10-29-1998

Contract manager anywhere and internet-enabled database application

Yao-Jen Chang
Florida International University

DOI: 10.25148/etd.FI14060141

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

CONTRACT MANAGER ANYWHERE

AN INTERNET-ENABLED DATABASE APPLICATION

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Yao-Jen Chang

1998

To: Dean Arthur W. Herriott
College of Arts and Sciences

This thesis written by Yao-Jen Chang, and entitled Contract Manager Anywhere --- An Internet-enabled Database Application, having been approved in respect to style and intellectual content, is referred to you for judgement.

We have read this thesis and recommended that it be approved.

Dawn Holmes

Nagarajan Prabhakaran

Wei Sun, Major Professor

Date of Defense: October 29, 1998

The thesis of Yao-Jen Chang is approved.

Dean Arthur W. Herriott
College of Arts and Sciences

Dean Richard L. Campbell
Division of Graduate Studies

Florida International University, 1998

This thesis is dedicated to my Father, Chang, Chi;

my Mother, Chen, Sue-I.

They teach me how to think.

ACNOWLEDGEMENTS

The author acknowledges the contribution and support of the members of my committee, Dr. Dawn Holmes and Dr. Nagarajan Prabhakaran, who offered their helpful comments and patience.

A special "thank you" goes to Dr. Wei Sun who gave me his guidance, inspiration, support and great help through all my graduate studying, especially for having the confidence in me.

I also wish to acknowledge my friends for their support and comments: Kuanwen Fang, Debra L. Davis, Maria I. Monteagudo, Nanchun Lin, Mr. and Mrs. Hsu with Cooperate Bank and Nick Baldwin with FLDOE.

I could not have accomplished this project without them. Thank you.

ABSTRACT OF THE THESIS

CONTRACT MANAGER ANYWHERE

AN INTERNET-ENABLED DATABASE APPLICATION

by

Yao-Jen Chang

Florida International University, 1998

Miami, Florida

Professor Wei Sun, Major Professor

This thesis describes the design and implementation of an Internet-enabled database application which uses ASP (Active Server Pages) and other related knowledge such as ADO (ActiveX Data Object), ODBC (Open DataBase Connectivity) and OLE-DB (Object Linking and Embed-DataBase). Because it is an Internet-enabled application, the subject of "Security" was also studied by implementing ASP technology. The database was created using Microsoft Access 97 and the web interface was built using HTML, VBScript and JavaScript. This system will be fully functioning at the Florida Department of Education (FLDOE) site and will enable remote access to allow users to monitor hundreds of contracts. Features include advanced scheduling, warning and accounting capability. The Internet-enabled feature allows nationwide users, including FLDOE project managers, FLDOE technical advisors and independent contractors, to follow the status of contracts, contractors, and payments very closely.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE **PAGE**

CHAPTER 1 INTRODUCTION

This thesis describes the development of the Projects and Grants Management (PGM) system. This system originally called State Administrative Expense Tracking System (SAETS) is an Internet-enabled database application which was developed for the Florida Department of Education (FLDOE). It is a PC-based client-server system used to effectively manage and trace contracts, contractors and payment schedules. The primary purpose of this system is to provide remote access for users from anywhere in the USA. Thus, users are able to implement this system from other computers and are no longer confined to one typical PC and software. Before the development of this Internet-enabled system, a desktop system had been used for the purpose of control and tracking for contracts and contractors.

1.1 Brief Project History

Before this system was developed, FLDOE staff members managed all the contracts, contractors and grants manually. Software, such as Microsoft Access and Lotus Organizer, was used solely to store data for which human resources were used to control and track the other necessary follow-up work. Secretaries played the important rolls in the whole filing process. Filing papers was one of the major tasks performed on a daily basis. The same information was often converted into different formats. This sometimes resulted in misleading information which caused extra workloads and time. Furthermore, there were problems with getting reports from the Comptroller in a timely fashion. Contractors also experienced difficulty with the old system. It was very difficult for them

to keep track of payment activities and a great deal of time was spent waiting for reports to come out. A software package designed in Microsoft Access 97 could solve these problems but would still have disadvantages such as a lack of client/server interaction and a limitation of where it can be implemented.

It is due to these inefficient processes and disadvantages that this project was born.

1.2 Why The Internet?

The Internet is the place where affordable computing and navigable, distributed information go to work on expanding knowledge. It is growing substantially each day and it is estimated that there are over 75 million users of this media. The number of people who get information on the Internet keeps increasing. The Internet is vastly and quickly becoming a way of doing everyday business and is expanding business hours to include more conventional "off hours". We now have the ability to communicate more extensively, author interactive Web pages, add audio components to our sites and so much more.

Statistics show that the Internet is growing by 10-15% per month. The Internet nowadays has more than 2 million nodes (i.e., computers which are always connected to each other) and is expected to increase to more than 100 million nodes by the year 2000. By the end of last year 1997, 82 million PCs were connected worldwide to the Internet. This is a 71 percent increase from 1996. Dataquest [1] predicts that by 2001, the Internet will be such

a major part of business operations worldwide that the number of "wired" computers will rise to 268 million.

Due to the increasing number of Internet users, the Internet software and services market produce revenues of US$12.2 billion in 1997, a 60 percent growth from 1996, when revenues came in at $7.5 billion. Dataquest estimates that by 2001, the Internet software and services market will reach $32.2 billion in revenue.

With those persuadable statistics and overwhelming growing popularity, there is little doubt that the Internet is the best communication medium for client/server interaction. This project indeed serves as one of the pioneering Internet-based projects that will eventually be the mainstream choice for deploying projects with boundless access, control and tracing capabilities for all data processing.

CHAPTER 2 SYSTEM OVERVIEW

This chapter gives a general overview of this system. This will enable us to have a better understanding of how this system works and its features. Details of design concepts will be presented in chapter 3.

2.1 System Operation

The major functions of this system include connecting user browsers to databases at the web server side, providing good user interfaces and granting various privileges for different types of users. There are two types of users a super user and an end user or contractor. All data information including a user's login accounts, contracts, contractors and payment details are stored in the database on the server side. A database is an example of a data store. A data store could be any store of information. The database tool used here is Microsoft Access 97.

On the browser side, all users have to establish their login accounts for navigating in the system. The identification of users is required for login. After successfully logging in, users will be able to place requests to the system. Once a user makes a request, the web server processes the request using Active Server Pages and sends the resulting information back in plain HTML format to the browser that requested it. Figure 2-1 illustrates a schematic overview of how this processing is done.

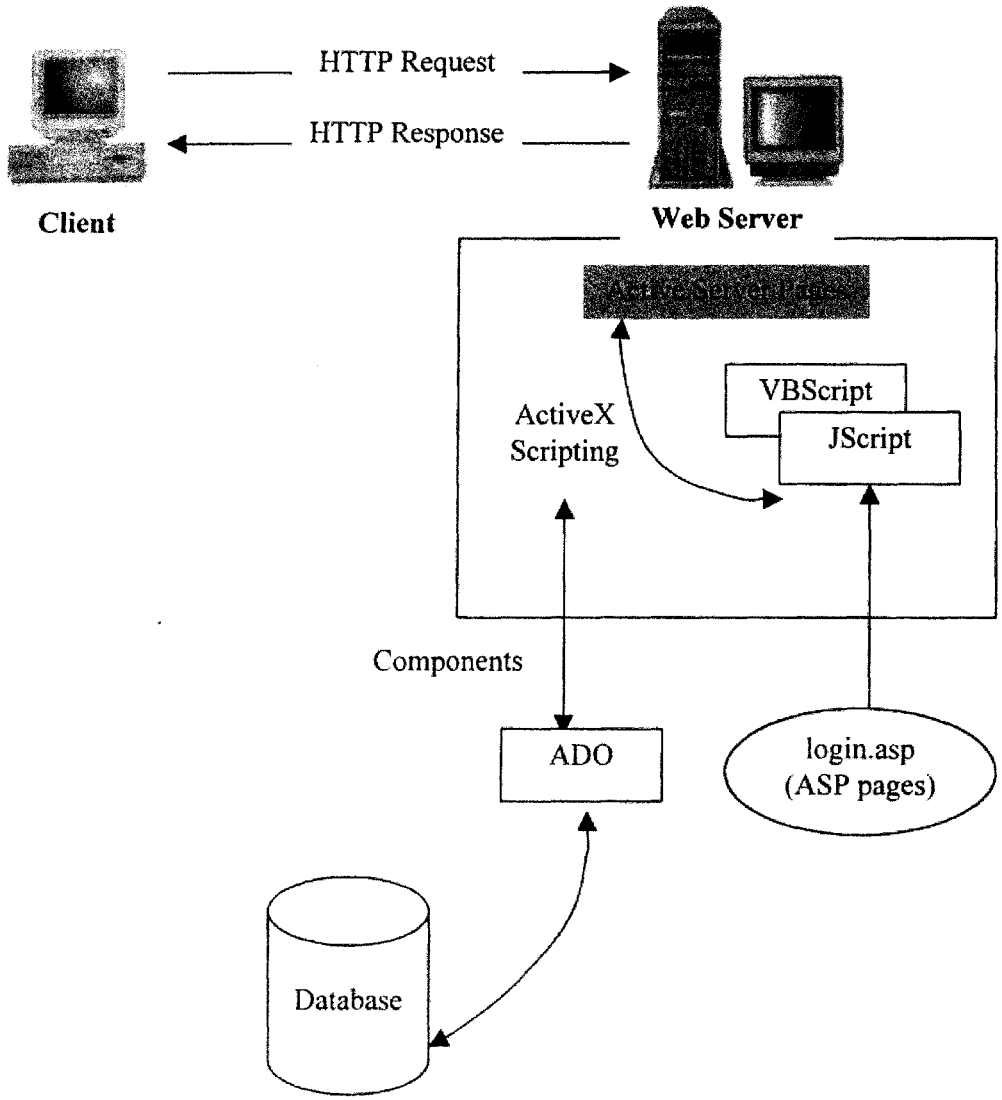

Figure 2-1 Processing of an ASP Request

2.2 System Technical Terms

The main technique used to develop this system is called ASP (Active Server Pages). ASP was officially announced to the world by Microsoft on July 16, 1996 and was codenamed "Denali" [2]. It is the latest sever-based technology designed to create dynamic and interactive HTML pages for WWW sites. ASP is revolutionizing the way Web applications are developed almost the same way Windows NT revolutionized client/server computing. ASP is designed to be used together with HTML to create dynamic pages. In fact, ASP actually creates HTML code. A Web page that uses ASP is likely to consist of a mixture of three types of syntax. Some of the page will be constructed from simple text, part will be HTML, and part will be ASP code. ASP is actually an extension to the web server that allows server-side scripting. At the same time it also provides a compendium of objects and components which manage interaction between the web server and the browser. Those objects can be manipulated by scripting languages. Figure 2-2 interprets that ASP neatly divides up into and uses different types of objects, each of which manages its own part of the interaction between client and server. ASP is a new technique for web developers to make a web site interactive.

Prior to ASP, the development of a typical interactive Web application meant compiling an executable application using a traditional application development environment such as Visual C++. After the application was compiled, it was copied to a CGI directory of the web server. Even the slightest change to the application meant recompiling the entire application and replacing the previous version of the executable file. This process is

unnecessarily resource intensive in a production environment. ASP solves this problem by providing a more direct and easier way to create web applications. [3]

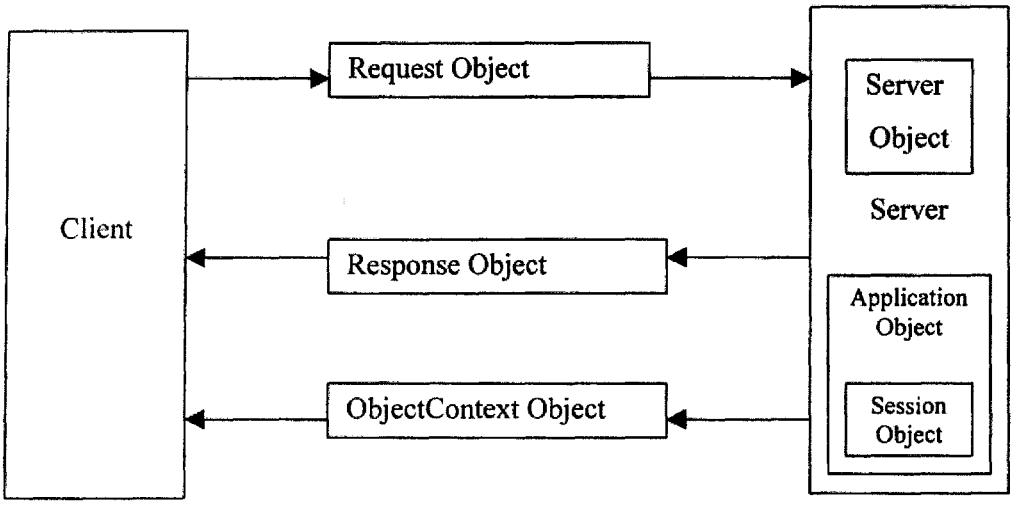

Figure 2-2 Client/Server Interaction in ASP

2.3 System Features

This system was designed and developed for the FLDOE. There are some features included in the system.

- Clear identification for users:

Once a user logons to the system successfully, the system shows clearly a column banner indicating what type of account the user has. Different types of users will be directed to different pages and given different privileges.

- Security control:

  This system is used to track Food and Nutrition Management Projects and Grants for the Florida Department of Education. Only specific users (i.e., FLDOE super users or valid contractors) have access to information on the Web page. Other than login page, all pages on the Web server can only be accessed by using a valid login account. And every page is protected from unauthorized users.

- Friendly and intelligent interface:

  This system provides easily, understandable, and user-friendly on-line instructions. With very little experiences of using browser, users can implement this system quickly and easily. Use of the system is particularly facilitated by providing feedback for the actions a user has taken, instructions concerning a user's alternative actions, message showing mistakes that have been made and available hot links to other useful and related pages.

- Online functions of add, delete, search and update:

  This was the primary motivation for developing this system. FLDOE super users can implement this system from any PC at anytime. The data can be updated and posted

simultaneously. Specialized software or a specific physical location is not required as long as the user has a browser connect to the Internet.

- Login account management:

  FLDOE super users have absolute control over all login accounts including contractors and other super users. The information such as the login username and password, the date an account was created and the date last time login a user logged in can be seen and monitored. If a user forgets his/her password, he/she can simply send a request to the system administrator and the login account will be reset with an assigned username and password and the user will be able to log onto the system and change the password.

- Remote access to all contractors:

  Just as with super users, contractors will have pre-setup login account information available so that they can log on from any Internet-enabled computer at any time.

CHAPTER 3 SYSTEM DESIGN

In this chapter, the core concepts involved in designing this system will be discussed. Topics include system architecture, system security, setting up the web server and system functions. The design details involved in using Microsoft Access are not the focus here since project's field of study is on the Internet client-server by implementing ASP. Other subjects such as system implementation will be introduced in the next chapter.

3.1 System Architecture and User Interfaces

Prominent and colorful images and icons contribute to making the user interfaces easy to use and visually appealing. All graphic designs were developed by modifying and updating the client's requests. Even at the time this paper is being delivered, the progress is continuing. The purpose of doing so is to ensure that the interfaces can be understandable from the client's points of view and to reduce the probability of a user making mistakes. The system architecture will be looked at first. The following illustration shows the architecture of the system:

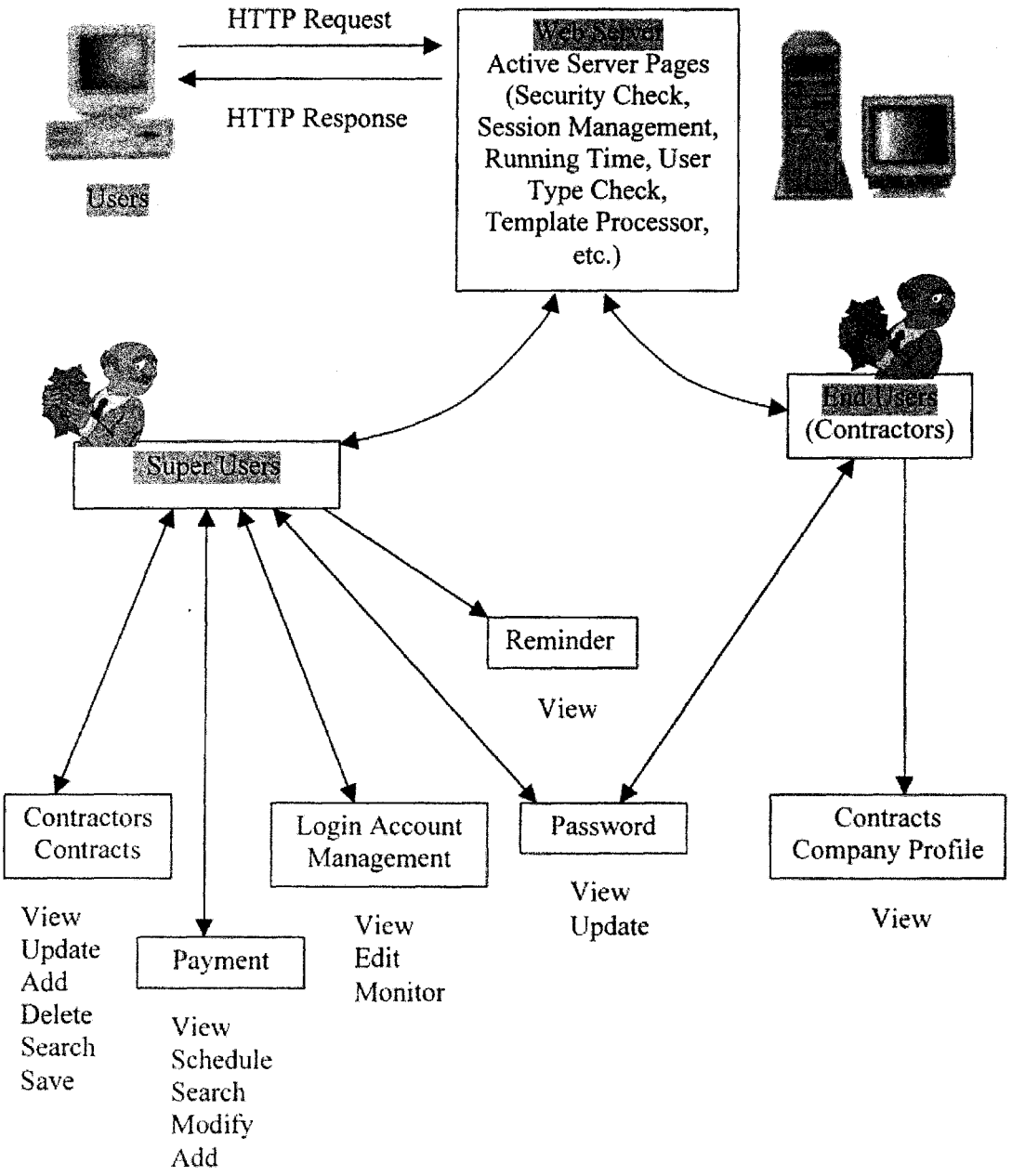

Figure 3-1 System Architecture

This system is designed to help FLDOE control and track all the projects and grants of its contractors. It also provides accounting capabilities to help FLDOE monitor and schedule payments. The system will automatically detect and determine the type of user and grant the appropriate privileges when the user logs in. Of course, a task of security check is performed here. Only valid and authorized users can successfully log onto this system.

