

# **ONLINE SHARE TRADING SYSTEM**

PROJECT WORK DONE AT

**MICROSIGN TECHNOLOGIES PVT. LTD., COCHIN.**

**PROJECT REPORT**

P-982

SUBMITTED IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE AWARD OF THE DEGREE OF  
**MASTER OF COMPUTER APPLICATIONS**  
OF BHARATHIAR UNIVERSITY, COIMBATORE.

SUBMITTED BY

**DANY AUGUSTINE**

**Reg.No. 0038M1022**



**Internal Guide**

**Mr. A. MUTHUKUMAR, M.Sc, M.Phil**

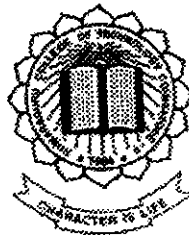
Assistant Professor,

Dept. of Computer Science and Engineering,  
Kumaraguru College of Technology, Coimbatore.

**External Guide**

**Mr. Rajan Varghese, MCA**

Microsign Technologies, Pvt.Ltd., Cochin



Department of Computer Science and Engineering  
**KUMARAGURU COLLEGE OF TECHNOLOGY**

Coimbatore – 641 006

APRIL 2003

# **Certificates**



Department of Computer Science and Engineering  
**KUMARAGURU COLLEGE OF TECHNOLOGY**  
COIMBATORE – 641 006.



### CERTIFICATE


This is to certify that the project work titled  
**ONLINE SHARE TRADING SYSTEM**

Done by


**DANY AUGUSTINE**  
**0038M1022**

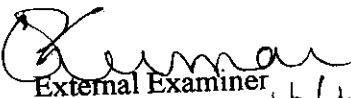
Submitted in partial fulfillment of the requirements for the award of the degree of  
**Master of Computer Applications of Bharathiar University.**

  
Professor and Head

  
Internal Guide

Submitted to University Examination held on 16-04-2003

  
Internal Examiner

  
External Examiner 16/4/03

osign Technologies Pvt. Ltd.  
www.microsigntech.com

Regd. Off : Financial Square  
Old Railway Station Road  
Ernakulam North, Cochin - 682 018  
Keralam, India  
Tel : 91-484-2395017, 2395634  
e-mail : info@microsigntech.com

Microsign

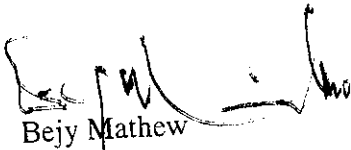
MT/2003/246

March 30, 2003

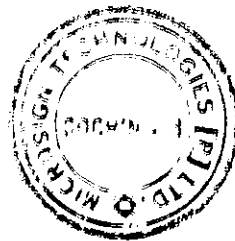
**CERTIFICATE**

This is to certify that Mr. Dany Augustine, Final year MCA, Kumaraguru College of Technology has undergone project work on ONLINE SHARE TRADING SYSTEM for Microsign Technologies Pvt.Ltd, Cochin, during the period December 2002 to March 2003.

The project has been completed successfully under my guidance and supervision and we are satisfied with his work.

  
Bejy Mathew

Managing Director



## DECLARATION

I hereby declare that the project entitled 'ONLINE SHARE TRADING SYSTEM' for Microsign Technologies Pvt. Ltd., Cochin, submitted to **Kumaraguru College of Technology, Coimbatore**, affiliated to **Bharathiar University** as the project work of **Master of Computer Applications Degree**, is a record of original work done by me under the supervision and guidance of **Mr. Rajan Varghese, MCA**, and **Mr. A. MUTHUKUMAR, M.Sc, M.Phil**, and this project work has not found the basis for the award of any Degree/Diploma/Associateship/Fellowship or similar title to any candidate of any university.

Place: Coimbatore

Signature of candidate



Date :

(DANY AUGUSTINE)

## **ACKNOWLEDGMENT**

### **I thank God Almighty for all his blessings**

With great pleasure, I sincerely thank a number of people who have provided understanding and assistance in this effort.

I express my sincere thanks to **Prof. Dr. S. THANGASAMY, B.E (Hon's), Ph.D.**, Head of Department of Computer Science and Engineering for making all necessary arrangements for the successful completion of the project.

I owe a great deal to my respected project guide **Mr. A. Muthukumar, MCA**, Lecturer Department of Computer Science and Engineering, for the motivation, constant encouragement and kind suggestions in every step throughout this project.

With profound gratitude, I sincerely thank **Mr. Bejy Mathew (M.D, Microsign Technologies Pvt. Ltd., Cochin)** and **Mr. Rajan Varghese**, project leader, for allowing me to do the project work at this prestigious organization. My sincere gratitude and thanks to **Mr. Bijoy Varghese** (programmer, Microsign) for his constant help and valuable guidance throughout the project work.

I express my warm thanks and gratitude to my parents, friends and well wishers who have directly and indirectly contributed a lot towards this project.

## **SYNOPSIS**

Shares are the rights of ownership in a limited liability company. Shares may be held by private individuals or by other companies. Shareholders have invested money in a company in return for part ownership of the company and for a share in its profit. Members buy and sell stocks and shares in a market place named a Stock Exchange.

Prices of securities on the stock exchange, as in other free markets, are determined by supply and demand. When more people want to buy than want to sell shares in particular company, the price of those shares will rise. When more people want to sell than want to buy, the price will fall. Hence the price index of the shares keeps on fluctuating during the trading time. So for a trader, it is very important to obtain a snapshot of the current market in order to place his/her orders efficiently. Often inadequate or incorrect information results in huge losses for the trader while very accurate and on-time information usually brings him financial gains.

The Online Share Trading System brings the information about the share market to the desktops of the traders and this information gets updated on milliseconds basis. The traders are able to get a view of the share market at any point of time during the trading session. Whenever a new order is placed into the system, the entire information gets updated and is available to all the clients logged onto the system. Based on this

information the clients can place their orders, they can view the orders that are in the pending state and can cancel/modify if they want. Also, they are able to see the executed orders for them and can proceed with the settlement operations. In overall the OnLine Share Trading System provides a platform for trading the shares.

All the transactions will be performed in a highly secured environment. There is a central office to control all the activities of the clients. The clients need to register with the central office and obtain an id and password in order to make use of the system. This id and password are required at the login time to authorize the client. The central office decides the shares to be traded within the system and they have an overall control over the entire operations of the system. During the trading if the central office wants to disable a user from trading they can do so.

In short, the Online Share Trading System strives to provide its users the ability to place, monitor and control all aspects of their trading from one interface that is available anywhere a user can log onto.



# CONTENTS

<b>1. INTRODUCTION</b>	
1.1 Project Overview -----	1
1.2 Organization Profile-----	3
<b>2. SYSTEM STUDY AND ANALYSIS</b>	
2.1 Existing System – Limitations-----	5
2.2 Proposed System-----	8
2.3 Requirements of the New System	
2.3.1 Functional Requirements-----	14
2.3.2 Performance Requirements-----	18
2.3.3 User Characteristics-----	20
<b>3. PROGRAMMING ENVIRONMENT</b>	
3.1 Hardware Configuration-----	23
3.2 Software Configuration-----	24
<b>4. SYSTEM DESIGN AND DEVELOPMENT</b>	
4.1 Input Design-----	42
4.2 Output Design-----	47
4.3 Database Design-----	48
4.4 Process Design-----	55
<b>5. SYSTEM TESTING AND IMPLEMENTATION</b>	
5.1 System Testing-----	61
5.2 System Implementation-----	66
<b>6. CONCLUSION-----</b>	67
<b>7. SCOPE FOR FUTURE DEVELOPMENT-----</b>	68
<b>8. BIBLIOGRAPHY-----</b>	69
<b>9. APPENDIX-----</b>	70

# 1.INTRODUCTION

## 1.1 PROJECT OVERVIEW

*Stock and Shares:* are the rights of ownership in a limited liability company. Each company has a *stock*, which is divided into a number of shares. Shares may be held by private individuals or by other companies. Shareholders have invested money in a company in return for part ownership of the company and for a share in its profit.

*Trading:* Stock Exchange is a market place in which members of the market buy and sell stocks and shares, and investments in companies or in governments. Traders buy and sell shares for companies, governments, organizations, and individuals. They are much interested in getting updated information about each of the shares being traded such as what is the current demand for a particular share, which shares are *falling*, which shares are *rising*, for what rate a particular share is being purchased etc. The price index of shares fluctuates very rapidly and hence it is necessary for the trader to get the most updated price index to place his/her orders.

Microsign Technologies (P) Ltd. Cochin, one of the leading IT firms in Kerala, provides IT related services like Software Development, Service, Network Implementation and Maintenance, especially capital market related. The company has a number of clients (share brokers) doing business in the capital market. These

organizations help their clients to trade through the Stock Exchanges by providing them necessary information about the shares and the current market demand for those shares.

The Online Share Trading System brings the trader the most updated information online and enables him to place the orders more efficiently. Also, some time later if they identify that their order details need to be updated or if it needs more priority, they are able to update the pending orders by editing the orders and providing the new details. And if the client wants to cancel his/her order, there is facility in the system to do so. If an order gets executed, the clients receive the details of the transaction in the very next moment. In short, the system helps the share traders to reduce the strains in the trading by providing them an efficient interface through which they can manage all the details of the trading.

## **FEATURES OF THE SYSTEM**

- ▶ Online Updated Information
- ▶ Facility to View the Share Details
- ▶ Facility to Place Buy/Sell Orders
- ▶ Facility to View, Update and/or Delete Pending Orders
- ▶ Facility to View the Executed Orders
- ▶ Enhanced Security for the Transactions

## **1.2 ORGANIZATION PROFILE**

A company situated at the heart of Cochin, the commercial capital of Kerala, for providing IT related services like Software Development, Service, Network Implementation and maintenance especially capital market related, having clients throughout India and a team of dedicated professionals, Microsign Technologies (P) Ltd is one of the leading IT firms in Kerala. Formed by a team of highly qualified and well-experienced IT professionals, in recognition of the growing need for extending information technology in India.

The firm has established its strong presence in the field of development in the state. Since its inception Quality rather than Quantity has been the benchmark by which all activities at Microsign Technologies has been carried out and evaluated. This, coupled with an approach to provide clients with Full and Maximum value for money rather than Cheapest Available Solution, has been the two guiding pillars of its operations. It is also engaged in support and service of leading capital market software in the state.

The company mainly focuses on Capital Market. The main product is a Capital Market BackOffice software-StockSERGE having around 50 clients throughout the country. StockSERGE is a multi-user back office software for stock brokers who deal with a large number of clients, branches, multiple stock exchanges with multiple segments etc. This is an Enterprise software, which can integrate the

transactions of a large enterprise with large number of branches spanned over different locations. The system is adequate to keep accounting standards & stock exchange regulations in maintaining records. Moreover, the company's experience in the field and technical expertise are combined to deliver such a powerful product.

## **2. SYSTEM STUDY AND ANALYSIS**

Microsign Technologies Pvt. Ltd., one of the leading IT firms in Cochin, the commercial capital of Kerala provides support for a number of stock market business organizations. These firms have brokers and their clients doing business through the Stock Exchanges. The Online Share Trading System is a client/server system that will be acting as the backbone of the trading transactions for these brokers and clients.

### **2.1 EXISTING SYSTEM**

The brokers are the authorized persons who can trade through the Stock Exchanges. These brokers trade for themselves and for other persons or clients. If somebody wants to trade through the Stock Exchange either he has to take a membership and thus become a broker or else he can authorize a person who is already a broker. Taking a membership in the Stock Exchange demands a very strong financial background and the procedures are very strict and difficult to go through. So for a common man who wants to trade through the Stock Exchange, the better way is trading through a broker.

Usually every broker will be having a number of clients doing the business through him. The clients place the orders into the exchange through their broker. All these clients need to know the current market position of the shares that they trade. They get this information from their brokers. Usually the clients meet the broker directly or

else they contact through the telephone. Obviously all the enquiries are forwarded towards the broker's office. Thus the office of the broker becomes a noisy place where all these enquiries are addressed and all the transactions are settled. The broker can avoid a lot of direct enquiries and unnecessary visits by providing an interface to these clients that brings them the most updated information about the stock market into their offices, allows them to place their orders and help them control these orders efficiently from within their offices. By transferring the core part of the trading operations to the clients' office, the staffs at the broker's office are able to concentrate on the transaction settlement operations rather than on addressing the customer enquiries. Only the cash settlements will have to be done at the broker's office.

### **DRAWBACKS OF THE EXISTING SYSTEM**

An elaborate study of the existing system brings into light the following drawbacks:

#### **From the Clients' Perspective:**

- Unavailability of updated information

The clients need to spend a long time at the broker's office to get the details about the current market. Only on rare occasions do they get the most updated market status.

- Time delay in placing orders

Once the clients got the details of the market condition of the shares they want to trade, they can place the orders to buy and/or sell shares.

At present they need to quote the quantity and price for the share they

want to trade. During the processing of these orders, the market index will have changed and the placed orders might end up in huge losses for the trader.

- **Difficulty in getting details of the pending orders**

Once an order is placed it is very difficult or sometimes impossible to trace the order. There is no way to know whether the order will get executed or not.

- **Inability to update/cancel the pending orders**

In the current system, once you place an order, you cannot modify or delete the order. Either it will get executed during the trading time or it will get deleted after the trading.

- **Time delay in getting the transaction reports from the broker**

If you have executed order(s), you will get reports about the transaction from the broker's office. At present it will take one or two days to get the reports from the broker's office.

#### **From the Broker's Perspective:**

- **No control over the overall operations**



The broker's office is supposed to have the overall control over the transactions between the clients. But in reality it is not so. They don't have the full control over the transactions and over the clients.

- Reduced customer satisfaction (Inability to satisfy all the customer enquiries in time)

There is no satisfactory method or system to address all the customer enquiries in time.

- No operational interactions with the clients
- Time delay in settling the transactions
- Difficulty in dealing with the orders and transaction details

## **2.2 PROPOSED SYSTEM**

The proposed system is a client/server system with the client part as the traders' interface and the server being the broker's office. As in every client/server architecture the overall control is at the server side. The server will be in the 'UP' state only during the trading period that is from 9:30 AM to 3:30 PM. After this time the no trading transactions takes place and the server will be shut down and the logged on clients receive appropriate messages indicating that the trading is over. The server provides memberships for the clients, authorizes the clients, decides the shares to be listed for trading and control the shares and clients during the trading time.

Also, the server sends the clients the most updated information about the stock market in order to enable them to place their orders in such a way that they get appropriate priority and the financial risks associated with the trading is less.

The client side interface brings the updated online information to the traders and they use this information as the basis for their trading. In order to use this information the traders or the users of the system has to login using the id and password they obtain during the registration at the server. Once the server authenticates the id and password the user becomes an authorized client and he has the rights of placing the orders into the system. The orders can be placed through the user-friendly interfaces and the client is able to view the progress of his orders. If at some other point of time he identifies that the orders need to be executed urgently he can set a higher priority by updating the details of the pending order. Once an order gets executed, that is a transaction takes place, the clients involved in the transaction gets an acknowledgement with the details of the transaction denoting that their order has been executed. At the same time they receive the message, the clients can view the payment details and they can go on with the settlement procedures at the broker's office.

## **ADVANTAGES OVER EXISTING SYSTEM**

### **From the Clients' Perspective:**

- Online Updated Information

The clients get the updated information from the share market through their interface. They are able to view the best existing orders, that is buy and sell orders, for the shares if at least one order for each share exists. Also they can query about the shares available for trading. And if some transaction occurs for the logged on client, he gets messages about the transaction.

- **Fast and efficient placement of orders**

The clients don't need the help of the staff at the broker's office to place an order. They are able to place 'buy' and 'sell' orders online through the interfaces provided.

- **Ability to control the transactions (updatation/cancellation)**

The updation and/or cancellation of the orders are really easy with the new system. The clients can view the pending orders for them from their interface and if they want, are able to update or cancel them.

- **Transaction reports without delay**

The clients get the reports about their transactions online with details such as the share, quantity, price, the customer, the brokerage amount etc. A client is able to view the reports at any point of time during the transaction.

- **Reduces Financial Losses**

The online updated information and the facility for placing the orders online reduces the financial risks involved in the traditional mode of trading.

#### **From the Broker's Perspective:**

- **Administrative control over the operations**

The server controls all the operations during the trading time. They are able to disable/enable the shares that are being traded and thus can suspend/resume the trading of these shares. The same operation is possible over the clients. If the broker wants a particular client to stop his trading, the staffs are able to do so from the server.

