Web-Based Inventory Database Application

By

Garald Emerson Seigla

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

May 2002

Web-Based Inventory Database Application

by

Garald Emerson Seigla

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

Garald Emerson Seigla	Date
Robert Schlemmer, Faculty Project Advisor	Date
Lawrence G. Gilligan, Department Head	Date

Acknowledgements/Dedications

I acknowledge Steve Young, Director of Information Technology at the University of Cincinnati Clermont College for his allowing me to write this new business software package for the IT department. I was fortunate enough to be able to work on this during business hours. I have learned and grown more from this experience than I have from any other project I have ever done before. It has truly been a memorable experience. I also wish to thank Dr. Sam Geonetta and Professor Bob Schlemmer for their help and guidance. The layout and structure of this capstone track demonstrates their dedication to the accountability and success of IET students. So, from pupil to professor, "Thank You".

Over the past eight years I have faced many challenges in my quest for education. None of these challenges has been greater than that of juggling my personal, professional and academic responsibilities. I am the father of three beautiful girls ranging in age from eight months to five years of age. My children have sought attention and understanding in my responsibilities from my devoted wife of seven years. This journey has been long and difficult at times not only for me personally, but for my entire family. However, I would not trade my experiences for anything in the world. My journey has been extremely rewarding both academically and personally. Therefore, I dedicate this project to my wife Stacy and our three daughters Kayla, Kassidy and Kimberly for their support and understanding during this entire degree program. I also wish to acknowledge the dedication my father, Gary L. Seigla, instilled in me at a young age and the strength God has given me to see this through.

Table of Contents

Table of Contents

List of Figures

Abstract

The University of Cincinnati Clermont College had a DOS-based inventory database application that lacked efficiency, expandability and remote accessibility. While this system, known as Alpha 3, was dependable, it was unable to keep up with the growing needs of the Information Technology staff. To resolve the deficiencies of this application, I have developed a web-based inventory database application that assists the IT department in tracking vital component information. The back-end of the application is maintained on Windows 2000 Server and SQL Server 2000. At the same time, the front-end of the application is housed on a Windows 2000 Server machine running Internet Information Services 5.0. The front-end was developed using HTML, Active Server Pages and JavaScript. All users have the ability to search and view records in the database while only authenticated users have the ability to add, change and delete records. This is a fully functional application that globalizes access to vital component information for IT staff. This application simplifies the process of inputting and extracting information by means of a simple graphical user interface.

Web-Based Inventory Database Application

1. Statement of the Problem

The University of Cincinnati Clermont has used a DOS-based inventory system to track electronic components for the Information Technology (IT) Department for the past eight years. While this system, known as Alpha 3, has been dependable, it has been unable to keep up with the growing needs of the IT staff at UC Clermont. Over the past two years UC Clermont has added two new buildings. Due to this expansion, the facility has added two hundred PC's and additional network devices and components. UC Clermont has also developed an off-site location in Mt. Caramel for educational outreach programs that has a computer lab and various support and network devices.

Various restrictions make it necessary to upgrade from the current inventory system to a more robust expandable system:

- Flat file database system
- Only supports a single user environment
- Processing is slow
- Limited reporting functionality
- No Windows conventions (cut, copy, paste, etc.)
- Lacks adequate remote accessibility

Alpha 3 is a flat file database system that was used to develop the IT department's device inventory at UC Clermont. The entire database is stored in a single table that has the ability to be indexed on any field sequentially. Since the College has expanded and the IT staff has quadrupled, this one-dimensional database system has become unable to

maintain the level of efficiency required to keep an IT department on the cutting edge of information(11).

Although Alpha 3 is shared over a Local Area Network (LAN), it is inefficient. Due to the increase in staff and technical support within the department, there is a greater need for individuals to access the database at the same time. The system is unable to maintain multi-user connections to add, change and view device information. This restriction has become more and more a hindrance to the department.

Another deficiency related to accessibility is the application's slow processing of user requests. Since this is a flat file system, the application searches the database sequentially. This proves to be time-consuming and processor retentive, which is typical of a legacy system.

A major benefit of any database system is the ability to format data in a reporting style that is beneficial to the user. While Alpha 3 supports the ability to add, change and view inventory data, it does not provide a meaningful reporting function. Such a function is a benefit the Director of the IT department has requested as it becomes more and more important to view specialized reports that provide fractal information(11).

Another major drawback of the Alpha 3 system is its inability to apply general Windows conventions. Any user of this system is unable to cut, copy and paste information from one screen to another. This makes certain aspects of data entry redundant and time-consuming. Steve Young, Director of Information Technology, pointed out this drawback when he recently received and installed one-hundred and seventy-five new PCs. These new PCs all had the same system specifications that had to be reentered for every new machine added to the database(11). Because it was not

possible to recompile the .exe of Alpha 3, the IT department did not have the ability to make a simple programming change to the front-end of the application. This simple change would have made it possible to enter only unique information for every machine to add it to the database. This, in turn, would have significantly reduced the data entry process of the new PCs.

Finally, the Alpha 3 system lacks adequate remote accessibility. With the current system, all users must make system modification to their machines and map a drive to a location on a central application server to run the software. Due to the development of an off-site training location, it is extremely slow and cumbersome to maintain a connection to the Alpha 3 system. However, support issues necessitate the need for some form of accessibility to this type of information.