Failed attempts to log in will result in the display of a red blinking error message on the bottom of the page. The type of error message returned reflect the type of mistakes made. Some examples are: "Please enter all information on the login fields!", "Right user name, Wrong password! Please try again!" and "Invalid account, Cannot login to this site!".

Another function executed here is the recording of the login date of each user in that user's account. This function will help the FLDOE project managers have better control of their contractor's login activities.

Figure 3-2 shows the login interface. The user type will be determined by processing some database lookup functions. Please see appendices.

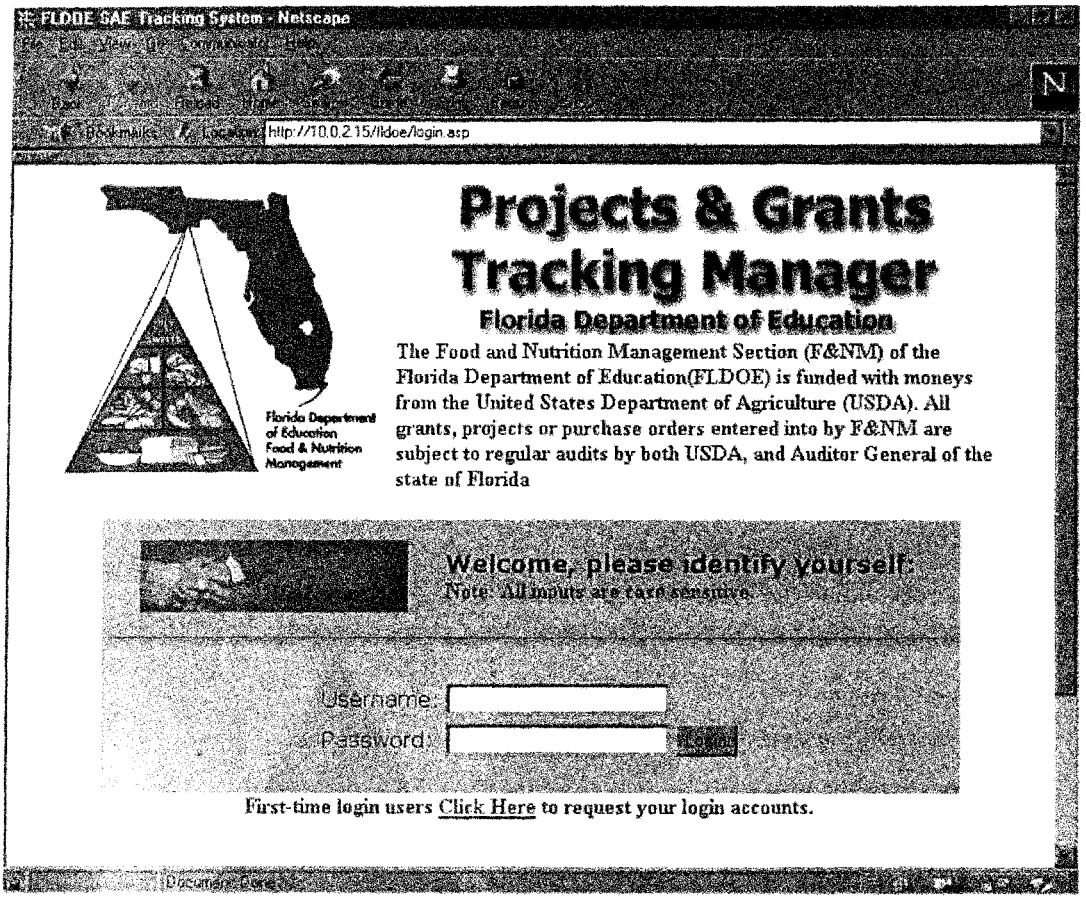

Figure 3-2 System Login Interface

3.1.1 Super Users

Super Users are FLDOE staff such as project managers and project technical advisors. When a super user log in, the system will bring the super user to his/her main page (Figure 3-3) where the function image icons are shown. These include contracts, contractors, payment information, reminders, login account management and update password icons.

A column banner on the left side of this page with big letters "SUPER USER" indicates the type of user currently logged in. This page is called the super user main page because of the specific function buttons, icons and hot links that appear on this page. It can also be considered a transfer center from which a super user can easily navigation through out the whole system. Of course access to this page is restricted to only authorized super users. The major functions will be discussed in the next sections that follow.

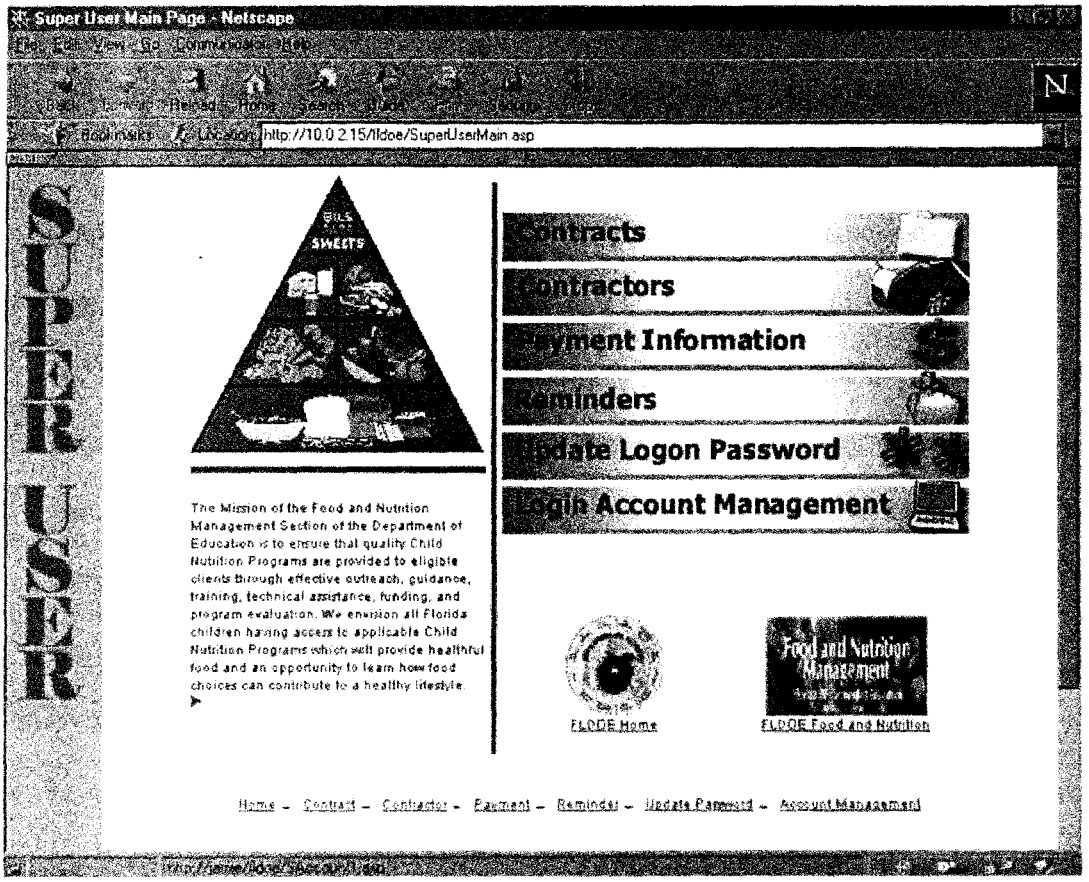

Figure 3-3 Super User Main Page

3.1.1.1 Contract and Contractor Controls

Contract and contractor controls are the major functions available to a super user. The super user can perform functions of view, update, add, delete and search on these pages. As can be seen in the Figure 3-4 and Figure 3-5, contract control interface and contractor control interface function pages are similar. The heading banner on the top of the screen is composed of three sections: heading and function buttons, instructions and system status.

- Heading and function buttons: The text heading indicates which interface the super user is currently using and, next to it, several function buttons including buttons of update, add, delete, search and list view. Those buttons are the essential parts of this page. They are going to perform the functions as they show on their face value such as, update, add, delete, search and list view. The list view button is particularly helpful as it provides another perspective that can be used to view the various of contracts and contractors. Furthermore, there is a batch link between the list view and form view where the page we came from. The batch link is a link to a typical record. We may consider it a short cut to the record the user wants.

- Instructions: This is a brief but clear instruction line for the users. For example, one instruction is "To update this record, make changes in any fields below and then click the Update button." This one line instruction will tutor the user how to perform the next possible action.

- System status: A line of text denotes the status of current system function. If the super user sees "Ready for search.", then the search button only needs to be clicked to perform the search function. Once the super user clicks the button the text "Ready for search criteria." will then replace "Ready for search." on this line and another instruction, like "Please enter the keywords in any fields below to perform the search.", will appear on the instruction line mentioned above. For example, if the search criteria is a contract whose number is "131-30450-60151", the instruction "Current Search: [Contract Number]='131-30450-60151' " will appear and the record will be found. Additionally, when the super user is in "Ready for search" status, another function button called "How to Search" will appear on the first line of the heading section. By clicking this button, a new pop-up window will appear containing a HTML-format instruction page regarding how to perform a search.

In the middle region of this page, the data or record were requested by the super user will be retrieved from the database and be posted. This is a display section.

At the bottom of the screen the record navigation buttons are displayed. These navigation buttons such as "The First", "Previous", "Next", "The Last", and "Post Data" will help the super user to move back and forth between records inside the database. The "Post Data" button will perform a re-query action for a modification action that has just been made by the super user. New data, if there is any, will not be posted onto the screen until this button has been clicked.

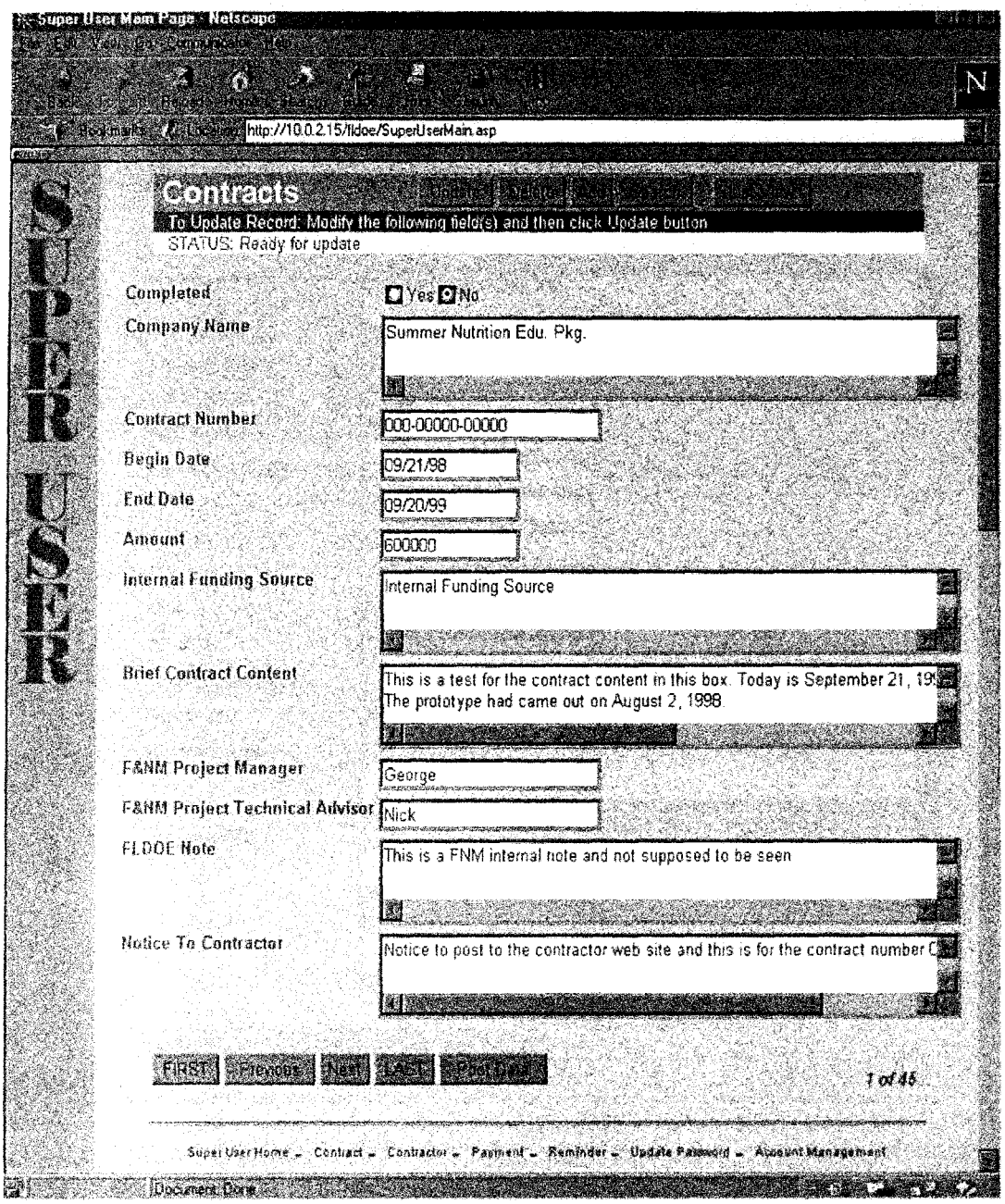

Figure 3-4 Super User Contracts Control Interface

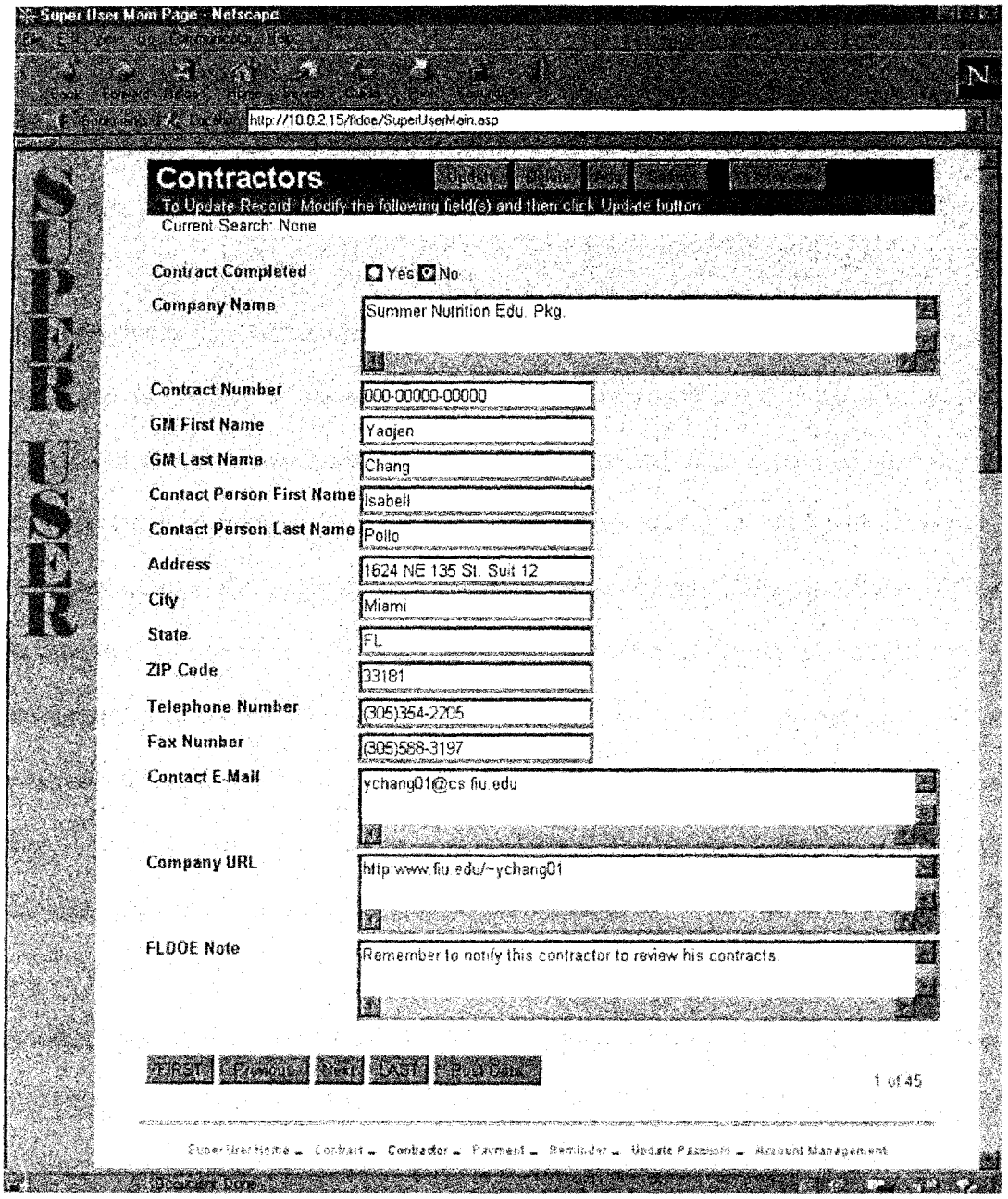

Figure 3-5 Super User Contractors Control Interface

3.1.1.2 Payment Scheduling

Accounting capabilities are included in this payment handling function. Function details will be discussed later. The payment scheduling interface will be introduced in this subsection.