- **Very good interaction with the clients**

Most of the information for the share trading is available for all the clients and the settlement is the remaining major activity at the broker's office. So the staffs get enough time to deal with the clients and to attend their problems.

- **Effective and fast transaction reports**

The transaction reports for the clients and for the server are available online and there is no involvement of the broker's staffs in preparing the reports for each clients.

- **Fast transaction settlements**

Since the clients get up-to-date transaction reports, the settlement of the transactions is very easy and fast.

## **FEASIBILITY STUDY**

To determine the feasibility of the system, we have to discuss three levels of feasibilities -- technical, financial and operational.

### **Technical Feasibility:**

The current available technical resources at the broker's office (server) include the Windows NT operating system that can be used to maintain the server part. The clients should have at least a machine with the Windows 2000/NT OS to maintain the client interface. Most of the clients of the Stock Broker have this basic requirement and others will have to enhance their existing operating systems to Windows 2000/NT. And to establish a link between the clients and the servers we need a kind of network and at present the broker has a VSAT link to his office.

### **Financial Feasibility:**

The financial feasibility of the system depends upon for how much amount the clients want to trade for. If a client is doing trades for a very small amount then the existing system is enough and he can follow that. For a client who carry out large transactions the, new system will be very helpful and with respect to the transactions he make the system will be feasible.

**Operational Feasibility:**

Operational feasibility is related to the easiness and simplicity of using the new system. The system is very user friendly and simple to use. What the client needs is an id and password to logon into the system. Once entered into it, he is able to access all the available information and he can place the orders.

## **2.3 REQUIREMENTS OF NEW SYSTEM**

### **2.3.1 FUNCTIONAL REQUIREMENTS**

#### **Introduction:**

The functional requirements of the system are:

- Bring the current market status to the client (Online Updated Information)
- Allow the client to place the orders
- Provide the facility to manage or control the orders
- Acknowledge the client transactions
- Send the client up-to-date reports

To satisfy the functional requirements the client must be a registered client. He should have a login id and password. After the validation process he can use the system to trade through the Exchange. He gets the share price indexes online. The client can place the orders through the screens provided for the purpose. At any time during the trade he is able to view the pending orders and trade summary. In addition to this the clients are also able to view the details about the shares of the listed companies.

#### **List of Inputs:**

There are inputs specific to the client and server. The client inputs will be the details of the orders placed by the clients. At the server the inputs are the details of the clients for registration and the details of the shares for listing them in the system. The inputs into the system are described below:

## Client Side Inputs:

1. User id
2. Password
3. Share Code
4. Quantity
5. Lot -Minimum Quantity (optional)
6. Price

The **USER ID** and **PASSWORD** are required for a client to login to the system. Also these inputs are required for the change of password. If the user wants to change the current password he must provide the current **USER ID** and **PASSSSWORD** along with the new ones. After the login process is complete the **USER ID** is used as identification of the client for placing the orders. The **USER ID** is taken as a default input whenever the client places an order. The other inputs at the client part are the **SHARE CODE**, **QUANTITY**, **LOT** and **PRICE** that form the details of an order. The client has to specify the share he wants to trade by choosing a **SHARE CODE** from the list provided to him. Along with the **SHARE CODE** he can include another data namely the **LOT** that is the minimum number of shares he will be dealing with. This field is optional and if not specified takes the value of **QUANTITY** that is the actual number of shares being traded by the client. **PRICE** is the most important data in an order and the client can use the price indexes provided by the system to place an order quoting the most appropriate price.



## Server Side Inputs:

1. Share Code
2. Share Description
3. Share Status (Enabled/Disabled)
4. User id
5. Password
6. Name
7. State
8. City
9. Phone Number
10. E-mail id
11. Client Status (Enabled/Disabled)

Most of the server side inputs are the details of the clients and shares except some exceptional inputs during the trading time that are used to control the overall operations. The important inputs are the SHARE CODE and the USER ID that are the identifications for the shares and clients respectively. During listing of a new share into the system the SHARE CODE, SHARE DESCRIPTION and its STATUS (enabled/disabled) must be provided. The client details are NAME, PASSWORD that must be specified along with the USER ID. The STATUS of the user indicates whether he is allowed to trade through the system or not. The other fields are optional.

## **Information Processing Required**

The information processing is performed through the following steps:

- The user login by providing the user id and password.
- After the validation, the user has the following options,
  - Access the Online Information
  - Place Orders
  - View Pending Orders
  - Update/Delete Pending Orders
- The user accesses the online share price index. If he wants to buy or sell any shares he can put the order details in the order screens. If pending orders exist for the user then he is able to view them and if he wants can delete and/or update them.
- If a match occurs for a buy and sell order then trading will occur. The clients receive appropriate messages from the server.
- At any time the clients are able to get the details of the shares being traded.
- They receive the details of the transactions; if there is any, at the very next moment the transactions occur.

### **2.3.2 PERFORMANCE REQUIREMENTS**

The performance of any system is measured using the following four factors. They are,

1. Security
2. Availability
3. Capacity
4. Response Time

For the Online Share Trading System the performance requirements are analyzed based on the above four factors. The analysis brings into light the following facts,

#### **Security:**

The Online Share Trading System is very secure. Only the valid users can enter into the system for trading. The users are registered at the server, i.e., at the broker office. The registered users are provided with a user id and password and this is required at the login time.

Also, the server is able to disable the user account if they find out that something is wrong with the client transactions. And the server has the overall control over the operations. The staff at the broker office will be give the administrative rights to control the system.

**Availability:**

The availability of the system depends upon the platform and since our programming environment is MS Visual Basic .NET currently it can run only on Windows 2000 machines because the .NET framework needs a Windows 2000 machine to operate.

**Capacity:**

There will be around 40 clients connected to the system during the trading period. Since we use the MS SQL Server 2000 as the database, this does not provide a problem to the capacity factor of the performance requirement. We will be able to provide a fairly well response time that is below 1.5 seconds for the clients.

**Response Time:**

The system is supposed to have a good response time that is acceptable to the user. In fact this factor decides the constraint of the Online Share Trading System since the information plays a fatal role in the share trade. The response time for our system depends upon the connection mode of the client with the server. Most of the clients will need a VSAT link, which some are already having. Some local clients can be connected using a LAN. Irrespective of the connections we will be able to provide a response time that is less than 1.5 seconds with a shorter response time to the LAN connected clients.

### **2.3.3 USER CHARACTERISTICS**

Basically there are two kinds of users in the Online Share Trading System.

They are,

- Administrators
- Registered Members (Traders)

The Administrators are the staff at the broker's office. They are responsible for maintaining the integrity of the system and should be proficient in database and administration related services. They have the authority to register new clients, remove existing user accounts, edit the user details, list new shares, edit/delete the shares, etc. They are given the authority to control the shares and clients during the trading period.

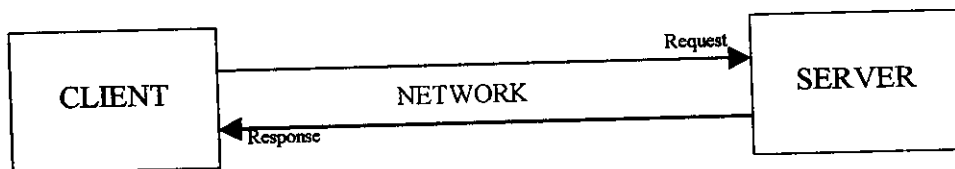
The Registered Members are the actual users of the system and they trade the shares by placing the orders and receiving the reports of the transactions. They get the online updated share index through the client interface that helps them in placing their orders.

### 3. PROGRAMMING ENVIRONMENT

#### CLIENT SERVER ARCHITECTURE

A client/server network consists of a group of user-oriented PCs called clients that issue requests to a server. The client PC is responsible for issuing requests for services to be rendered. The server's function on the network is to service these requests. Servers generally are high performance systems that are optimized to provide network services to other PCs. The server machine often has a faster CPU, more memory, and more disk space than a typical client machine.

#### General Structure



#### THE BASIC FEATURES OF CLIENT/SERVER MODEL

Clients and Servers are functional modules with well-defined interfaces (i.e they hide internal information). The functions performed by a client and a server can be implemented by a set of software modules, hardware components, or a combination of these. Clients and/or servers may run on dedicated machines, if needed.

- 1) Each client/server relationship is established between two functional modules when one module (client) initiates a service request and the other (server) chooses to respond to the service request. For a given service request clients and servers do not reverse roles. However, a server for request R1 may become a client for service request R2 when it issues requests to another server.
  
- 2) Information exchange between clients and servers is strictly through messages (i.e., no information is exchanged through global variables). The service request and additional information is placed into a message that is sent to the server. The server's message is similarly another message that is sent back to the client. This is an extremely crucial feature of client/server model.
  
- 3) Clients and servers typically reside on separate machines connected through a network. Conceptually, clients and servers may run on the same machine or on separate machines. In a Distributed Client/Server system, clients and servers reside on separate machines. The implication of the last two features is that client/server requests are real time messages that are exchanged through network services. This feature increases the appeal of the client/server model (i.e., flexibility, scalability) but introduces several technical issues such as portability, interoperability, security, and performance.

### **3.1 HARDWARE CONFIGURATION**

The hardware and peripheral configurations used for the project are listed below:

#### **SERVER CONFIGURATION**

<b>PROCESSOR</b>	<b>:Intel Pentium III</b>
<b>RAM</b>	<b>:128 MB</b>
<b>HDD</b>	<b>:20 GB</b>

#### **CLIENT**

<b>PROCESSOR</b>	<b>:Intel Pentium III</b>
<b>RAM</b>	<b>:128 MB</b>
<b>HDD</b>	<b>:10 GB</b>

#### **SQL SERVER**

<b>PROCESSOR</b>	<b>:Intel Pentium III</b>
<b>RAM</b>	<b>:128 MB</b>
<b>HDD</b>	<b>:20 GB</b>



## **3.2 SOFTWARE CONFIGURATION**

### **SERVER**

<b>OPERATING SYSTEM</b>	<b>:Windows 2000 Server</b>
<b>DATABASE</b>	<b>:MS SQL 2000</b>
<b>PROGRAMMING ENVIRONMENT</b>	<b>:MS Visual Basic .NET</b>

### **CLIENT**

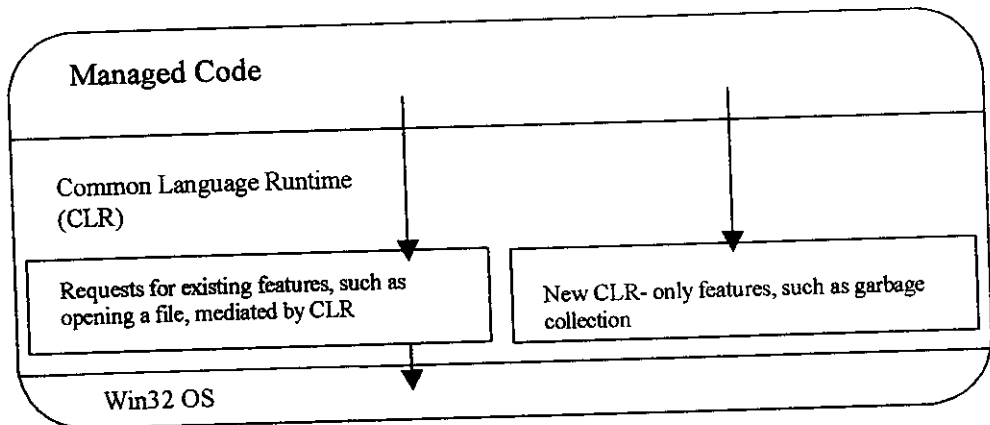
<b>OPERATING SYSTEM</b>	<b>:Windows 2000 Server</b>
<b>PROGRAMMING ENVIRONMENT</b>	<b>:ms Visual Basic .NET</b>

## FEATURES OF SOFTWARE

### MICROSOFT .NET

The .NET framework is Microsoft's operating system product that provides prefabricated solutions to the programming problems. It is an add-on run-time environment that runs on Windows 2000 operating system. The key to the framework is managed code. Managed code runs in an environment, called the *Common Language Runtime (CLR)* that provides a richer set of services than the standard Win32 operating system as shown in figure.

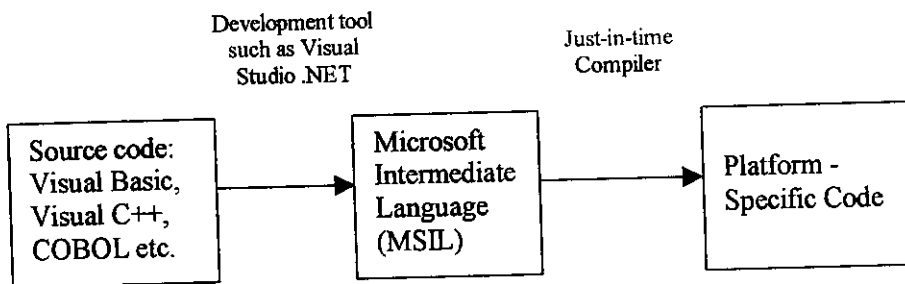
Figure: Managed execution in the Control Language Runtime



The CLR environment is the higher power that we have to turn our code over to in order to make platform independent applications. Every CLR compliant development tool compiles its own source code into a standard *Microsoft Intermediate Language (MSIL, or IL for short)*. Because all development tool produce the same IL,

regardless of the language in which their source code is written, differences in implementation are gone by the time they reach the CLR. Microsoft Visual Studio .NET provides CLR-compliant versions of Visual Basic, C#, Jscript, and C++.

Figure: Different source code programming languages are compiled into MSIL



The IL code produced by the development tool can't run directly on any computer. A second step is required, called *just-in-time* (JIT) compilation, as shown in the above figure. A tool called a just-in-time compiler, or JITter, reads the IL and produces actual machine code that runs on that platform. This provides .NET with a certain amount of platform independence, as each platform can have its own JITter.

## Microsoft .NET Services

Microsoft .NET provides the following services:

- 1) **.NET framework** –a new run-time environment

The .NET framework is a run-time environment that makes it much easier for programmers to write good, robust code quickly, and to manage, deploy, and revise the code. The programs and components execute inside this environment. It provides programmers with cool run-time features such as automatic memory management (garbage collection) and easier access to all system services. It adds many utility features such as easy Internet and database access. It also provides a new mechanism for code reuse-easier to use and at the same time more powerful and flexible than COM. The .NET framework is easier to deploy because it doesn't require registry settings. It also provides standardized, system-level support for versioning. All of these features are available to programmers in any .NET-compliant language.

## 2) **Windows Forms** - a new way of writing rich client applications

A dedicated client application needs to provide a good user interface. A high-quality interface can provide a much better user experience. Microsoft .NET provides a new package, called .NET Windows Forms, that makes it easy to write dedicated Windows client applications using the .NET framework.

## 3) **ADO.NET** – good support for database access within the .NET framework

No programming environment would be complete without some mention of database access. Most programs spend most of their time gathering

information from a client, making a database query, and presenting the results to the client. .NET provides good support for database operations using ADO.NET.

In short, the important services provided by the .NET framework are:

- Provides automatic memory management via garbage collection
- Supports explicit standardized version management
- Extends rich object oriented programming features to all languages
- Organizes system functionality into a hierarchical namespace
- Supports code security
- Provides seamless interoperability with COM, both as client and as server

## **Object-Oriented Programming Features of .NET**

.NET provides all languages with the object-oriented features of inheritance and constructors.

### **Inheritance:-**

The .NET framework uses inheritance to provide all kinds of system functionality, from the simplest string conversions to the most sophisticated Web Services. All objects in the .NET system, without exception, derive from System.Object or another class that in turn derives from it. If you don't specify a different base class, System.Object is implied. If you prefer a different base class, you specify it by using the keyword *inherits* in Visual Basic or the colon operator in C#. .NET has the ability to provide cross-language inheritance, that is, to allow a class written in one language,

Visual Basic, for example, to derive from a base class written in another language, say C#.

### **Object Constructors:-**

Object Oriented Programming uses the concept of class constructor, a function called when the object is created which usually is a standard place for putting the initialization code. In .NET, the constructor takes the C++ model so that parameterized constructors can be created in order to support inheritance. Every .NET class can have one or more constructor methods. This method has the name *New* in Visual Basic.NET. The constructor function is called when a client creates object using the new operator.

## **Microsoft SQL Server 2000**

Microsoft® SQL Server™ 2000 is a set of components that work together to meet the data storage and analysis needs of the largest Web sites and enterprise data processing systems, yet at the same time can provide easy-to-use data storage services to an individual or small business.