2. Flat File Databases versus Relational Databases

There are two general methods of storing data. One is a flat file database and the other is a relational database. Each method was reviewed for its ability to meet UC Clermont's needs. ZDNet Webopedia defines flat file database as "A relatively simple database system in which each database is contained in a single table. In contrast, relational database systems can use multiple tables to store information, and each table can have a different record format. Relational systems are more suitable for large applications, but flat databases are adequate for many small applications."(See Figure 1)(9) For large applications, the disadvantages far outweigh the advantages of a flat file database.

Invoices
Invoice Number
Date
Customer Name
Address
City
State
Zip Code
Telephone Number
Product Name
Quantity
Color
Price
Total

Figure 1. Flat File Database

Advantages of Flat File Databases:

- **Inexpensive** – flat file data is typically stored in text files. The only software required is a front-end to access the data.

- **Platform Independent** – text files are universally accepted by all server platforms, so there is no problem migrating a database from server to server.

- **Easy to Understand** – records in a flat file are stored in a straight line separated by a common delimiter.

Disadvantages of Flat File Databases:

- **Low Reliability and Integrity** – if flat files grow beyond what the server resources are prepared to handle, they are prone to corruption.

- **Low Security** – a flat file may be opened by anyone who knows where to look. There is no built-in security.

- **Limited Data Structuring** – programmers are not offered the flexibility to create relationships between data.

- **Difficult Integration with Other Programs** – once a flat file is created for use by one program, it becomes impossible to have another program use it. This is because succeeding programs must conform to the structure of the flat file.

An alternate system is the use of relational database management systems (RDBMS). Relational databases store data as related tables. (See Figure 2) This feature makes relational databases more powerful and efficient because how data is related or extracted from the database is clearly defined. One database can be spread across multiple tables. As a result, the same database can be viewed in many different ways.

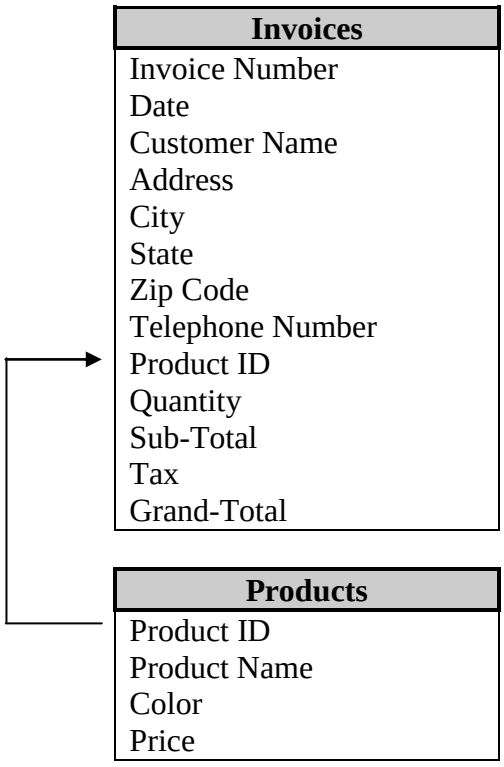


Figure 2. Relational Database

Advantages of Relational Databases:

- **Structured Query Language (SQL)** – this is a standardized query language for requesting information from a database. SQL allows for the creation of stored procedures. Stored procedures are reusable operations stored on the server in a compiled structure. Query results are much more efficient because there is no need to parse the queries.

- **Support for Large Amounts of Data** – due to their structure, relational databases are designed to handle larger amounts of data compared to flat files.

- **High Security and Reliability** – passwords and user levels are usually built-in features of relational databases. Relational databases also have a record locking feature that prevents data corruption if and when a second user accesses the same record.

- **Optimized Performance** – relational databases can take advantage of hardware platform architectures such as parallel processing or clustered environments.

- **Third Party Support** – many third party tools are available to users to simplify database management.

Disadvantages of Relational Databases:

- **Complex** – RDBMS require administrators to manage them. This requires a high level of proficiency in SQL, RDBMS and security.

- **Expensive** – Most relational databases require the users to purchase software and licenses. In addition, they require more server resources and the proficiency of a database administrator to maintain/manage the system.

3. Description of the Solution

After reviewing UC Clermont's infrastructure, Microsoft SQL Server was chosen as the RDBMS solution. This solution satisfied the first three restrictions that made it necessary to upgrade to a more robust expandable system. These restrictions were a flat file database, single user environment and slow data processing. SQL Server runs on Windows NT/2000, which provides multiprocessing[A], multitasking[B] and multithreading[C] functionality. Because of these operating system (OS) benefits, data processing and data retrieval are more efficient. This also allows multiple users to gain access to SQL Server by using their client computer system(10, pp. 37-38). Since UC Clermont is an NT/2000 shop, the implementation of this back-end solution was seamless.

The last three restrictions are addressed by the use of hypertext markup language (HTML), active server pages (ASP) and JavaScript for the front-end. These restrictions

are: limited reporting functionality, no Windows conventions and inadequate remote accessibility. Through the combination of HTML, ASP and JavaScript, I have created a dynamic user interface that allows for on-line reporting, Windows conventions and remote accessibility. This type of front-end provides a seamless portal to the back-end database through cutting edge technology.