The interface of this page is still similar to the contracts and contractors pages. Figure 3-6 illustrates the similarity between them. Same construction of this page is for the convenient usage of super users. All payment schedules can be viewed, monitored, scheduled and updated on this page. As with other super user pages, this page has restricted access. The heading banner contains three buttons. The first button "First Scheduling" is used to perform first time scheduling for a contractor who already has a valid contract or contracts with the FLDOE but is not yet receiving any payments. When it is clicked, a drop down list in a new page provides all contracts which are ready for the first time scheduling. This design was developed due to the desire of FLDOE to distinguish the initial payments from other old payments.

Another major function here is to edit and add payment schedules for a contractor. Before allowing a user to edit or add payment schedule for a certain contractor, the system will retrieve the contractor's payment history (or past payment activities). This is done to provide the super user with all the necessary information before taking any further scheduling actions. A payment history is a list of information of past payment activities such as the total amount being paid, current balance, amount of each payment, date of each payment, payment scheduler and who approved the payment. Based on the

information of payment history, the super user will be able to perform more accurate scheduling.

There is also a search function on this page. The search criterion of this function based upon the payment due date. For purpose of convenience, the super user can just select "On", "Prior to", and "After" from a drop down list and enter a date to perform the search. For example, if the user selects "Prior to" on the list and then enters the date "09/20/98" or "9/20/1998", then the system will return all the payments and associated information due prior to September 20, 1998. The result of this function can provide valuable information that will help FLDOE project managers and technical advisors control, track and monitor all the payment activities.

This function is slightly similar to the reminder function which will be introduced in the follow subsection. The difference is to inquiry the payment due date in different query requests. This function is performed on the base of the "date" of payment due and reminder function is executed on the base of "number of date" before due date.

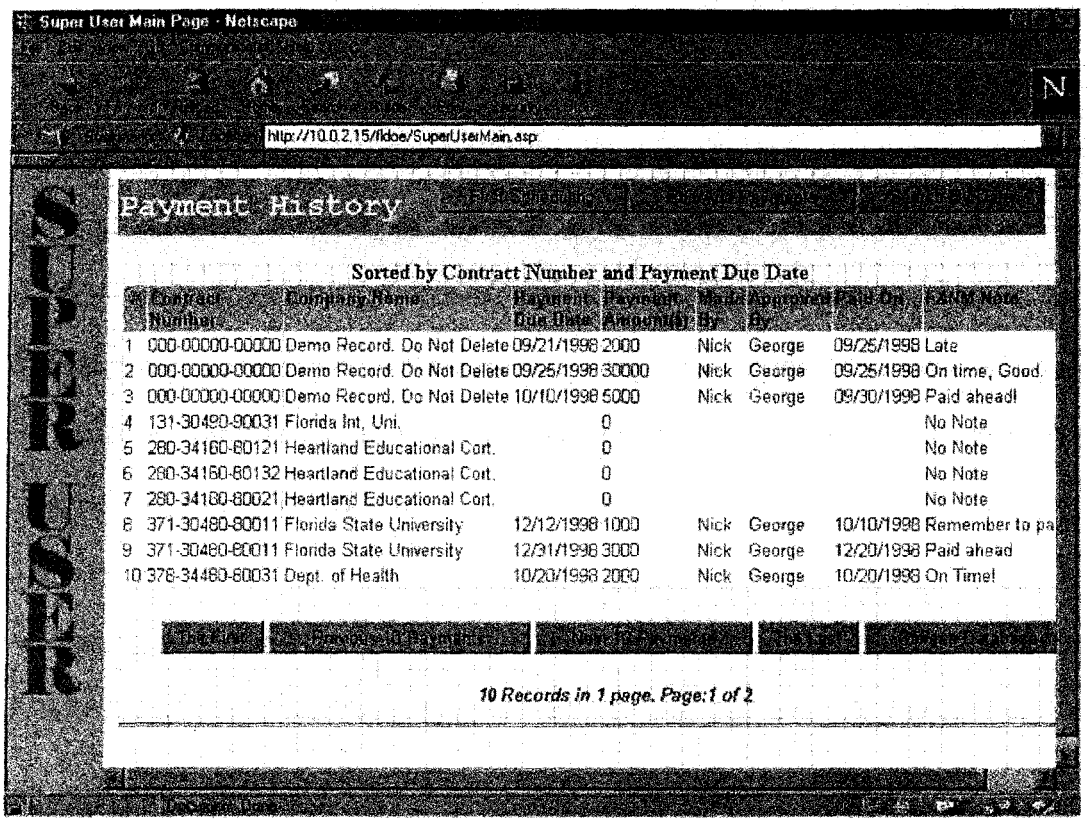

Figure 3-6 Super User Payment Scheduling Interface

3.1.1.3 Others

Other available and thoughtful designs for the super users are "Reminders", "Update Password", and "Login Account Management".

- Reminders:

A reminder is a time sensitive notice or memo. It allows the super user to view all the payments due within certain number of days entered by the super user. It is a slightly

different from the search function on the payment information page mentioned above. The super user doesn't have to know the exact date a payment is due. For example, if a super user wishes to know if there are any payments due within next 3 days, 3 is entered into the input box and the "Go" button clicked. The system will return all payments that are due within the next 3 days. Indeed, within the Microsoft Access environment, the desktop version of the system will automatically execute this reminder function with pop up reminders when a super user just logs onto the system. The default setting for the number of the threshold days is 10.

- Update Passwords:

In the original design, users were to be allowed to create their own login account names and passwords online the first time they logged in, if they could provide all accurate information required about himself/herself. However, for the security reasons, this was not implemented currently, all login accounts for both super users and contractors are pre-setup. The system administrator creates all login accounts including login usernames and passwords in advance. Of course, users themselves can update their passwords anytime they log in.

- Login Account Management:

Currently login account management allows the project managers or technical advisors to retrieve all users' login account information and login activities and to edit accounts. The super user will be able to know the last login date of a user and can obtain all the login names and passwords.

For the convenience of usage, the interface of this function is designed using a drop down list (See Figure 3-7). All users whose valid accounts had been pre-setup will be displayed on the drop down list in an alphabetic order. All super users' names, for examples, will appear in a format of "Last Name, First Name" and for all end users, the list information will show in a format of "Contract Number, Company Name". The system administrator just has to simply select one account from the list and click the "Go" button. The system will retrieve the data and display the account information of the administrator's selection on the screen. The account information includes user type, super user's real name, contract number and company name of a contractor, login username, login password, the date of account created and the date of last time login activity.

Once a super user obtained the account information, if necessary, he/she can edit the account by clicking on the button of "Edit Password of This Account". A pop-up window interface will allow the user to update the password.

Not only does this function provide a way of retrieving account information, but also can it prevent users from forgetting the login passwords.

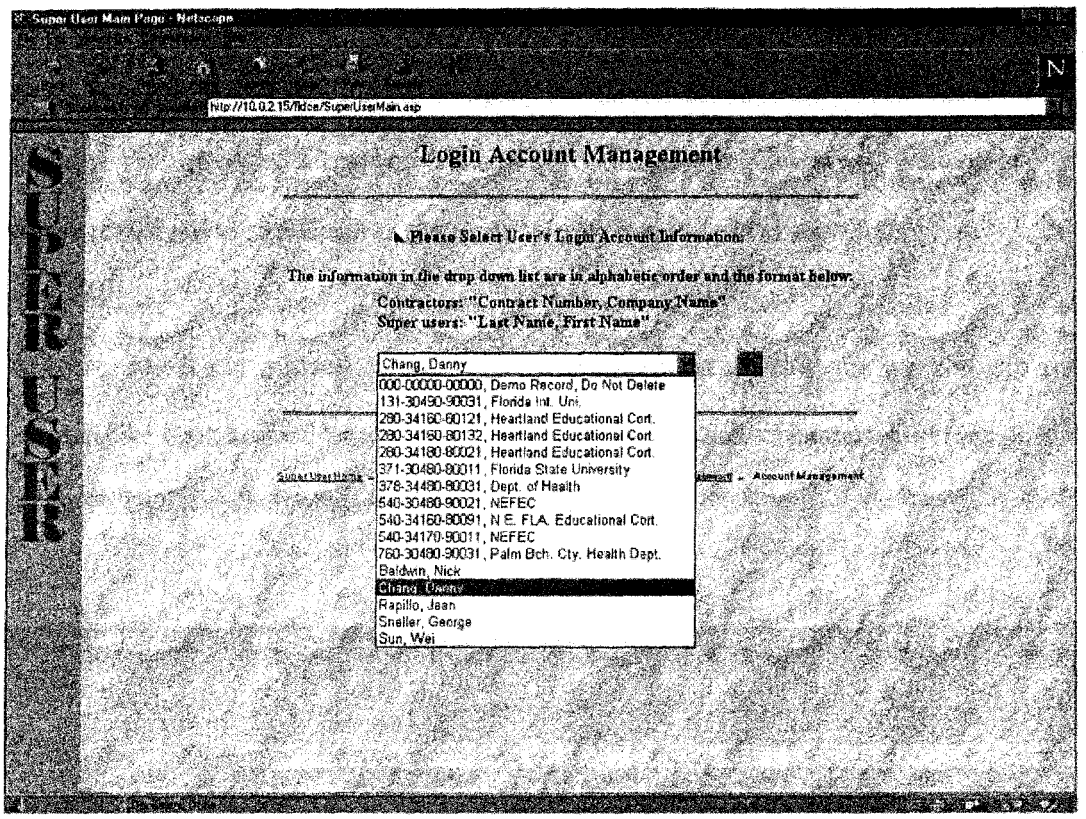

Figure 3-7 Login Account Management Interface

3.1.2 End Users

The end-users of this system are the contractors of FLDOE. A contractor is a party who has at least one contract with FLDOE. For security reasons, all end-users' login accounts will be created by the system administrator in advance. When an end-user logs onto the system, the system will know that the current user is an end user and will bring the user to the "End User Main Page". See Figure 3-8. As with super users, the column banner on the left-hand side indicates the current type of user. Through this page, the end user can

only view regarding his/her own record. For example, information regarding that user's contract or contracts with FLDOE, company profile and the payment information binding with the typical contract can be obtained. Of course, as was mentioned previously, the end user still has the privilege to change his/her own password at any time.

Another interaction between FLDOE super users and contractors is that through the Internet, the end user will be able to see any information, announcement or notice posted by the FLDOE immediately and simultaneously.

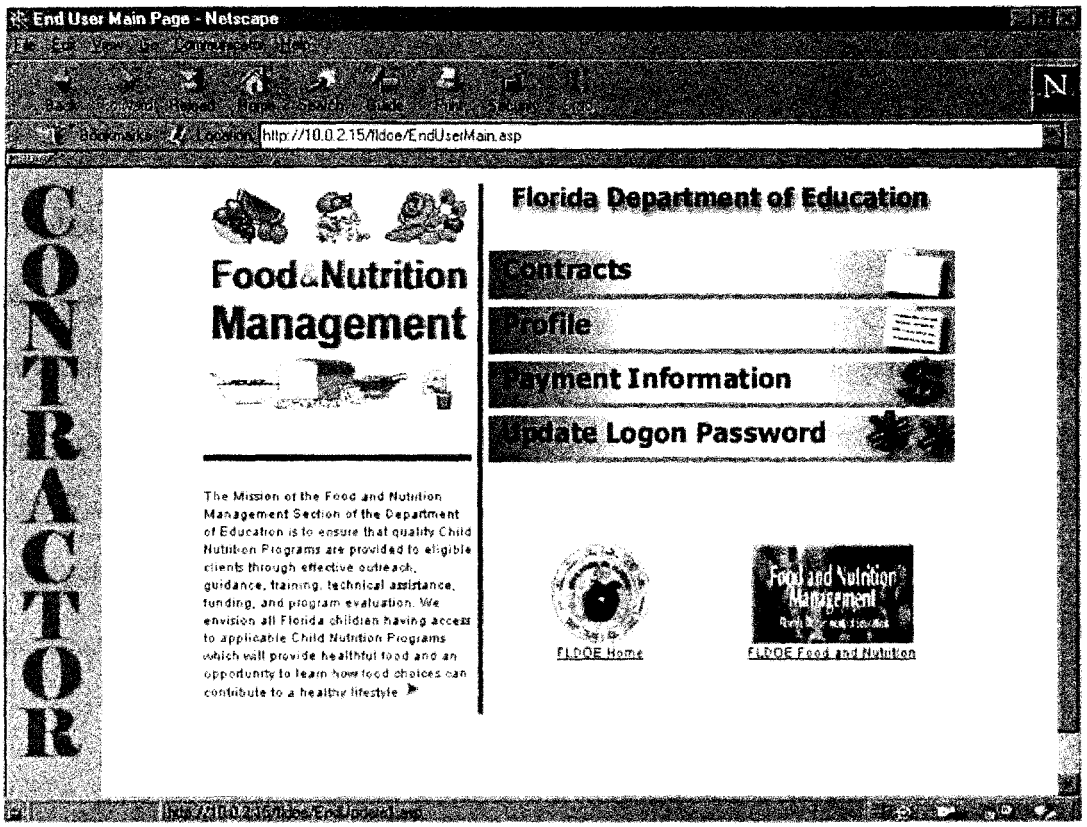

Figure 3-8 End User Main Page

3.2 System Security

Security has always been one of the big issues on the Internet. In fact, thousands of papers have been published studying this topic. Of course, we also need to consider security in the design of this system. This is a system used to control, track and monitor all contracts and contractors of FLDOE. It is not a system that could or should be public to the world. Building a membership-based community is necessary and important. The system administrator, super users, and end users (or contractors) are the three types of users of this system. In addition to a valid and pre-setup user account mentioned previously, there are other kinds of security protections applied in this system.

One protection used in this system is called "time out". If any page is left untouched, unattended or unused for over 30 seconds, then this page will expire. If the user then try to reload or refresh that page, the system will automatically bring the user back to the login page and the user will have to re-login. In other words, the system does not know whether or not the current user is the same one who successfully logged on 30 seconds ago. Thus, the login process must be carried out again. Time-out has to be set in the global.asa file and will be applied to every page in the system. As the global.asa file is an optional file that relates directly to the Application and Session objects, it is not a concern in this project.

There are two other major security protections implemented in this system. One protection used is the implementation of IIS (Internet Information Server) and another is the implementation of ASP.

3.2.1 User Access

The Internet Information Server (IIS) enables access to resources only after the user's access privileges have been verified. Figure 3-9 shows the security checks performed before a user is permitted to access a requested page.

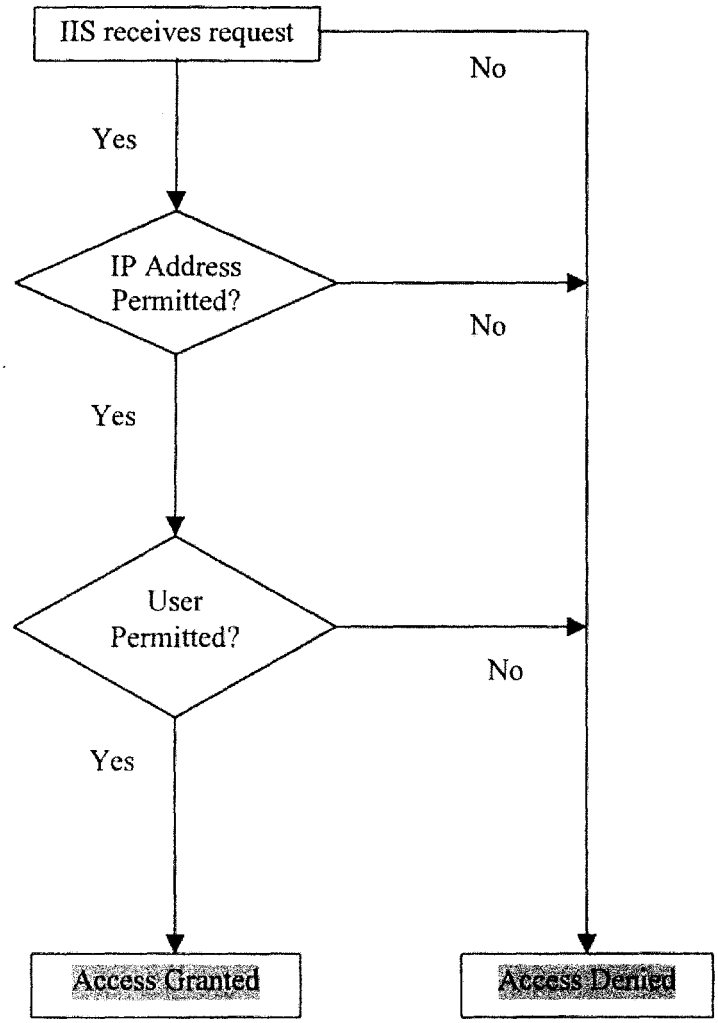

Figure 3-9 System Security Process

The IIS web service can be set to enable only users on computers in a given range of IP addresses to access files on this server. This provides additional security on an Intranet web server by disabling all IP addresses other than the local ones. It also is possible to disallow specific IP address for the web server, for example, addresses you know hacking attempts were made from.