Microsoft SQL Server 2000 features include:

- Internet Integration.

The SQL Server 2000 database engine includes integrated XML support. It also has the scalability, availability, and security features required to operate as the data storage component of the largest Web sites. The SQL Server 2000 programming model is integrated with the Windows DNA architecture for developing Web applications, and SQL Server 2000 supports features such as English Query and the Microsoft Search Service to incorporate user-friendly queries and powerful search capabilities in Web applications.

- Scalability and Availability.

The same database engine can be used across platforms ranging from laptop computers running Microsoft Windows® 98 through large, multiprocessor servers running Microsoft Windows 2000 Data Center Edition. SQL Server 2000 Enterprise Edition supports features such as federated servers, indexed views, and

large memory support that allow it to scale to the performance levels required by the largest Web sites.

- Enterprise-Level Database Features.

The SQL Server 2000 relational database engine supports the features required to support demanding data processing environments. The database engine protects data integrity while minimizing the overhead of managing thousands of users concurrently modifying the database. SQL Server 2000 distributed queries allow you to reference data from multiple sources as if it were a part of a SQL Server 2000 database, while at the same time, the distributed transaction support protects the integrity of any updates of the distributed data. Replication allows you to also maintain multiple copies of data, while ensuring that the separate copies remain synchronized. You can replicate a set of data to multiple, mobile, disconnected users, have them work autonomously, and then merge their modifications back to the publisher.

- Ease of installation, deployment, and use.

SQL Server 2000 includes a set of administrative and development tools that improve upon the process of installing, deploying, managing, and using SQL Server across several sites. SQL Server 2000 also supports a standards-based programming model integrated with the Windows DNA, making the use of SQL Server databases and data warehouses a seamless part of building powerful and scalable systems. These features allow you to rapidly deliver SQL Server



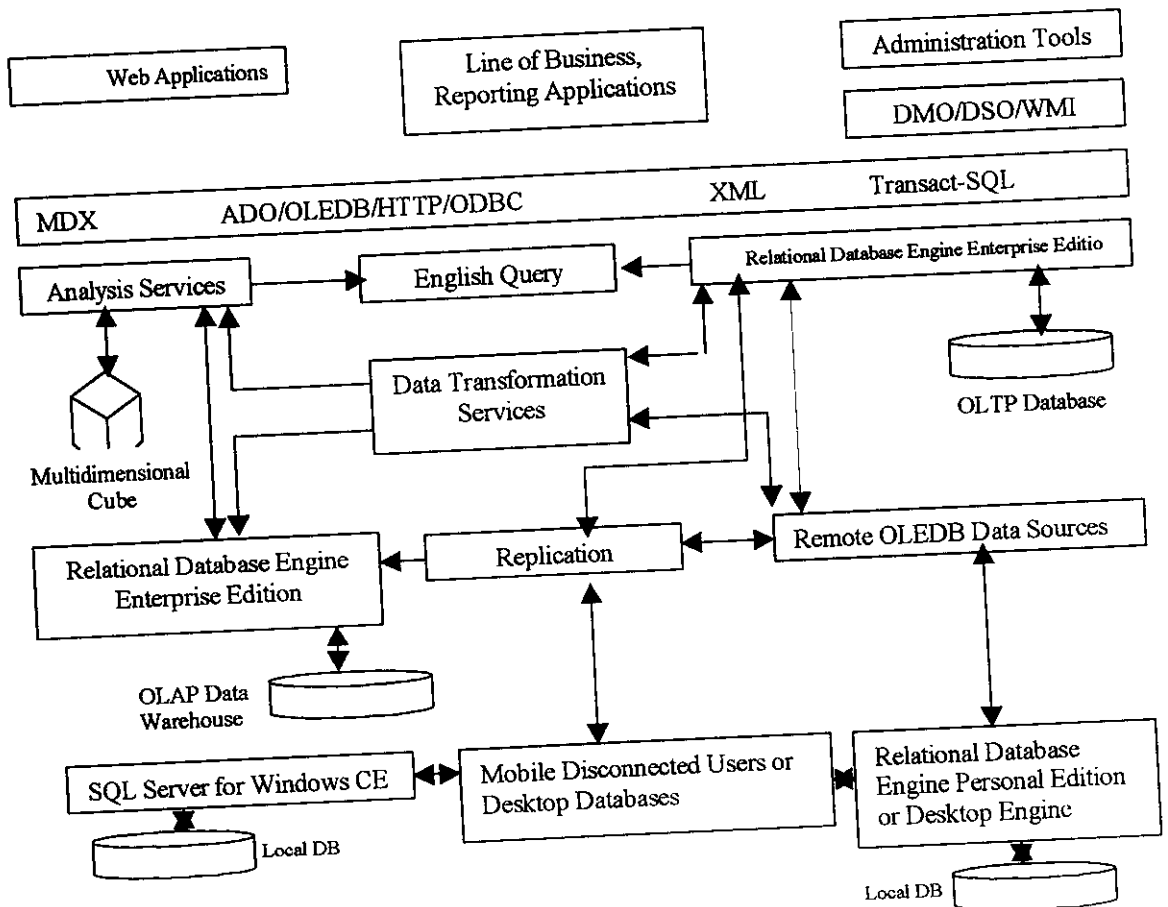
applications that customers can implement with a minimum of installation and administrative overhead.

- Data warehousing.

SQL Server 2000 includes tools for extracting and analyzing summary data for online analytical processing. SQL Server also includes tools for visually designing databases and analyzing data using English-based questions.

### SQL Server 2000 Component Overview

This diagram is an illustration of the relationships between the major components of Microsoft SQL Server 2000.



SQL Server 2000 provides two fundamental services to applications in a Windows DNA environment:

- The SQL Server 2000 **relational database engine** is a modern, highly scalable, highly reliable engine for storing data. The database engine stores data in tables. Each table represents some object of interest to the organization. Each table has columns that represent an attribute of the object modeled by the table and rows that represent a single occurrence of the type of object modeled by the table. Applications can submit Structured Query Language (SQL) statements to the database engine, which returns the results to the application in the form of a tabular result set. The specific dialect of SQL supported by SQL Server is called Transact-SQL. Applications can also submit either SQL statements or XPath queries and request that the database engine return the results in the form of an XML document.

The relational database engine is highly **scalable**. The SQL Server 2000, Enterprise Edition can support groups of database servers that cooperate to form terabyte-sized databases accessed by thousands of users at the same time. The database engine also tunes itself, dynamically acquiring resources as more users connect to the database, and then freeing the resources as the users log off. This means that the smaller editions of SQL Server can be used for individuals or small workgroups that do not have dedicated database administrators. SQL Server for Windows CE even extends the SQL Server programming model to Windows CE devices used by mobile, disconnected users. Even large Enterprise Edition

database servers running in production are easy to administer using the graphical user interface (GUI) administration utilities that are a part of the product.

The relational database engine is highly **reliable** and capable of running for long periods without down time. Administrative actions that required stopping and starting in earlier versions of the database engine can now be performed while the engine is running, increasing availability. The integration of the database engine with Windows 2000 and Windows NT failover clustering allows you to define virtual servers that keep running even if one of the physical servers in the node fails. Where appropriate, log shipping can be used to maintain a warm standby server that can replace a production server within minutes of a failure.

The relational database engine is also highly **secure**. Login authentication can be integrated with Windows Authentication, so that no passwords are stored in SQL Server or sent across the network where they could be read by network sniffers. Sites can set up C2-level auditing of all users accessing a database, and can use Secure Sockets Layer (SSL) encryption to encrypt all data transferred between applications and the database.

The distributed query feature of the database engine allows you to access data from any source of data that can be accessed using OLE DB. The tables of the remote OLE DB data source can be referenced in Transact-SQL statements just like tables that actually reside in a SQL Server database. In addition, the full-text search feature allows you to perform sophisticated pattern matches against textual data stored in SQL Server databases or Windows files.

The relational database engine is capable of storing detailed records of all the transactions generated by the top online transaction processing (OLTP) systems. The database engine can also support the demanding processing requirements for fact tables and dimension tables in the largest online analytical (OLAP) data warehouses.

- Microsoft SQL Server 2000 **Analysis Services** provides tools for analyzing the data stored in data warehouses and data marts. Certain analytical processes, such as getting a summary of the monthly sales by product of all the stores in a district, take a long time if run against all the detail records of an OLTP system. To speed up these types of analytical processes, data from an OLTP system is periodically summarized and stored in fact and dimension tables in a data warehouse or data mart. Analysis Services presents the data from these fact and dimension tables as multidimensional cubes that can be analyzed for trends and other information that is important for planning future work. Processing OLAP queries on multidimensional Analysis Services cubes is substantially faster than attempting the same queries on the detail data recorded in OLTP databases.

### **Application Support**

Both the relational database engine and Analysis Services provide native support for the common Windows DNA or Win32 data access interfaces, such as ActiveX Data Objects (ADO), OLE DB, and Open Database Connectivity (ODBC). Applications can use any of these application programming interfaces (APIs) to send SQL or XML statements to the relational database engine using a native OLE DB

provider or ODBC driver. SQL Server 2000 also introduces the ability to use HTTP to send SQL or XML statements to the relational database engine. Applications can use the multidimensional extensions of either ADO or OLE DB to send Multidimensional Expressions (MDX) queries to Analysis Services. Because SQL Server uses the standard Windows DNA data access APIs, the development of SQL Server applications is well supported by the Microsoft application development environments. In addition, interactive query tools, such as Query Analyzer, provide templates, interactive debuggers, and interactive test environments that speed the ability of your programmers to deliver SQL Server applications.

In addition to supporting the data storage and OLAP processing needs of applications, SQL Server 2000 provides a full set of easy to use, graphical administration tools and wizards for creating, configuring, and maintaining databases, data warehouses, and data marts. SQL Server also documents the administration APIs used by the SQL Server tools, giving you the ability to incorporate SQL Server administration functionality directly into your own applications. The SQL Server administration APIs include:

- SQL Distributed Management Objects (SQL-DMO), a set of COM objects that encapsulates the administration functions for all of the entities in the relational database engine and databases.
- Decision Support Objects (DSO), a set of COM objects that encapsulates the administration functions for all of the entities in Analysis Services engine and multidimensional cubes.

- Windows Management Instrumentation (WMI), SQL Server 2000 provides a SQL Server WMI provider that lets WMI applications get information on SQL Server databases and instances.

### **Additional Components**

SQL Server 2000 provides several components that support important requirements of modern data storage systems. The data storage needs of today's large enterprises are very complex, and go beyond having a single OLTP system integrated with a single data warehouse or data mart. Increasing numbers of field personnel need to load sets of data, disconnect from the network, record their work autonomously during the day, then plug back in to the network and merge their records into the central data store at the end of the day. OLTP systems have to support the needs of both internal employees operating through an intranet and hundreds of thousands of customers placing orders through your Web portal. Keeping data close to the workgroups or even individuals who primarily work on the data, and then replicating the data to a primary data store may minimize the overall processing load of your system.

- SQL Server 2000 replication allows sites to maintain multiple copies of data on different computers in order to improve overall system performance while at the same time making sure the different copies of data are kept synchronized. For example, a department could maintain the department sales data on a departmental server, but use replication to update the sales data in the corporate computer. Several mobile disconnected users can disconnect from the network, work throughout the day, and at the end of the day use merge replication to merge

their work records back into the main database. These workers can be using SQL Server Personal Edition on notebook or laptop computers, or using SQL Server for Windows CE on Windows CE devices; all are supported by SQL Server replication. SQL Server replication also supports replicating data to data warehouses, and can replicate data to or from any data source that supports OLE DB access.

- SQL Server 2000 Data Transformation Services (DTS) greatly improves the process of building OLAP data warehouses. Large OLTP databases are finely tuned to support the entry of thousands of business transactions at the same time. OLTP databases are also structured to record the details of every transaction. Trying to perform sophisticated analysis to discover trends in sales over a number of months and years would require scanning huge numbers of records, and the heavy processing load would drag down the performance of the OLTP databases. Data warehouses and data marts are built from the data in one or more OLTP systems that is extracted and transformed into something more useful for OLAP processing. OLTP detail rows are periodically pulled into a staging database, where they are summarized and the summary data is stored in a data warehouse or data mart. Data Transformation Services supports extracting data from one source of data, performing sometimes complex transformations of the data, and then storing the summarized, transformed data in another data source. The component greatly simplifies the process of extracting data from multiple OLTP systems and building it into an OLAP data warehouse or data mart.

DTS is not limited to being used to build data warehouses. It can be used any time you have to retrieve data from one data source, perform complex transformations on the data, and then store it in another data source. DTS is also not limited to working with SQL Server databases or Analysis Services cubes, DTS can work with any data source that can be accessed using OLE DB.

- SQL Server 2000 English Query allows you to build applications that can customize themselves to ad hoc user questions. An English Query administrator defines for the English Query engine all of the logical relationships between the tables and columns of a database or the cubes in a data warehouse or data mart. An application can then present the user with a box where she can enter a character string with a question (written in English) about the data in the database or data warehouse. The application passes the string to the English Query engine, which analyzes the string against the relationships defined between the tables or cubes. English Query then returns to the application a SQL statement or MDX (multidimensional expression) query that will return the answer to the user's question.
- Meta Data Services provides facilities for storing, viewing, and retrieving descriptions of the objects in your applications and system. Meta Data Services supports the MDC Open Information Model (OIM) specification defining a common format for storing descriptions of entities such as tables, views, cubes, or transformations, as well as the relationships between these entities. Application development tools that support OIM can use these descriptions to facilitate rapid development and interchange with other tools and applications. SQL Server



components, such as Data Transformation Services packages and Analysis Services databases, can also be stored in the Meta Data Services repository.

## **Database Architecture**

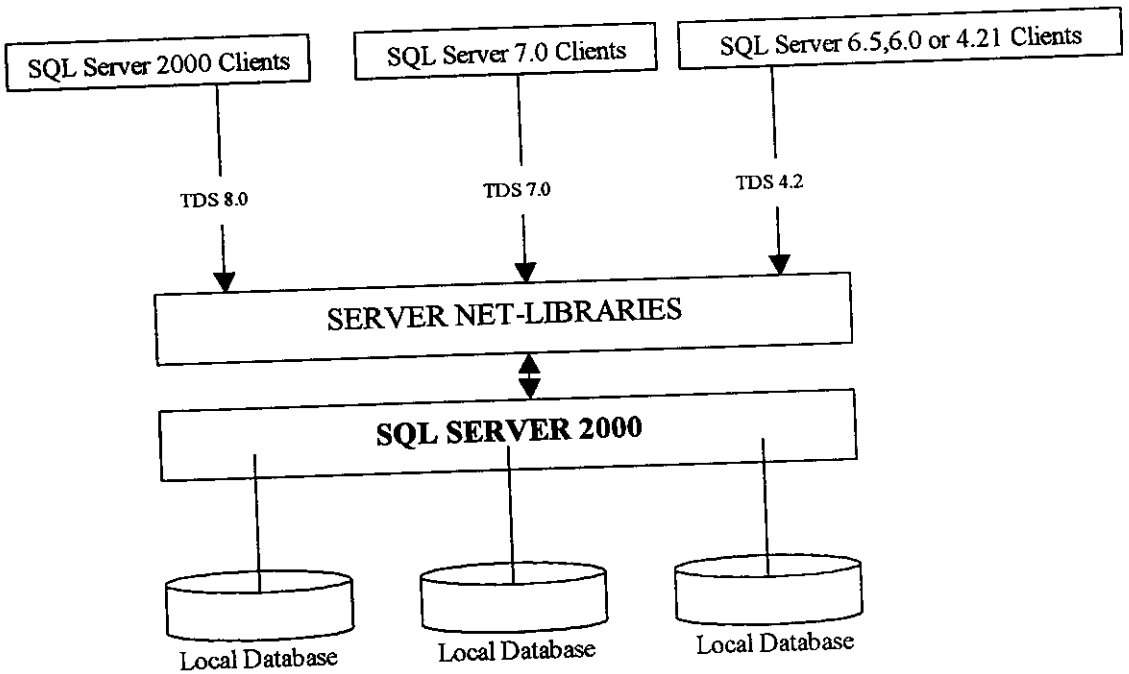
Microsoft SQL Server 2000 data is stored in databases. The data in a database is organized into the logical components visible to users. A database is also physically implemented as two or more files on disk. When using a database, you work primarily with the logical components such as tables, views, procedures, and users. The physical implementation of files is largely transparent. Typically, only the database administrator needs to work with the physical implementation.