3.1 User Profile

All Web-Based Inventory Database Application users have the necessary knowledge to interact with this application. All but the student assistants are IT professionals that deal with technology issues on a daily basis, and the student assistants are all from computer software or hardware programs at UC Clermont.

The web interface is developed from standard web programming techniques that are acceptable and easy to use. These techniques compliment the components of any standard web browser and build on popular windows conventions.

All members of the IT staff use this application on a daily basis. This is a seamless transition from what is currently used for the Help Call application. Due to experience with this application, users are able to identify familiar layouts and techniques necessary to add, change and search records in the database.

Some key interface design requirements build on standard interactive Internet components, such as: text boxes, command buttons and general navigation menus.

3.2 Design Protocols

3.2.1 Flowcharts

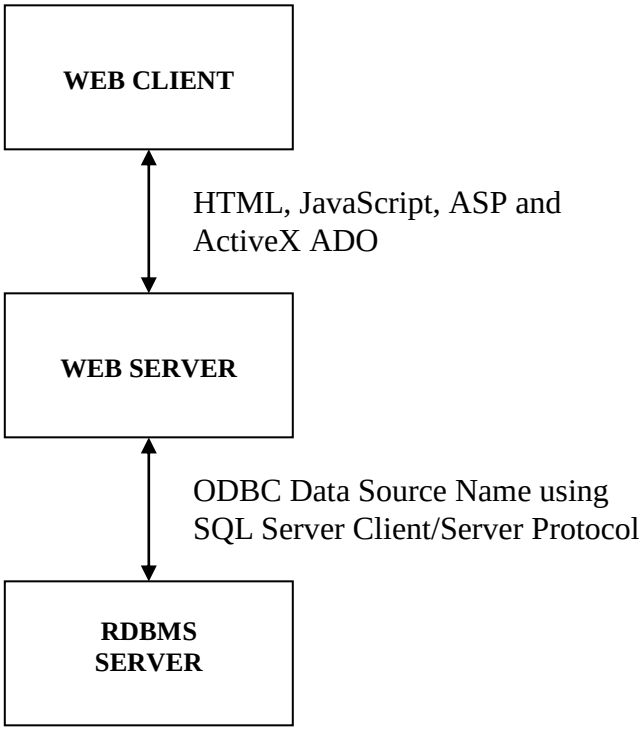


Figure 3. Back-End Flowchart

Figure 3 displays the back-end components and the communication protocols used between these components. This lays the foundation for data access and data storage for this application.

On the following page, figure 4 illustrates the dynamic layout of the front-end of the application. This shows how the user navigates through the application. This illustration also shows what authenticated users have access to compared to access for general users.

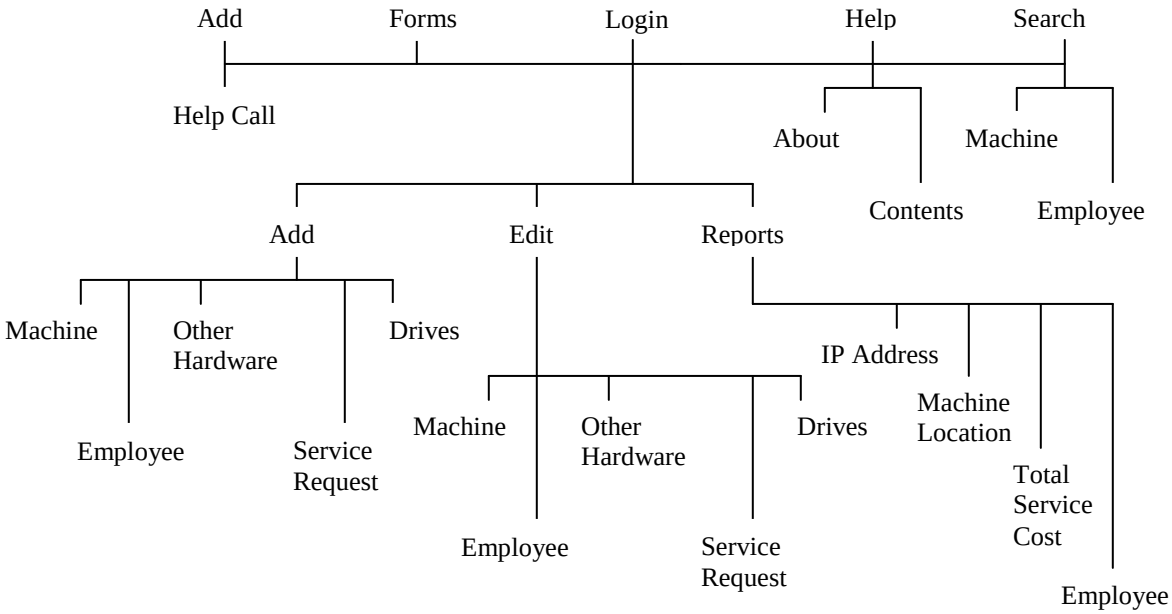


Figure 4. Front-End Flowchart

3.2.2 Server Specifications

See Appendix A.

3.2.3 Database Design

See Appendix B.

3.2.4 Interface Design and Navigation

Upon entering the application administrative users are required to login. As long as these users offer a valid username and password, use of the application is granted. Once in the application, all users are introduced to a general homepage stating the purpose of the application, general navigation techniques and application designs and logos. The application is navigated with pull-down menus. Users simply select one of seven menus and then click on the item in the menus that they are interested in accessing. These pull-down menus are available on all subsequent pages of the application.