3.2.2 Valid Users-Only Areas on The System Server

The next level of access control is to verify the user's login information according to the user data stored in the database. The login user table stores information used to define various types of users.

Indeed, there are three approaches that can be used to create members-only areas:

- Use NTFS file security to enable access to the members' area for specific user accounts or groups. This is a good approach for Intranets. However, the disadvantage is that an NT administrator has to create an NT user account for every member.

- Create an ISAPI (Internet Server API) filter that implements a custom authentication scheme. Multiple members are mapped to a single NT user account which is granted access to a specific members' area on the server.

- Create a custom authentication based on Active Server Pages. Every user is validated before access to any resources in the valid users-only areas on the system server.

The third approach is the one used to provide security for this system. Every page of the system has to check whether the user has already been validated. Figure 3-10 shows how the validation process works.

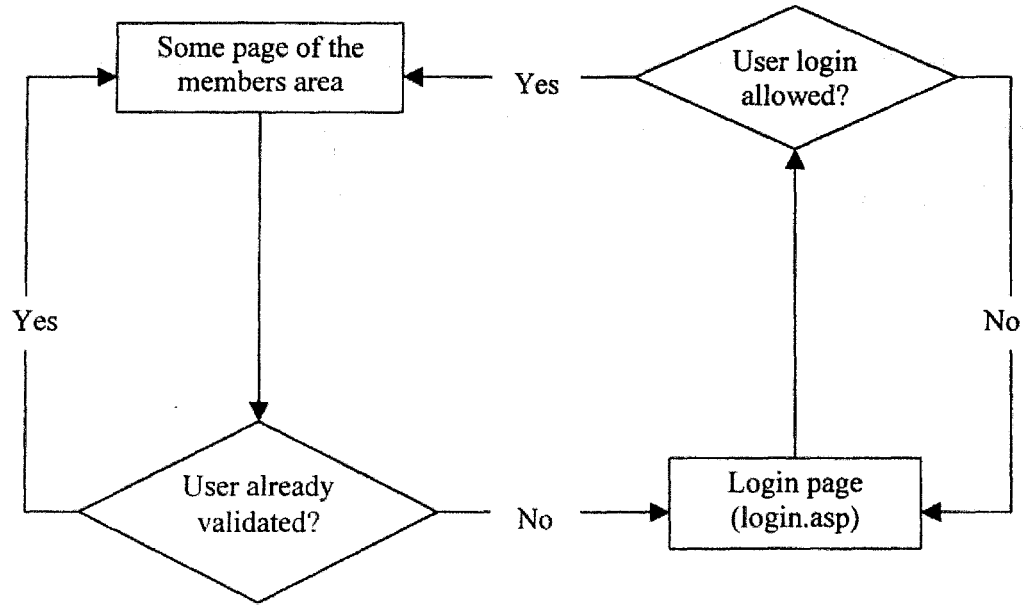

Figure 3-10 User Validation Process

The user validation process is based on the Microsoft Access server table and a stored procedure to validate the validation of a user trying to log on. It is advantageous to use a database because of the easy maintenance of user information. The table that contains the user information could be very simple and easily updated in the future. The table which

stores all users' login account information will be introduced in the next section. First the concept of database connectivity will be discussed.

3.3 System Database

Databases play a key role when we want to provide personalized and live content. Databases can be used to store any kind of information, from quotes of the day to full-featured online stores. The database behind this system was constructed by using Microsoft Access 97. Microsoft provides access to the database with the ActiveX Data Objects (ADO), which builds upon OLE-DB. OLE-DB will be introduced in chapter 4.

A well-known technique that is often used to connect to a database is ODBC (Open DataBase Connectivity). ODBC is a single, well-defined interface for uniformly accessing different database management systems regardless of the provider-specific interface. The ODBC programming interface (API) defines a database-independent programming model that provides a single API interface. It is designed to allow a common set of routines to be used to access databases, although it was primarily aimed at relational databases. This allows a programmer to connect to a database using ODBC and manipulate the data without worrying exactly where the data was stored, or what particular database was storing it. (See Figure 3-11)

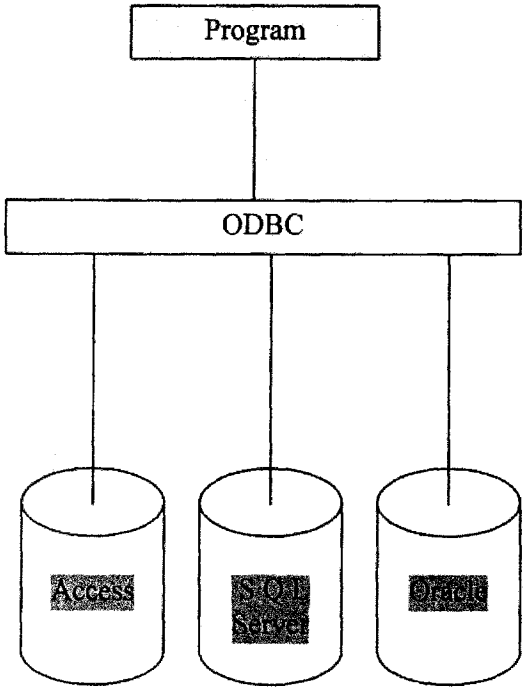

Figure 3-11 ODBC

For example, Figure 3-12 illustrates a table which stores user login information. It stores the user type, first name and last name, company name, login user name, login password, date the login account created and the last login date. For security reasons, the initial login account set up has to be done by the system administrator in an internal-use desktop system but the values in the password field are on-line updateable. The user type field is used to determine the type of login user and where the program should direct the user as well as what the appropriate privileges are.

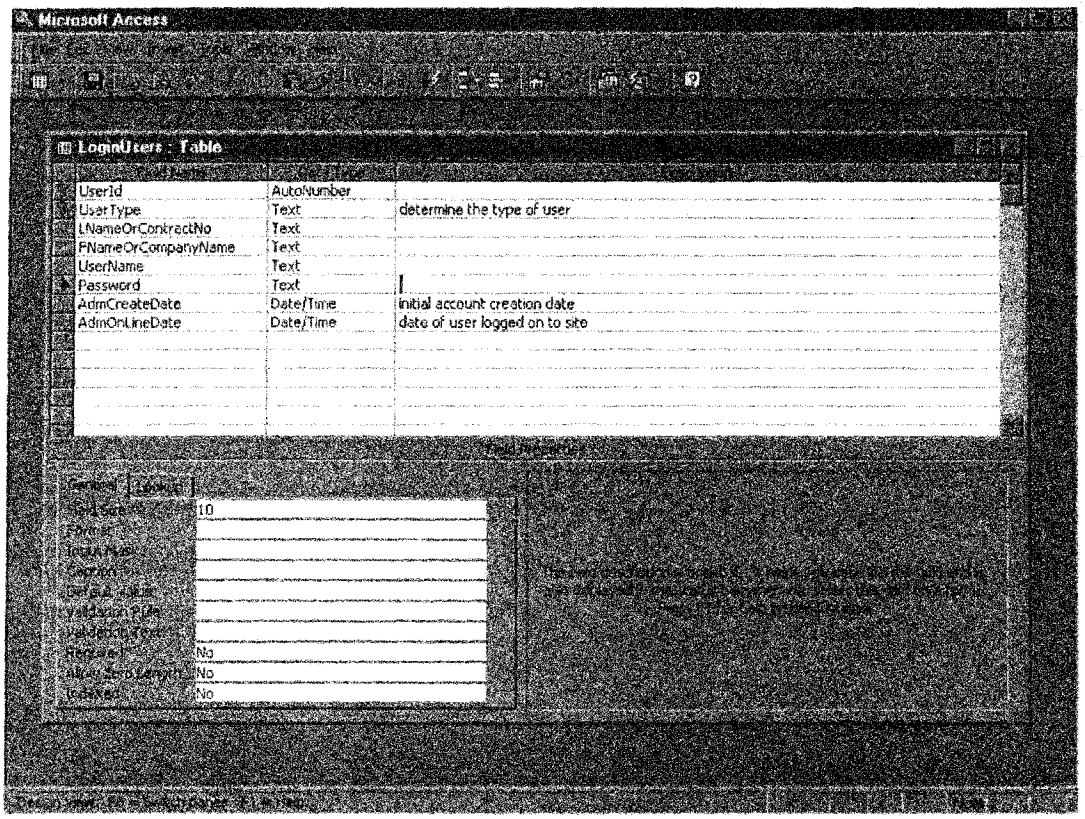

Figure 3-12 Login User Table Design

3.4 System Major Functionality

The user interfaces and associated function operations were discussed in section 3.1 This section discusses the functionality of the system. The functionality can be broken down into several areas:

- Different privileges for different types of users: As was previously mentioned, there are two types of users with legal accounts allowed to log onto this system. One type of user is the Super User. Super users are the FLDOE system administrators, project managers and technical advisors. The other type of user is the Contractor or End User

who has at least one valid contract with the FLDOE. The system will determine what privileges to give the current user by bringing the user to the correct location. Contractors will be able to update their login account passwords but will only be in the system's View mode --- i.e., they can only view their own information.

- Contracts and contractors control: The system will be in Edit mode when a super user logs in. Therefore, the super users will have full control (such as add, delete, update and save) of all records in the database.
- Advanced payment scheduling: The system will allow super users to schedule payment dates in advance and monitor the delivery status for each payment. Accounting capabilities were applied to allow calculation of the total amount paid and the remaining balance of the contracts. Of course all payment information such as who made the payment and who approved the payment can also be kept track of and edited in future follow-up work.
- Reminders of payment schedules: A reminder function will execute if a payment is due within a certain number of days which depends upon a super user's inquiry criteria.
- On-line login account management: Super users will have control over all login accounts. All login usernames and passwords can be retrieved should a super user request it. This function will handle the situation in which users forget their passwords.

CHAPTER 4 SYSTEM IMPLEMENTATION

We have discussed the technologies like ASP and ODBC in the previous chapters. Now, in this chapter, we are moving to the issue of system implementation. Various technologies in different fields had been implemented to develop this system. We will take a look on each of them.

4.1 Platform

4.1.1 Software and Hardware

Since ASP is the advanced technique used to develop the web application, its software and hardware requirements are critical. The minimum software and hardware requirements for developing ASP applications are:

- A Pentium-based computer
- 32 MB of RAM
- 100 MB of free hard drive space
- Windows NT Server 4.0 with TCP/IP networking support properly installed and configured.
- Internet Information Server (IIS) 3.0 or better is required for Windows NT Server; Microsoft Personal Web Server (PWS) is required for Windows 95; Microsoft Peer Web Services is required for Windows NT Workstation
- A database that supports ODBC (such as Microsoft Access or Microsoft SQL Server)

For a user (client), the minimal and recommended system configuration is:

- Windows 3.1 (Windows 95 or Windows NT is highly recommended)
- 486 CPU (a Pentium-based computer is highly recommended)
- 8 MB RAM (16 MB RAM or better is highly recommended)
- 10 MB of free hard drive space (50 MB is highly recommended)
- 28.8 kbps data modem (33.3 kbps or better is highly recommended)
- VGA adapter and monitor

4.1.2 Operating System

The best choice of an operating system is Windows NT. Windows NT in conjunction with IIS and ASP provides a very powerful platform for developing and deploying web applications. Further, there are some advantages we can benefit from by using Windows NT: [9]

- Better performance: Windows NT Server has been optimized to provide the best performance for network-intensive server applications. On the other hand, Windows 95 and Windows NT Workstation have been optimized to provide the best performance for productivity applications. Therefore, Windows NT Server yields better performance when hosting ASP applications.
- More secure: Because IIS uses NTFS security when running under Windows NT Server 4.0, Windows NT Server is a more secure platform to host ASP applications. Windows 95 cannot implement security using NTFS security permissions because it uses PWS (Personal Web Server), a watered-down version of Internet Information Server.

- Easy integration with enterprise-quality applications: Enterprise-quality applications, such as applications in the Microsoft BackOffice Suite, require Windows NT Server. Therefore, choosing Windows NT Server to develop ASP applications will make it easier for a developer to integrate ASP applications with various components of BackOffice to develop sophisticated Web applications.

4.1.3 Web Server

A web site is composed of three components: the hardware (the computer), the software, and a network connection. Each of these three plays a symmetrical role in establishing a reliable web server. Because the technique used to develop the system here is ASP, certainly the relationship between IIS, Windows NT Server, ASP and the database need to be discussed.

Windows NT is not actually the only choice. There are three kinds of servers that can be used to develop an ASP application. Internet Information Server (IIS) is required if we are using Windows NT Server, Microsoft Personal Web Server (PWS) is required if we are using Windows 95, and Microsoft Peer Web Services is required if we are using Windows NT Workstation. Why? Because the ASP component is actually an ISAPI (Internet Server API) application, the web developer should be able to develop ASP applications with any ISAPI-compliant Web server by simply downloading the ASP component of IIS and installing it. [6] Although Windows NT Workstation as well as Windows 95 can be used to develop ASP applications, the Windows NT is still considered the preferred choice. The details will be discussed later.

4.1.4 Programming Language

The choice of programming language running at the server side is Visual Basic. Visual Basic is the most compatible development system to use in conjunction with Microsoft Access that is the database driver used for the system. In industry, Visual Basic is considered a development tool that is easy to use and has built in local area network (LAN) and Internet deployment models. Rich component encapsulation and reuse are an industry-leading array of third-party components and the ability to use existing Visual Basic code and technology. Visual Basic offers advanced features such as optimized native code compilation and enhanced database access. The Microsoft Transaction Server provides state-of-the art application performance and scalability for the additional client/server, three-tier distributed and Internet application architectures. The new released version of Visual Basic has the ability to merge client/server and Internet technologies. Developers are no longer required to choose between performance and productivity.

4.2 ASP Implementation

We briefly introduced the client/server interaction in ASP in an earlier in chapter 2 section (See Figure 2-2). But why choose ASP? How does it work with HTML? What is the difference between it and dynamic HTML? Those topics are discussed in the following sections.

4.2.1 Why Use ASP?

The primary difference between ASP and the other new generation technologies is that ASP must be executed on the Web server, while the pages generated by other technologies are interpreted by the browser (or client). The advantages that ASP enjoys over CGI and Perl are those of simplicity and speed.

At one time, the browser could do everything you needed --- it interpreted HTML pages, displayed graphics in a certain way and handled errors. However, with the passing of time, browsers have had to cope with an ever-increasing list of tasks such as handling scripts and having built in controls. Consequently, browsers have become bigger and slower.

The idea behind ASP is to decrease the demand on browsers by getting the server to do some of the work instead. A large central machine can be used to take some of the load, performing some of these tasks itself instead of relaying them to the browser.

Some significant advantages are:

- minimizes network traffic by limiting the need for the browser and server to talk to each other
- makes for quicker loading time since, in the end, you're only actually downloading a page of HTML
- allows you to run programs in languages that aren't supported by your user's browser
- can provide the client with data that does not reside on the client's machine
- provides improved security measures since you can code things which can never be viewed from the browser
- enables Visual Basic developers to perform functions that previously required CGI or ISAPI programming
- integrate ActiveX server components

4.2.2 ASP and HTML

The Hyper Text Markup Language (HTML) is used for the design layout and exhibition of each page. All the forms and user interfaces are created by using HTML. As we mentioned earlier, ASP is designed to be used together with HTML to create dynamic pages. In fact, ASP actually creates HTML code. A web page that uses ASP is likely to consist of a mixture of three types of syntax. Part of the page will be constructed from simple text, part will be HTML and part will be ASP code. The following table summarizes each of these aspects: [7]

Type	Purpose	Interpreter	Hallmarks
Text	Information to be shown on the page	Viewer's browser on their PC shows the text	Simple ASCII text
HTML tags	Instructions to the browser about how to format text and display images	Viewer's browser on their PC interprets the tags to format the text	Each tag within < > delimiters; usually has open and close tags, such as <TABLE>, </TABLE>
ASP statements	Instructions to the Web server running ASP about how to create portions of the page to be sent out	Web site host's Web server software with ASP extensions performs the instructions of the ASP code	Each ASP section contained within <% %> delimiters; ASP statements have a flavor of Visual Basic, with the appearance of programming code with variables, decision trees, etc.

4.2.3 ASP and Dynamic HTML

ASP and Dynamic HTML can both be thought of as extensions to scripting languages and HTML; however, neither of them are programming languages in their own right. ASP takes the scripting language code and converts it into HTML on the server, before sending it back to the browser.

On the other hand, Dynamic HTML is just like scripting in that the script is interpreted by the browser level that creates a representation of the page in HTML. In fact, the only way in which Dynamic HTML differs from scripting is that it allows access to extra features such as the ability to animate pages and position graphics and text precisely by using (X, Y) type coordinates. [8] We could put it this way: ASP is a server-side technology, while Dynamic HTML is a closely related client-side technology.

4.3 WWW

The best choice of interface between all users (client) and the system (server), doubtless, is the World Wide Web (WWW).

When the WWW [9] was introduced to the world in 1989, it added browsing capability to the Internet. Users are able to access text, images, video and audio in a consistent manner. Moreover, users can retrieve information using a graphical user interface rather then using a dumb text terminal. The WWW operates on a client-server model. Every web page has an address called its Uniform Resource Locator (URL). A URL contains the object name, its address and the protocol used to find the object. A client sends a

service request to a server, and the server searches the web page locally and then sends the result back to the client. Client and server communication is via Hypertext Transfer Protocol (HTTP), which defines how documents are referenced and exchanged.