Each instance of SQL Server has four system databases (**master**, **model**, **tempdb**, and **msdb**) and one or more user databases. It is not necessary to run multiple copies of the SQL Server database engine to allow multiple users to access the databases on a server. An instance of the SQL Server Standard or Enterprise Edition is capable of handling thousands of users working in multiple databases at the same time. Each instance of SQL Server makes all databases in the instance available to all users that connect to the instance, subject to the defined security permissions.

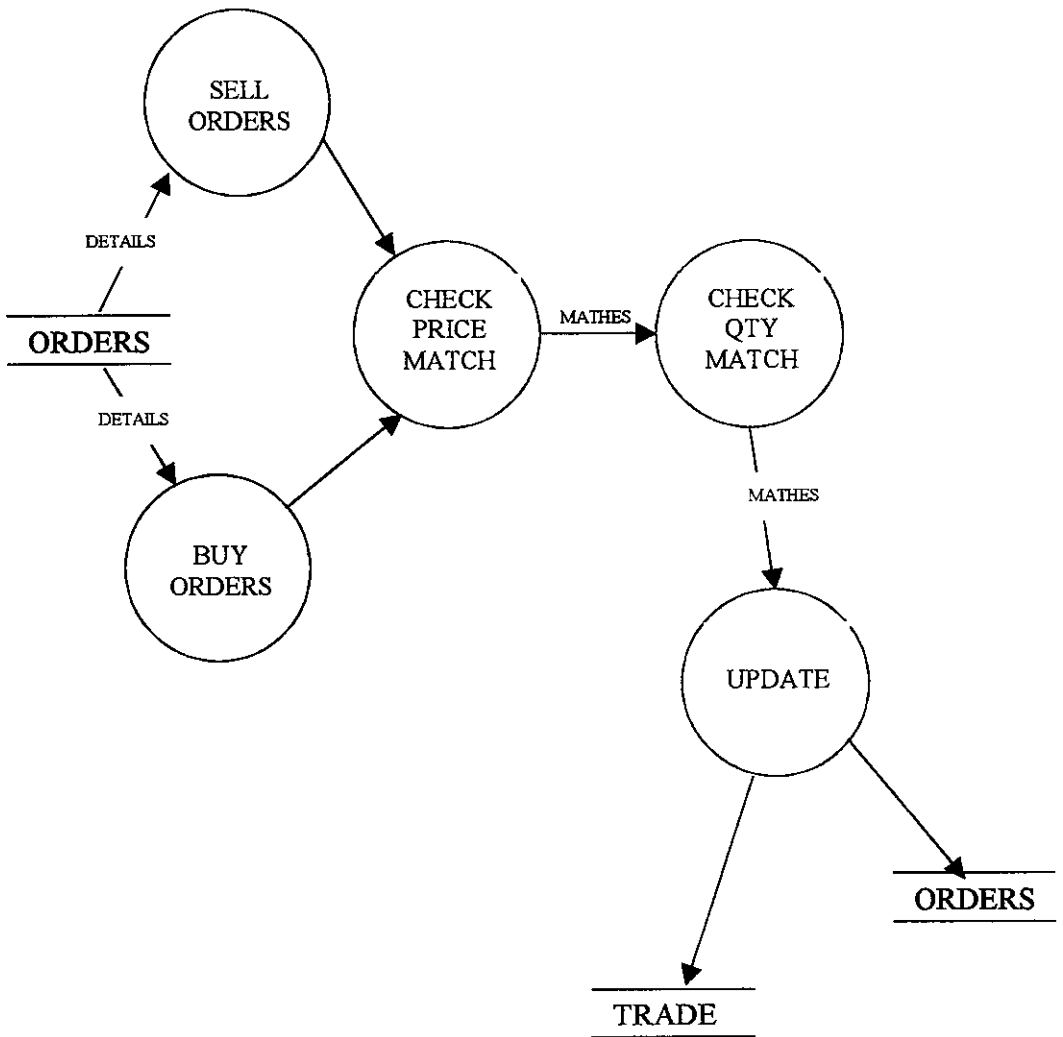
## **Relational Database Engine Architecture Overview**

The server components of Microsoft SQL Server 2000 receive SQL statements from clients and process those SQL statements. This illustration shows the major components involved with processing an SQL statement received from a SQL

Server client. The database server processes all requests passed to it from the server Net-Libraries. It compiles all the SQL statements into execution plans, and then uses the plans to access the requested data and build the result set returned to the client.



## LEVEL 2 – ORDER MATCHING



## **5. SYSTEM IMPLEMENTATION AND TESTING**

### **5.1 SYSTEM TESTING**

Software testing is the process of executing the program with the intent of finding an error. Testing demonstrates that software functions appear to be working according to specification, that behavioral and performance requirements appear to have been met. In addition, data collected as testing is conducted provide a good indication of software reliability and some indication of software quality as a whole. As for every software products, the Online Share Trading System was tested using the two basic testing techniques,

- Black-Box Testing
- White-Box Testing

#### **Black-Box Testing**

Black- Box testing also called behavioral testing, focuses on the functional requirements of the system. Black- Box tests are used to demonstrate that software functions are operational, that input is properly accepted and output is correctly produced, and that the integrity of external information (database) is maintained. Black- Box testing attempts to find errors in the following categories:

- 1) Incorrect or missing functions
- 2) Interface errors
- 3) Errors in data structures or external database access
- 4) Behavior or performance errors

5) Initialization errors

6) Termination errors

The Black- Box testing was conducted successfully for the Online Share Trading System. The system has been tested by giving various kinds of inputs and the expected output was obtained. A set of test data was prepared for verifying the order execution operation and was found successful. The clients got the messages from the server for their executed transactions and the detailed reports were available.

### **White- Box Testing**

White- Box testing, also called glass-box testing, is conducted to ensure that the internal operations are performed according to specifications and all internal components have been adequately exercised. It is a test case design method that uses the control structure of the procedural design to derive test cases. Logical paths through the software are tested by providing test cases that exercise specific sets of conditions and/or loops. Using the White- Box testing method test cases are derived that,

1. Guarantee that all independent paths within a module have been exercised at least once
2. Exercise all logical decisions on their true and false sides
3. Execute all loops at their boundaries and within their operational bounds
4. Exercise internal data structures to ensure their validity

A limited number of important logical paths have been selected and exercised as specified in the White- Box testing method and thus testing the OnLine Share Trading System according to White-Box method. The important data structures have been probed for their validity and were found to be successful. The attributes of both the Black Box testing and the White- Box testing were combined to validate the interfaces and ensured that the internal workings of the software are correct.

### **Testing of Client / Server Architecture**

Online Share Trading System being developed according to client – server architecture, was tested according to client server testing strategies. The distributed nature of client/server environments, the performance issues associated with transaction processing the potential presence of a number of different hardware platforms, the complexities of network communication, the need to service multiple clients from a centralized/distributed data bases and the coordination requirements imposed on the server all combine to make testing of client/server architectures and the software that reside within them considerably more difficult than testing stand alone applications. The testing of the system according to client server architectural strategies was found to be successful.

The system was tested at three different levels:

- 1) Individual client application was tested in a disconnected mode, the operation of the server and the underlying network were not considered

- 2) The client and the server were tested in concert, the network operations were not explicitly exercised
- 3) The complete client/server architecture, including network operation and performance, was tested

In addition to these tests, the **database tests** to test the accuracy and integrity of data stored by the server, and **transaction tests** to ensure that each class of transactions is processed according to the requirements were performed.

### **Unit Testing**

Unit testing focuses on verification effort on the smallest unit of software design – the software component or module. The module interface was tested to ensure that information properly flows into and out of the program unit under test. The local data structure was examined to ensure that data stored temporarily maintains its integrity during all steps in the execution of the algorithms execution Boundary conditions were tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

### **Integration Testing**

Integration testing is a systematic technique for constructing the program structure while constructing tests to uncover errors associated with interfacing. The top-down incremental integration approach was followed and the program was built and

tested in small increments. The unit-tested modules were taken and were used as the building blocks in constructing a program structure that has been dictated by design.

### **Recovery Testing**

The recovery testing was conducted for the server since it must be fault tolerant, i.e., processing faults must not cause overall system function to cease. The system was force to fail in a variety of ways and the data recovery was evaluated and found to be satisfactory.

### **Stress Testing**

Stress tests are designed to confront programs with abnormal situations. Stress testing executes the system in a manner that demands resources in abnormal quantity, frequency, or volume. The Online Share Trading System was Stress Tested by allowing ten clients to logon to the system at the same time. Each of the clients was able to get the online share index and the server messages. They placed a large number of orders simultaneously and the performance of the system was found to be satisfactory. The server control over the operations was exercised and was verified to be working as per the design.



## **5.2 IMPLEMENTATION**

A crucial phase of the software development cycle is the successful implementation of the new system. Implementation means converting a new system design into operation. This involves creating computer compatible files, training the staff, and installing the hardware requirements.

The Online Share Trading System has to be implemented by replacing the manual system. This will take some time since we cannot replace the existing system immediately. We need to conduct a step-by-step change since if some data is lost during the transfer we will be able to trace it. So training the users, and the change from the current system to the new one are the major operations in the implementation.

## **6. CONCLUSION**

OnLine Share Trading System strives to provide its clients the ability to place, monitor and control all aspects of their trading from one interface that is available anywhere a dealer can log onto. It has the ability to quickly access filled, working, cancelled and rejected orders. Online Share System provides you the fastest, most efficient order execution possible. The system routes your orders directly into the matching engine and you can expect the confirmations within seconds.

The system was developed in such a way that it will help both the clients and their brokers in trading the shares. This factor will be a plus point as far as the feasibility of the system is concerned. Within a local network the system will be really cost effective and the infrequent clients will be able to afford it.

Under the current environments the system is found working effectively. The system was thoroughly tested by giving test data and was found to be fully satisfactory. The maintenance of the system would not impose much difficulty since it has simple interfaces and the logic applied is very straightforward.

## **7. SCOPE FOR FUTURE DEVELOPMENT**

The system has been given all the functionalities required by today's trading environment and some additional functionality have also been provided foreseeing the needs in future. The system is developed in such a way that the scalability factor of the system is given high priority and future enhancements will not impose much difficulty. The system can be used as part of fully functional financial solution software for the share traders that keep all their financial details. The system can be used for faster transaction settlements and thus will be helpful for the share brokers.

## **8. BIBLIOGRAPHY**

1. James A. Senn, *Analysis & Design Of Information Systems*, McGraw Hill International, 1989.
2. Roger S. Pressman, *Software Engineering*, McGraw Hill International, 1997.
3. Ron Soukup and Kalen Delaney, *Microsoft SQL Server 7.0*, WP Publishers and Distributors, 2001.
4. David S. Platt, *Microsoft .NET*, WP Publishers and Distributors, 2001.
5. William Perry, *Effective Methods for Software Testing*, Wiley –QED Publication, 1995.
6. Evangelos Petroustos, *Visual Basic 6.0*, BPB Publications, 1998.

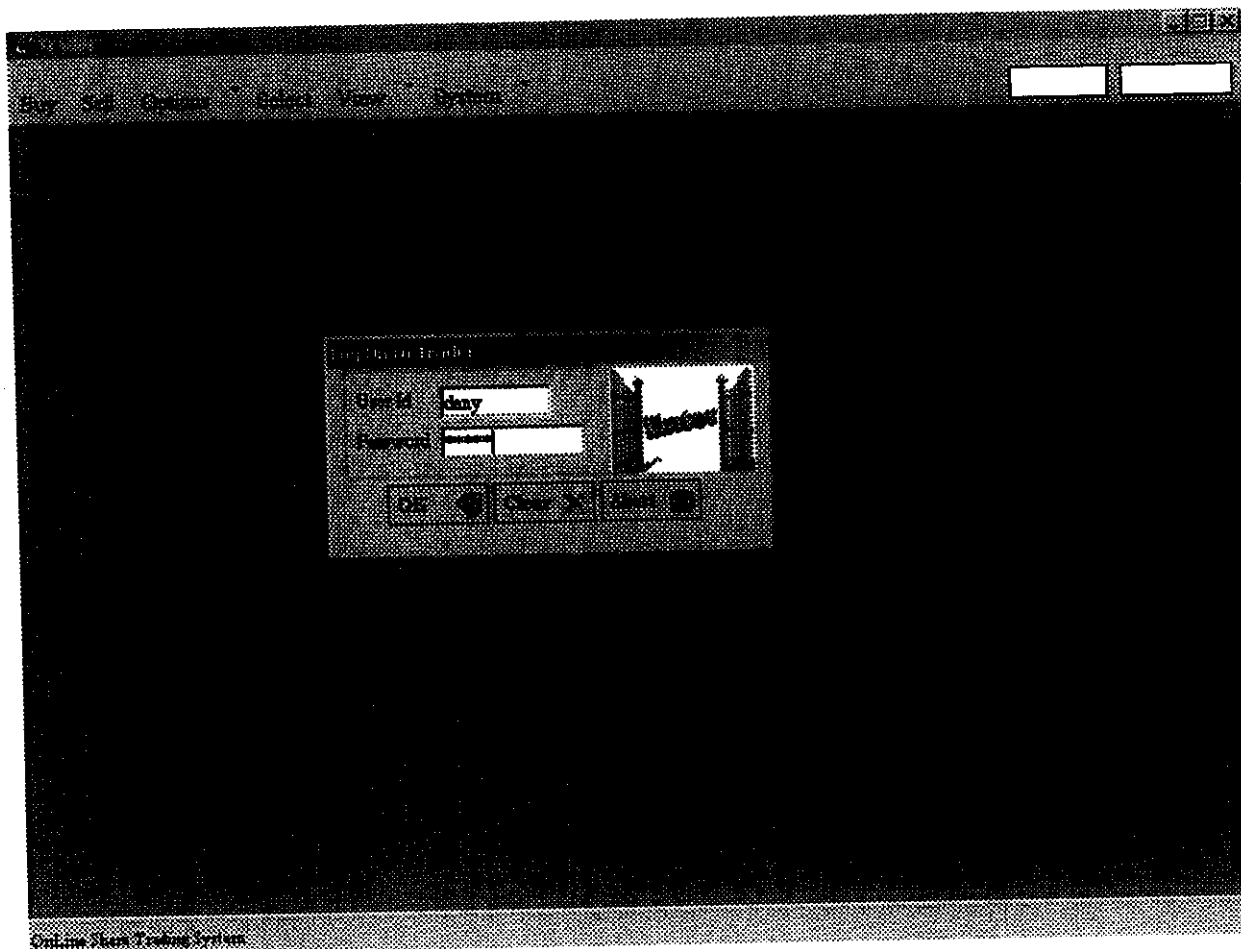
# Appendices

## 9. APPENDIX

### 9.1 SAMPLE SCREENS

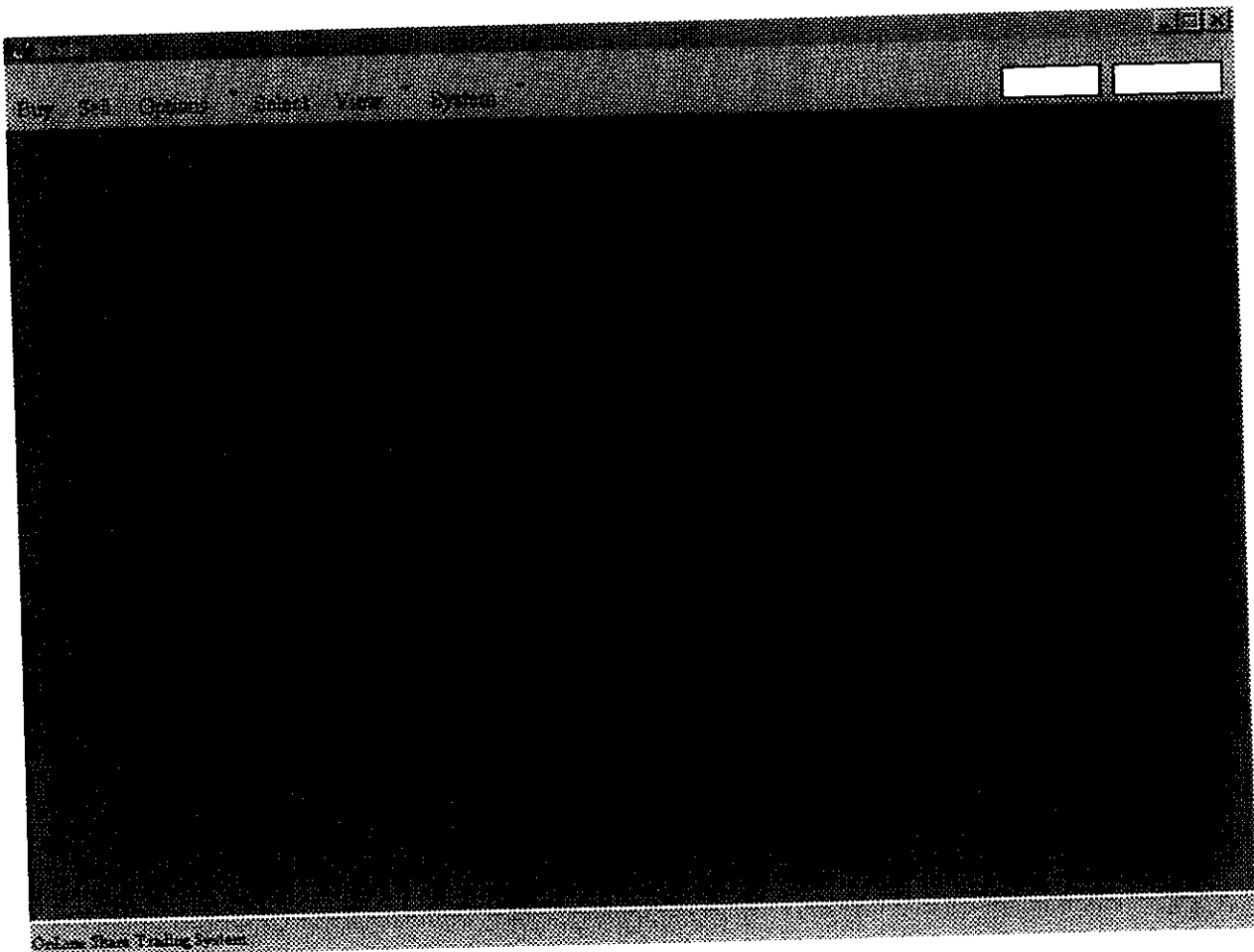
#### Login Form

The login form collects the login details from the user i.e., the user id and password. This information is given to the server for validation. If the user is not an authorized user then appropriate messages are given to the client. If the user id and password are correct the user is given the access to the system.



