Exhibit D

All of the Above, Inc.
P.O. Box 2765
Sausalito, California 94966
(415)460-0642
aota@woz.org

TRADING GAME

DESIGN DOCUMENT

version 1.0
March 5, 1998

All of the Above, Inc.
P.O. Box 2765
Sausalito, California 94966
(415)460-0642
aota@woz.org

TRADING GAME DESIGN DOCUMENT

1. INTRODUCTION

The Trading Game is an Internet based trading system that presents the user with a real time, interactive trading environment. Through a Java client interface, the user trades products with a number of other users via a server-maintained market environment.

The user will be presented with product information, including price, volume, and trading position in a compact and easy to read format. The user will be able to send buy and sell requests easily and see results as they happen in real time, browse account information, access a chat room, news and other pertinent information all from the convenience of a web browser.

The first stage of this system is a simulation using simulated money and simulated products. The final evolution of this system will be to use real products and real money in a secure transaction environment. The secure version of this system will require account verification, audit trails, transaction and account security, zero fault tolerance, and encryption technologies.

This Design Document is intended to be a detailed description of the Graphical User Interface and the structure of the trading architecture necessary to service the end user GUI. It is not intended to be a detailed description of the secure account management system or audit system.

2. TARGET PLATFORM

The Trading Game is expected to be released no sooner than Q4 1998. Therefore, the target platform for which it is being created is somewhat ambitious by January 1, 1998 standards. Windows95 or NT, Internet Explorer 4.0, Java enabled, Pentium 133, 24 MB RAM, 28.8K modem.

The Trading Game demonstration program may be designed to operate with different specifications from the above target due to limitations of time.

3. HOW IT WORKS[1]

The initial problem with any trading system is that some people have an inherent advantage in the market due to the latency of the market status to trade execution loop. This is why some people pay a small fortune for a place on the stock exchange floor. A trading system which only exists online can eliminate nearly all of this disparity and simultaneously enhance the thrill of competition to the betterment of all participants.

We have developed an event-driven architecture[2] which attempts to put all traders on an even footing with an equal opportunity to participate in trades regardless of latency. The system works on time-stamped windows of events we call TTime. TTime is used as the update window for trades. Each T is sequentially numbered. We adjust TTime duration depending upon the number of users and

[1] See Appendix for process diagrams
[2] This architecture may be patentable

the second standard deviation (adjustable) of their average response times. All trades are events synchronized to TTime and executed in the same Ttime. There is no trading advantage to having a T1 connection, although users on 14.4 modems will probably fall outside the second standard deviation.

TTime is derived from requests coming from the client applet. This approach means that the client applet can tell if a TTime was lost by comparing the sequential time stamped event received to the one requested. It also enables us to reconstruct trades at the server or the client. TTime durations may vary as more users log on, or if the Internet experiences widespread delays, but the TTimes will still be sequential. Having the client applet request information updates also provides for session timeouts if the client goes away or if the line goes dead.

When a user clicks the TRADE button for a buy or sell, the data is sent to the transaction server with information about which TTime is current at the user. If the user made the trade within the TTime window, the trade will be put into the queue for processing. At the end of TTime, all trades which have been submitted will be sorted with a set of arbitration rules on the basis of TTime (with delayed Ts given first priority), other priority metrics (possibly including parameters such as "is this an important customer?"), volume and price. Then the trades will be matched, executed, updated to the various product and account databases, and distributed to users as an updateT. (see figure X)

A single updateT requires a data set of less than 255 bytes, the minimum size of a TCP/IP block, will be sent to all users. This display data set will be optimized to update the display with relevant data and step to the next TTime.

Each product is traded separately with its own unique TTime. The object-oriented architecture will enable rapid creation of new products with very simple point and click decisions. The connections to the account information and real-word stock ticker information will be made as objects which are also quickly replicated. It is not unreasonable to expect that new products will be able to be created, tested and deployed in less than a day since nearly all of the code will be identical.

Graphics, user appearance and other features are also independent objects which can be regularly replaced without affecting other objects.