3.2.5 Icons and Graphics

This Web-Based Inventory Database Application is representative of the University of Cincinnati colors and logos. All pages display the UC logo married to the Clermont tagline. Unique icons are used accordingly to further associate the difference in application screens for intended user interaction.

3.2.6 Color Scheme

The application is segregated into three basic modes of interaction: Add screens, Edit screens and View screens. These different modes are identified by their unique background colors. All application colors meet the guidelines for acceptable color use on university represented web pages. This unique blend of colors, logos and icons allow the user to build a visual association in connection with where they are and what they are doing in the application. See appendices C, D and E for examples.

3.2.7 Help System

This Web-Based Inventory Database Application offers a basic Help System. Users can access information from the general navigation bar that describes unique facets of the application. Users can find information on required fields specific to the type of mode in which they are interfacing. Users can also find information on specific form validation that is used throughout the application. Finally, users can find information on how to navigate the application using the general navigation bar at the top of each and every page.

4. Deliverables

1. A Web-Based Inventory Database Application
2. A back-end RDBMS that will facilitate the development of a dynamic front-end interface. This RDBMS will be developed on SQL Server 2000.
3. A dynamic front-end interface developed in HTML, ASP and JavaScript.
4. Authentication for administrative users of the database.
5. A general navigation bar on every page for easy navigation of the application.
6. Unique forms and reports requested by the client.
7. Ability for authenticated users to:
    - Add devices, drives, other hardware and employees into the database
    - View devices and their associative information
    - Update devices, drives, other hardware and employees
    - Search on multiple criteria for devices in the database
    - Link to internal and external resources within the application
    - Learn basic application mannerisms through the Help section

5. Designs and Development

5.1 Timeline

Senior Design I Autumn 2001

Weeks 1-5

- Met with Steve Young, Director of IT Department, to discuss project
- Gathered information on data storage and retrieval processes
- Defined input criteria for desired application
- Defined output products for given input criteria

Weeks 6-10

- Defined database structure
- Defined how objects of database structure were to interact
- Defined size and data types for elements of defined database objects
- Presented summary of information gathering process and proposed solution **12/16**

Senior Design II Winter 2002

Weeks 1-5

- Developed SQL Server Database Solution from defined criteria
- Developed Active Server Pages from input criteria for the application
- Developed HTML pages for required forms and reports
- Completed documentation

Weeks 6-10

- Developed JavaScript for client-side form validation
- Integrated background colors and graphics for application interface
- Developed general navigation menu for all application pages
- Presented working prototype **3/15**

Senior Design III Spring 2002

Weeks 1-5

- Rolled out prototype for testing
- Refined and reassessed application functionality
- Further developed Help section of application.
- Reviewed and refined preliminary documentation

Weeks 6-10

- Completed testing and functionality
- Finalized preliminary documentation and suggestions
- Presented final working project and submitted all documentation **5/30**

5.2 Budget

5.2.1 Hardware

Hardware	Quantity	Proposed	Price
Server	1	Dell PowerEdge 2500	$4,039.00
UPS	1	APC Back-UPS C350 – BK350	$ 89.99
		Total Hardware Cost:	$4,128.99

Figure 5. Hardware Requirements

5.2.2 Software

Software Description	Quantity	Price
SQL Server 2000 Standard Edition	1	$ 499.00
Backup Agent for MS SQL Server	1	$ 535.00
Norton AntiVirus Software	1	$ 49.99
Diskeeper 6.0 Server	1	$ 259.95
Total Software Cost:		$1,343.94

Figure 6. Software Requirements

Hardware and software costs total $5,472.93. The funding for this project came from administrative funds by virtue of the Dean of UC Clermont. The Maintenance Analyst/System Developer for UC Clermont did proprietary development of the application.

6. Proof of Design

This Web-based version of UC Clermont's inventory database established the transition from a DOS-based 16-bit application to a cutting edge, state of the art Internet application. This was achieved by developing a robust back-end database with the interactive capabilities for a web-based application. The back-end database was developed using SQL Server 2000 on a Windows 2000 Server machine. This solution addresses the requirements to develop a dynamic front-end interface and also offers expandability for future projects. This solution also integrates seamlessly into the current network infrastructure.

The front-end for this web-based inventory database application offers interactivity built on standard web programming techniques. This is achieved through the use of HTML, ASP and JavaScript. The combination of these three languages facilitates three major areas of development: 1) standard web look and feel, 2) database connectivity and manipulation and 3) user input validation.

HTML presents the required functionality to provide a standard web look and feel. An HTML document is an electronic file that contains familiar elements, such as hyperlinks, buttons, text boxes and option buttons that provide the necessary components to develop an interactive web application.

Combined with HTML, ASP grants this application the ability to connect to SQL Server and manipulate data in the database. It does this by using active data objects (ADO) and SQL to communicate with a predefined data source name on the web server. Refer to Appendix F for an example.

Finally, combined with HTML and ASP, JavaScript offers the ability to validate end user input via the client machine. This moves input validation from the server to the client for efficiency and decentralized processing. JavaScript is also used in this application to provide the general navigation bar at the top of all active server pages.