## Main Form

The main form is an MDI form that acts as a container for other forms. Here the typical child forms are the Refresh Screen that brings the online share index to the client, the forms for placing the orders, form for updating the orders, form for changing the current password, and other reports to the client. The Refresh Screen is always displayed during the trading time and once the trading time is over it is no longer available. The current date and server-time gets displayed in the upper right part of the screen. The server-time is very important because the trading occurs based on this time.



## Buy Order Screen

The buy orders for the shares are placed through the buy order screen provided for the purpose. The details for the orders such as scrip code, price, and quantity etc. has to be provided at the respective fields. The scrip code can be selected from the list of available shares provided to the user. The quantity is necessary while the lot is optional which denotes the minimum quantity the user will be trading. If not specified it takes the value of the quantity.

Scrip Code	Quantity (No. of)	Lot (No. of)	Price (Rs.)
ite	20	20	10
asa			
wsiagn			
ovacle			
premier			

Buy Order Form Loaded



## Sell Order Screen

Similar to the Buy Order Screen we have the Sell Order Screen that is used to place the Sell Orders for the different shares. Here also the details such as scrip code, price, and quantity etc. must be provided.

The screenshot shows a window titled "Trade" with a menu bar containing "Buy", "Options", "Market View", and "System". The date and time are displayed as "25/02/2003" and "2:46:43 PM". The main area is dark. At the bottom, there is a "SELL ORDER" form with the following fields:

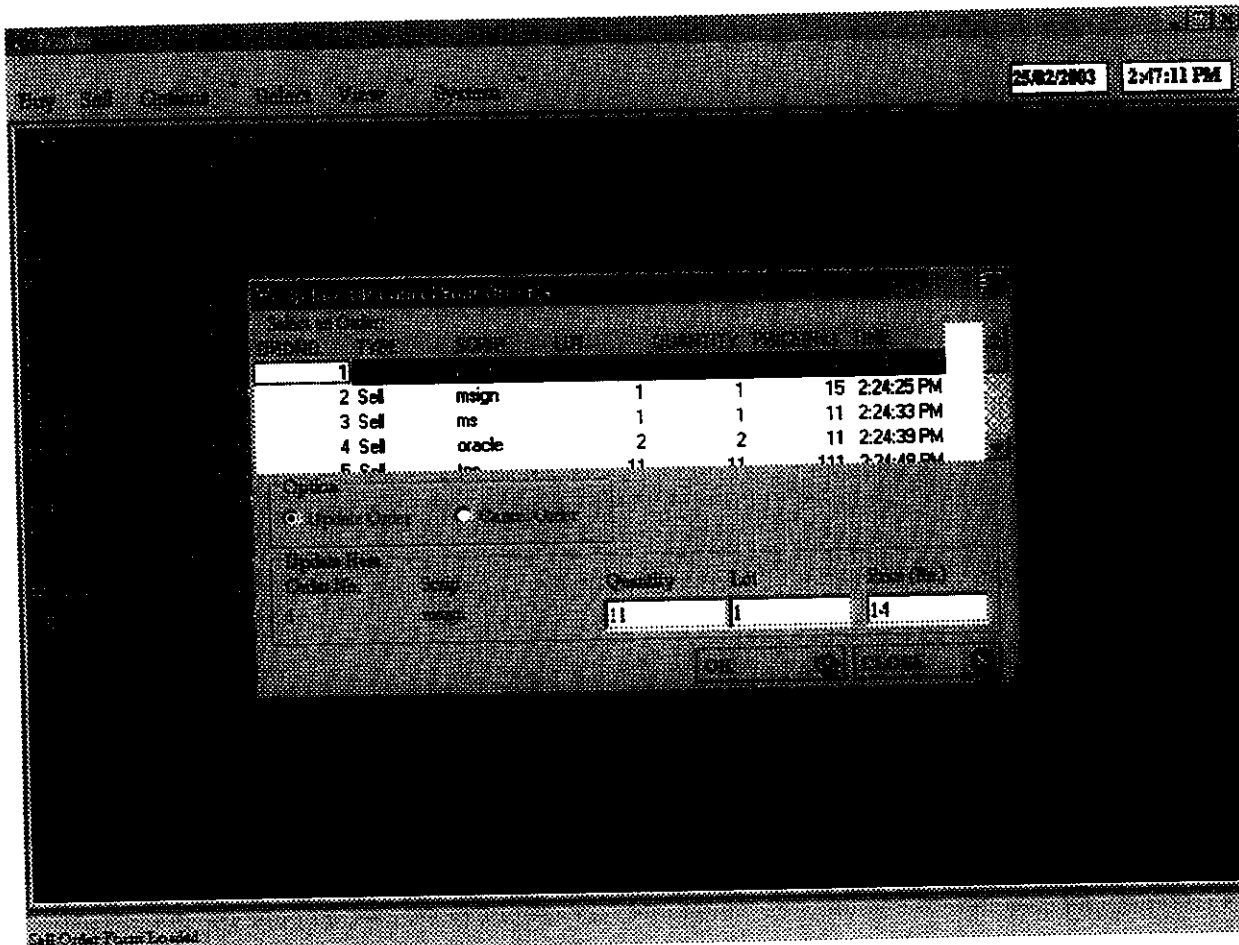
Scrip Code	Quantity (No. S)	Lot (No. S)	Price (Rs. S)
Mr Sun Scrip Oracle Tender	1	1	11.11

Buttons for "SUBMIT", "CANCEL", and "CLOSE" are located on the right side of the form.



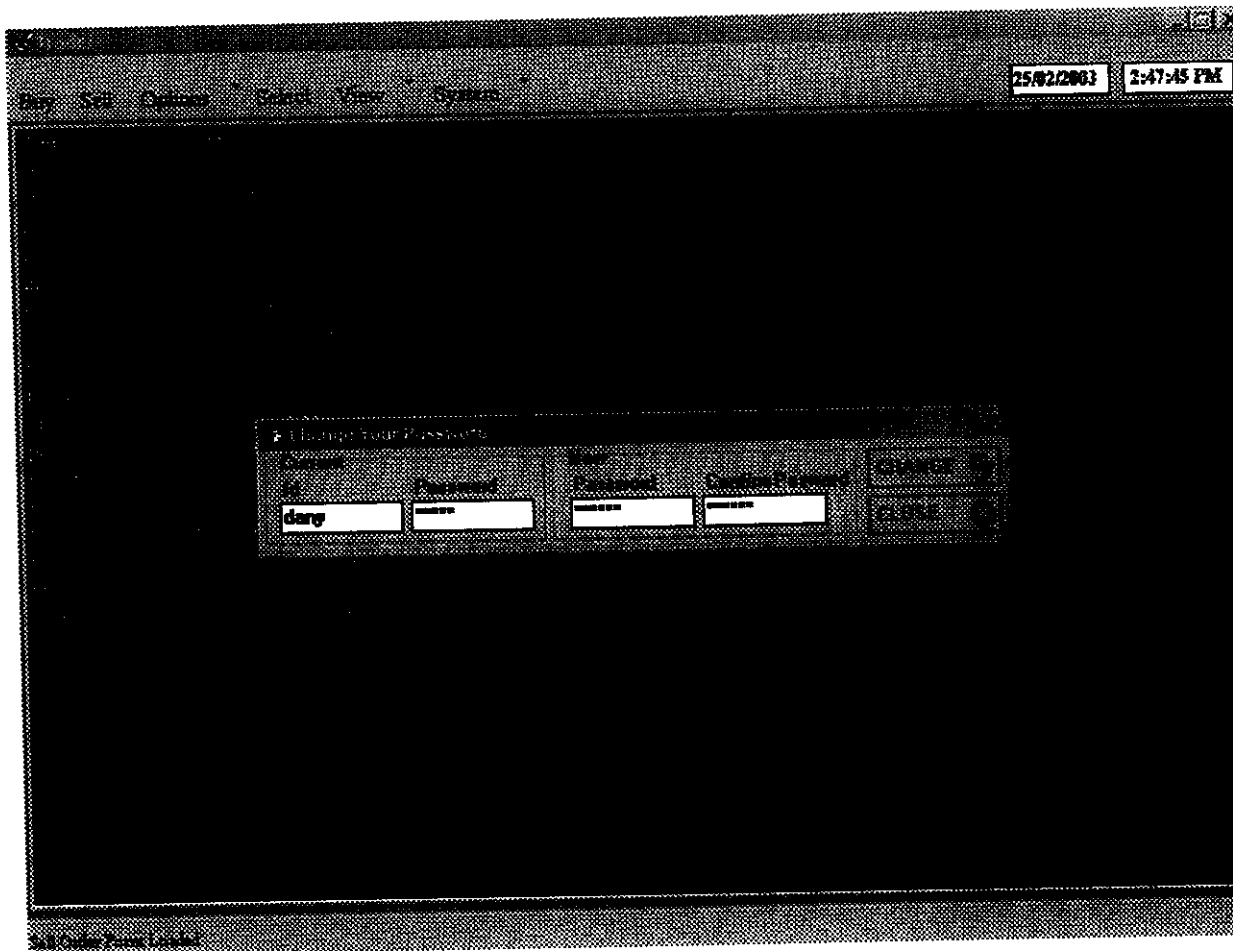
## Update/Cancel Order Screen

During the trading time there will be pending orders existing for the user. The ability to update and/or cancel these orders is of very importance with respect to the trading nature of the share system. In our system the clients are able to update and/or cancel their orders through the Update/Cancel Screen provided for the purpose. They get the details of the pending orders and can select the order to be updated or cancelled. If the operation is updation, they can edit the details and submit the order.



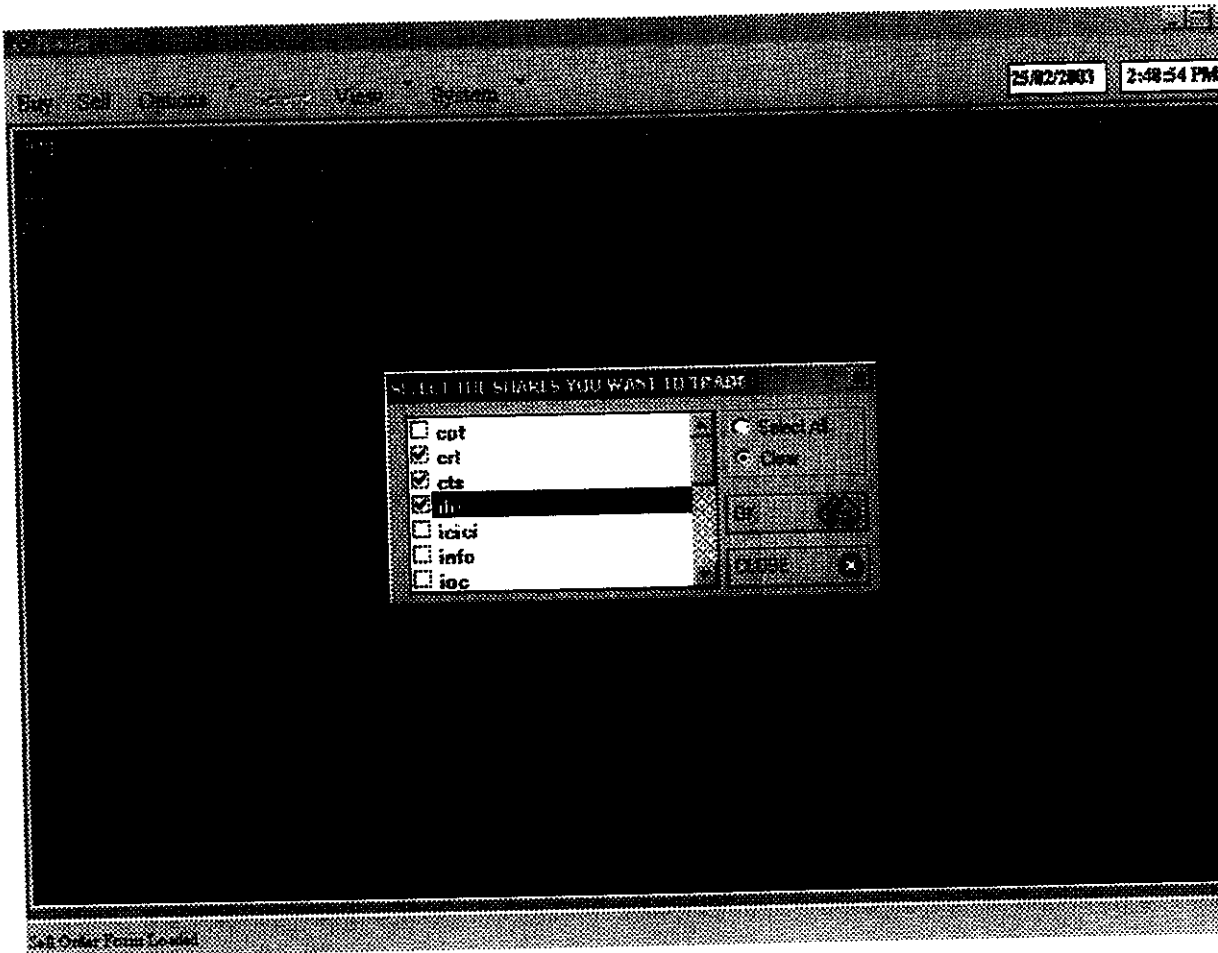
## Password Change

The password of a user can be changed from the client side by providing the current password details and the new details. The user gets an acknowledgment from the server once the updation is complete.



