../../index.html"
              class="page_title">Home</a> | '
        . '<a href="../../../help/whocanplay.htm"
        class="page_title">Quick Introduction</a> | '
        . '<a href="../../../help/trading_rules.htm"
        class="page_title"><B>Rules</B></a> | '
        . '<a href="../../../login_signup/register.php"
        class="page_title">Register</a> | '
        . '<a href="../../../help/contact.htm"
        class="page_title">Contact Us<a> | '
        . '<a href="../../../login_signup/login.php"
        class="page_title">Login</a></Font>  ';
    }

    print '</TD>
          </TR>
          </TABLE>
          <BR>';
}

```

```

function box_top($title, $url) {

    $html = '
    <TABLE cellpadding="1" cellspacing="0" border="0"
    width="180" bgcolor="#'.color().'">
    <TR><TD>';
    $html .= '
    <TABLE cellpadding="2" cellspacing="2" border="0"
    width="100%" bgcolor="#FFFFFF">';

    /* Title row */
    $html .= '
        <TR bgcolor="'.color().'">

```

```

        <TD class="box_title"><SMALL><A href="'. $url.'"
            class="box_title">'. $title.'</A></SMALL>
        </TD>
    </TR>';

/* content row */
$html .= '
        <TR bgcolor="#FFFFFF">
            <TD>';
print $html;
}
function box_bottom() {
    $html = '<table width="100%" border="0"
        cellpadding="0" cellspacing="0">
        <tr height="25">
            <td bgcolor=#003366 align="center" nowrap width="750"
            colspan="3">
            <font color=#FFFFFF size=-2 face=verdana> Comments or
            Bug Report?
            <a href="../../../help/comments.htm" target="commentFrame">
            <font face=Arial><font color=#FFFFFF size =2
            face=Arial>Comment on this page
            </font> </font></a> </font></td> </tr>
            <tr> <td align="center"></td> <td align="center"
            nowrap><font size=2 face=Arial> <small>
            <a href="../../../index.html" class="page_title">Home</a> |
            <a href="../../../help/whocanplay.htm" class="page_title">
            Introduction</a> |<a href="../../../help/trading_rules.htm"
            class="page_title"><B>Rules</B></a> |
            <a href="../../../login_signup/register.php"
            class="page_title"> Register</a> | <a
            href="../../../help/contact.htm" class="page_title">Contact
            Us<a>|<a href="../../../login_signup/login.php"
            class="page_title">Login </a>
            <br><br> Copyright &#169; 2001 VCOST Inc. All rights
            reserved.<br> Please read our
            <a href="../../../help/privacy.htm" target="_top">Privacy
            Statement</a></small> </font></td> </tr></table>';
    print $html;
}
function warning($message) {
print '<FONT color="#FF0000">'. $message.'</FONT><BR>';
}
?>

```



```

//*****
// File name: udfunctions.php
//*****

<?php
session_start();
include "../include/database.php";
//get trader data: lname, fname
function get_user_data($acct_id) {
    global $s_first_name, $s_last_name;
    $query = "SELECT fname, lname
              FROM trader a, account b
              WHERE a.trader_id=b.trader_id
              AND b.account_id = '$acct_id'";
    $result = mysql_query($query) or die ("Query
              failed");
    if (mysql_num_rows($result) > 0)
    {
        session_register('s_first_name');
        session_register('s_last_name');
        $row = mysql_fetch_array($result);
        $s_first_name = $row['fname'];
        $s_last_name = $row['lname'];
    }
}

function logout()
{
    session_unregister('s_account_id');
    session_unregister('s_first_name');
    session_unregister('s_last_name');
    header("Location: ../index.html");
}

function logged_in()
{
    if (session_is_registered('s_account_id'))
    { return 1; }
    else { return 0; }
}

function show_user()
{
    global $s_account_id, $s_first_name, $s_last_name;
    print "AccountID: $s_account_id<br>";
}

```



```

        print "FirstName: $s_first_name<br>";
        print "LastName: $s_last_name<br>";
    }

?>

//*****
// File name: confirm.php
//*****

<?php
session_start();

include "../include/site.php";
include "../include/database.php";
include "../util/account_funcs.php";

site_header(array('title'=>'Confirmation Page'));

if (($action == "confirm") && ($email != "") && ($password
!= ""))
{
    $query = "SELECT trader_id, password, email
              FROM trader
              WHERE
              confirm_hash='".$confirm_hash.'" ";

    $result = mysql_query($query) or die ("Query
              failed");
    if (mysql_num_rows($result) > 0) {
        $row = mysql_fetch_array($result);
        $trader_id = $row['trader_id'];

        if (($email == $row['email']) &&
            (md5($password) == $row['password']))
        {
            mysql_query("UPDATE trader SET active='Y',
            confirm_hash='' WHERE email='$email'") or die
            ("User update failed");

            $result = mysql_query("SELECT account_id,
            account_type FROM account
            WHERE trader_id = '$trader_id' ")
            or die ("Query failed");

```

```

while ($row = mysql_fetch_array($result))
{
    session_register('s_account_id');
    $s_account_id = $row['account_id'];
    $account_type = $row['account_type'];
}
get_user_data($s_account_id);

    if ( $account_type == "Regular" )
    {
        header("Location: regMainPage.php");
    }
    else if ( $account_type == "Premium" )
    {
        header("Location: premMainPage.php");
    }
}
else {
    warning("Sorry, the information you entered is
        not valid. Please try again.");
}
}
else {
    warning("Sorry, your confirmation code is
        invalid");
}
}

//site_header(array('title'=>'Please log in to confirm
your new account.'));

if ($action == "confirm") {
    if ($email == "")
        { warning("You must enter your email address."); }
    if ($password == "")
        { warning("You must enter your password."); }
}
}
?>

<HTML>
<HEAD>
<TITLE>Confirm your account</TITLE>
</HEAD>

```

<P>To confirm your account, please enter your email address and your password below.</P>

```
<FORM method="post" action="confirm.php">
Email :<br>
<INPUT type="text" name="email" value="<? print $email;
?>" size="20" maxlength="50"><br>
Password :<br>
<INPUT type="password" name="password" value="<? print
$password; ?>" size="20" maxlength="10"><br>
<INPUT type="hidden" name="confirm_hash" value="<? print
$confirm_hash; ?>">
<INPUT type="hidden" name="action" value="confirm">
<INPUT type="submit" value="Submit">
</FORM>
</HTML>
<?php
box_bottom();
?>
```

```
//*****
// File name: login.php
//*****
```

```
<?php

session_start();
include "../include/site.php";
include "../include/database.php";

if (($action == "login") && ($form1_email != "") &&
($form1_password != ""))
{
    global $s_account_id ;
    $query = "SELECT trader_id, password FROM trader
              WHERE
              email='$form1_email' AND active='Y'";
    $result = mysql_query($query);

    if (mysql_num_rows($result) > 0) {
        $row = mysql_fetch_array($result);

        if (md5($form1_password) == $row['password'])
        {
            $trader_id = $row['trader_id'];
```