Because the client software interacts with the server according to a predefined protocol, the client software can be customized for the user's particular computer host. Therefore the server doesn't have to worry about the hardware particularities of the client software. Separate versions of the information do not need to be developed for any particular hardware platform since the customizations necessary are written into the client software

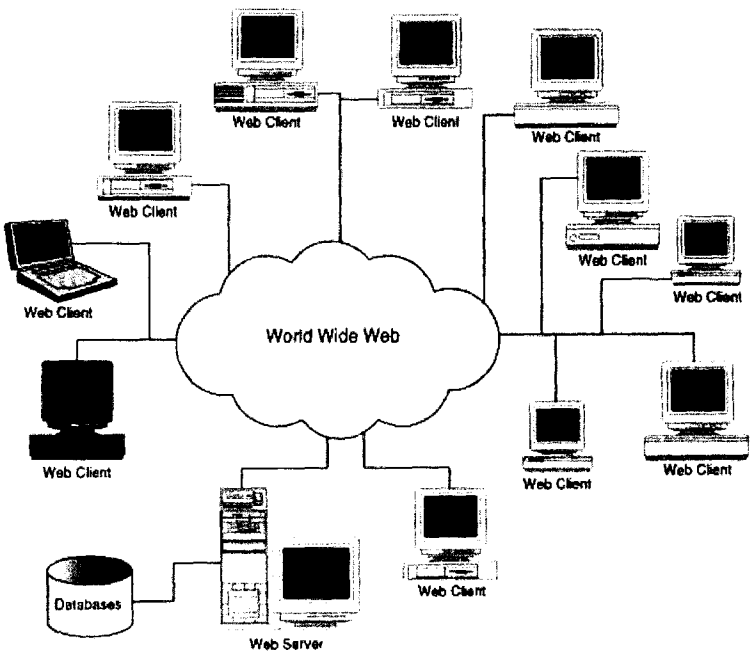

Figure 4-1 The WWW Communication

for each platform. Figure 4-1 is a simple diagram of how web server and web clients are connected through the WWW.

In the WWW, information is organized using hypertext/hypermedia. Users can use web browsers to travel between documents. There are many web browsers, such as Netscape, Microsoft Explorer, Mosaic and Lynx, available today. Netscape is probably the most popular accounting for over 70% of all browsers currently in use. [10]

The WWW is the most exciting development in the Internet. It supports not only text but also images, videos, sounds as well as graphics. Users can easily connect to web sites by using any kind of browser. By taking advantage of the WWW, the system definitely can use it as the interface.

4.4 Database Connectivity

In this section, we are going to discuss the system database connection. What is a connection? A connection is what links the ASP script code to the database; it is a way to tie them together. Once we have a way of connecting to a database, we need to know next is how to identify the database and the Data Source Name (DSN). In this case, the Access database is identified as the .mdb file.

We mentioned that the database used in this system was constructed by using Microsoft Access 97. We also know that Microsoft provides access to a database with the ActiveX

Data Objects (ADO), which builds upon OLE-DB. OLE-DB is a very similar idea to ODBC (Figure 4-2), but, in fact, it has a much broader range of data stores and can sit on the top of ODBC. That means that we will be able to keep our existing ODBC connections and use the new OLE-DB drivers. It introduces two new items: data providers and data consumers. A data provider is something that provides data and a data consumer is something that uses that data. In this contract manager system, the data consumer is ASP. In another context, the data consumer could well be an application that

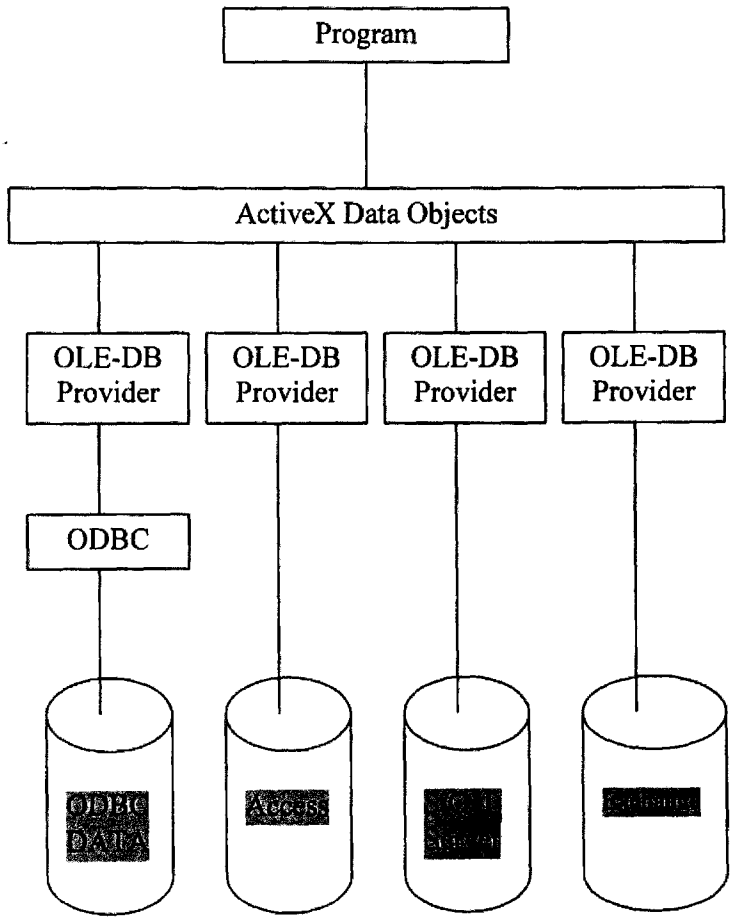

Figure 4-2 OLE-DB Architecture

is written in another language, such as Visual Basic or Visual C++. In fact, ADO is the actual consumer, because ADO talks to OLE-DB and we talk to ADO.

Indeed, we don't need to know anything about OLE-DB because ADO will hide all of the complexity from us and give us a simple way of accessing data from the database. It is the way we actually get data to and from a database. Figure 4-3 shows a diagram of how the objects provided by ADO relate to each other. The topics, such as how the recordset or fields collections work, are not our concern here. In this database connectivity section, we just need to know how the system and backstage database got connected. Simple, we just need to write one line of codes as below in our ASP file and then we will be able to connect the system with our Access database:

```
Set Conn = Server.CreateObject("ADODB.Connection")
```

Amazingly, this one-line code is actually the interaction between the database, ADO and ASP.

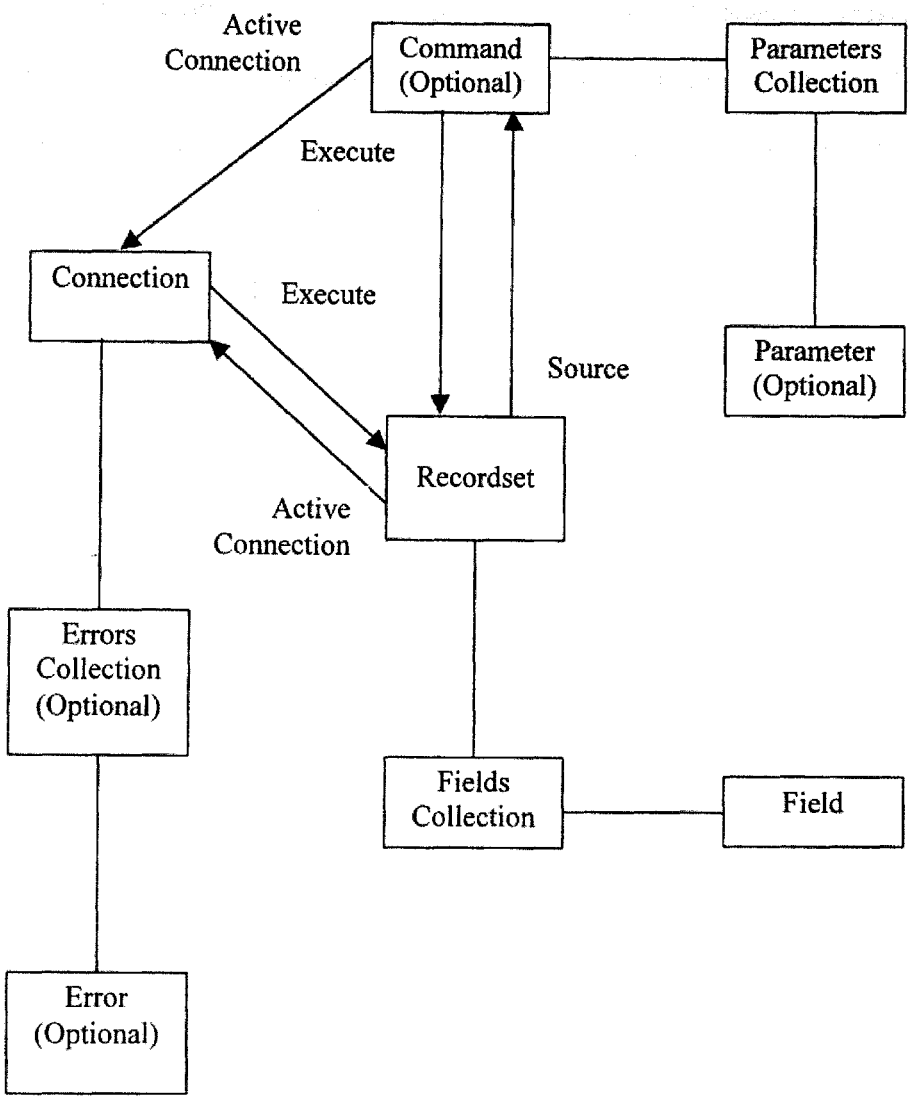

Figure 4-3 The Architecture of ADO Objects

CHAPTER 5 CONCLUSION

In this thesis, we designed and implemented an Internet-enabled database application by using ASP and other related technologies. As shown in this thesis, we have studied how to develop an on-line Internet application and, because it is Internet-enabled application, we also have included a detail interpretation on the subject of security.

An on-line Internet application is very efficient for the both super users and end users for a number of reasons:

- always on-line
- user does not get involved with system administration
- no installation needed
- no constraints on using specific software and PC
- no limitation on the location of the implementation

This system will be fully functioning at the FLDOE site. It enables remote access to monitor hundreds of contracts with advanced scheduling, warning and accounting capabilities. Because of the Internet-enabled feature, nationwide users, including FLDOE project managers, FLDOE technical advisors and contractors, can follow on the status of the contracts and contractors very closely.

To our best knowledge, the browsers still have some shortcomings need to be improved. For example, instead of using reload function on each page, browsers cannot work simultaneously with the database update function to post data. In the future, we may use some newer technologies to enhance the functionality of the system on every part of it.

Overall, this project will make dramatic improvements on the contract tracking process and more fully utilize the precious human resources available. Using this system is the fastest and easiest way to accomplish the tasks for both contractors and the government of the state Florida.

LIST OF REFERENCES

[1] NetscapeWorld; http://www.netscapeworld.com/netscapeworld/nw-09-1997/nw-09-dataquest.html; Kathleen Ohlson, IDG News Service Boston Bureau

[2] Knowledge Base on http://www.microsoft.com

[3] Sanjaya Hettihewa and Kelly Held, *Active Server Pages,* Sams.net Publishing, 1997, IN., USA

[4] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann, World-Wide Web: The Information Universe, in: *Electronic Networking: Research, Applications and Policy,* vol. 1, No. 2 (1992) 52-58.

[5] Microsoft Corporation: Windows NT.

[6] Brian Francis, John Kauffmann, Juan I. Llibre, David Sussman, and Chris Ullman, *Active Server Pages,* Wrox Publishing, 1998, Chicago, Illinois, USA.

[7] Hyper Text Markup Language v3.2 Reference, http://www2.wvitcoe.wvnet.edu/~sbolt/html3.

[8] Internet Magazine, March, 1998

[9] Mark Handley and Jon Crowcroft, *World-Wide Web: Beneath the Surf,* UCL Press, 1994.

[10] Internet White Paper (EARTHWEB), http://www.idl.com/whitepap.html.

[11] Brent Baccala, *Connected: An Internet Encyclopedia,* http://www.freesoft.org/connected/index.html, 1996

[12] Tracy LaQuey, *The Internet Companion (A Beginner's Guide to Global networking),* ddison-Wesley, 1996

[13] D. Comer, *Internetworking with TCP/IP: Principles, Protocols, Architectures,* Prentice Hall, 1988

[14] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, *Object-Oriented Requirements Analysis and Logical Design,* John Wiley and Sons, New York, 1993.

[15] Microsoft Corporation: *Building Applications with Microsoft Access for Window95 Programming with Visual Basic for Application*, 1997.

[16] R. Elmasri, S.B. Navathe: *Fundamentals of Database Systems.* The Benjamin Cammings Publishing Company, 1994

[17] Microsoft Corporation: *Building Web Applications with Microsoft Active Server Pages,* 1997.

[18] Microsoft Corporation: Microsoft SQL Server version 6.0 Transact - SQL Reference, 1997.

APPENDICES

```
<% msg=session("msg")%>
<HTML>
<HEAD>
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-
1">
  <META NAME="Author" CONTENT="Yaojen Chang">
  <TITLE>FLDOE SAE Tracking System</TITLE>
</HEAD>
<BODY>

<Table align=center cellspacing=0 border=0>
<tr>
   <td rowspan=2><image src=images/DoeLogoTop1.gif>
        <td align=left><image src=images/DoeLogoTop22.jpg><tr>
        <td><font color=darkblue><b>
        <p>The Food and Nutrition Management Section (F&NM) of the Florida
        Department of Education(FLDOE) is funded with moneys from the United States
        Department of Agriculture (USDA). All grants, projects or purchase orders
        entered into by F&NM are subject to regular audits by both USDA, and Auditor
        General of the state of Florida</b></font><tr>
        </td>
</tr>
</Table>
<br>

<TABLE align=center WIDTH=640 BORDER=0 CELLPADDING=5
CELLSPACING=0 BGCOLOR=0>
<TR><TD BGCOLOR="#B0D0BA" ALIGN=CENTER>

<TABLE>
<TR><TD>
        <TABLE>
        <TR>
                <TD VALIGN=CENTER><IMG SRC="images/handshake.jpg"></TD>
                <TD WIDTH=16></TD>
                <TD VALIGN=CENTER><STRONG><FONT SIZE=+1
                FACE="Verdana, Arial, Helvetica">Welcome, please identify
                yourself:</FONT></STRONG> <br><font color=red><b>Note: <u>All
                inputs are case sensitive.</u></b></font></TD>
```

```
        </TR>
        </TABLE>
</TD>
</TR>
</TABLE>
<hr>
<FORM Method="Post" Action="LoginCheck.asp">
<TABLE BORDER=0>
<TR>
   <TD><FONT FACE="Verdana,Arial,Helvetica">Username:</FONT></TD>
   <TD><INPUT Type="Text" name="UserName" value="" maxlength=20></TD>
</TR>
<TR>
        <TD><FONT FACE="Verdana,Arial,Helvetica">Password:</FONT></TD>
        <TD><INPUT Type="Password" name="Password" value=""
        MAXLENGTH=20></TD>
        <TD align=center><INPUT Type="Submit" value="Login"></TD>
</TR>
</TABLE>
</FORM>
</TD></TR>
</TABLE>

<Center><b>First-time login users <a href="mailto:ychang01@cs.fiu.edu">Click
Here</a> to request your login accounts.</b></Center>

<br>

<blink><%=msg%></blink>
<%
  if session("msg")="" then
        session.abandon
  End If
  session("msg")=""
%>
</BODY>
</HTML>
```

Source Code of Validation Check for Typical Users

```
<Script Language=VBScript runat=server>
Function Redirect()
        Response.Redirect "Login.asp"
End function
```

```
</Script>

<%
set Conn=Server.CreateObject("ADODB.Connection")
Conn.Open("SAEforASP")

If Request.Form("UserName")="" or Request.Form("Password")="" then
        session("msg")="<h3><center><Font color=red>Oops! You must enter all
        information needed.</font></center></h3>"
        Redirect
Else
        sqlLogin="Select * From LoginUsers Where
        ((LoginUsers.UserName='"&Request.form("UserName")&"'));"
        set rsLogin=Conn.Execute(sqlLogin)
End If

If rsLogin.eof then
        rsLogin.close
        session("msg")="<H3><center><Font color=red>INVALID Account. Please try
        again or quit.</font></center></H3>"
        Redirect
Elseif not rsLogin("Password")=Request.Form("Password") then 'match the username,
check password
        rsLogin.close
        session("msg")="<h3><center><Font color=red>Right user name but Wrong
        password. Please try again or quit.</font></center></H3>"
        Redirect
Else
        session("UserType")=rsLogin("UserType")
        session("UserName")=rsLogin("UserName")
        session("Password")=rsLogin("Password")
        session("ContractNumber")=rsLogin("LNameOrContractNo")
        session("CompanyName")=rsLogin("FNameOrCompanyName")
        session("LogonStatus")=1
        session("AdmOnlineDate")=rsLogin("AdmOnlineDate")
        sql="Update LoginUsers Set LoginUsers.AdmOnlineDate=#"&Date()&"# Where
        ((LoginUsers.Password)='"&Request.form("Password")&"');"
        set rs2=Conn.Execute(sql)

        SELECT CASE session("UserType")
                case "S"
                        Response.Redirect "SuperUserMain.asp"
                case "s"
                        Response.Redirect "SuperUserMain.asp"
                case "C"
```

```
                    Response.Redirect "EndUserMain.asp"
                case "c"
                    Response.Redirect "EndUserMain.asp"
        END SELECT
End If

rsLogin.close
Conn.Close

%>
```