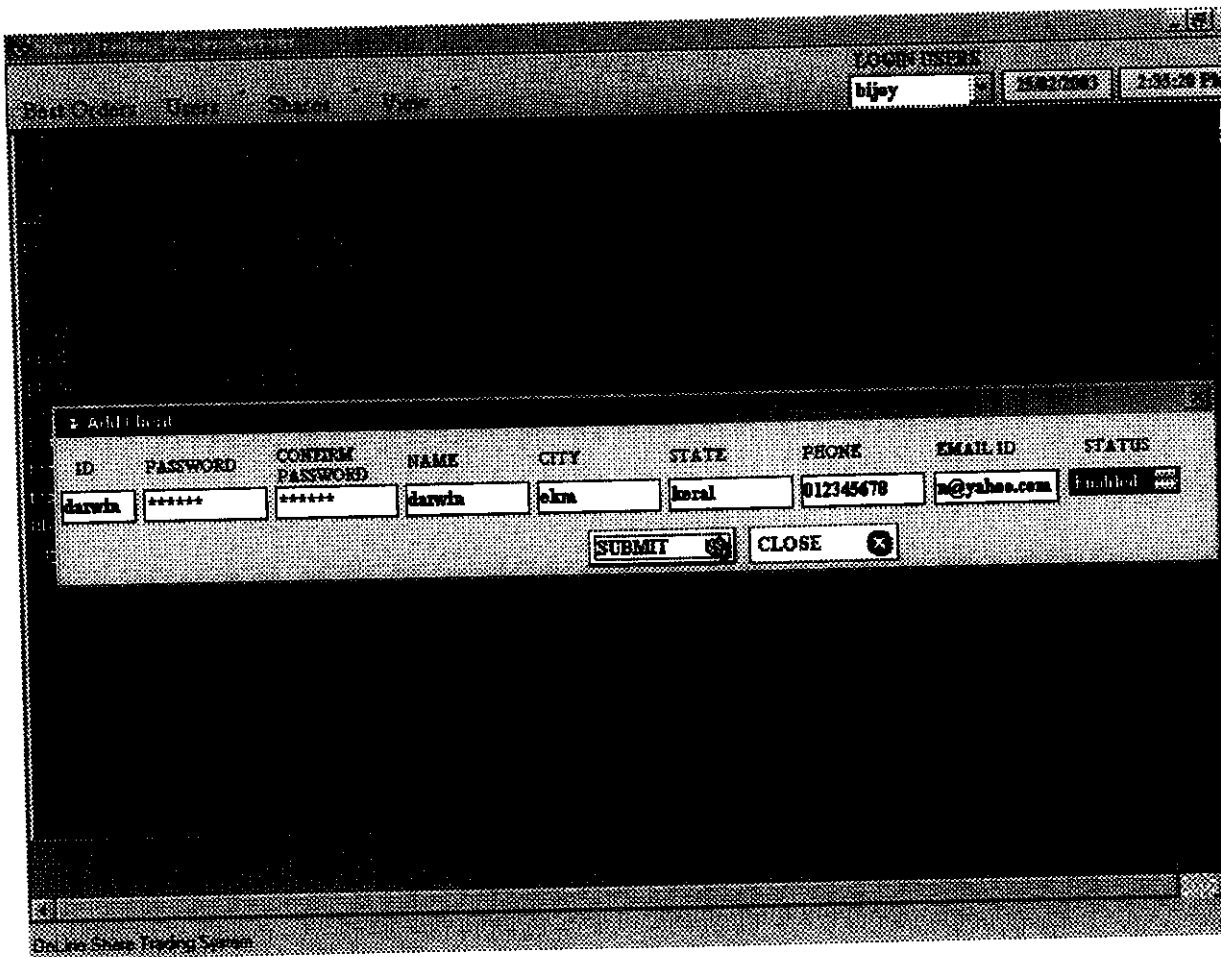
## Share Select Screen

Usually the Refresh Screen displays the details of all of the shares listed by the system. Sometimes the user may find it difficult to search for the share he wants to trade. And he will be interested only in the details of the shares that he is trading. The Share Select Screen allows the user to select the shares, the details of which only will be displayed in the Refresh Screen.



## Membership Creation

Through this screen a client gets registered. The details of the clients are the inputs to this screen. The client can suggest a user id and password and if a similar one doesn't exist then the details gets approved or else he has to give another id and password.



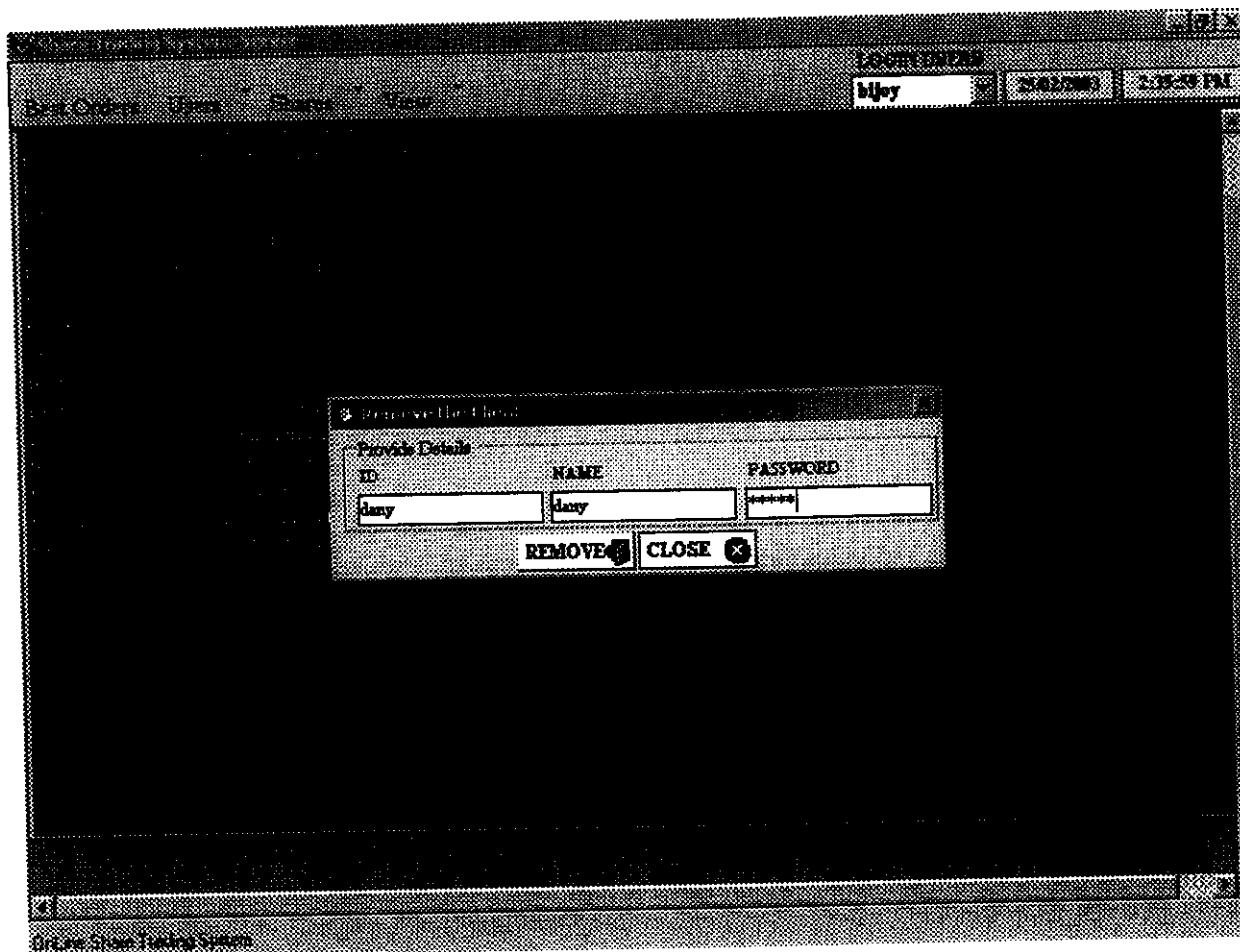
The screenshot shows a web application interface for creating a membership. At the top right, there are navigation links: "HOME", "ABOUT US", "CONTACT US", "LOGIN", "REGISTER", and "LOGOUT". Below these is a search bar with the text "bipoy". The main content area is titled "Add Member" and contains a form with the following fields:

ID	PASSWORD	CONFIRM PASSWORD	NAME	CITY	STATE	PHONE	EMAIL ID	STATUS
darwin	*****	*****	darwin	okna	horal	012345678	m@yahoo.com	Enabled

At the bottom of the form, there are two buttons: "SUBMIT" and "CLOSE".

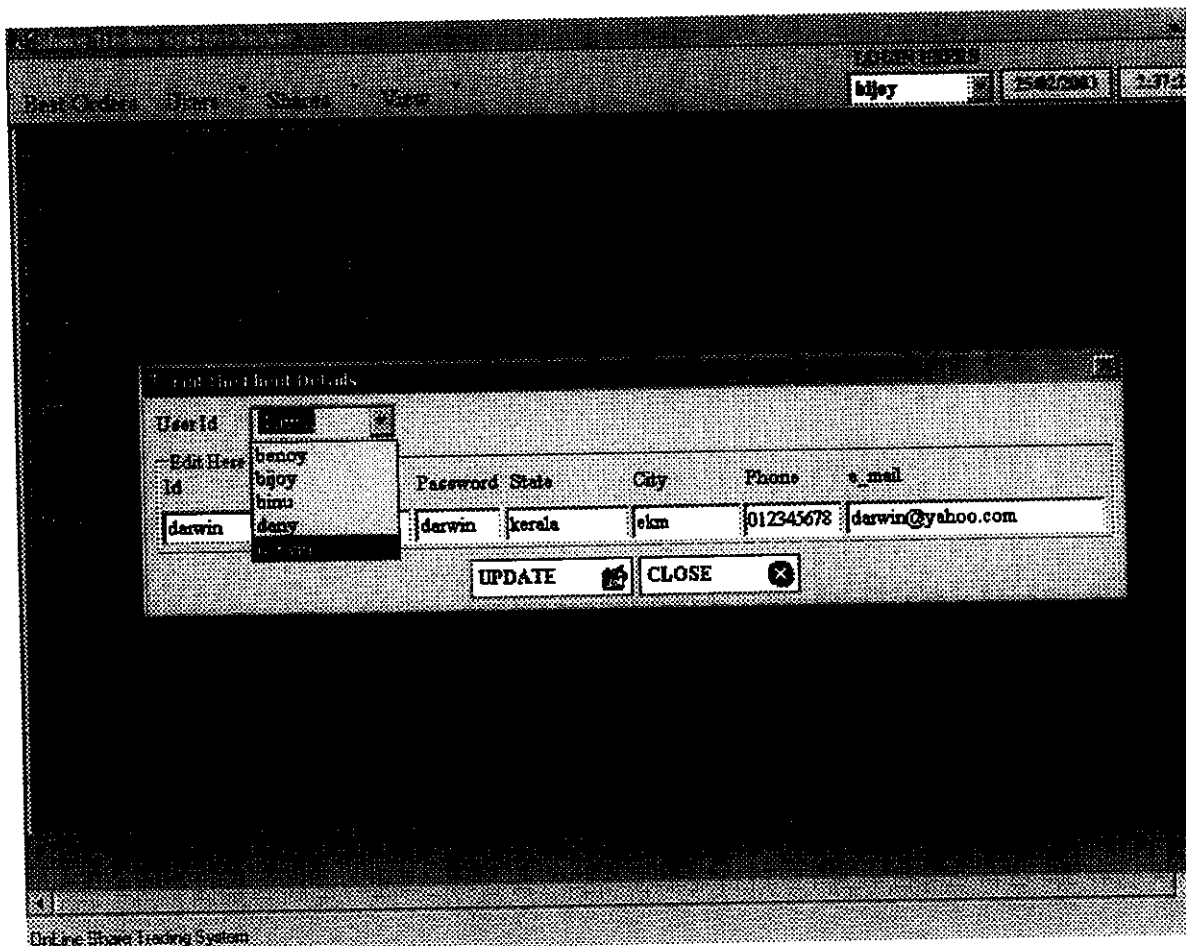
## Client Deletion Screen

The server side staffs are able to remove a client from the system at any point of time. The detail of the client such as name, password, and user id has to be provided at the fields in the screen.



## Client Membership Editing

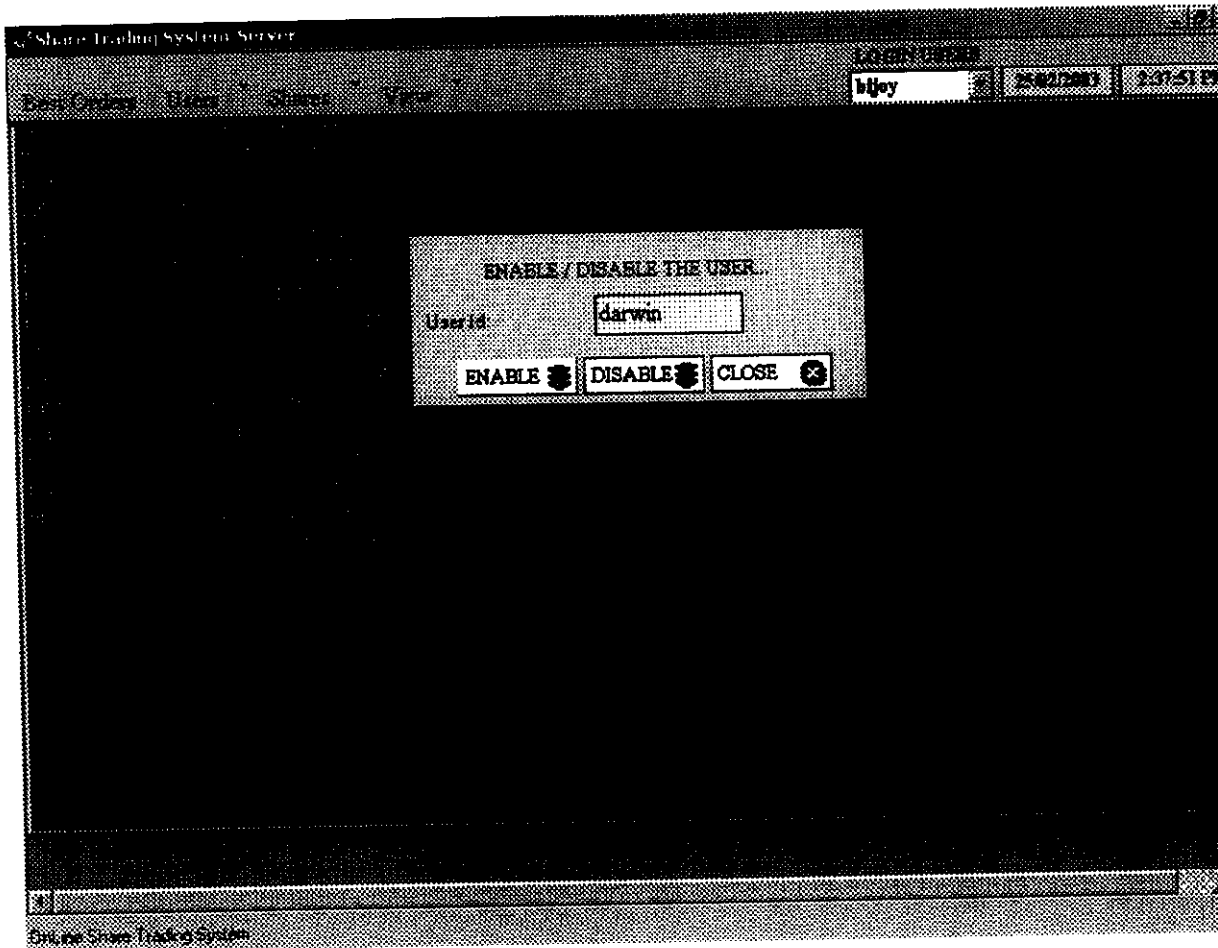
Once created the membership of the clients will need to be edited at a later time. The server part has the screen Client Membership Editing for this purpose. All the details furnished during the creation time can be edited including the user id and password. The screen is,