```

$query1 = "SELECT account_id, account_type
           FROM account
           WHERE trader_id = '$trader_id'";
$result1 = mysql_query($query1) or die
           ("Query failed");

session_register('s_account_id');
while ($row1 = mysql_fetch_array($result1))
{
    $s_account_id = $row1['account_id'];
    $account_type = $row1['account_type'];
}
get_user_data($s_account_id);

if ( $account_type == "Regular" )
{
    header("Location: regMainPage.php");
}
else if ( $account_type == "Premium" )
{
    header("Location: premMainPage.php");
}
}
else { warning("Sorry, the information you entered is
              not valid. Please try again."); }
}
else { warning("Sorry, the information you entered is
              not valid. Please try again."); }
}

site_header(array('title'=>'Please log in'));

if ($action == "login") {
    if ($form1_email == "")
    { warning("Your email cannot be left blank"); }
    if ($form1_password == "")
    { warning("You must enter a password"); }
}
?>

<TABLE width="90%" align="center"><TR><TD>
<font face="arial" size="2" color="black" >First Time
Visitors?&nbsp;
<A HREF="../../../login_signup/register.php">Please register a
new account</A>. It is free. You will learn how the stock

```

```

market works with the most realistic stock market
simulation on the Web.
<p>If you have previously registered at VCOST, Inc.,
please enter your e-mail address and password:
</font><br>
<FONT SIZE="-1" FACE="Arial, Helvetica"><BR>
<A href="../help/forgotpasswd.php">If you have forgotten
your password, you can request that we e-mail it to you.
<table border=0 cellpadding=3 cellspacing=4
width="600"><font size=-1></p>
<FORM method="post" action="login.php">
<tr>
    <td><font size=-1>Email:</font></td>
    <td nowrap>
        <INPUT type="text" name="form1_email" value="<? print
        $form1_email; ?>" size="33" maxlength="50">
    </td>
</tr>
<tr>
    <td><font size=-1>Password:</font></td>
    <td nowrap>
        <INPUT type="password" name="form1_password" value="<?
        print $form1_password; ?>" size="12" maxlength="20">
    </td>
</tr>
<tr>
    <td>
        <INPUT type="hidden" name="action" value="login">
        <INPUT type="submit" value="Login"></td>
    <td>
        <INPUT type="reset"></center>
    </td>
</tr>
</FORM>
</table>
<P><FONT FACE="Arial" SIZE="2">Problem Logging-in? Click
<A HREF="../help/forgotpasswd.htm">here</A></FONT></P>
<?php
box_bottom();
?>

```

```

//*****
// File name: logout.php
//*****

<?php

include "../include/udfunctions.php";
logout();

$message = "You have been successfully logout. If you want
to access your account, please re-login.";
print '<FONT color="#FF0000">'.$message.'</FONT><BR>';

?>

//*****
// File name: premMainPage.php
//*****

<?php

session_start();
include "../include/site.php";
include "../util/account_funcs.php";

site_header(array('title'=>'User Account Information'));

echo "<h3><i>$s_first_name $s_last_name, enjoy your stock
trading using VCOSTS Premium Account. Earn the real money!
Good Luck!!!</i></h3>";
get_prem_acc_summary ($s_account_id);
get_stock_summary ($s_account_id);

?>

<HTML>
<HEAD>
<TITLE>VCOSTS Account Summary</TITLE>
</HEAD>

<br><hr>
<table width="600" align="center" border="0"
cellspacing="0" cellpadding="0" bgcolor="#FFFFFF">
<TR BGCOLOR="#FFFFFF">
    <TD HEIGHT=15 WIDTH=60% ALIGN=center VALIGN=top>

```

```

        <FONT SIZE=1>
        <FORM ACTION="../transaction/buy_page.php"
method=Post><INPUT TYPE=SUBMIT VALUE="Buy Stock">
        </FORM>
        </FONT>
        </TD>
        <TD HEIGHT=15 WIDTH=40% ALIGN=left VALIGN=top>
        <FONT SIZE=1>
        <FORM ACTION="../transaction/sell_page.php"
method=Post>
        <INPUT TYPE=SUBMIT name=sell_stock VALUE="Sell
        Stock">
        </FORM>
        </FONT>
        </TD>
    </TR>
</table>

```

```
<?php
```

```

box_bottom();
?>

```

```

//*****
// File name: premRegister.php
//*****

```

```
<?php
```

```

include "../include/site.php";
include "../include/database.php";
include "../util/validate_funcs.php";

```

```

if ($action == "new")
{
    if (($form3_pass1 != "") &&
        ($form3_pass2 != "") &&
        ($form3_email != "") &&
        ($form3_pass1 == $form3_pass2)) {

        if ( validate_email($form3_email) )
        {
            //check if the email that user input is unique
            $query = "SELECT email FROM trader";
            $result = mysql_query($query) or die ("Query

```

```

failed");
$existed = 0;
for ($i=0; $i<mysql_num_rows($result); $i++)
{
    $row = mysql_fetch_array($result);
    $existed_email = $row['email'];
    if ( $existed_email == $form3_email )
    {
        $existed = 1;
        warning("Sorry!! You have already had an
        account with this email address \n"
        . "Please try using another email address
        if you still want \n"."a new account.");
    } //end of if
} //end of for

if ( $existed == 0 )
{
    //update trader table
    $confirm_hash = md5(strval(time()) .
                        strval(rand()));
    $query = "INSERT INTO trader (fname, middle_initial,
    lname, addr1, addr2, city, state, zcode, home_phone,
    work_phone, cell_phone, fax, email, password,
    confirm_hash)
    VALUES ('".$form3_fname."', '".$form3_mid."',
    '".$form3_lname."', '".$form3_addr1."',
    '".$form3_addr2."', '".$form3_city."',
    '".$form3_state."', '".$form3_zcode."',
    '".$form3_hphone."',
    '".$form3_wphone."', '".$form3_cphone."',
    '".$form3_fax."',
    '".$form3_email."',
    '".md5($form3_pass1)."', '".$confirm_hash."')";
    $result = mysql_query($query) or die ("Query
    failed");

    //update prem_trader table
    $query1 = "SELECT trader_id
    FROM trader
    WHERE email='".$form3_email'";
    $result1 = mysql_query($query1) or die ("Query
    failed");
    $trader_id = mysql_result($result1, 0,
    "trader_id");

```



```

$query = "INSERT INTO prem_trader (trader_id, SSN,
birthdate) VALUES ('".$trader_id."', ' ".$form3_ssn."', '
 ".$form3_birth."')";
$result = mysql_query($query) or die ("Query failed");

//update account table
$query = "INSERT INTO account (trader_id, account_type,
cash_balance) VALUES ('".$trader_id."', 'Premium',
'100000.00')";
$result = mysql_query($query) or die ("Query failed");

warning("An email has been sent to you. Please follow the
instructions to active your account.");