Included in the front-end is authentication for administrative users of this application. The Director of the IT department requested that only administrative users be able to add, change and delete items in the database through use of the front-end application. Using session variables that store valid usernames and passwords fulfilled this requirement. It was also requested that all users be able to view devices and their associative information, search the database on multiple criteria, link to internal and

external resources and learn basic application mannerisms. HTML and ASP pages were used to fulfill these requirements.

7. Conclusions

The IT department at UC Clermont was faced with having to upgrade a 16-bit, DOS-based inventory database application. The requirements for the new application centered on an expandable back-end, user-friendly front-end, off-site accessibility, authentication for valid users and customized reporting modules.

Due to the nature of the project, I was able to draw on skills I acquired from my academic experiences. I successfully set up and administered a SQL Server 2000 machine that houses the database for this project. I also successfully designed and implemented a fully functional front-end using HTML, ASP and JavaScript.

The front-end of the application was developed through solicited input from the end users. By carefully documenting operational requirements from interviews and final reports, the process for developing an acceptable interface was simple. I had a unique opportunity to design an application that was reflective of my knowledge and experience with web-based application development. Having only had purely cosmetic exposure to the languages necessary to complete this interface, I found the greatest challenge to be learning syntax that would translate into necessary characteristics of the application.

Due to defined methodologies in the front-end, remote access to this application is seamless. Access from outside of the college is no different than sitting at the campus interacting with the application. This was a key point in the project development.

Another key point in the project development was valid user authentication. Certain modules of the application are accessible to all employees of UC Clermont and

other modules are only available to IT staff. This made authentication a necessity in this project. I achieved this by use of session variables in active server pages.

Finally, it was necessary to provide customized reports for the IT director and UC Clermont administration. I achieved this by using SQL Server stored procedures and active server pages.

This project meets and exceeds deliverables set in Senior Design II. I have successfully designed and implemented a fully functional web-based inventory database application that is currently in production by the IT department at UC Clermont.

8. Recommendations

Overall I feel this is a solid application. I spent a lot of time fine tuning the application for efficiency and ease of use. Because of complications I faced early on with this project, I found it necessary to define custom attributes for the entire application. The design time controls in Visual Studio 6.0 were not stable enough to use drag and drop programming techniques in this development suite. Therefore, I resolved this by simply going back to the basics and hand coding all these features. This proved to be very time consuming and frustrating at times. My limited knowledge of active server pages and JavaScript has grown to an intermediate level of experience with these languages.

I would not recommend changing any of the basic functionality of the application in the future. The basic design is simple and easy to follow. There may be consideration to expand on this functionality and add more dynamics behind the scenes.

The back-end has been developed on a platform that is adaptable and most of all expandable for the next five to ten years. Depending on the direction of UC Clermont,

the back-end is robust enough to handle enormous growth in the field of real-time data access. This was an intricate piece of the development of this application. This flexibility also offers a lot of room for development of applications such as this one.

As far as recommendations for improvements to this application, I would recommend reviewing and restructuring access security, converting the entire application to ASP.NET and developing a maintenance plan for the back-end SQL Server.

I was able to implement an acceptable level of security for the application, but I feel my security techniques need to be reviewed and restructured to better define security policies and procedures for future projects. In other words, I would re-examine what network accounts are used to communicate between the IIS web server and the SQL server, the permissions of these accounts and how secure are the accounts in the embedded ASP coding.

I would also recommend converting the current application to embrace the new Microsoft .NET technology. Due to preliminary research, I feel that this technology is far superior to the current technology used to develop this application. For example, end user validation can be achieved through user controls in ASP. So, what took thousands of lines of JavaScript coding using embedded files can now be produced with a couple hundred lines of code embedded in active server pages. Also, from initial exposure, the design time controls seem to work seamlessly in this development environment. This makes designing applications such as this one much easier and more reliable. These are only a few benefits of converting to the .NET technology.

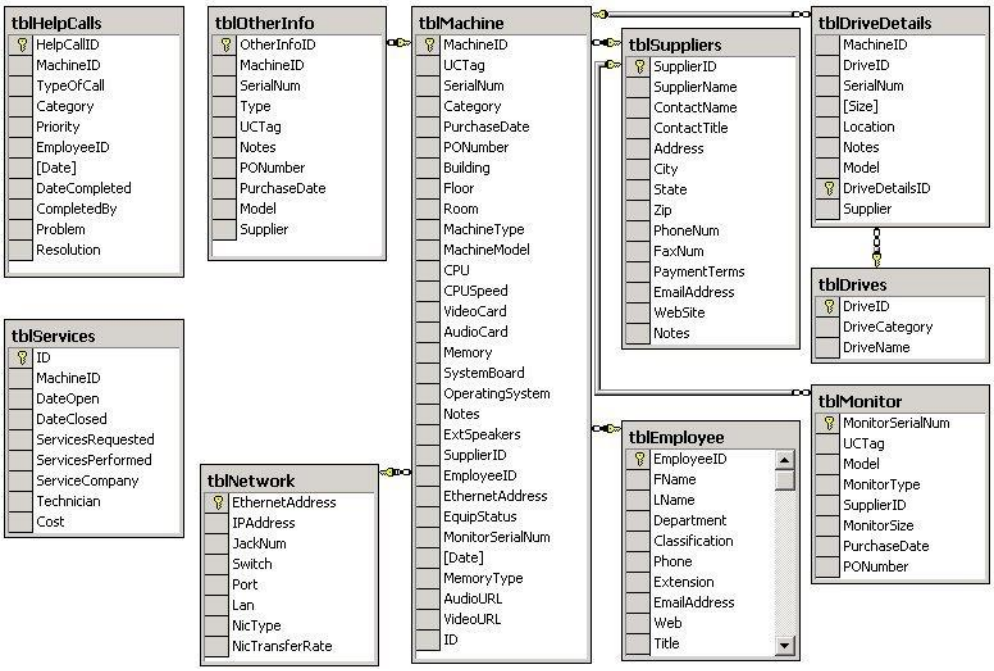
Finally, I would also recommend developing an effective maintenance plan for the database server. By developing maintenance policies and procedures now we can lay