## Enable/Disable User

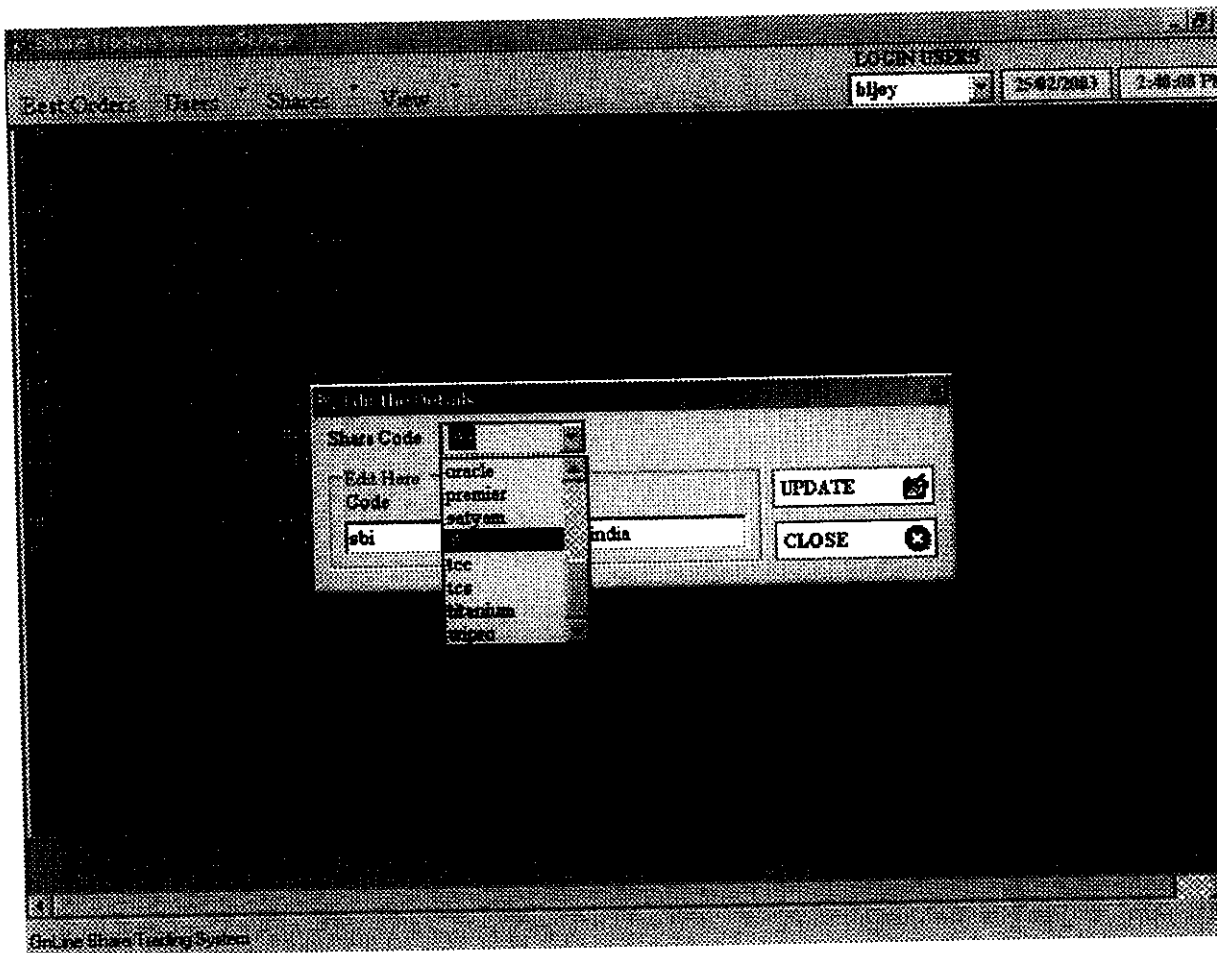
During the trading time the server need the overall control over the operations performed by the clients. The control over the clients is obtained mainly through the Enable/Disable User. Here by providing the user id and selecting the appropriate action the corresponding user can be enabled or disabled.





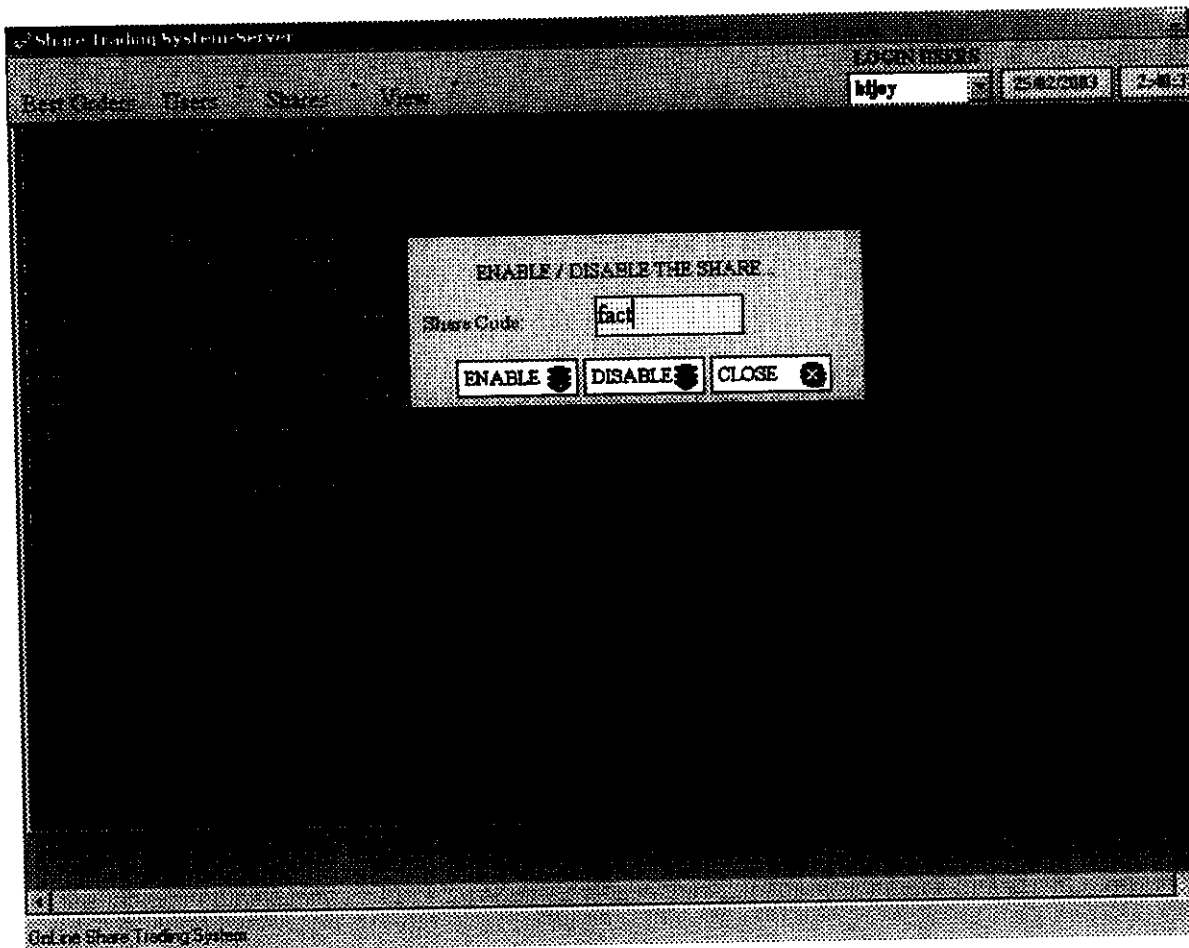
## Edit Share Details

The share details can be edited through the screen Edit the Details. All the details can be edited including the share code.



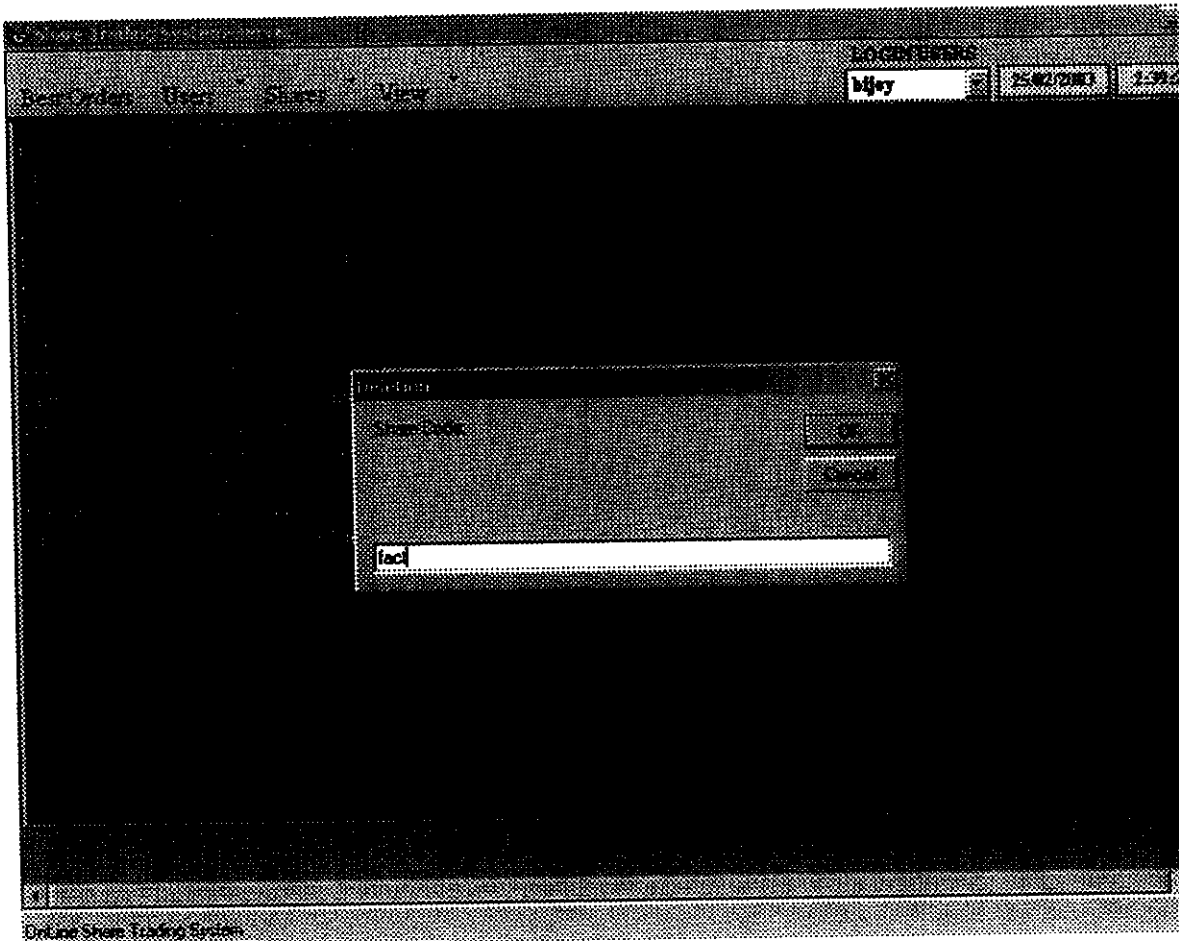
## Enable/Disable Share

The shares can be disabled/enabled during the trading if the broker's office wants to do so. By disabling the share it is no more available for trading and no details related to that share is available to the clients. To list it for trading and for the people to perform transactions on that share the server has to enable it again.



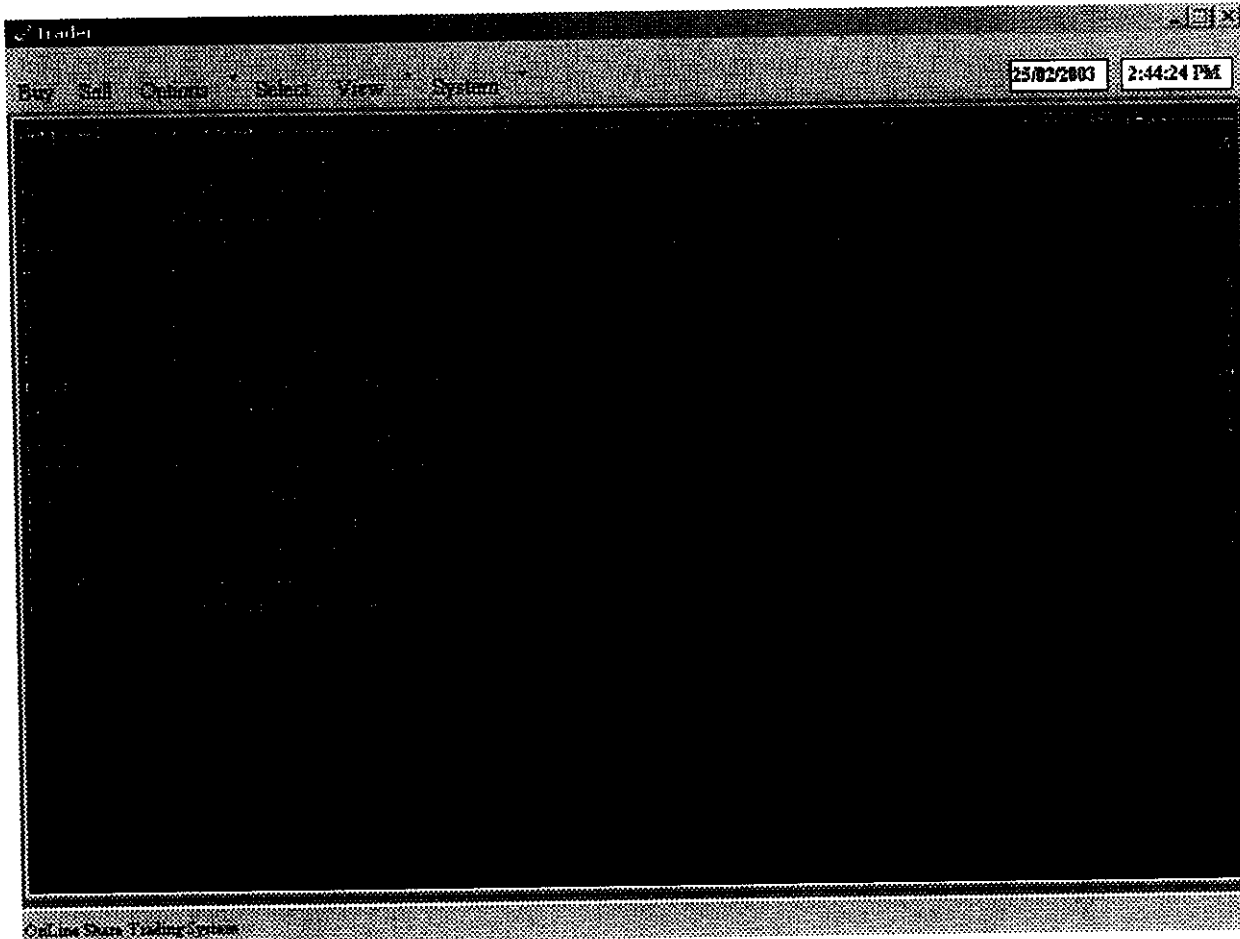
## Share Deletion

A share already being traded in the system can be deleted from the server. The share code of the share must be provided as the input. All the details for the share get deleted and the share gets deleted permanently from the system.



## Refresh Screen

The refresh screen displays the best of the orders placed for each type of share. The clients to place the orders use the price index from this screen. Thus the screen helps the user in analyzing the current market and finding out the appropriate transaction for him. The user is able to query the status of any displayed share at any point of time during the trading period.



## 9.2 SAMPLE CODE

The two important processes of the Online Share Trading System are the Order Matching process and the Online Information process. The source code for these processes are given which is written in the MS Visual Basic .NET programming language.