    $mail_text = "Thanks for requesting an account at
                    our website.\n". "Please click the
                    link below which contains
                    your account confirmation code:\n\n"
    ..../login_signup/confirm.php\?confirm_hash=".$confirm_has
h."\n\n". "Once you have confirmed your account you can
start to play with real\n". "time stocks immediately";
    mail($form3_email, "Your confirmation code for
        VCOSTS Inc.", $mail_text);
    }//end of if($existed=0)
} //end of if (the input email with validate format)

else
    { warning("Sorry!! Your email format is not
        correct. Please type a validate email
        address.");
    }
    } //end of if email and password box are not empty
} //end of if($action == "new")
site_header(array('title'=>'Please register with us'));

?>
<h1 align="center"><b><font color="#000080">Premium
Account Registration</font></b></h1>
<table border=0 cellpadding=3 cellspacing=4 width="604">
<font size=-1>
<FORM method="post" action="premRegister.php">
<tr>
    <td width="79"><font size=-1>Email:</font></td>
    <td nowrap width="292">
    <INPUT type="text" name="form3_email" value="<? print

```

```

        $form3_email; ?>" size="33" maxlength="50">
    </td>
    <td bgcolor="#eeeeed" width="193">
        <font class="s">
Please type your active email address in order to get the
confirmation code and further instructions to access your
account.</font>
        </td>
</tr>
<tr>
    <td width="79">
        <font size=-1>Password:</font>
    </td>
    <td nowrap width="292">
<INPUT type="password" name="form3_pass1" value="<? print
$form3_pass1; ?>" size="10" maxlength="22">
    </td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>Re-enter Password:</font>
    </td>
    <td width="292">
<INPUT type="password" name="form3_pass2" value="<? print
$form3_pass2; ?>" size="10" maxlength="22">
    </td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>First Name:</font>
    </td>
    <td width="292">
<INPUT type="text" name="form3_fname" value="<? print
$form3_fname; ?>" size="33" maxlength="50">
    </td>
    <td width="193">
<font size=-1>MI:<INPUT type="text" name="form3_mid"
value="<? print $form3_mid; ?>" size="2"
maxlength="5"></font></td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>Last Name:</font>
    </td>
    <td width="292">

```

```

<INPUT type="text" name="form3_lname" value="<? print
$form3_lname; ?>" size="33" maxlength="50">
</td>
</tr>
<tr>
<td width="79" >S.S.N.:</td>
<td width="292">
<INPUT type="text" name="form3_ssn" value="<? print
$form3_ssn; ?>" size="33" maxlength="50">
</td>
</tr>
<tr>
<td width="79" >
<font size=-1>Date of </font>Birth:&nbsp;
</td>
<td width="292">
<INPUT type="text" name="form3_birth" value="<? print
$form3_birth; ?>" size="33" maxlength="50">
</td>
</tr>
<tr>
<td width="79" >
<font size=-1>Address1:</font>
</td>
<td width="292">
<INPUT type="text" name="form3_addr1" value="<? print
$form3_addr1; ?>" size="40" maxlength="60">
</td>
</tr>
<tr>
<td width="79" >
<font size=-1>Address2:</font>
</td>
<td width="292">
<INPUT type="text" name="form3_addr2" value="<? print
$form3_addr2; ?>" size="40" maxlength="60">
</td>
</tr>
<tr>
<td width="79" >
<font size=-1>City:</font>
</td>
<td width="292">
<INPUT type="text" name="form3_city" value="<? print
$form3_city; ?>" size="14" maxlength="30">

```

```

        </td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>State:</font>
    </td>
    <td width="292">
        <INPUT type="text" name="form3_state" value="<? print
        $form3_state; ?>" size="14" maxlength="25">
    </td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>Zip:</font>
    </td>
    <td width="292">
        <INPUT type="text" name="form3_zcode" value="<? print
        $form3_zcode; ?>" size="12" maxlength="20">
    </td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>Home Phone Number:</font>
    </td>
    <td width="292">
        <INPUT type="text" name="form3_hphone" value="<?
        print $form3_hphone; ?>" size="9" maxlength="20">
    </td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>Work Phone Number:</font>
    </td>
    <td width="292">
        <INPUT type="text" name="form3_wphone" value="<?
        print $form3_wphone; ?>" size="9" maxlength="20">
    </td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>Cell Phone Number:</font>
    </td>
    <td width="292">
        <INPUT type="text" name="form3_cphone" value="<?
        print $form3_cphone; ?>" size="9" maxlength="20">
    </td>
</tr>

```

```

        </td>
</tr>
<tr>
    <td width="79" >
        <font size=-1>Fax Number:</font>
    </td>
    <td width="292">
        <INPUT type="text" name="form3_fax" value="<? print
        $form3_fax;?>" size="9" maxlength="20">
    </td>
</tr>
<tr>
    <td width="79"></td>
    <td width="292"><center>
        <INPUT type="hidden" name="action" value="new">
        <INPUT type="submit" value="Submit">
        <INPUT type="reset"></center>
    </td>
    <td width="193"></td>
</tr>
</FORM>
</font>
</table>

```

```

//*****
// File name: register.php
//*****

```

```

<?php
session_start();

include "../include/site.php";
include "../include/database.php";
include "../util/validate_funcs.php";

if ($action == "new")
{
    if (($form2_pass1 != "") &&
        ($form2_pass2 != "") &&
        ($form2_email != "") &&
        ($form2_pass1 == $form2_pass2)) {

        if ( validate_email($form2_email) )
        {
            //check if the email that user input is unique

```

```

$query = "SELECT email FROM trader";
$result = mysql_query($query) or die ("Query
failed");
$existed = 0;
for ($i=0; $i<mysql_num_rows($result); $i++)
{
    $row = mysql_fetch_array($result);
    $existed_email = $row['email'];

    if ( $existed_email == $form2_email ) {
        $existed = 1;
        warning("Sorry!! You have already had
an account with this email
address \n"."Please try
using another email address
if you still want \n"."a new
account.");
    } //end of if
} //end of for

if ( $existed == 0 )
{
    //update trader table
    $confirm_hash = md5(strval(time()) .
strval(rand()));
    $query = "INSERT INTO trader (fname,
middle_initial, lname, addr1, addr2,city, state, zipcode,
home_phone, work_phone, cell_phone, fax, email,password,
confirm_hash) VALUES ('".$form2_fname."',
'".$form2_mid."', '".$form2_lname."', '".$form2_addr1."',
'".$form2_addr2."', '".$form2_city."', '".$form2_state."',
'".$form2_zipcode."', '".$form2_hphone."', '".$form2_wphone.
"', '".$form2_cphone."', '".$form2_fax."', '".$form2_email.
"', '".md5($form2_pass1)."', '".$confirm_hash."')";
    $result = mysql_query($query) or die ("Query
failed");