the foundation for future projects to follow the same guidelines. This is important for replication and disaster recovery. Setting such guidelines will take a skilled DBA with experience in this type of administration to create an effective plan.

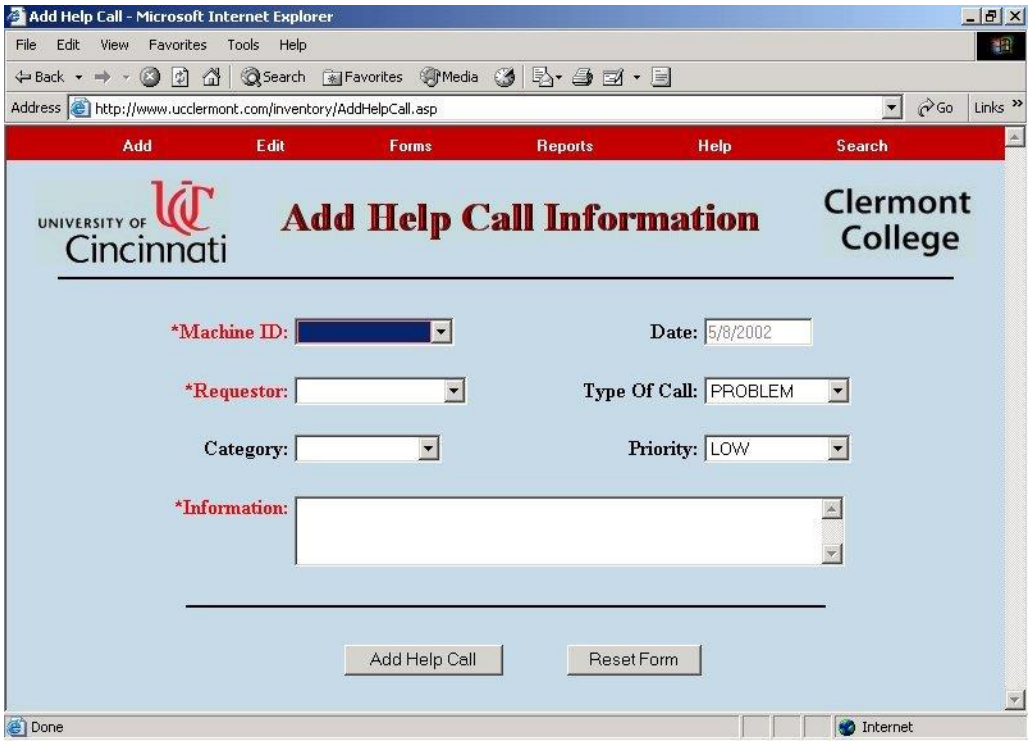
Appendix A.
Server Specifications

Catalog Number:	04 04
PowerEdge 2500:	PowerEdge 2500SC Intel Pentium III 1.0GHZ w/256K Cache 2510BSD - [461-4303]
Additional Processors:	Single Processor Only 1P - [311-1193]
Memory:	512MB SDRAM, 133 MHz, 4X128MB DIMMs 512M4D - [311-6568]
Keyboard:	Standard Windows Keyboard, Gray S - [310-4100]
Monitor:	No Monitor Option N - [320-0058]
1st Hard Drive:	18GB,U160M,SCSI,1 in,10K HD 18GB10 - [340-1937]
Primary Controller:	PERC3-DI, 128MB, 2 Internal Channels - Embedded RAID ROMB128 - [340-2485]
Diskette Drive:	3.5 in, 1.44MB Floppy Drive FD - [340-2557]
Operating System:	NO Factory Installed Operating System NOOS/O - [420-4106]
Mouse:	Logitec System Mouse, Gray LDN - [310-3776]
Tape Backup Unit:	$200 OFF! PV100T, DDS4, 20/40G, TBU, NC, Internal DD4BSDP - [461-2182]
CD-ROM/DVD-ROM:	24X, IDE CD-ROM CD24X - [313-8993]
Hard Drive Backplane:	1X6 Hot-Pluggable HDD Backplane 1X6BKPL - [311-6578]
Documentation:	Electronic Documentation for PowerEdge 2500 EDOCS - [310-1989]
2nd Hard Drive:	18GB,U160M,SCSI,1 in,10K HD 18GB10 - [340-1937]
Tape Backup Software:	CA Arcserve Standard PSCASTD - [420-2837]
Hard Drive Configuration:	Drives attached to PERC3-DI, RAID 5 - Min. of 3 drives required MR5N - [340-2578]
Chassis Orientation:	Tower TOWER - [310-0841]
Hardware Support Services:	3Yrs Same Day 4Hr Response Parts + Onsite Labor (M-F 8am-6pm) U3Y5X10 - [900-6160] [900-6162]
Installation Services:	No Installation NOINSTL - [900-9997]
3rd Hard Drive:	18GB,U160M,SCSI,1 in,10K HD 18GB10 - [340-1937]
Power Supply Kit:	Redundant Power Supply 2+1 REDPWR - [310-0843]
Networking Accessories:	10/100 PCI Combo Card PCICC - [026412]
Networking Accessories:	Category 5 RJ-45 cable (25 ft.) RJ4525 - [506183]