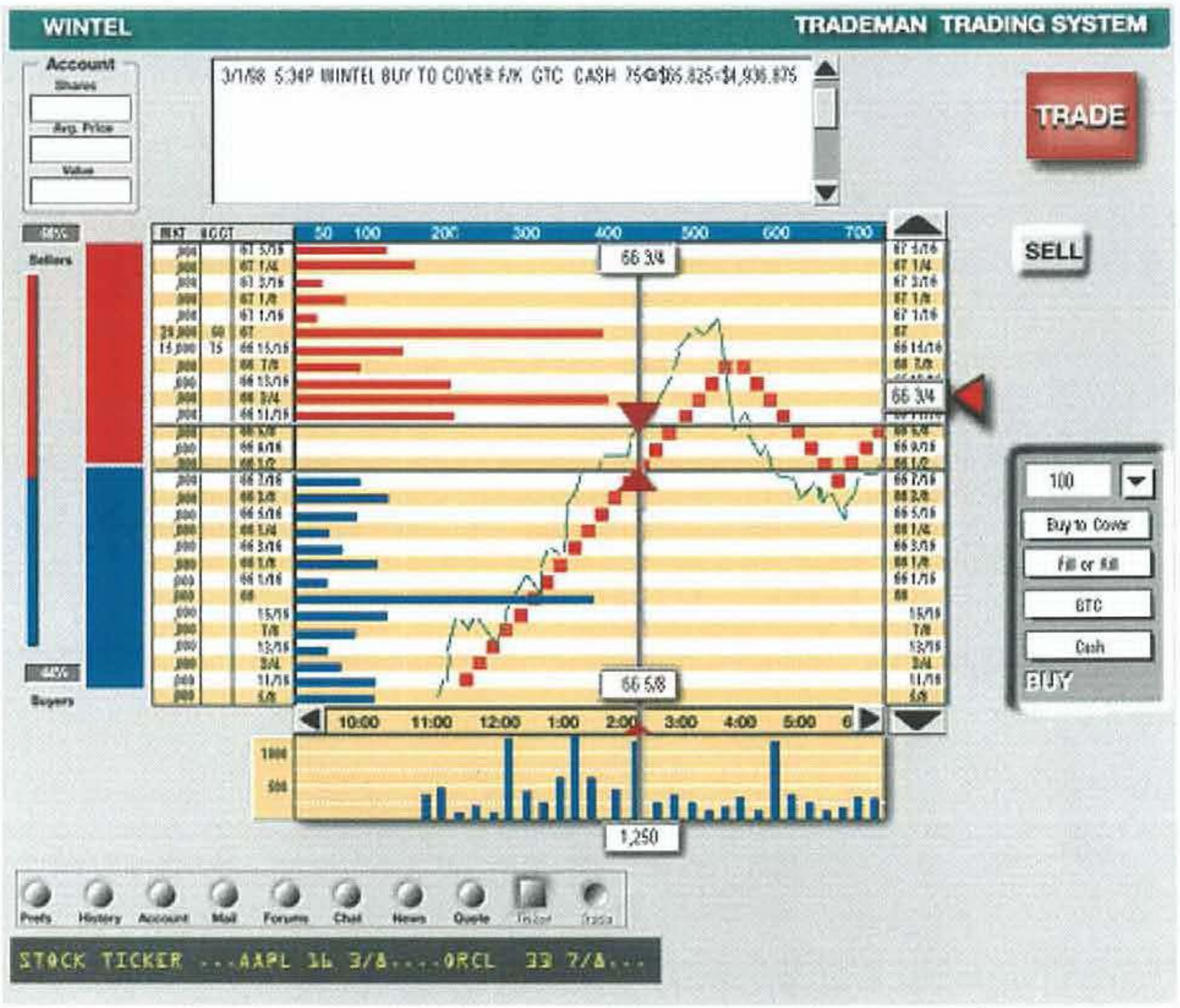
Trading cycles should be very short, on the order of every 10 seconds with under 1,000 simultaneous traders. Since each product is traded separately, popular products can be given their own server and connections for better performance. Multiple products will be able to be traded on a single low-cost server as long as the number of users remains small. The entire trading system could run on a single low-cost server, although we don't recommend this for robust financial transactions.

4. CLIENT SOFTWARE

The client software breaks down into many different components. Quite a few of these components are self displaying, Others draw themselves within another component. This allows modularity within the GUI, enabling it to only redraw the sections of the graphics that require updating, resulting in dramatically increased efficiency. Double buffering will be used to eliminate flicker in all critical sections.

The graphic example shown (ver 2.1) is designed to be an easily replaced set of image objects. We envision customers having a choice of interface styles from staid to wild. The system is designed so that the underlying data is identical to all users, regardless of their display objects.

The following section is an item by item breakdown of the Graphical User Interface, describing each item, its function and any other relevant function.



```
3/1/98 5:34P WINTEL BUY TO COVER F/K GTC CASH 75@$65.825=$4,936.875
```

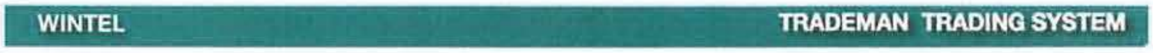
```
STOCK TICKER ...AAPL 36 3/8....ORCL 33 7/8...
```

Screen capture of complete sample Trading System GUI [3]

MainApplet This is the applet's display area itself. Its size is determined by the height and width parameters within the <applet> HTML tag. The applet will fetch the graphics from the server and distribute them to the various objects when the applet loads. This object functions as a container for all the functional parts of the Applet.

- The applet will use a border layout.
- North: Title
- South: ButtonPanel
- Center: DisplayCardPanel

Title The Title object will consist of a simple bar at the top of the screen that displays the name of the product being traded at the left, and another label at the right that either says: "TRADING GAME TRADING SYSTEM" or "PREFERENCES", depending on which card is being displayed below.



Title Object

[3] Unless otherwise noted all images are from graphic design version 2.1 at 65% of scale

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.