    //update account table
    $query = "SELECT trader_id
FROM trader
WHERE email='".$form2_email'";
    $result = mysql_query($query) or die ("Query
failed");
    $trader_id = mysql_result($result, 0,
"trader_id");

```

```

        $query = "INSERT INTO account (trader_id,
                                account_type, cash_balance)
                                VALUES ('".$trader_id."',
                                'Regular', '100000.00')";
        $result = mysql_query($query) or die ("Query
        failed");
        warning("An email has been sent to you.
        Please follow the instructions to active your account.");
        $mail_text = "Thanks for requesting an
        account at our website.\n" . "Please click the link below
        which contains your account confirmation code:\n\n".
        "../login_signup/confirm.php\?confirm_hash=".$confirm_hash
        ."\n\n" . "Once you have confirmed your account
        you can start to play with real\n". "time stocks
        immediately";

        mail($form2_email,"Your confirmation code for
        VCOSTS Inc.", $mail_text);
        }//end of if($existed=0)
    }//end of if (the input email with validate format)

    else
    { warning("Sorry!! Your email format is not
    correct. Please type a validate email address.");
    }
    }//end of if email and password box are not empty
} //end of if($action == "new")
site_header(array('title'=>'Please register with us'));

?>
<h1 align="center"><b><font color="#000080">Regular
Account Registration</font></b></h1>
<TABLE width="90%" align="center"><TR><TD>
<P>If you do not have an account with us, please enter
your email address in the box below. We will send you a
confirmation code and further instructions.</P>

<?
if ($action == "new") {
    if ($form2_email == "") { warning("Sorry!! Your email
    cannot be left blank"); }
    if ($form2_pass1 == "") { warning("Sorry!! You must
    enter a password"); }
    if ($form2_pass2 == "") { warning("Sorry!! You must
    re-enter your password"); }

```

```

        if ($form2_pass1 != $form2_pass2) { warning("Sorry!!
The passwords did not match. Please re-enter them.");
    }
}

```

```

?>

```

```

<table border=0 cellpadding=3 cellspacing=4
width="590"><font size=-1>
<FORM method="post" action="register.php">
<tr>
    <td>
        <font size=-1>Email:</font>
    </td>
    <td nowrap>
        <INPUT type="text" name="form2_email" value="<? print
$form2_email; ?>" size="33" maxlength="50">
    </td>
    <td bgcolor="#eeeeed">
        <font class="s">Please type your active email address in
order to get the confirmation code and further
instructions to access your account.</font>
    </td>
</tr>
<tr>
    <td>
        <font size=-1>Password:</font>
    </td>
    <td nowrap>
        <INPUT type="password" name="form2_pass1" value="<? print
$form2_pass1; ?>" size="10" maxlength="22">
    </td>
</tr>
<tr>
    <td >
        <font size=-1>Re-enter Password:</font>
    </td>
    <td>
        <INPUT type="password" name="form2_pass2" value="<? print
$form2_pass2; ?>" size="10" maxlength="22">
    </td>
</tr>
<tr>
    <td >
        <font size=-1>First Name,MI:</font>

```



```

        </td>
        <td>
<INPUT type="text" name="form2_fname" value="<? print
$form2_fname; ?>" size="33" maxlength="50">
        </td>
        <td>
<INPUT type="text" name="form2_mid" value="<? print
$form2_mid; ?>" size="2" maxlength="5">
        </td>
</tr>
<tr>
        <td >
        <font size=-1>Last Name:</font>
        </td>
        <td>
<INPUT type="text" name="form2_lname" value="<? print
$form2_lname; ?>" size="33" maxlength="50">
        </td>
</tr>
<tr>
        <td >
        <font size=-1>Address1:</font>
        </td>
        <td>
<INPUT type="text" name="form2_addr1" value="<? print
$form2_addr1; ?>" size="40" maxlength="60">
        </td>
</tr>
<tr>
        <td >
        <font size=-1>Address2:</font>
        </td>
        <td>
<INPUT type="text" name="form2_addr2" value="<? print
$form2_addr2; ?>" size="40" maxlength="60">
        </td>
</tr>
<tr>
        <td >
        <font size=-1>City:</font>
        </td>
        <td>
<INPUT type="text" name="form2_city" value="<? print
$form2_city; ?>" size="14" maxlength="30">
        </td>

```

```

</tr>
<tr>
  <td >
    <font size=-1>State:</font>
  </td>
  <td>
    <INPUT type="text" name="form2_state" value="<? print
    $form2_state; ?>" size="14" maxlength="25">
  </td>
</tr>
<tr>
  <td >
    <font size=-1>Zip:</font>
  </td>
  <td>
    <INPUT type="text" name="form2_zcode" value="<? print
    $form2_zcode; ?>" size="12" maxlength="20">
  </td>
</tr>
<tr>
  <td >
    <font size=-1>Home Phone Number:</font>
  </td>
  <td>
    <INPUT type="text" name="form2_hphone" value="<? print
    $form2_hphone; ?>" size="9" maxlength="20">
  </td>
</tr>
<tr>
  <td >
    <font size=-1>Work Phone Number:</font>
  </td>
  <td>
    <INPUT type="text" name="form2_wphone" value="<? print
    $form2_wphone; ?>" size="9" maxlength="20">
  </td>
</tr>
<tr>
  <td >
    <font size=-1>Cell Phone Number:</font>
  </td>
  <td>
    <INPUT type="text" name="form2_cphone" value="<? print
    $form2_cphone; ?>" size="9" maxlength="20">
  </td>

```

```

</tr>
<tr>
    <td >
        <font size=-1>Fax Number:</font>
    </td>
    <td>
        <INPUT type="text" name="form2_fax" value="<? print
        $form2_fax; ?>" size="9" maxlength="20">
    </td>
</tr>
<tr>
    <td></td>
    <td><center>
        <INPUT type="hidden" name="action" value="new">
        <INPUT type="submit" value="Submit">
        <INPUT type="reset"></center>
    </td>
    <td></td>
</tr>
</FORM>
</font>
</table>

```

```

<?php

```

```

box_bottom();
?>

```

```

//*****
// File name: regMainPage.php
//*****

```

```

<?php

```

```

session_start();
include "../include/site.php";
include "../util/account_funcs.php";

site_header(array('title'=>'User Account Information'));
echo "<h3><i>$s_first_name $s_last_name, enjoy your stock
trading using VCOSTS! Good Luck!!</i></h3>";
get_regl_acc_summary ($s_account_id);
get_stock_summary ($s_account_id);