### Order Matching Process

```
Dim DAshare As New OleDbDataAdapter()
Dim DAsel As New OleDbDataAdapter()
Dim DAbuy As New OleDbDataAdapter()
Dim DATime As New OleDbDataAdapter()

Dim DSbuy As New DataSet()
Dim DSsel As New DataSet()
Dim DSshare As New DataSet()
Dim DSTime As New DataSet()

Private Sub OrderMatch()
    Dim i, j, RemQty As Integer
    Dim CurDate, CurTime As String

    CurDate = GetDate()
    CurTime = GetTime()

    Dim InsComm, UpComm, DelSel, DelBuy As OleDbCommand

    Dim SellSel As New OleDbCommand("select * from orderdet_tns
where ordtype='sell' order by ordno")
    SellSel.CommandType = CommandType.Text
    SellSel.Connection = Conn
    DAsel.SelectCommand = SellSel
    DSsel.Clear()
    DAsel.Fill(DSsel)

    Dim BuySel As New OleDbCommand("select * from orderdet_tns
where ordtype='buy ' order by ordno")
    BuySel.CommandType = CommandType.Text
```

```

BuySel.Connection = Conn
DAbuy.SelectCommand = BuySel
DSbuy.Clear()
DAbuy.Fill(DSbuy)

Dim DAttr As New OleDbDataAdapter()
Dim DStr As New DataSet()

Dim PriComm As New OleDbCommand("select * from trade_tns order
by tradeno", Conn)
PriComm.CommandType = CommandType.Text
DAttr.SelectCommand = PriComm
DStr.Clear()
DAttr.Fill(DStr)
If DStr.Tables(0).Rows.Count > 0 Then
    TradeNo = DStr.Tables(0).Rows(DStr.Tables(0).Rows.Count -
1).Item(0)
Else
    TradeNo = 1
End If

For i = 0 To DSsel.Tables(0).Rows.Count - 1
    For j = 0 To DSbuy.Tables(0).Rows.Count - 1
        If Trim(DSsel.Tables(0).Rows(i).Item(5)) =
Trim(DSbuy.Tables(0).Rows(j).Item(5)) And
DSsel.Tables(0).Rows(i).Item(8) <= DSbuy.Tables(0).Rows(j).Item(8) Then
'And Not DSsel.Tables(0).Rows(i).Item(4) =
DSbuy.Tables(0).Rows(j).Item(4)
            If DSsel.Tables(0).Rows(i).Item(7) =
DSbuy.Tables(0).Rows(j).Item(7) Then
                TradeNo = TradeNo + 1
                InsComm = New OleDbCommand("insert into
trade_tns values(" & TradeNo & ", '" & DSsel.Tables(0).Rows(i).Item(5) &
"', '" & DSsel.Tables(0).Rows(i).Item(7) & "', '" &
DSsel.Tables(0).Rows(i).Item(8) & "', '" &
DSbuy.Tables(0).Rows(j).Item(4) & "', '" &
DSsel.Tables(0).Rows(i).Item(4) & "', '" & CurDate & "', '" & CurTime &
"',0,0)", Conn)
                    InsComm.ExecuteNonQuery()
                    DelSel = New OleDbCommand("delete from
orderdet_tns where ordno=" & DSsel.Tables(0).Rows(i).Item(0), Conn)
                    DelSel.ExecuteNonQuery()
                    DelBuy = New OleDbCommand("delete from
orderdet_tns where ordno=" & DSbuy.Tables(0).Rows(j).Item(0), Conn)
                    DelBuy.ExecuteNonQuery()

                    Exit For
                    Exit For
            Else
                If DSsel.Tables(0).Rows(i).Item(7) >
DSbuy.Tables(0).Rows(j).Item(7) And DSbuy.Tables(0).Rows(j).Item(7) >=
DSsel.Tables(0).Rows(i).Item(6) Then

                    RemQty = DSsel.Tables(0).Rows(i).Item(7) -
DSbuy.Tables(0).Rows(j).Item(7)
                    If RemQty < DSsel.Tables(0).Rows(i).Item(6)
Then

```



```

Dim Comm As New OleDbCommand("update
orderdet_tns set mingty=" & RemQty & " where ordno=" &
DSsel.Tables(0).Rows(i).Item(0), Conn)
Comm.ExecuteNonQuery()
End If
TradeNo = TradeNo + 1
InsComm = New OleDbCommand("insert into
trade_tns values(" & TradeNo & ",'" & DSsel.Tables(0).Rows(i).Item(5) &
"', " & DSbuy.Tables(0).Rows(j).Item(7) & ", " &
DSsel.Tables(0).Rows(i).Item(8) & ",'" &
DSbuy.Tables(0).Rows(j).Item(4) & "','" &
DSsel.Tables(0).Rows(i).Item(4) & "','" & CurDate & "','" & CurTime &
"',0,0)", Conn)
InsComm.ExecuteNonQuery()

UpComm = New OleDbCommand("update
orderdet_tns set ordqty=" & RemQty & " where ordno=" &
DSsel.Tables(0).Rows(i).Item(0), Conn)
UpComm.ExecuteNonQuery()

DelBuy = New OleDbCommand("delete from
orderdet_tns where ordno=" & DSbuy.Tables(0).Rows(j).Item(0), Conn)
DelBuy.ExecuteNonQuery()

Exit For
Exit For
Else
If DSsel.Tables(0).Rows(i).Item(7) <
DSbuy.Tables(0).Rows(j).Item(7) And DSsel.Tables(0).Rows(i).Item(7) >=
DSbuy.Tables(0).Rows(j).Item(6) Then
RemQty =
DSbuy.Tables(0).Rows(j).Item(7) - DSsel.Tables(0).Rows(i).Item(7)
If RemQty <
DSbuy.Tables(0).Rows(j).Item(6) Then
Dim Comm As New
OleDbCommand("update orderdet_tns set mingty=" & RemQty & " where
ordno=" & DSbuy.Tables(0).Rows(j).Item(0), Conn)
Comm.ExecuteNonQuery()
End If
TradeNo = TradeNo + 1
InsComm = New OleDbCommand("insert into
trade_tns values(" & TradeNo & ",'" & DSsel.Tables(0).Rows(i).Item(5) &
"', " & DSsel.Tables(0).Rows(i).Item(7) & ", " &
DSsel.Tables(0).Rows(i).Item(8) & ",'" &
DSbuy.Tables(0).Rows(j).Item(4) & "','" &
DSsel.Tables(0).Rows(i).Item(4) & "','" & CurDate & "','" & CurTime &
"',0,0)", Conn)
InsComm.ExecuteNonQuery()

UpComm = New OleDbCommand("update
orderdet_tns set ordqty=" & RemQty & " where ordno=" &
DSbuy.Tables(0).Rows(j).Item(0), Conn)
UpComm.ExecuteNonQuery()

DelSel = New OleDbCommand("delete from
orderdet_tns where ordno=" & DSsel.Tables(0).Rows(i).Item(0), Conn)
DelSel.ExecuteNonQuery()

```

```

Exit For
Exit For
End If
End If
End If
End If
Next
Next
End Sub

```

## Online Information Process

```

Private Sub FillGrid()
    FlexGrid1.Clear()
    InitFlex()
    DSsell.Clear()
    Dim CntT As Integer
    For CntT = 0 To DSsell.Tables.Count - 1
        DSsell.Tables.RemoveAt(CntT)
    Next

    Dim SelComm As New OleDbCommand("select * from orderdet_tns
where status='1' order by ordno")
    SelComm.CommandType = CommandType.Text
    SelComm.Connection = Conn
    DAorder.SelectCommand = SelComm
    DSorder.Clear()
    DAorder.Fill(DSorder)
    FlxRow = 1

    myDataColumn = New DataColumn()
    myDataTable.Columns.Add(myDataColumn)
    Dim sr As StreamReader = File.OpenText("select.txt")
    Dim Input As String
    Dim FlagSelect As Boolean
    FlagSelect = True
    Input = sr.ReadLine()
    If Not Input Is Nothing Then
        FlagSelect = False

        DSsell.Tables.Add(myDataTable)
        While Not Input Is Nothing
            myDataRow = myDataTable.NewRow()
            myDataRow(0) = Trim(Input)
            myDataTable.Rows.Add(myDataRow)
            Input = sr.ReadLine()
        End While
    End If
    sr.Close()
    If FlagSelect Then

```

```

        Dim SelSel As New OleDbCommand("SELECT DISTINCT share FROM
(SELECT * FROM orderdet_tns) DERIVEDTBL")
        SelSel.CommandType = CommandType.Text
        SelSel.Connection = Conn
        DAsell.SelectCommand = SelSel
        DSsell.Clear()
        DAsell.Fill(DSsell)
    End If
    Dim i, j, k, l, min, max, m As Integer
    Dim SlNoSel As Integer
    Dim SlNoBuy As Integer

    Dim MinPrice As Double
    Dim MaxPrice As Double
    Dim str1, str2 As String
    For i = 0 To DSsell.Tables(0).Rows.Count - 1 '1
        str1 = DSsell.Tables(0).Rows(i).Item(0)
        For j = 0 To DSorder.Tables(0).Rows.Count - 1 '2
            str2 = DSorder.Tables(0).Rows(j).Item(5)
            SlNoSel = 0 '3
            SlNoBuy = 0

            If Trim(DSsell.Tables(0).Rows(i).Item(0)) =
Trim(DSorder.Tables(0).Rows(j).Item(5)) Then
                If Trim(DSorder.Tables(0).Rows(j).Item(3)) = "sell"
Then
                    min = j
                    MinPrice = DSorder.Tables(0).Rows(j).Item(8)
                    SlNoSel = DSorder.Tables(0).Rows(j).Item(0)
                    MaxPrice = -1
                End If

                If Trim(DSorder.Tables(0).Rows(j).Item(3)) = "buy"
Then
                    max = j
                    MaxPrice = DSorder.Tables(0).Rows(j).Item(8)
                    SlNoBuy = DSorder.Tables(0).Rows(j).Item(0)
                    MinPrice = 1000000000
                End If

                For k = j To DSorder.Tables(0).Rows.Count - 1 '7
                    If Trim(DSsell.Tables(0).Rows(i).Item(0)) =
Trim(DSorder.Tables(0).Rows(k).Item(5)) And
Trim(DSorder.Tables(0).Rows(k).Item(3)) = "sell" And MinPrice >
DSorder.Tables(0).Rows(k).Item(8) Then '8
                        min = k '9
                        MinPrice =
DSorder.Tables(0).Rows(k).Item(8) '10
                        SlNoSel = DSorder.Tables(0).Rows(k).Item(0)
                    '11
                    End If '14

                    If Trim(DSsell.Tables(0).Rows(i).Item(0)) =
Trim(DSorder.Tables(0).Rows(k).Item(5)) And

```

```

Trim(DSorder.Tables(0).Rows(k).Item(3)) = "buy" And MaxPrice <
DSorder.Tables(0).Rows(k).Item(8) Then '8
    max = k '9
    MaxPrice =
DSorder.Tables(0).Rows(k).Item(8) '10
    SlNoBuy = DSorder.Tables(0).Rows(k).Item(0)
'11
        End If '14
    Next k '15
End If '

If (SlNoSel > 0) Then '17
    Dim SelItem As New OleDbCommand("Select * from
orderdet_tns where ordno=" & SlNoSel) '19
    SelItem.CommandType = CommandType.Text '20
    SelItem.Connection = Conn '21

    DAselitem.SelectCommand = SelItem '22

    DSselitem.Clear() '23
    DAselitem.Fill(DSselitem) '24
End If

If SlNoBuy > 0 Then
    Dim BuyItem As New OleDbCommand("Select * from
orderdet_tns where ordno=" & SlNoBuy)
    BuyItem.CommandType = CommandType.Text
    BuyItem.Connection = Conn

    DAbuyitem.SelectCommand = BuyItem

    DSbuyitem.Clear()
    DAbuyitem.Fill(DSbuyitem)
End If

If SlNoSel > 0 Or SlNoBuy > 0 Then
    Dim Cnt As Integer
    Dim Flg As Boolean
    Flg = False

    If SlNoSel > 0 And SlNoBuy > 0 Then
        If DSbuyitem.Tables(0).Rows.Count > 0 And
DSselitem.Tables(0).Rows.Count > 0 Then
            If
Trim(DSbuyitem.Tables(0).Rows(0).Item(5)) =
Trim(DSselitem.Tables(0).Rows(0).Item(5)) Then
                For Cnt = 0 To
DSshare.Tables(0).Rows.Count - 1
                    If
Trim(DSbuyitem.Tables(0).Rows(0).Item(5)) =
Trim(DSshare.Tables(0).Rows(Cnt).Item(0)) Then
                        Flg = True
                        Exit For
                    End If
                Next Cnt
            End If
        End If
    End If

```

```

FlexGrid1.Rows = FlxRow + 1
If Flg = True Then
    FlexGrid1.set_TextMatrix(FlxRow, 1,
DSshare.Tables(0).Rows(Cnt).Item(1))
    FlexGrid1.set_TextMatrix(FlxRow, 0,
DSbuyitem.Tables(0).Rows(0).Item(5))
    Flg = False
End If
For m = 2 To 3
    FlexGrid1.set_TextMatrix(FlxRow, m,
DSbuyitem.Tables(0).Rows(0).Item(m + 4))
Next

Dim b As Integer
Dim pr As String
Dim PrFlgm As Boolean
prflgm = False
pr =
Trim(DSbuyitem.Tables(0).Rows(0).Item(8))
For b = 0 To pr.Length - 1
    If pr.Chars(b) = "." Then
        PrFlgm = True
        If pr.Length = b + 1 Then
            pr = pr + "00"
        Else
            If pr.Length = b + 2 Then
                pr = pr + "0"
            End If
        End If
    End If
Next
If PrFlgm = False Then
    pr = pr + ".00"
End If
FlexGrid1.set_TextMatrix(FlxRow, 4, pr)
pr = Nothing
For m = 5 To 6
    FlexGrid1.set_TextMatrix(FlxRow, m,
DSselitem.Tables(0).Rows(0).Item(m + 1))
Next
prflgm = False
pr =
Trim(DSselitem.Tables(0).Rows(0).Item(8))
For b = 0 To pr.Length - 1
    If pr.Chars(b) = "." Then
        PrFlgm = True
        If pr.Length = b + 1 Then
            pr = pr + "00"
        Else
            If pr.Length = b + 2 Then
                pr = pr + "0"
            End If
        End If
    End If
Next

```

```

        If PrFlgm = False Then
            pr = pr + ".00"
        End If
        FlexGrid1.set_TextMatrix(FlxRow, 7, pr)
        pr = Nothing

        FlxRow = FlxRow + 1
    End If
Else
    If SlNoSel = 0 And SlNoBuy > 0 Then
        For Cnt = 0 To DSshare.Tables(0).Rows.Count
- 1
            If
Trim(DSbuyitem.Tables(0).Rows(0).Item(5)) =
Trim(DSshare.Tables(0).Rows(Cnt).Item(0)) Then
                Flg = True
                Exit For
            End If
        Next Cnt

        FlexGrid1.Rows = FlxRow + 1
        If Flg = True Then
            FlexGrid1.set_TextMatrix(FlxRow, 0,
DSbuyitem.Tables(0).Rows(0).Item(5))
            FlexGrid1.set_TextMatrix(FlxRow, 1,
DSshare.Tables(0).Rows(Cnt).Item(1))
            Flg = False
        End If
        For m = 2 To 3
            FlexGrid1.set_TextMatrix(FlxRow, m,
DSbuyitem.Tables(0).Rows(0).Item(m + 4))
        Next

        Dim b As Integer
        Dim pr As String
        Dim PrFlgm As Boolean
        prflgm = False
        pr =
Trim(DSbuyitem.Tables(0).Rows(0).Item(8))
        For b = 0 To pr.Length - 1
            If pr.Chars(b) = "." Then
                PrFlgm = True
                If pr.Length = b + 1 Then
                    pr = pr + "00"
                Else
                    If pr.Length = b + 2 Then
                        pr = pr + "0"
                    End If
                End If
            End If
        Next
        If PrFlgm = False Then
            pr = pr + ".00"
        End If
        FlexGrid1.set_TextMatrix(FlxRow, 4, pr)
        pr = Nothing
    End If

```

```

        For m = 5 To 6
            FlexGrid1.set_TextMatrix(FlxRow, m, "")
        Next
        FlxRow = FlxRow + 1
    Else
        If SlNoSel > 0 And SlNoBuy = 0 Then

            For Cnt = 0 To
DSshare.Tables(0).Rows.Count - 1
                If
Trim(DSselitem.Tables(0).Rows(0).Item(5)) =
Trim(DSshare.Tables(0).Rows(Cnt).Item(0)) Then
                    Flg = True
                    Exit For
                End If
            Next Cnt

            FlexGrid1.Rows = FlxRow + 1
            If Flg = True Then
                FlexGrid1.set_TextMatrix(FlxRow, 0,
DSselitem.Tables(0).Rows(0).Item(5))
                FlexGrid1.set_TextMatrix(FlxRow, 1,
DSshare.Tables(0).Rows(Cnt).Item(1))
                Flg = False
            End If

            For m = 2 To 3
                FlexGrid1.set_TextMatrix(FlxRow, m,
"" )

                Next
                '*****
                Dim b As Integer
                Dim pr As String
                Dim PrFlgm As Boolean
                prflgm = False
                For m = 5 To 6
                    FlexGrid1.set_TextMatrix(FlxRow, m,
DSselitem.Tables(0).Rows(0).Item(m + 1))
                Next
                prflgm = False
                pr =
Trim(DSselitem.Tables(0).Rows(0).Item(8))
                For b = 0 To pr.Length - 1
                    If pr.Chars(b) = "." Then
                        PrFlgm = True
                        If pr.Length = b + 1 Then
                            pr = pr + "00"
                        Else
                            If pr.Length = b + 2 Then
                                pr = pr + "0"
                            End If
                        End If
                    Exit For
                End If
            End If
            Next
            If PrFlgm = False Then
                pr = pr + ".00"
            End If
        End If
    End If

```

```
End If
FlexGrid1.set_TextMatrix(FlxRow, 7, pr)
pr = Nothing
FlxRow = FlxRow + 1
End If
End If
End If
Exit For
End If
Next j '32
Next i '33
End Sub
```