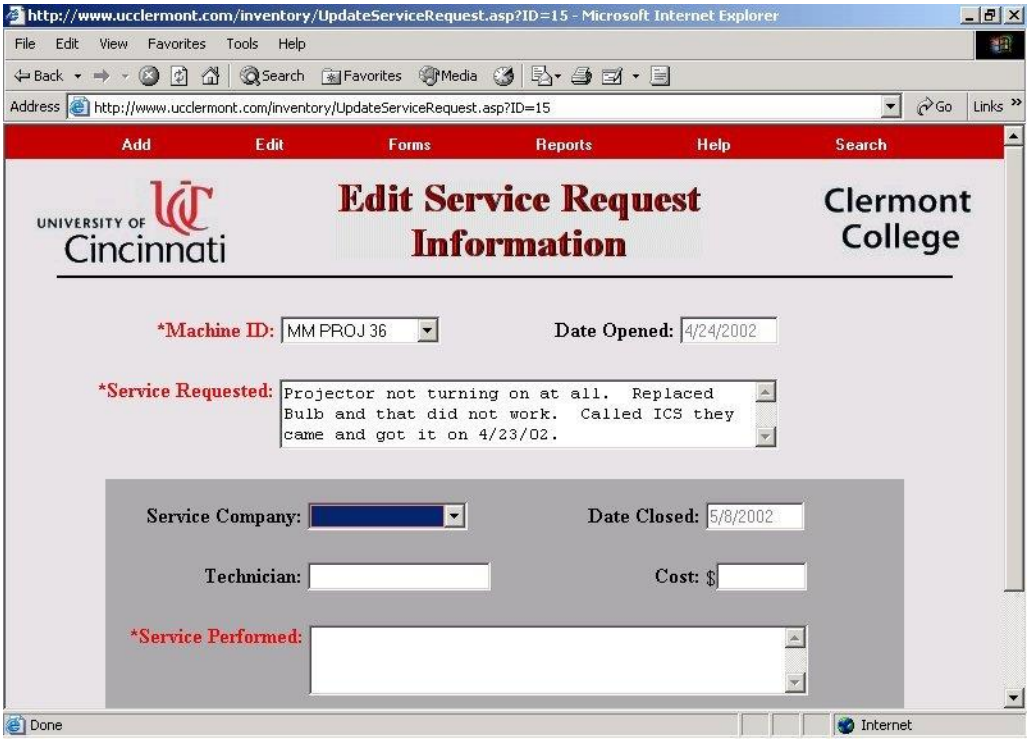
Appendix B.
Database Design
Entity-Relationship (ER) Diagram



Appendix C.
Screen Shot of Add Screen



Appendix D.
Screen Shot of Edit Screen



Appendix E.
Screen Shot of View Screen

Appendix F.
Sample Code
ASP, ADO and Javascript

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
<!--#include file ="login.asp"-->
<!--#include file ="menu.asp"-->
<!--#include file ="AddServiceRequestValidation.asp"-->
</HEAD>
<%
'Begin code to add a service request to the database
If Request.ServerVariables("REQUEST_METHOD") = "POST" Then
        %><br><br><%
        Dim SQLService

        'Define SQL to communicate with Database
        SQLService = "Select MachineID, DateOpen, ServicesRequested,
        ServicesPerformed from tblServices"

        'Instantiate ADO Connection and Recordset objects
        Set conn =  Server.CreateObject("ADODB.Connection")
        Set rs = Server.CreateObject("ADODB.Recordset")

        'Open connection via data source name and populate recordset
        conn.Open "dsn=<data source name>;uid=<ID>;pwd=<Password>"
        rs.Open SQLService, conn, 2, 3

        'Add record to tblServices
        rs.AddNew
        rs.Fields("MachineID")=Request("cboMachineID")
        rs.Fields("DateOpen")=Date()
        rs.Fields("ServicesRequested")=Request("txtServiceRequested")
        rs.Fields("ServicesPerformed")=""
        rs.Update

        Response.Write "<H2 align=center>Your service request has been posted to the
        database.</H2><br><br>"

        'Close connection and set objects equal to Nothing
        rs.Close
        conn.Close
        Set rs=Nothing
```

```
        Set conn=Nothing
        Response.End
End If

'Begin code for add service request page
'Declare variables for connection information
Dim SQL
Dim rst
Dim dbconn

'Define SQL to communicate with Database
SQL = "SELECT MachineID FROM tblMachine"

'Instantiate ADO Connection and Recordset objects
Set dbconn=server.CreateObject("ADODB.CONNECTION")
dbconn.Open " dsn=<data source name>;uid=<ID>;pwd=<Password>"

'Populate recordset with SQL parameters
Set rst=dbconn.Execute(SQL)