?>

```

```

<HTML>
<HEAD>
<TITLE>VCOSTS Account Summary</TITLE></HEAD>
<table width="600" align="center" border="0"
cellspacing="0" cellpadding="0" bgcolor="#FFFFFF">
<TR BGCOLOR="#FFFFFF">
    <TD HEIGHT=15 WIDTH=60% ALIGN=center VALIGN=top>
        <FONT SIZE=1>
        <FORM ACTION="../../../transaction/buy_page.php"
        method=Post>
        <INPUT TYPE=SUBMIT VALUE="Buy Stock">
        </FORM>
        </FONT>
    </TD>
    <TD HEIGHT=15 WIDTH=40% ALIGN=left VALIGN=top>
        <FONT SIZE=1>
        <FORM ACTION="../../../transaction/sell_page.php"
        method=Post>
        <INPUT TYPE=SUBMIT name=sell_stock VALUE="Sell
        Stock">
        </FORM>
        </FONT>
    </TD>
</TR>
</table>

<?php
box_bottom();
?>

//*****
// File name: buy_page.php
//*****

<?php
session_start();

include "../include/site.php";
include "../include/database.php";
include "../util/validate_funcs.php";
include "../util/account_funcs.php";

site_header(array('title'=>'Buy Page'));

```

```

if (($action == "buy") && ($symbol != "") && ($quantity !=
"")&& ($price != ""))
{
//check if account balance is enough to buy, and hold the
money for this buy
require("../util/ans_funcs.php");
$buy = new CBuy;
$buy->checkBalance_beforeBuy ($quantity, $price,
$s_account_id);

if ( $buy->enough == 1 )
{
    switch($expire) {
        case "Until Market Close Today":

            $today = getdate();
            $hours = $today[hours];
            $minutes = $today[minutes];

            if( ($hours > 16) || (($hours == 16) && ($minutes
>= 30)) || ($hours < 9) || (($hours == 9) && ($minutes <
30) ) )
            {
                warning ("Sorry, the Market has already been
closed today, please use extended hours trading.");
            }

            else
            {
//get the current year
                $year = $today[year];

//get the current month
                if ($today[mon]<10) $mon = "0".$today[mon];
                else $mon = $today[mon];

//get the current day
                if ($today[mday]<10) $mday = "0".$today[mday];
                else $mday = $today[mday];

//get the end_time for today market close time --16:30
                $end_time = $year . $mon . $mday. "163000";

                $query = "INSERT INTO transaction (account_id,

```

```

        transaction_type, symbol,
        stock_quantity,
        trader_price, start_time, end_time,
        status, trans_time_type )
        VALUES ('".$s_account_id."', 'buy',
        '".$symbol."', '".$quantity."',
        '".$price."',
        NOW(), '".$end_time."', 'pending',
        'day')";
        $result = mysql_query($query) or die ("Query
        failed");
        warning("Your buy request has been submitted
        successfully. Thanks!");
    }
    break;

    case "1 (Day + Ext. Hours)" :
        $showmanydays = 1;
        $end_time = get_end_time ($showmanydays);
        $query = "INSERT INTO transaction (account_id,
        transaction_type,symbol, stock_quantity, trader_price,
        start_time, end_time, status, trans_time_type )
        VALUES ('".$s_account_id."', 'buy', '".$symbol."',
        '".$quantity."', '".$price."', NOW(), '".$end_time."',
        'pending', 'day_ext')";
        $result = mysql_query($query) or die ("Query
        failed");

        warning("Your buy request has been submitted
        successfully. Thanks!");
    break;

    case "2 (Day + Ext. Hours)" :
        $showmanydays = 2;
        $end_time = get_end_time ($showmanydays);
        $query = "INSERT INTO transaction (account_id,
        transaction_type, symbol, stock_quantity, trader_price,
        start_time, end_time, status, trans_time_type )
        VALUES ('".$s_account_id."', 'buy',
        '".$symbol."', '".$quantity."', '".$price."', NOW(),
        '".$end_time."', 'pending', 'day_ext')";
        $result = mysql_query($query) or die ("Query
        failed");
        warning("Your buy request has been submitted
        successfully. Thanks!");

```

```

break;

case "3 (Day + Ext. Hours)" :
    $showmanydays = 3;
    $end_time = get_end_time ($showmanydays);
    $query = "INSERT INTO transaction (account_id,
transaction_type,symbol, stock_quantity, trader_price,
start_time, end_time, status, trans_time_type )
VALUES ('".$s_account_id."', 'buy',
'".$symbol."', '".$quantity."', '".$price."',
NOW(), '".$end_time."', 'pending', 'day_ext')";
    $result = mysql_query($query) or die ("Query
failed");
    warning("Your buy request has been submitted
successfully. Thanks!");
break;

case "1 Week" :
    $showmanydays = 7;
    $end_time = get_end_time ($showmanydays);

    $query = "INSERT INTO transaction (account_id,
transaction_type, symbol, stock_quantity,trader_price,
start_time, end_time, status, trans_time_type )
VALUES ('".$s_account_id."', 'buy',
'".$symbol."', '".$quantity."', '".$price."',
NOW(), '".$end_time."', 'pending', 'day_ext')";
    $result = mysql_query($query) or die ("Query failed");
    warning("Your buy request has been submitted
successfully. Thanks!");
} //end of switch(expire)
} //end of if ( $enough_money = 1 )

else { warning("Sorry, you don't have enough money to
finish this trading.");
}

} //end of if $action == "buy"

if ($action == "buy") {
    if ($symbol == "") { warning("You must enter a stock
symbol that you want to buy."); }
    if ($quantity == "") { warning("You must enter the
quantity of the stock that you want to buy."); }

```

```

        if ($price == "") { warning("You must enter the price
that you want to buy this stock.");}
    }
?>

<?php
//echo "<P>Welcome $s_first_name $s_last_name! </P>";
?>

<h2 align="center"><b><span style="background-color:
#8080c0"><font color="#FFFFCC">Please fill out this form
to BUY stock.</font> </span></b></h2>
<form method="POST" action="buy_page.php" onSubmit="">
<table>
<TR>
<TD align=center>Symbol</TD>
<TD align=center>Quantity</TD>
<TD align=center>Price</TD>
<TD align=center>Expire</TD>
</TR>
<TR>
<TD align=center><input type="text" name="symbol"
size="20"></TD>
<TD align=center><input type="text" name="quantity"
size="20"></TD>
<TD align=center><input type="text" name="price"
size="20"></TD>
<TD align=center><select size="1" name="expire">
    <option selected value="Until Market Close
Today">Until Market Close Today</option>
    <option value="1 (Day + Ext. Hours)">1 (Day + Ext.
Hours)</option>
    <option value="2 (Day + Ext. Hours)">2 (Day + Ext.
Hours)</option>
    <option value="3 (Day + Ext. Hours)">3 (Day + Ext.
Hours)</option>
    <option value="1 Week">1 Week</option>
</select></TD>
</TR>
</table>
<p>
<INPUT type="hidden" name="action" value="buy">
<input type="submit" value="Submit" name="B1">
<input type="reset" value="Reset" name="B2"></p>
</form>

```



```

<?php

get_stock_summary ($s_account_id);
echo "<p>&nbsp;</p><p>&nbsp;</p>";
box_bottom();