Some Functions Defined in the Contracts page
(Intact Source Code is of 760 lines)

```
<%@ LANGUAGE="vbscript" %>
<%
'-----------------------------------------------------------------------------------------------------
' Modes: The form mode can be controlled by passing the following
'         name/value pairs using POST or GET:
'             FormMode=Edit
'             FormMode=Search
'             FormMode=Add
' - If a field contains a URL to an image and has a name that begins with "img_"
'   (case-insensitive), the image will be displayed using the IMG tag.
'- If a field contains a URL and has a name that begins with "url_"
'  (case-insensitive), a jump will be displayed using the Anchor tag.
'-----------------------------------------------------------------------------------------------------

'-----------------------------------------------------------------------------------------------------
' Purpose:  Substitutes Empty for Null and trims leading/trailing spaces
' Inputs:    varTemp   - the target value
' Returns:  The processed value
'-----------------------------------------------------------------------------------------------------

Function ConvertNull(varTemp)
        If IsNull(varTemp) Then
                ConvertNull = ""
        Else
                ConvertNull = Trim(varTemp)
        End If
End Function
```

```
'-----------------------------------------------------------------------------------------------
' Purpose:  Embeds bracketing quotes around the string
' Inputs:   varTemp   - the target value
' Returns:  The processed value
'-----------------------------------------------------------------------------------------------

Function QuotedString(varTemp)
        If IsNull(varTemp) Then
                QuotedString = Chr(34) & Chr(34)
        Else
                QuotedString = Chr(34) & CStr(varTemp) & Chr(34)
        End If
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Tests string to see if it is a URL by looking for protocol
' Inputs:   varTemp   - the target value
' Returns:  True - if is URL, False if not
'-----------------------------------------------------------------------------------------------

Function IsURL(varTemp)
        IsURL = True
        If UCase(Left(Trim(varTemp), 6)) = "HTTP:/" Then Exit Function
        If UCase(Left(Trim(varTemp), 6)) = "FILE:/" Then Exit Function
        If UCase(Left(Trim(varTemp), 8)) = "MAILTO:/" Then Exit Function
        If UCase(Left(Trim(varTemp), 5)) = "FTP:/" Then Exit Function
        If UCase(Left(Trim(varTemp), 8)) = "GOPHER:/" Then Exit Function
        If UCase(Left(Trim(varTemp), 6)) = "NEWS:/" Then Exit Function
        If UCase(Left(Trim(varTemp), 7)) = "HTTPS:/" Then Exit Function
        If UCase(Left(Trim(varTemp), 8)) = "TELNET:/" Then Exit Function
        If UCase(Left(Trim(varTemp), 6)) = "NNTP:/" Then Exit Function
        IsURL = False
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Tests whether the field in the recordset is updatable
' Assumes:  That the recordset containing the field is open
' Inputs:   strFieldName      - the name of the field in the recordset
' Returns:  True if updatable, False if not
'-----------------------------------------------------------------------------------------------

Function CanUpdateField(strFieldName)
        Dim intUpdatable
        intUpdatable = (adFldUpdatable Or adFldUnknownUpdatable)
        CanUpdateField = True
```

```
        If (rsSContractContractors(strFieldName).Attributes And intUpdatable) = False
Then
                CanUpdateField = False
        End If
End Function
```

Display Handler
(Intact Source Code is of 132 lines)

```
'-------------------------------------------------------------------------------------------------
' Purpose:  Handles the display of a field from a recordset depending on its data type,
'           attributes, and the current mode.
' Assumes: That the recordset containing the field is open That strFormMode is initialized
' Inputs:   strFieldName - the name of the field in the recordset
'           strLabel - the label to display
'           blnIdentity - identity field flag
'           avarLookup - array of lookup values
'-------------------------------------------------------------------------------------------------

Sub ShowField(strFieldName, strLabel, blnIdentity, avarLookup)
        Dim blnFieldRequired
        Dim intMaxSize
        Dim intInputSize
        Dim strOption1State
        Dim strOption2State
        Dim strFieldValue
        Dim nPos
        strFieldValue = ""
        nPos=Instr(strFieldName,".")
        Do While nPos > 0
                strFieldName= Mid (strFieldName, nPos+1)
                nPos=Instr(strFieldName,".")
        Loop
        ' If not in Edit form mode then set value to empty so doesn't display
        strFieldValue = ""
        If strFormMode = "Edit" Then strFieldValue =
RTrim(rsSContractContractors(strFieldName))

        ' See if the field is required by checking the attributes
        blnFieldRequired = False
        If (rsSContractContractors(strFieldName).Attributes And adFldIsNullable) = 0
        Then
                blnFieldRequired = True
        End If
```

```
' Set values for the MaxLength and Size attributes
intMaxSize = dfMaxSize
intInputSize = rsSContractContractors(strFieldName).DefinedSize + 2
If strFormMode <> "Search" Then intMaxSize = intInputSize - 2

' Write the field label and start the value cell
Response.Write "<TR VALIGN=TOP>"
Response.Write "<TD HEIGHT=25 ALIGN=Left NOWRAP><FONT SIZE=-
1><B>  " & strLabel & "</B></FONT></TD>"
Response.Write "<TD WIDTH=100% ><FONT SIZE=-1>"

' If the field is not updatable, then handle
' it like an Identity column and exit
If Not CanUpdateField(strFieldName) Then
        ' Special handling if Binary
        Select Case rsSContractContractors(strFieldName).Type

                Case adBinary, adVarBinary, adLongVarBinary      'Binary
                        Response.Write "[Binary]"
                Case Else
                        Select Case strFormMode

                                Case "Edit"
                                        Response.Write ConvertNull(strFieldValue)
                                        Response.Write "<INPUT TYPE=Hidden
                                        NAME=" & QuotedString(strFieldName)
                                        Response.Write " VALUE=" &
                                        QuotedString(strFieldValue) & " >"

                                Case "Add"
                                        Response.Write "[AutoNumber]"
                                        Response.Write "<INPUT TYPE=Hidden
                                        NAME=" & QuotedString(strFieldName)
                                        Response.Write " VALUE=" &
                                        QuotedString(strFieldValue) & " >"

                                Case "Search"
                                        Response.Write "<INPUT TYPE=Text
                                        NAME=" & QuotedString(strFieldName)
                                        Response.Write " SIZE=" & intInputSize
                                        Response.Write " MAXLENGTH=" &
                                        intMaxSize
                                        Response.Write " VALUE=" &
                                        QuotedString(strFieldValue) & " >"
                        End Select
```

```
        End Select
        Response.Write "</FONT></TD></TR>"
        Exit Sub
End If
' Handle lookups using a select and options
If Not IsNull(avarLookup) Then
        Response.Write "<SELECT NAME=" & QuotedString(strFieldName) &
        ">"
        ' Add blank entry if not required or in search mode
        If Not blnFieldRequired Or strFormMode = "Search" Then
                If (strFormMode = "Search" Or strFormMode = "Add") Then
                        Response.Write "<OPTION SELECTED>"
                Else
                        Response.Write "<OPTION>"
                End If
        End If

        ' Loop thru the rows in the array
        For intRow = 0 to UBound(avarLookup, 2)
                Response.Write "<OPTION VALUE=" &
                QuotedString(avarLookup(0, intRow))
                If strFormMode = "Edit" Then
                        If ConvertNull(avarLookup(0, intRow)) =
                        ConvertNull(strFieldValue) Then
                        Response.Write " SELECTED"
                        End If
                End If
                Response.Write ">"
                Response.Write ConvertNull(avarLookup(1, intRow))
        Next
        Response.Write "</SELECT>"
        If blnFieldRequired And strFormMode = "Add" Then
                Response.Write "  Required"
        End If
        Response.Write "</FONT></TD></TR>"
        Exit Sub
End If

' Evaluate data type and handle appropriately
Select Case rsSContractContractors(strFieldName).Type

        Case adBoolean, adUnsignedTinyInt          'Boolean
                If strFormMode = "Search" Then
                        strOption1State = " >Yes"
                        strOption2State = " >No"
```

```
        Else
            Select Case strFieldValue
                Case "True", "1", "-1"
                    strOption1State = " CHECKED>Yes"
                    strOption2State = " >No"
                Case "False", "0"
                    strOption1State = " >Yes"
                    strOption2State = " CHECKED>No"
                Case Else
                    strOption1State = " >Yes"
                    strOption2State = " >No"
            End Select
        End If
        Response.Write "<INPUT TYPE=Radio VALUE=1 NAME=" &
        QuotedString(strFieldName) & strOption1State
        Response.Write "<INPUT TYPE=Radio VALUE=0 NAME=" &
        QuotedString(strFieldName) & strOption2State
        If strFormMode = "Search" Then
            'Response.Write "<INPUT TYPE=Radio NAME=" &
            QuotedString(strFieldName) & " CHECKED>Neither"
            Response.Write "<INPUT TYPE=hidden NAME=" &
            QuotedString(strFieldName) & " CHECKED> <===
            MUST Check One Of These For The Status Of Contract."
        End If

Case adBinary, adVarBinary, adLongVarBinary      'Binary

        Response.Write "[Binary]"

Case adLongVarChar, adLongVarWChar               'Memo

        Response.Write "<TEXTAREA NAME=" &
        QuotedString(strFieldName) & " ROWS=3 COLS=80>"
        Response.Write
        Server.HTMLEncode(ConvertNull(strFieldValue))
        Response.Write "</TEXTAREA>"

Case Else

        Dim nType
        nType=rsSContractContractors(strFieldName).Type
        If (nType <> adVarChar) and (nType <> adWVarChar) and
        (nType <> adBSTR) and (nType <> adChar) and (nType <>
        adWChar)  Then intInputSize = (intInputSize-2)*3+2
```

```
                    If strFormMode <> "Search" Then intMaxSize =
                    intInputSize - 2
                    End If
```

Form Mode Handler
(Intact Source Code is of 62 lines)

```
If blnIdentity Then
        Select Case strFormMode

                Case "Edit"
                        Response.Write ConvertNull(strFieldValue)
                        Response.Write "<INPUT TYPE=Hidden NAME=" &
                        QuotedString(strFieldName)
                        Response.Write " VALUE=" & QuotedString(strFieldValue) & "
                        >"
                Case "Add"
                        Response.Write "[AutoNumber]"
                        Response.Write "<INPUT TYPE=Hidden NAME=" &
                        QuotedString(strFieldName)
                        Response.Write " VALUE=" & QuotedString(strFieldValue) & "
                        >"
                Case "Search"
                        Response.Write "<INPUT TYPE=Text NAME=" &
                        QuotedString(strFieldName) & " SIZE=" & tInputSize
                        Response.Write " MAXLENGTH=" & tMaxSize & " VALUE="
                        & QuotedString(strFieldValue) & " >"
        End Select
Else
```

Length of Display Fields Handler

```
        If intInputSize =50 Then

                intInputSize = 9
                Response.Write "<INPUT TYPE=Text NAME=" &
                QuotedString(strFieldName)
                Response.Write " SIZE=" & intInputSize
                Response.Write " MAXLENGTH=" & intMaxSize
                Response.Write " VALUE=" & QuotedString(strFieldValue) & " >"
                elseif intinputsize =26 then
                intInputSize = 9
                Response.Write "<INPUT TYPE=Text NAME=" &
                QuotedString(strFieldName)
                Response.Write " SIZE=" & intInputSize
```

```
        Response.Write " MAXLENGTH=" & intMaxSize
        Response.Write " VALUE=" & QuotedString(strFieldValue) & " >"
        elseif intinputsize>50 then
        Response.Write "<TEXTAREA NAME=" & QuotedString(strFieldName)
        & " ROWS=2 COLS=34>"
        Response.Write Server.HTMLEncode(ConvertNull(strFieldValue))
        Response.Write "</TEXTAREA>"

Else
        intInputSize=15
        Response.Write "<INPUT TYPE=Text NAME=" &
        QuotedString(strFieldName)
        Response.Write " SIZE=" & intInputSize
        Response.Write " MAXLENGTH=" & intMaxSize
        Response.Write " VALUE=" & QuotedString(strFieldValue) & " >"

        'If intInputSize > 80 Then intInputSize = 80
        'Response.Write "<INPUT TYPE=Text NAME=" &
        QuotedString(strFieldName)
        'Response.Write " SIZE=" & intInputSize
        'Response.Write " MAXLENGTH=" & intMaxSize
        'Response.Write " VALUE=" & QuotedString(strFieldValue) & " >"

        ' Check for special field types
        Select Case UCase(Left(rsSContractContractors(strFieldName).Name, 4))

                Case "IMG_"
                        If strFieldValue <> "" Then
                        Response.Write "<BR><BR><IMG SRC=" &
                        QuotedString(strFieldValue) & "><BR> <BR>"
                        End If

                Case "URL_"
                        If strFieldValue <> "" Then
                        Response.Write "  <A HREF=" &
                        QuotedString(strFieldValue) & ">"
                        Response.Write "Go"
                        Response.Write "</A>"
                        End If

                Case Else
                        If IsURL(strFieldValue) Then
                        Response.Write "  <A HREF=" &
                        QuotedString(strFieldValue) & ">"
                        Response.Write "Go"
```

```
                        Response.Write "</A>"
                        End If
                End Select
        End If
End If
End Select

If blnFieldRequired And strFormMode = "Add" Then
        'Response.Write "    <<=== Required; You MUST indicate the status for the
        contract."
End If
Response.Write "</FONT></TD></TR>"

End Sub
</SCRIPT>
```