%>
<BODY bgcolor="C5DBE0"><br><br>
<Table align=center width=95% border=0>
        <TR>
                <TD width=23% valign=middle align=left><IMG
SRC="http://www.clc.uc.edu/inventory/images/uclogoblue.jpg" ALIGN=Middle
ALT="UC Logo"></TD>
                <TD width=52% valign=middle align=center><IMG
SRC="http://www.clc.uc.edu/inventory/images/AddServiceRequestInformation.gif"
ALIGN=Middle ALT="Add Service Request Information"></TD>
                <TD width=20% valign=middle align=right><IMG
SRC="http://www.clc.uc.edu/inventory/images/ccblue.jpg" ALIGN=Middle
ALT="Clermont College"></TD>
        </TR>
</Table>

<HR style="HEIGHT: 2px; WIDTH: 700px" color=black>

<form action="AddServiceRequest.asp" method="post" name="AddServiceForm">

<Table align=center width=75% border=0>
        <TR>
                <TD width=15% align=right><b><Font color=red>*Machine
ID:</Font></b><br><br></TD>
```

```
                <TD width=20%><SELECT name="cboMachineID" style="width:
125px" onfocus="document.all.cboMachineID.style.background='#FF8B8B';"
onblur="document.all.cboMachineID.style.background='#FFFFFF'">
                                <Option value=""></Option>
                                <%do while not rst.EOF
                                        If strMachineID = rst("MachineID") Then%>
                                                <Option Selected
value="<%=strMachineID%>"><%=strMachineID%></Option>
                                                <%Else%><OPTION
value="<%=rst("MachineID")%>"><%=rst("MachineID")%></Option>
                                                <%End If%>
                                                <%rst.movenext
                                loop%></SELECT><br><br></TD>
                <TD width=20% align=right><b>Date:</b><br><br></TD>
                <TD width=20%><input type="text" name="txtDate" size=9 value="<%
=Date()%>" disabled="true"><br><br></TD>
        </TR>
</Table>

<Table align=center width=85% border=0>
        <TR valign=top>
                <TD width=20% align=right><b><Font color=red>*Service
Requested:</Font></b></TD>
                <TD width=66% align=left><TEXTAREA Name=txtServiceRequested
style="WIDTH: 390px; HEIGHT: 55px"
onfocus="document.all.txtServiceRequested.style.background='#FF8B8B';"
onblur="document.all.txtServiceRequested.style.background='#FFFFFF'">
</TEXTAREA></TD>
        </TR>
</Table><br>

<HR style="HEIGHT: 2px; WIDTH: 500px" color=black><br>

<Table align=center width=50% border=0>
        <TR align=center>
                <TD width=25%><input type="button" value="Add Service Request"
onClick="validateForm(this)" size=16></TD>
                <TD width=25%><input type="reset" value="Reset Form"
name="cmdReset"></TD>
        </TR>
</Table>
<%

'Close all connections and recordsets
rst.close
dbconn.close
```

```
'Set ADO variables to nothing
Set rst=Nothing
Set dbconn=Nothing
%>

</BODY>
</HTML>

<SCRIPT LANGUAGE="JavaScript">
        <!-- document.AddServiceForm.cboMachineID.focus(); -->
</SCRIPT>
```

Notes

A. Multitasking – The ability to execute more than one task at the same time, a task being a program. There are two basic types of multitasking: *preemptive* and *cooperative.* In preemptive multitasking, the operating system parcels out CPU *time slices* to each program. In cooperative multitasking, each program can control the CPU for as long as it needs it. If a program is not using the CPU, however, it can allow another program to use it temporarily.(9)

B. Multiprocessing – A computer system's ability to support more than one process (program) at the same time. Multiprocessing operating systems enable several programs to run concurrently.(9)

C. Multithreading – The ability of an operating system to execute different parts of a program, called *threads,* simultaneously.(9)

Sources

1. Ashenfelter, John. "Database Design For the Web". http://www.webreview.com/1999/03_26/developers/03_26_99_1.shtml. March 26, 1999.

2. Atkinson, James and Scott Mitchell. *Teach Yourself Active Server Pages 3.0*. Indiana: Sams Publishing, 2000.

3. "Data Modeling 301: Normalization". http://www.4guysfromrolla.com/webtech/042699-1.shtml

4. Dietrick, Brian. Programmer, Dow Corning Precision Lens. Personal Interview. October, 2001.

5. Go, Peter. "Flat Files vs. Relational Databases". http://www.icthus.net/cgi-city/articles/adding.a.database.2.shtml

6. Meade, John and John Sudds. Chp.6, "*Developing Web Applications*", Chp.7, "*Data Access and Transactions*". Washington: Microsoft Press, 2000.

7. Shepker, Mathew. *Teach Yourself Microsoft SQL Server*. Indiana: Sams Publishing, 2000.

8. "SQL Database Normalization Rules" http://msdn.microsoft.com/library/default.asp?url=/library/en-us/eqbol/eqintbol_008c.asp

9. Webopedia - An online dictionary and search engine for computer and Internet technology. http://www.webopedia.com/

10. Wynkoop, Stephen. *Using Microsoft SQL Server 7.0*. Indiana: Que, 1999.

11. Young, Steve. Director of Information Systems Department, Clermont College. Personal Interview. October, 2001.