?>

//*****
// File name: sell_page.php
//*****

<?php

session_start();

include "../include/site.php";
include "../include/database.php";
include "../util/validate_funcs.php";
include "../util/account_funcs.php";

if (($action == "sell") && ($symbol != "") && ($quantity
!= "") && ($price != ""))
{

//check if the seller has enough stock for sale
require("../util/ans_funcs.php");
$sell = new CSell;
$sell->checkStock_beforeSell ($symbol, $quantity,
$s_account_id);

if ( $sell->enoughStock == 1 )
{
    switch($expire) {
    case "Until Market Close Today":
        $today = getdate();
        $hours = $today[hours];
        $minutes = $today[minutes];
        if( ($hours > 16) || (($hours == 16) && ($minutes
            >= 30)) || ($hours < 9) || (($hours == 9) &&
            ($minutes < 30) ) )
        {
            warning ("Sorry, the Market has already been
                closed today, please use extended

```

```

        hours trading.");
    }

    else
    {
//get the current year
        $year = $today[year];
//get the current month
        if ($today[mon]<10) $mon = "0".$today[mon];
        else $mon = $today[mon];
//get the current day
        if ($today[mday]<10) $mday="0".$today[mday];
        else $mday = $today[mday];
//get the end_time for today market close time --16:30
        $end_time = $year . $mon . $mday. "163000";
        $query = "INSERT INTO transaction
            (account_id, transaction_type,
            symbol, stock_quantity,
            trader_price, start_time,
            end_time,status, trans_time_type )
            VALUES ('".$ss_account_id."', 'sell',
            '".$symbol."', '".$quantity."',
            '".$price."',NOW(), '".$end_time."',
            'pending', 'day')";
        $result = mysql_query($query) or die ("Query
            failed");
        warning("Your sell request has been submitted
            successfully. Thanks!");
    }
break;

case "1 (Day + Ext. Hours)" :
    $showmanydays = 1;
    $end_time = get_end_time ($showmanydays);
    $query = "INSERT INTO transaction (account_id,
transaction_type, symbol, stock_quantity, trader_price,
start_time, end_time, status, trans_time_type )
            VALUES ('".$ss_account_id."', 'sell',
            '".$symbol."', '".$quantity."', '".$price."', NOW(),
            '".$end_time."', 'pending', 'day_ext')";
    $result = mysql_query($query) or die ("Query
        failed");
    warning("Your sell request has been submitted
        successfully. Thanks!");
break;

```

```

case "2 (Day + Ext. Hours)" :
    $showmanydays = 2;
    $end_time = get_end_time ($showmanydays);
    $query = "INSERT INTO transaction (account_id,
transaction_type,symbol, stock_quantity, trader_price,
start_time, end_time, status, trans_time_type )
VALUES ('".$s_account_id."', 'sell', '".$symbol."',
'".$quantity."', '".$price."',NOW(), '".$end_time."',
'pending', 'day_ext')";
    $result = mysql_query($query) or die ("Query
        failed");

    warning("Your sell request has been submitted
        successfully. Thanks!");
    break;

case "3 (Day + Ext. Hours)" :
    $showmanydays = 3;
    $end_time = get_end_time ($showmanydays);
    $query = "INSERT INTO transaction (account_id,
transaction_type, symbol, stock_quantity, trader_price,
start_time, end_time, status, trans_time_type )
VALUES ('".$s_account_id."', 'sell',
'".$symbol."', '".$quantity."', '".$price."',
NOW(), '".$end_time."', 'pending', 'day_ext')";
    $result = mysql_query($query) or die ("Query failed");
    warning("Your sell request has been submitted
        successfully. Thanks!");
    break;

case "1 Week" :
    $showmanydays = 7;
    $end_time = get_end_time ($showmanydays);
    $query = "INSERT INTO transaction (account_id,
transaction_type, symbol, stock_quantity, trader_price,
start_time, end_time, status, trans_time_type )
VALUES ('".$s_account_id."', 'sell',
'".$symbol."', '".$quantity."', '".$price."',
NOW(), '".$end_time."', 'pending', 'day_ext')";
    $result = mysql_query($query) or die ("Query
        failed");
    warning("Your sell request has been submitted
        successfully. Thanks!");
} //end of switch(expire)

```

```

    } //end of if ( $enoughStock = 1 )

else
{ warning("Sorry, you don't have enough stock to finish
this trading."); }
} //end of if $action == "sell"

site_header(array('title'=>'Sell Page'));

if ($action == "sell") {
    if ($symbol == "")
        { warning("You must enter a stock symbol that you
            want to buy.");
        }
    if ($quantity == "")
        { warning("You must enter the quantity of the stock
            that you want to buy.");
        }
    if ($price == "")
        { warning("You must enter the price that you want to
            buy this stock."); }
}

?>

<h2 align="center"><b><span style="background-color:
#8080c0"><font color="#FFFFCC">Please fill out this form
to SELL the stock in your account.</font></span></b></h2>
<form method="POST" action="sell_page.php" onSubmit="">
<table>
<TR>
<TD align=center>Symbol</TD>
<TD align=center>Quantity</TD>
<TD align=center>Price</TD>
<TD align=center>Expire</TD>
</TR>
<TR>
<TD align=center><input type="text" name="symbol"
size="20"></TD>
<TD align=center><input type="text" name="quantity"
size="20"></TD>
<TD align=center><input type="text" name="price"
size="20"></TD>
<TD align=center><select size="1" name="expire">
    <option selected value="Until Market Close

```

```

        Today">Until Market Close Today</option>
        <option value="1 (Day + Ext. Hours)">1 (Day + Ext.
        Hours)</option>
        <option value="2 (Day + Ext. Hours)">2 (Day + Ext.
        Hours)</option>
        <option value="3 (Day + Ext. Hours)">3 (Day + Ext.
        Hours)</option>
        <option value="1 Week">1 Week</option>
    </select></TD>
</TR>
</table>

<p>
    <INPUT type="hidden" name="action" value="sell">
    <input type="submit" value="Submit" name="B1">
    <input type="reset" value="Reset" name="B2"></p>
</form>

<?php
get_stock_summary ($s_account_id);
echo "<p>&nbsp;</p><p>&nbsp;</p>";
box_bottom();

?>

//*****
// File name: account_funcs.php
//*****

<?php

include "../include/database.php";
function get_stock_value($acct_id)
{
    $query = "SELECT sum(a.stock_quantity *
                b.last_trade_price)
                as stock_total
                FROM account_stock a, stock b
                WHERE b.symbol=a.stock_symbol and
                        account_id='$acct_id'";
    $result = mysql_query($query) or die ("Get stock
                value Query failed");

    while ($row = mysql_fetch_array($result))
    {

```

```

        return $stock_total = $row['stock_total'];
    }
}

function get_account_value($acct_id)
{
    $stock_total = get_stock_value($acct_id);
    $result = mysql_query( "SELECT cash_balance
                           FROM account
                           WHERE account_id='$acct_id'");
    while ($row = mysql_fetch_array($result)) {
        $cash_balance = $row['cash_balance'];
    }

    return $account_value = $cash_balance + $stock_total;
}

/*
function update_regl_account($acct_id)
{
    $query = "SELECT sum(a.stock_quantity *
                    b.last_trade_price)
              as stock_total
              FROM account_stock a, stock b
              WHERE b.symbol=a.stock_symbol and
                    account_id='$acct_id'";
    $result = mysql_query($query) or die ("Query failed");
    while ($row = mysql_fetch_array($result))
    {
        $stock_total = $row['stock_total'];
    }
    mysql_query( "UPDATE account
                  SET stock_value = $stock_total
                  WHERE account_id='$acct_id'");
}

function update_prem_account($acct_id) {
    $reward_rate = 0.002;