Some Functions defined in the Contracts action page
(Intact Source Code is of 512 lines)

```
<%@ LANGUAGE="VBScript" %>
<%
'-----------------------------------------------------------------------------------------
' Action Page
' This file is an Active Server Page that contains the server script that handles filter,
' update, insert, and delete commands from the form view of a Data Form. It can also
' echo back confirmation of database operations and report errors.
' Some commands are passed through and redirected.
'-----------------------------------------------------------------------------------------
%>

<SCRIPT RUNAT=Server LANGUAGE="VBScript">
'-----------------------------------------------------------------------------------------
' Purpose:  Substitutes Null for Empty
' Inputs:    varTemp   - the target value
' Returns:  The processed value
'-----------------------------------------------------------------------------------------

Function RestoreNull(varTemp)
        If Trim(varTemp) = "" Then
                RestoreNull = Null
        Else
                RestoreNull = varTemp
        End If
End Function
```

```
Sub RaiseError(intErrorValue, strFieldName)
        Dim strMsg
        Select Case intErrorValue
                Case errInvalidPrefix
                strMsg = "Wildcard characters * and % can only be used at the end of the
                criteria"
                Case errInvalidOperator
                        strMsg = "Invalid filtering operators - use <= or >= instead."
                Case errInvalidOperatorUse
                        strMsg = "The 'Like' operator can only be used with strings."
                Case errNotEditable
                        strMsg = strFieldName & " field is not editable."
                Case errValueRequired
                        strMsg = "A value is required for " & strFieldName & "."
        End Select
        Err.Raise intErrorValue, "DataForm", strMsg
End Sub

'---------------------------------------------------------------------------------------
' Purpose:  Converts to subtype of string - handles Null cases
' Inputs:   varTemp   - the target value
' Returns:  The processed value
'---------------------------------------------------------------------------------------

Function ConvertToString(varTemp)
        If IsNull(varTemp) Then
                ConvertToString = Null
        Else
                ConvertToString = CStr(varTemp)
        End If
End Function

'---------------------------------------------------------------------------------------
' Purpose:  Tests to equality while dealing with Null values
' Inputs:   varTemp1  - the first value
'           varTemp2  - the second value
' Returns:  True if equal, False if not
'---------------------------------------------------------------------------------------

Function IsEqual(ByVal varTemp1, ByVal varTemp2)
        IsEqual = False
        If IsNull(varTemp1) And IsNull(varTemp2) Then
                IsEqual = True
        Else
```

```
            If IsNull(varTemp1) Then Exit Function
            If IsNull(varTemp2) Then Exit Function
        End If
        If varTemp1 = varTemp2 Then IsEqual = True
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Tests whether the field in the recordset is required
' Assumes: That the recordset containing the field is open
' Inputs:     strFieldName      - the name of the field in the recordset
' Returns:   True if updatable, False if not
'-----------------------------------------------------------------------------------------------

Function IsRequiredField(strFieldName)
        IsRequiredField = False
        If (rsSContractContractors(strFieldName).Attributes And adFldIsNullable) = 0
Then
            IsRequiredField = True
        End If
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Tests whether the field in the recordset is updatable
' Assumes: That the recordset containing the field is open
' Effects:    Sets Err object if field is not updatable
' Inputs:     strFieldName - the name of the field in the recordset
' Returns:   True if updatable, False if not
'-----------------------------------------------------------------------------------------------

Function CanUpdateField(strFieldName)
        Dim intUpdatable
        intUpdatable = (adFldUpdatable Or adFldUnknownUpdatable)
        CanUpdateField = True
        If (rsSContractContractors(strFieldName).Attributes And intUpdatable) = False
Then
            CanUpdateField = False
        End If
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Insert operation - updates a recordset field with a new value
'           during an insert operation.
' Assumes: That the recordset containing the field is open
' Effects:    Sets Err object if field is not set but is required
' Inputs:     strFieldName - the name of the field in the recordset
```

```
' Returns:  True if successful, False if not
'-------------------------------------------------------------------------------

Function InsertField(strFieldName)
        InsertField = True
        If IsEmpty(Request(strFieldName)) Then Exit Function
        Select Case rsSContractContractors(strFieldName).Type
                Case adBinary, adVarBinary, adLongVarBinary      'Binary
                Case Else
                        If CanUpdateField(strFieldName) Then
                                If IsRequiredField(strFieldName) And
                                IsNull(RestoreNull(Request(strFieldName))) Then
                                        RaiseError errValueRequired, strFieldName
                                        InsertField = False
                                        Exit Function
                                End If
                                rsSContractContractors(strFieldName) =
                                RestoreNull(Request(strFieldName))
                        End If
        End Select
End Function

'-------------------------------------------------------------------------------
' Purpose:  Update operation - updates a recordset field with a new value
' Assumes: That the recordset containing the field is open
' Effects:    Sets Err object if field is not set but is required
' Inputs:    strFieldName - the name of the field in the recordset
' Returns:  True if successful, False if not
'-------------------------------------------------------------------------------

Function UpdateField(strFieldName)
        UpdateField = True
        If IsEmpty(Request(strFieldName)) Then Exit Function
        Select Case rsSContractContractors(strFieldName).Type
                Case adBinary, adVarBinary, adLongVarBinary      'Binary
                Case Else
                        ' Only update if the value has changed
                        If Not
                        IsEqual(ConvertToString(rsSContractContractors(strFieldName)),
                        RestoreNull(Request(strFieldName))) Then
                        If CanUpdateField(strFieldName) Then

                                If IsRequiredField(strFieldName) And
                                IsNull(RestoreNull(Request(strFieldName))) Then
                                RaiseError errValueRequired, strFieldName
```

```
                    UpdateField = False
                        Exit Function
                    End If
                    rsSContractContractors(strFieldName) =
                    RestoreNull(Request(strFieldName))
                Else
                        RaiseError errNotEditable, strFieldName
                        UpdateField = False
                End If
                End If
        End Select
End Function

'-------------------------------------------------------------------------------------------
' Purpose:  Criteria handler for a field in the recordset. Determines
'           correct delimiter based on data type
' Effects:  Appends to strWhere and strWhereDisplay variables
' Inputs:   strFieldName - the name of the field in the recordset
'           avarLookup - lookup array - null if none
'-------------------------------------------------------------------------------------------

Sub FilterField(ByVal strFieldName, avarLookup)
        Dim strFieldDelimiter
        Dim strDisplayValue
        Dim strValue
        Dim intRow
        strValue = Request(strFieldName)
        strDisplayValue = Request(strFieldName)

        ' If empty then exit right away
        If Request(strFieldName) = "" Then Exit Sub
'IF request(strFieldName)="Content" then
                'strFieldName=""
'ELSE

        ' Concatenate the And boolean operator
        If strWhere <> "" Then strWhere = strWhere & " And"
        If strWhereDisplay <> "" Then strWhereDisplay = strWhereDisplay & " And"

        ' If lookup field, then use lookup value for display
        If Not IsNull(avarLookup) Then
                For intRow = 0 to UBound(avarLookup, 2)
                        If CStr(avarLookup(0, intRow)) = Request(strFieldName) Then
                                strDisplayValue = avarLookup(1, intRow)
                                Exit For
```

```
                End If
            Next
        End If
'END IF
        ' Set delimiter based on data type
        Select Case rsSContractContractors(strFieldName).Type
            Case adBSTR, adChar, adWChar, adVarChar, adVarWChar   'string types
                strFieldDelimiter = "'"
            Case adLongVarChar, adLongVarWChar                    'long string types
                strFieldDelimiter = "'"
            Case adDate, adDBDate, adDBTimeStamp                  'date types
                strFieldDelimiter = "#"
            Case Else
                strFieldDelimiter = ""
        End Select

        ' Modifies script level variables
        strWhere = strWhere & " " & PrepFilterItem(strFieldName, strValue,
        strFieldDelimiter)
        strWhereDisplay = strWhereDisplay & " " & PrepFilterItem(strFieldName,
        strDisplayValue, strFieldDelimiter)

End Sub

'-------------------------------------------------------------------------------------------------
' Purpose:  Constructs a name/value pair for a where clause
' Effects:  Sets Err object if the criteria is invalid
' Inputs:   strFieldName - the name of the field in the recordset
'           strCriteria - the criteria to use
'           strDelimiter - the proper delimiter to use
' Returns:  The name/value pair as a string
'-------------------------------------------------------------------------------------------------

Function PrepFilterItem(ByVal strFieldName, ByVal strCriteria, ByVal strDelimiter)
        Dim strOperator
        Dim intEndOfWord
        Dim strWord

        ' Char, VarChar, and LongVarChar must be single quote delimited.
        ' Dates are pound sign delimited.
        ' Numerics should not be delimited.
        ' String to Date conversion rules are same as VBA.
        ' Only support for ANDing.
        ' Support the LIKE operator but only with * or % as suffix.
```

```
strCriteria = Trim(strCriteria) 'remove leading/trailing spaces
strOperator = "="                    'sets default
strValue = strCriteria               'sets default

' Get first word and look for operator
intEndOfWord = InStr(strCriteria, " ")
If intEndOfWord Then
        strWord = UCase(Left(strCriteria, intEndOfWord - 1))
        ' See if the word is an operator
        Select Case strWord
                Case "=", "<", ">", "<=", ">=", "<>", "LIKE"
                        strOperator = strWord
                        strValue = Trim(Mid(strCriteria, intEndOfWord + 1))
                Case "=<", "=>"
                        RaiseError errInvalidOperator, strFieldName
        End Select
Else
        strWord = UCase(Left(strCriteria, 2))
        Select Case strWord
                Case "<=", ">=", "<>"
                        strOperator = strWord
                        strValue = Trim(Mid(strCriteria, 3))
                Case "=<", "=>"
                        RaiseError errInvalidOperator, strFieldName
                Case Else
                        strWord = UCase(Left(strCriteria, 1))
                        Select Case strWord
                                Case "=", "<", ">"
                                        strOperator = strWord
                                        strValue = Trim(Mid(strCriteria, 2))
                        End Select
        End Select
End If

' Make sure LIKE is only used with strings
If strOperator = "LIKE" and strDelimiter <> "'" Then
        RaiseError errInvalidOperatorUse, strFieldName
End If

' Strip any extraneous delimiters because we add them anyway
' Single Quote
If Left(strValue, 1) = Chr(39) Then strValue = Mid(strValue, 2)
If Right(strValue, 1) = Chr(39) Then strValue = Left(strValue, Len(strValue) - 1)

' Double Quote - just in case
```

```
        If Left(strValue, 1) = Chr(34) Then strValue = Mid(strValue, 2)
        If Right(strValue, 1) = Chr(34) Then strValue = Left(strValue, Len(strValue) - 1)

        ' Pound sign - dates
        If Left(strValue, 1) = Chr(35) Then strValue = Mid(strValue, 2)
        If Right(strValue, 1) = Chr(35) Then strValue = Left(strValue, Len(strValue) - 1)

        ' Check for leading wildcards
        If Left(strValue, 1) = "*" Or Left(strValue, 1) = "%" Then
                RaiseError errInvalidPrefix, strFieldName
        End If

        PrepFilterItem = "[" & strFieldName & "]" & " " & strOperator & " " &
strDelimiter & strValue & strDelimiter
End Function

</SCRIPT>


<%
If Not IsEmpty(Request("DataAction")) Then
        strDataAction = Trim(Request("DataAction"))
Else
        Response.Redirect "SContractForm.asp?FormMode=Edit"
End If
```

Action handler
(Intact Source Code is of 176 lines)

```
Select Case strDataAction

        Case "List View"

                Response.Redirect "SContractList.asp"

        Case "Cancel"

                Response.Redirect "SContractForm.asp?FormMode=Edit"

        Case "Search"

                On Error Resume Next
                Session("rsSContractContractors_Filter") = ""
                'Session("rsSContractContractors_FilterDisplay") = ""
                Session("rsSContractContractors_Recordset").Filter = ""
```

```
        Response.Redirect "SContractForm.asp?FormMode=" & strDataAction

Case "Add"

        On Error Resume Next
        Session("rsSContractContractors_Filter") = ""
        Session("rsSContractContractors_FilterDisplay") = ""
        Session("rsSContractContractors_Recordset").Filter = ""
        Response.Redirect "SContractForm.asp?FormMode=" & strDataAction

Case "Find"

        Session("rsSContractContractors_PageSize") = 1 'So we don't do standard
        page conversion
        Session("rsSContractContractors_AbsolutePage") =
        CLng(Request("Bookmark"))
        Response.Redirect "SContractForm.asp"

Case "All Records"

        On Error Resume Next
        Session("rsSContractContractors_Filter") = ""
        Session("rsSContractContractors_FilterDisplay") = ""
        Session("rsSContractContractors_Recordset").Filter = ""
        Session("rsSContractContractors_AbsolutePage") = 1
        Response.Redirect "SContractForm.asp"

Case "Go"

        On Error Resume Next
        ' Make sure we exit and re-process the form if session has timed out
        If IsEmpty(Session("rsSContractContractors_Recordset")) Then
                Response.Redirect "SContractForm.asp?FormMode=Edit"
        End If

        Set rsSContractContractors =
        Session("rsSContractContractors_Recordset")

        strWhere = ""
        strWhereDisplay = ""
        FilterField "Complete", Null
        FilterField "Title", Null
        FilterField "ContractorNumber", Null
        FilterField "BeginDate", Null
        FilterField "EndDate", Null
```

```
        FilterField "Amount", Null
        FilterField "IntFundingSource", Null
        FilterField "Content", Null
        FilterField "DOEStaff", Null
        FilterField "DOETech", Null
        FilterField "DoeNote", Null
        FilterField "DoeNotice", Null

        'Filter the recordset
        If strWhere <> "" Then
                Session("rsSContractContractors_Filter") = strWhere
                Session("rsSContractContractors_FilterDisplay") =
                strWhereDisplay
                Session("rsSContractContractors_AbsolutePage") = 1
        Else
                Session("rsSContractContractors_Filter") = ""
                Session("rsSContractContractors_FilterDisplay") = ""
        End If

        ' Jump back to the form
        If Err.Number = 0 Then Response.Redirect "SContractForm.asp"

Case "Save"

        On Error Resume Next
        ' Make sure we exit and re-process the form if session has timed out
        If IsEmpty(Session("rsSContractContractors_Recordset")) Then
                Response.Redirect "SContractForm.asp?FormMode=Edit"
        End If

        Set rsSContractContractors =
        Session("rsSContractContractors_Recordset")
        rsSContractContractors.AddNew

        Do
                If Not InsertField("Complete") Then Exit Do
                If Not InsertField("Title") Then Exit Do
                If Not InsertField("ContractorNumber") Then Exit Do
                If Not InsertField("BeginDate") Then Exit Do
                If Not InsertField("EndDate") Then Exit Do
                If Not InsertField("Amount") Then Exit Do
                If Not InsertField("IntFundingSource") Then Exit Do
                If Not InsertField("Content") Then Exit Do
                If Not InsertField("DOEStaff") Then Exit Do
                If Not InsertField("DOETech") Then Exit Do
```

```
            If Not InsertField("DoeNote") Then Exit Do
            If Not InsertField("DoeNotice") Then Exit Do

            rsSContractContractors.Update
            Exit Do
        Loop

        If Err.Number <> 0 Then
            If rsSContractContractors.EditMode Then
            rsSContractContractors.CancelUpdate
        Else
            If IsEmpty(Session("rsSContractContractors_AbsolutePage")) Or
            Session("rsSContractContractors_AbsolutePage") = 0 Then
            Session("rsSContractContractors_AbsolutePage") = 1
            End If
            ' Requery static cursor so inserted record is visible
            If rsSContractContractors.CursorType = adOpenStatic Then
            rsSContractContractors.Requery
            Session("rsSContractContractors_Status") = "Record has been
            inserted"
        End If

Case "Update"

        On Error Resume Next
        ' Make sure we exit and re-process the form if session has timed out
        If IsEmpty(Session("rsSContractContractors_Recordset")) Then
            Response.Redirect "SContractForm.asp?FormMode=Edit"
        End If

        Set rsSContractContractors =
        Session("rsSContractContractors_Recordset")
        If rsSContractContractors.EOF and rsSContractContractors.BOF Then
        Response.Redirect "SContractForm.asp"

        Do

            If Not UpdateField("Complete") Then Exit Do
            If Not UpdateField("Title") Then Exit Do
            If Not UpdateField("ContractorNumber") Then Exit Do
            If Not UpdateField("BeginDate") Then Exit Do
            If Not UpdateField("EndDate") Then Exit Do
            If Not UpdateField("Amount") Then Exit Do
            If Not UpdateField("IntFundingSource") Then Exit Do
            If Not UpdateField("Content") Then Exit Do
```

```
            If Not UpdateField("DOEStaff") Then Exit Do
            If Not UpdateField("DOETech") Then Exit Do
            If Not UpdateField("DoeNote") Then Exit Do
            If Not UpdateField("DoeNotice") Then Exit Do

            If rsSContractContractors.EditMode Then
            rsSContractContractors.Update
            Exit Do
        Loop

        If Err.Number <> 0 Then
            If rsSContractContractors.EditMode Then
            rsSContractContractors.CancelUpdate
        End If

Case "Delete"

        On Error Resume Next

        ' Make sure we exit and re-process the form if session has timed out
        If IsEmpty(Session("rsSContractContractors_Recordset")) Then
            Response.Redirect "SContractForm.asp?FormMode=Edit"
        End If

        Set rsSContractContractors =
        Session("rsSContractContractors_Recordset")
        If rsSContractContractors.EOF and rsSContractContractors.BOF Then
        Response.Redirect "SContractForm.asp"

        rsSContractContractors.Delete

        ' Proceed if no error
        If Err.Number = 0 Then
            ' Requery static cursor so deleted record is removed
            If rsSContractContractors.CursorType = adOpenStatic Then
            rsSContractContractors.Requery

            ' Move off deleted rec
            rsSContractContractors.MoveNext

            ' If at EOF then jump back one and adjust AbsolutePage
            If rsSContractContractors.EOF Then
                rsSContractContractors.MovePrevious
```

```
                    Session("rsSContractContractors_AbsolutePage") =
                    Session("rsSContractContractors_AbsolutePage") - 1

                    If rsSContractContractors.BOF And
                    rsSContractContractors.EOF Then
                    rsSContractContractors.Requery
                End If
            End If

End Select
%>

<%
Select Case strDataAction

    Case "Save"

        Response.Write("Unable to save the record into Contractors.")

    Case "Update"

        Response.Write("Unable to post the updated record to Contractors.")

    Case "Delete"

        Response.Write("Unable to delete the record from Contractors.")

End Select
%>
```