    //calculate all stocks value, then update stock_value in
    account table
    $query = "SELECT sum(a.stock_quantity *
                    b.last_trade_price)
              as stock_total, original_value

```

```

        FROM account_stock a, stock b
        WHERE b.symbol=a.stock_symbol and
              account_id='$acct_id';
$result = mysql_query($query) or die ("Query failed");
while ($row = mysql_fetch_array($result))
{
    $stock_total = $row['stock_total'];
    $orig_val = $row['original_value'];
}

mysql_query( "UPDATE account
             SET stock_value = $stock_total
             WHERE account_id='$acct_id'");

//update reward in prem_trader
get_account_value($acct_id);

if (($earn=$account_value-$orig_val)>0) {
    $reward = $earn * $reward_rate;
}
else { $reward = 0; }

mysql_query( "UPDATE prem_trader
             SET reward = $reward
             WHERE account_id='$acct_id'");
}
*/

function get_reward ($acct_id)
{
    $query = "SELECT reward_pct, original_value
             FROM account
             WHERE account_id='$acct_id'";
    $result = mysql_query($query) or die ("Query
             failed");
    while ($row = mysql_fetch_array($result)) {
        $reward_pct = $row['reward_pct'];
        $original_value = $row['original_value'];
    }

    $account_value = get_account_value($acct_id);
    return $reward = ($account_value -
        $original_value) * reward_pct;
}

```

```

function get_regl_acc_summary ($acct_id)
{
    $stock_value = get_stock_value($acct_id);
    $account_value = get_account_value($acct_id);
    $cash_balance = $account_value - $stock_value;
    echo "<h3><font color='#000080'><b>Account
    Summary</b></font></h3>
<table>
    <tr>
        <td width='128\' bgcolor='#99CCFF'>Available
Cash</td>
        <td width='228\' bgcolor='#99CCFF'> $cash_balance
        </td>
    </tr>
    <tr>
        <td width='128\' bgcolor='#FFCCCC'>Stock Value</td>
        <td width='228\' bgcolor='#FFCCCC'>$stock_value</td>
    </tr>
    <tr>
        <td width='128\' bgcolor='#C0C0C0'><span
        style='background-color:
        #C0C0C0'>Account Value </span></td>
        <td width='228\' bgcolor='#C0C0C0'><span
        style='background-color:
        #C0C0C0'>$account_value</span></td>
    </tr>
</table>
";
} //end of function get_regl_acc_summary ($acct_id)

function get_stock_summary ($acct_id)
{
    $query = "SELECT a.stock_symbol, a.stock_quantity ,
                b.last_trade_price, b.day_change,
                b.day_high, b.day_low
    FROM account_stock a, stock b
    WHERE b.symbol=a.stock_symbol and
            account_id='$acct_id'";

    $result = mysql_query($query) or die ("Query failed");
    //the title of stock summary table
    echo "<h3><font color='#000080'><b>Stock in Your
    Account</b></font> </h3>";
    echo "<TABLE BORDER=1 width='555'>";
    echo "<TR>

```



```

<TH width='92' align='center' style='background-color:
#8080C0'>
<font color='#FFFFFF'>Symbol</font></TH>
<TH width='150' align='center' style='background-color:
#8080C0'>
<font color='#FFFFFF'>Original Quantity</font></TH>
<TH width='92' align='center' style='background-color:
#8080C0'>
<font color='#FFFFFF'>Last Price</font></TH>
<TH width='92' align='center' style='background-color:
#8080C0'>
<font color='#FFFFFF'>Day Change</font></TH>
<TH width='92' align='center' style='background-color:
#8080C0'>
<font color='#FFFFFF'>Day High</font></TH>
<TH width='92' align='center' style='background-color:
#8080C0'>
<font color='#FFFFFF'>Day Low</font></TH>
<TH width='93' align='center' style='background-color:
#8080C0'>
<font color='#FFFFFF'>Value($)</font></TH></TR>";
//display the content of the stock summary table
// while ($row = mysql_fetch_array($result))
    for ($i=0; $i<mysql_num_rows($result); $i++)
    {
        $row = mysql_fetch_array($result);
        $stock = $row['stock_symbol'];
        $quantity = $row['stock_quantity'];
        $last = $row['last_trade_price'];
        $day_change = $row['day_change'];
        $day_high = $row['day_high'];
        $day_low = $row['day_low'];
        $value = $last * $quantity;
echo "<TR>
<TD width='92' align='center'
bgcolor='#FFFFFF'>$stock</TD>
<TD width='150' align='center'
bgcolor='#FFFFFF'>$quantity</TD>
<TD width='92' align='center' bgcolor='#FFFFFF'>$last</TD>
<TD width='92' align='center'
bgcolor='#FFFFFF'>$day_change</TD>
<TD width='92' align='center'
bgcolor='#FFFFFF'>$day_high</TD>
<TD width='92' align='center'
bgcolor='#FFFFFF'>$day_low</TD>

```

```

<TD width='93' align='center'
bgcolor='#FFFFFF'>$value</TD></TR>";

    }//end of while
    echo "</table>";
} // end of function get_stock_summary ($acct_id)

function get_prem_acc_summary ($acct_id) {
    $stock_value = get_stock_value($acct_id);
    $account_value = get_account_value($acct_id);
    $cash_balance = $account_value - $stock_value;
    $reward = get_reward($acct_id);
    echo "<h3><font color='#000080'><b>Account
Summary</b></font></h3>
<table>
    <tr>
        <td width='128\' bgcolor='#99CCFF'>Available
Cash</td>
        <td width='228\' bgcolor='#99CCFF'> $cash_balance
</td>
    </tr>
    <tr>
        <td width='128\' bgcolor='#FFFFCC'>Stock Value</td>
        <td width='228\' bgcolor='#FFFFCC'>$stock_value</td>
    </tr>
    <tr>
        <td width='128\' bgcolor='#FFCCCC'>Account Value</td>
        <td width='228\' bgcolor='#FFCCCC'>$account_value
        </td>
    </tr>
    <tr>
        <td width='128\' bgcolor='#C0C0C0'>Reward</td>
        <td width='228\' bgcolor='#C0C0C0'>$reward</td>
    </tr>
</table>";
} //end of function get_prem_acc_summary ($acct_id)

?>

//*****
// File name: ans_funcs.php
//*****
<?php

include "../include/database.php";

```

```

class CBuy
{
    function updateStock_afterBuy($symbol, $quantity,
                                   $acct_id)
    {
        $query = "SELECT  stock_symbol FROM account_stock
                    WHERE account_id='$acct_id' ";
        $result = mysql_query($query) or die ("Query
                    failed");
        $this->flag = 0;
        for ($i=0; $i<mysql_num_rows($result); $i++)
        {
            $row = mysql_fetch_array($result);
            $this->symb = $row['stock_symbol'];
            // printf("%s\n", $symb);
            if ( $this->symb == $symbol )
            {
                mysql_query( "UPDATE account_stock
                             SET stock_quantity +=
                             $quantity
                             WHERE account_id='$acct_id'
                             AND stock_symbol='$symbol'");
                $this->flag = 1;
            } //end of if
        } //end of for

        if ($this->flag == 0)
        {
            mysql_query( "INSERT INTO account_stock
                        (account_id, stock_symbol,
                         stock_quantity)
                        VALUES ('$acct_id', '$symbol',
                         '$quantity')");
        }
    } //end of function updateStock_afterBuy

    function checkBalance_beforeBuy ($quantity, $price,
                                      $acct_id)
    {
        $this->value = $price * $quantity;
        $query = "SELECT cash_balance
                  FROM account
                  WHERE account_id='$acct_id' ";
        $result = mysql_query($query) or die ("Query

```

```

        cash_balance from account failed");
        while ($row = mysql_fetch_array($result))
        {
            // $bal = $row['cash_balance'];
            $this->balance = $row['cash_balance'];
        }

        if ( $this->balance >= $this->value )
        {
            mysql_query( "UPDATE account
            SET cash_balance = cash_balance - $this->value
                        WHERE account_id='$acct_id'");
            $this->enough = 1;
        }

        else if ( $this->balance < $this->value )
        {
            $this->enough = 0;
        }
    } //end of function checkAcct_beforeBuy

    function returnBalance_ifnotBuy ($quantity, $price,
                                    $acct_id)
    {
        $value = $price * $quantity;
        $query = "UPDATE account
                  SET cash_balance += $value
                  WHERE account_id='$acct_id'";
        $result = mysql_query($query) or die ("Query
        returnBalance_ifnotBuy failed");
    }

} //end of class CBuy

class CSell{
    function checkStock_beforeSell ($symbol, $quantity,
                                    $acct_id)
    {
        $query = "SELECT stock_quantity FROM account_stock
                  WHERE account_id='$acct_id'
                  AND stock_symbol='$symbol'";
        $result = mysql_query($query) or die ("Query
        failed");

        if (mysql_num_rows($result) > 0)

```

```

        {
            $row = mysql_fetch_array($result);
            $stock_quantity = $row['stock_quantity'];
            if ( $stock_quantity >= $quantity )
            {
                return $this->enoughStock = 1;
            }
            else
                return $this->enoughStock = 0;
        }

        else
        {
            return $this->enoughStock = 0;
        }
    } //end of function checkStock_beforeSell ($symbol,
    $quantity, $acct_id)

    function updateStock_afterSell($symbol, $quantity,
                                   $acct_id)
    {
        $query = "UPDATE account_stock
                  SET stock_quantity = stock_quantity -
                    $quantity
                  WHERE account_id='$acct_id'
                    AND stock_symbol='$symbol' ";
        $result = mysql_query($query) or die ("Query111
        failed");
    } //end of function updateStock_afterSell

    function updateBalance_afterSell($quantity, $price,
                                     $acct_id)
    {
        $value = $price * $quantity;
        $query = "UPDATE account
                  SET cash_balance = cash_balance + $value
                  WHERE account_id='$acct_id'";
        $result = mysql_query($query) or die ("Query account
        failed");
    }
} //end of class CSell
?>

```

```

/*****
// File name: validate_funcs.php
/*****
<?php

function get_end_time ($expire_dates)
{
    $today = getdate();
    $year = $today[year];
    $mon = $today[mon];
    $mday1 = $today[mday];
    $mday = $mday1 + $expire_dates;

    if ( ($mon == 01 || 03 || 05 || 07 || 08 || 10 || 12) &&
($mday > 31))
    {
        $mon ++;
        $mday = $mday - 31;
    }
    if ($mon == 13 ) $mon =1;
    if ( ($mon == 02 || 04 || 06 || 09 || 11) && ($mday >
30))
    {
        $mon ++;
        $mday = $mday - 30;
    }
    if ($mon<10) $mon = "0".$mon;
    if ($mday<10) $mday = "0".$mday;
    //get time
    if ($today[hours]<10) $hours = "0".$today[hours];
    else $hours = $today[hours];
    if ($today[minutes]<10)
        $minutes = "0".$today[minutes];
    else $minutes = $today[minutes];

    if ($today[seconds]<10)
        $seconds = "0".$today[seconds];
    else $seconds = $today[seconds];

    //get the end_time
    $end_time =
        $year.$mon.$mday.$hours.$minutes.$seconds;
    return $end_time;
}

```

```

//email confirm function
function validate_email ($aemail) {
//return true if it is a valid email, otherwise return
false
    return (ereg('^[~!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+'.'
        '@'.'
        '[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+'.'.'
        '[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+'.$',
        $aemail));
}
//numbers confirmation
function validate_numbers ($string) {
//return true if string is numbers, return false if it is
not all numbers
    return (eregi("[0-9]+$", $string));
}
?>

```

BIBLIOGRAPHY

- Klein,A.(1998). *Fat cat investing at the click of a mouse: how andy klein and the internet can give everyone a seat on the exchange*. New York: Henry Holt and Company.
- Farley,A.(2000). *The master swing trader: tools and techniques to profit from outstanding short-term trading opportunities*. New York: McGraw-Hill Professional Publishing.
- Labier,R., & LaBier,R.(2000). *The nasdaq trader's toolkit*. New York: John Wiley and Sons.
- Edwards,R., & Magee,J.(1997). *Technical analysis of stock trends*(7th ed.). Atlanta: American Management Association.
- Bruner,R.(1998). *Analytical approaches to stock selection*. Chicago, Glenlake Publishing Company.
- DuBois,P.(2000). *Mysql*. Indiana: New Riders Publishing.
- Harlow,E.(2000). *Developing linux application*. Indiana: New Riders Publishing.
- Maxfield,W. (2000). *Mysql and php from scratch*. Indiana: Que.
- Meloni,J.(2000). *Essentials php*. California: Prima Publishing.
- Blaha,M., & Premerlani,W. (1998). *Object-oriented modeling and design for database applications*. New Jersey: Prentice Hall.
- Elmasri,R., & Navathe,S. (2000). *Fundamentals of Database Systems*(3rd ed.). Massachusetts: Addison-Wesley.