Some functions defined in Payment action page
(Intact Source Code is of 225 lines)

```
'-----------------------------------------------------------------------------------------------
' Purpose:  Substitutes Null for Empty
' Inputs:    varTemp   - the target value
' Returns:  The processed value
'-----------------------------------------------------------------------------------------------

Function RestoreNull(varTemp)
    If Trim(varTemp) = "" Then
        RestoreNull = Null
    Else
        RestoreNull = varTemp
    End If
```

```
End Function

Sub RaiseError(intErrorValue, strFieldName)
        Dim strMsg
        Select Case intErrorValue
                Case errInvalidPrefix
                        strMsg = "Wildcard characters * and % can only be used at the end
                        of the criteria"
                Case errInvalidOperator
                        strMsg = "Invalid filtering operators - use <= or >= instead."
                Case errInvalidOperatorUse
                        strMsg = "The 'Like' operator can only be used with strings."
                Case errNotEditable
                        strMsg = strFieldName & " field is not editable."
                Case errValueRequired
                        strMsg = "A value is required for " & strFieldName & "."
        End Select
        Err.Raise intErrorValue, "DataForm", strMsg
End Sub

'---------------------------------------------------------------------------------------
' Purpose:  Converts to subtype of string - handles Null cases
' Inputs:   varTemp - the target value
' Returns:  The processed value
'---------------------------------------------------------------------------------------

Function ConvertToString(varTemp)
        If IsNull(varTemp) Then
                ConvertToString = Null
        Else
                ConvertToString = CStr(varTemp)
        End If
End Function

'---------------------------------------------------------------------------------------
' Purpose:  Tests to equality while dealing with Null values
' Inputs:   varTemp1  - the first value
'           varTemp2  - the second value
' Returns:  True if equal, False if not
'---------------------------------------------------------------------------------------

Function IsEqual(ByVal varTemp1, ByVal varTemp2)
        IsEqual = False
                If IsNull(varTemp1) And IsNull(varTemp2) Then
                        IsEqual = True
```

```
        Else
                If IsNull(varTemp1) Then Exit Function
                If IsNull(varTemp2) Then Exit Function
        End If
        If varTemp1 = varTemp2 Then IsEqual = True
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Tests whether the field in the recordset is required
' Inputs:    strFieldName - the name of the field in the recordset
' Returns:  True if updatable, False if not
'-----------------------------------------------------------------------------------------------

Function IsRequiredField(strFieldName)
        IsRequiredField = False
        If (rsSPaymentSQLQuery(strFieldName).Attributes And adFldIsNullable) = 0
Then
                IsRequiredField = True
        End If
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Tests whether the field in the recordset is updatable
' Inputs:    strFieldName - the name of the field in the recordset
' Returns:  True if updatable, False if not
'-----------------------------------------------------------------------------------------------

Function CanUpdateField(strFieldName)
        Dim intUpdatable
        intUpdatable = (adFldUpdatable Or adFldUnknownUpdatable)
        CanUpdateField = True
        If (rsSPaymentSQLQuery(strFieldName).Attributes And intUpdatable) = False
Then
                CanUpdateField = False
        End If
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Insert operation - updates a recordset field with a new value
'           during an insert operation.
' Inputs:    strFieldName - the name of the field in the recordset
' Returns: True if successful, False if not
'-----------------------------------------------------------------------------------------------

Function InsertField(strFieldName)
```

```
        InsertField = True
        If IsEmpty(Request(strFieldName)) Then Exit Function
        Select Case rsSPaymentSQLQuery(strFieldName).Type
                Case adBinary, adVarBinary, adLongVarBinary
                Case Else
                        If CanUpdateField(strFieldName) Then
                                If IsRequiredField(strFieldName) And
                                IsNull(RestoreNull(Request(strFieldName))) Then
                                        RaiseError errValueRequired, strFieldName
                                        InsertField = False
                                        Exit Function
                                End If
                                rsSPaymentSQLQuery(strFieldName) =
                                RestoreNull(Request(strFieldName))
                        End If
        End Select
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Update operation - updates a recordset field with a new value
' Inputs:   strFieldName - the name of the field in the recordset
' Returns:  True if successful, False if not
'-----------------------------------------------------------------------------------------------

Function UpdateField(strFieldName)
        UpdateField = True
        If IsEmpty(Request(strFieldName)) Then Exit Function
        Select Case rsSPaymentSQLQuery(strFieldName).Type
                Case adBinary, adVarBinary, adLongVarBinary
                Case Else
                        ' Only update if the value has changed
                        If Not
                        IsEqual(ConvertToString(rsSPaymentSQLQuery(strFieldName)),
                        RestoreNull(Request(strFieldName))) Then
                                If CanUpdateField(strFieldName) Then

                                        If IsRequiredField(strFieldName) And
                                        IsNull(RestoreNull(Request(strFieldName))) Then
                                                RaiseError errValueRequired, strFieldName
                                                UpdateField = False
                                                Exit Function
                                        End If
                                        rsSPaymentSQLQuery(strFieldName) =
                                        RestoreNull(Request(strFieldName))
                                Else
```

```
                    RaiseError errNotEditable, strFieldName
                    UpdateField = False
                End If
            End If
    End Select
End Function

'-----------------------------------------------------------------------------------------------
' Purpose:  Criteria handler for a field in the recordset. Determines correct delimiter
'  based on data type
' Inputs:   strFieldName - the name of the field in the recordset
'           avarLookup - lookup array - null if none
'-----------------------------------------------------------------------------------------------

Sub FilterField(ByVal strFieldName, avarLookup)
    Dim strFieldDelimiter
    Dim strDisplayValue
    Dim strValue
    Dim intRow
    strValue = Request(strFieldName)
    strDisplayValue = Request(strFieldName)

    ' If empty then exit right away
    If Request(strFieldName) = "" Then Exit Sub

    ' Concatenate the And boolean operator
    If strWhere <> "" Then strWhere = strWhere & " And"
    If strWhereDisplay <> "" Then strWhereDisplay = strWhereDisplay & " And"

    ' If lookup field, then use lookup value for display
    If Not IsNull(avarLookup) Then
        For intRow = 0 to UBound(avarLookup, 2)
            If CStr(avarLookup(0, intRow)) = Request(strFieldName) Then
                strDisplayValue = avarLookup(1, intRow)
                Exit For
            End If
        Next
    End If

    ' Set delimiter based on data type
    Select Case rsSPaymentSQLQuery(strFieldName).Type
        Case adBSTR, adChar, adWChar, adVarChar, adVarWChar
            strFieldDelimiter = "'"
        Case adLongVarChar, adLongVarWChar
            strFieldDelimiter = "'"
```

```
            Case adDate, adDBDate, adDBTimeStamp
                strFieldDelimiter = "#"
            Case Else
                strFieldDelimiter = ""
        End Select

        ' Modifies script level variables
        strWhere = strWhere & " " & PrepFilterItem(strFieldName, strValue,
        strFieldDelimiter)
        strWhereDisplay = strWhereDisplay & " " & PrepFilterItem(strFieldName,
        strDisplayValue, strFieldDelimiter)

End Sub

'-------------------------------------------------------------------------------------------------
' Purpose:  Constructs a name/value pair for a where clause
' Inputs:   strFieldName - the name of the field in the recordset
'           strCriteria - the criteria to use
'           strDelimiter - the proper delimiter to use
' Returns:  The name/value pair as a string
'-------------------------------------------------------------------------------------------------

Function PrepFilterItem(ByVal strFieldName, ByVal strCriteria, ByVal strDelimiter)
        Dim strOperator
        Dim intEndOfWord
        Dim strWord

        strCriteria = Trim(strCriteria)
        strOperator = "="
        strValue = strCriteria

        ' Get first word and look for operator
        intEndOfWord = InStr(strCriteria, " ")
        If intEndOfWord Then
            strWord = UCase(Left(strCriteria, intEndOfWord - 1))
            ' See if the word is an operator
            Select Case strWord
                Case "=", "<", ">", "<=", ">=", "<>", "LIKE"
                    strOperator = strWord
                    strValue = Trim(Mid(strCriteria, intEndOfWord + 1))
                Case "=<", "=>"
                    RaiseError errInvalidOperator, strFieldName
            End Select
        Else
            strWord = UCase(Left(strCriteria, 2))
```

```
            Select Case strWord
                Case "<=", ">=", "<>"
                    strOperator = strWord
                    strValue = Trim(Mid(strCriteria, 3))
                Case "=<", "=>"
                    RaiseError errInvalidOperator, strFieldName
                Case Else
                    strWord = UCase(Left(strCriteria, 1))
                    Select Case strWord
                        Case "=", "<", ">"
                            strOperator = strWord
                            strValue = Trim(Mid(strCriteria, 2))
                    End Select
            End Select
        End If

        ' Make sure LIKE is only used with strings
        If strOperator = "LIKE" and strDelimiter <> "'" Then
            RaiseError errInvalidOperatorUse, strFieldName
        End If

        ' Single Quote
        If Left(strValue, 1) = Chr(39) Then strValue = Mid(strValue, 2)
        If Right(strValue, 1) = Chr(39) Then strValue = Left(strValue, Len(strValue) - 1)

        ' Double Quote - just in case
        If Left(strValue, 1) = Chr(34) Then strValue = Mid(strValue, 2)
        If Right(strValue, 1) = Chr(34) Then strValue = Left(strValue, Len(strValue) - 1)

        ' Pound sign - dates
        If Left(strValue, 1) = Chr(35) Then strValue = Mid(strValue, 2)
        If Right(strValue, 1) = Chr(35) Then strValue = Left(strValue, Len(strValue) - 1)

        ' Check for leading wildcards
        If Left(strValue, 1) = "*" Or Left(strValue, 1) = "%" Then
            RaiseError errInvalidPrefix, strFieldName
        End If

        PrepFilterItem = "[" & strFieldName & "]" & " " & strOperator & " " &
strDelimiter & strValue & strDelimiter
End Function

'-----------------------------------------------------------------------------------------------
' Purpose: Display field involved in a database operation for feedback.
'          strFieldName - the name of the field in the recordset
```

```
'-----------------------------------------------------------------------------------------

Sub FeedbackField(strFieldLabel, strFieldName, avarLookup)
        Dim strBool
        Dim intRow
        Response.Write "<TR VALIGN=TOP>"
        Response.Write "<TD ALIGN=Left><FONT SIZE=-1><B>  " &
        strFieldLabel & "</B></FONT></TD>"
        Response.Write "<TD BGCOLOR=White WIDTH=100% ALIGN=Left><FONT
        SIZE=-1>"

        ' Test for lookup
        If Not IsNull(avarLookup) Then
                For intRow = 0 to UBound(avarLookup, 2)
                        If CStr(avarLookup(0, intRow)) = Request(strFieldName) Then
                                Response.Write Server.HTMLEncode(avarLookup(1,
                                intRow))
                                Exit For
                        End If
                Next
                Response.Write "</FONT></TD></TR>"
                Exit Sub
        End If

        ' Test for empty
        If Request(strFieldName) = "" Then
                Response.Write " "
                Response.Write "</FONT></TD></TR>"
                Exit Sub
        End If

        ' Test the data types and display appropriately
        Select Case rsSPaymentSQLQuery(strFieldName).Type
                Case adBoolean, adUnsignedTinyInt
                        strBool = ""
                        If Request(strFieldName) <> 0 Then
                                strBool = "True"
                        Else
                                strBool = "False"
                        End If
                        Response.Write strBool
                Case adBinary, adVarBinary, adLongVarBinary
                        Response.Write "[Binary]"
                Case adLongVarChar, adLongVarWChar
                        Response.Write Server.HTMLEncode(Request(strFieldName))
```

```
            Case Else
                If Not CanUpdateField(strFieldName) Then
                    Response.Write "[AutoNumber]"
                Else
                    Response.Write
                    Server.HTMLEncode(Request(strFieldName))
                End If
        End Select
        Response.Write "</FONT></TD></TR>"
End Sub
```

Action Handler for Payment Page (Intact Source Code is of 246 lines)

```
Select Case strDataAction

    Case "List View"

        Response.Redirect "SPaymentList.asp"

    Case "Cancel"

        Response.Redirect "SPaymentForm.asp?FormMode=Edit"

    Case "Filter"

        On Error Resume Next
        Session("rsSPaymentSQLQuery_Filter") = ""
        Session("rsSPaymentSQLQuery_FilterDisplay") = ""
        Session("rsSPaymentSQLQuery_Recordset").Filter = ""
        Response.Redirect "SPaymentForm.asp?FormMode=" & strDataAction

    Case "New"

        On Error Resume Next
        Session("rsSPaymentSQLQuery_Filter") = ""
        Session("rsSPaymentSQLQuery_FilterDisplay") = ""
        Session("rsSPaymentSQLQuery_Recordset").Filter = ""
        Response.Redirect "SPaymentForm.asp?FormMode=" & strDataAction

    Case "Find"

        Session("rsSPaymentSQLQuery_PageSize") = 1 'So we don't do standard
        page conversion
```

```
        Session("rsSPaymentSQLQuery_AbsolutePage") =
        CLng(Request("Bookmark"))
        Response.Redirect "SPaymentForm.asp"
        'Response.Redirect "SPaymentEdit2.asp"

Case "All Records"

        On Error Resume Next
        Session("rsSPaymentSQLQuery_Filter") = ""
        Session("rsSPaymentSQLQuery_FilterDisplay") = ""
        Session("rsSPaymentSQLQuery_Recordset").Filter = ""
        Session("rsSPaymentSQLQuery_AbsolutePage") = 1
        Response.Redirect "SPaymentForm.asp"

Case "Apply"

        On Error Resume Next

        ' Make sure we exit and re-process the form if session has timed out
        If IsEmpty(Session("rsSPaymentSQLQuery_Recordset")) Then
                Response.Redirect "SPaymentForm.asp?FormMode=Edit"
        End If

        Set rsSPaymentSQLQuery = Session("rsSPaymentSQLQuery_Recordset")

        strWhere = ""
        strWhereDisplay = ""
        FilterField "ContractorNumber", Null
        FilterField "Title", Null
        FilterField "PayDate", Null
        FilterField "Pay", Null
        FilterField "MadeBy", Null
        FilterField "ApprovedBy", Null
        FilterField "Status", Null
        FilterField "Note", Null

        ' Filter the recordset
        If strWhere <> "" Then
                Session("rsSPaymentSQLQuery_Filter") = strWhere
                Session("rsSPaymentSQLQuery_FilterDisplay") =
                strWhereDisplay
                Session("rsSPaymentSQLQuery_AbsolutePage") = 1
        Else
                Session("rsSPaymentSQLQuery_Filter") = ""
                Session("rsSPaymentSQLQuery_FilterDisplay") = ""
```

```
        End If

        ' Jump back to the form
        If Err.Number = 0 Then Response.Redirect "SPaymentForm.asp"

Case "Insert"

        On Error Resume Next

        ' Make sure we exit and re-process the form if session has timed out
        If IsEmpty(Session("rsSPaymentSQLQuery_Recordset")) Then
                Response.Redirect "SPaymentForm.asp?FormMode=Edit"
        End If

        Set rsSPaymentSQLQuery = Session("rsSPaymentSQLQuery_Recordset")
        rsSPaymentSQLQuery.AddNew

        Do
                If Not InsertField("ContractorNumber") Then Exit Do
                If Not InsertField("Title") Then Exit Do
                If Not InsertField("PayDate") Then Exit Do
                If Not InsertField("Pay") Then Exit Do
                If Not InsertField("MadeBy") Then Exit Do
                If Not InsertField("ApprovedBy") Then Exit Do
                If Not InsertField("Status") Then Exit Do
                If Not InsertField("Note") Then Exit Do

                rsSPaymentSQLQuery.Update
                Exit Do
        Loop

        If Err.Number <> 0 Then
                If rsSPaymentSQLQuery.EditMode Then
                rsSPaymentSQLQuery.CancelUpdate
        Else
                If IsEmpty(Session("rsSPaymentSQLQuery_AbsolutePage")) Or
                Session("rsSPaymentSQLQuery_AbsolutePage") = 0 Then
                Session("rsSPaymentSQLQuery_AbsolutePage") = 1
                End If
                ' Requery static cursor so inserted record is visible
                If rsSPaymentSQLQuery.CursorType = adOpenStatic Then
                rsSPaymentSQLQuery.Requery
                Session("rsSPaymentSQLQuery_Status") = "Record has been
                inserted"
        End If
```

```
Case "Update"

        On Error Resume Next

        ' Make sure we exit and re-process the form if session has timed out
        If IsEmpty(Session("rsSPaymentSQLQuery_Recordset")) Then
                Response.Redirect "SPaymentForm.asp?FormMode=Edit"
        End If

        Set rsSPaymentSQLQuery = Session("rsSPaymentSQLQuery_Recordset")
        If rsSPaymentSQLQuery.EOF and rsSPaymentSQLQuery.BOF Then
        Response.Redirect "SPaymentForm.asp"

        Do

                If Not UpdateField("ContractorNumber") Then Exit Do
                If Not UpdateField("Title") Then Exit Do
                If Not UpdateField("PayDate") Then Exit Do
                If Not UpdateField("Pay") Then Exit Do
                If Not UpdateField("MadeBy") Then Exit Do
                If Not UpdateField("ApprovedBy") Then Exit Do
                If Not UpdateField("Status") Then Exit Do
                If Not UpdateField("Note") Then Exit Do

                If rsSPaymentSQLQuery.EditMode Then
                rsSPaymentSQLQuery.Update
                Exit Do
        Loop

        If Err.Number <> 0 Then
                If rsSPaymentSQLQuery.EditMode Then
                rsSPaymentSQLQuery.CancelUpdate
        End If

Case "Delete"

        On Error Resume Next

        ' Make sure we exit and re-process the form if session has timed out
        If IsEmpty(Session("rsSPaymentSQLQuery_Recordset")) Then
                Response.Redirect "SPaymentForm.asp?FormMode=Edit"
        End If

        Set rsSPaymentSQLQuery = Session("rsSPaymentSQLQuery_Recordset")
```

```
                If rsSPaymentSQLQuery.EOF and rsSPaymentSQLQuery.BOF Then
                Response.Redirect "SPaymentForm.asp"

                rsSPaymentSQLQuery.Delete

                ' Proceed if no error
                If Err.Number = 0 Then
                        ' Requery static cursor so deleted record is removed
                        If rsSPaymentSQLQuery.CursorType = adOpenStatic Then
                        rsSPaymentSQLQuery.Requery

                        ' Move off deleted rec
                        rsSPaymentSQLQuery.MoveNext

                        ' If at EOF then jump back one and adjust AbsolutePage
                        If rsSPaymentSQLQuery.EOF Then
                                rsSPaymentSQLQuery.MovePrevious
                                Session("rsSPaymentSQLQuery_AbsolutePage") =
                                Session("rsSPaymentSQLQuery_AbsolutePage") - 1

                                If rsSPaymentSQLQuery.BOF And
                                rsSPaymentSQLQuery.EOF Then
                                rsSPaymentSQLQuery.Requery
                        End If
                End If

End Select
```

Error Handler

```
<%
If Err Then
        Select Case Err.Number
                Case -2147467259
                        strErrorAdditionalInfo = "  This may be caused by an attempt to
                        update a non-primary table in a view."
                Case Else
                        strErrorAdditionalInfo = ""
        End Select
%>

<HTML>
<HEAD>
<META NAME="GENERATOR" CONTENT="Microsoft Visual InterDev">
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
```

```
<META NAME="keywords" CONTENT=" Payment History --- For View Only Form">
<TITLE>Payment History --- For View Only Form</TITLE>
</HEAD>
<BASEFONT FACE="Arial, Helvetica, sans-serif">
<LINK REL=STYLESHEET HREF="./Stylesheets/Grid/Style2.css">
<BODY BACKGROUND="./Images/Grid/Background/Back2.jpg" BGCOLOR=White>
<TABLE WIDTH=100% CELLSPACING=0 CELLPADDING=0 BORDER=0>
<TR>
        <TH COLSPAN=2 NOWRAP ALIGN=Left BGCOLOR=Silver
        BACKGROUND="./Images/Grid/Navigation/Nav1.jpg">
        <FONT SIZE=6> Message: </FONT>
        </TH>
</TR>
<TR>
        <TD BGCOLOR=#FFFFCC COLSPAN=2>
        <FONT SIZE=3><B>
        <%
        Select Case strDataAction
                Case "Insert"
                        Response.Write("Unable to insert the record into SQLQuery.")
                Case "Update"
                        Response.Write("Unable to post the updated record to
                        SQLQuery.")
                Case "Delete"
                        Response.Write("Unable to delete the record from SQLQuery.")
                        End Select
        %>
        </B></FONT>
        </TD>
</TR>
</TABLE>

<TABLE WIDTH=100% CELLSPACING=1 CELLPADDING=2 BORDER=0>
<TR>
        <TD ALIGN=Left BGCOLOR=Silver><FONT SIZE=-
        1><B>  Item</B></FONT></TD>
        <TD WIDTH=100% ALIGN=Left BGCOLOR=Silver><FONT SIZE=-
        1><B>Description</B></FONT></TD>
</TR>
<TR>
        <TD><FONT SIZE=-1><B>  Source:</B></FONT></TD>
        <TD BGCOLOR=White><FONT SIZE=-1><%= Err.Source %></TD>
</TR>
<TR>
```

```
        <TD NOWRAP><FONT SIZE=-1><B>  Error
        Number:</B></FONT></TD>
        <TD BGCOLOR=White><FONT SIZE=-1><%= Err.Number
        %></FONT></TD>
</TR>
<TR>
        <TD><FONT SIZE=-1><B>  Description:</B></FONT></TD>
        <TD BGCOLOR=White><FONT SIZE=-1><%=
        Server.HTMLEncode(Err.Description & strErrorAdditionalInfo)
        %></FONT></TD>
</TR>
<TR>
        <TD COLSPAN=2><HR></TD>
</TR>
<TR>
        <TD>
        <% Response.Write "<FORM ACTION=""SPaymentForm.asp""
        METHOD=""POST"">" %>
        <INPUT TYPE="Hidden" NAME="FormMode" VALUE="Edit">
        <INPUT TYPE="SUBMIT" VALUE="Form View">
        </FORM>
        </TD>
        <TD>
        <FONT SIZE=-1>
        To return to the form view with the previously entered
        information intact, use your browsers &amp;quot;back&amp;quot; button
        </FONT>
        </TD>
</TR>
</TABLE>
</BODY>
</HTML>

<% Else %>
<%

Response.Redirect "SPaymentForm.asp"
%>
<%
End If
Set rsSPaymentSQLQuery = Nothing
%>
```