

# **Information Management System for the OCAS Professional Practice and Career Placement Office**

By

Joseph Bartels  
Doug Troxell

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  
March 2001

# Information Management System for the OCAS Professional Practice and Career Placement Office

By

Joe Bartels  
Doug Troxell

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

© Copyright 2001 Joe Bartels and Doug Troxell

The author grants to the Information Engineering Technology Program permission to reproduce and distribute copies of this document in whole or in part.

---

Author: Joe Bartels

---

Date

---

Author: Doug Troxell

---

Date

---

Faculty Project Advisor

---

Date

---

Department Head

---

Date

# Table of Contents

<b>Section</b>	<b>Page</b>
Table of Contents	i
List of Figures	ii
Abstract	iii
1. Statement of the Problem	1
1.1 Definition of the Need	1
2. Review of the Literature	2
3. Description of the Solution	3
3.1 User Profiles	4
3.2 Design Protocols	4
3.2.1 Company Interface	4
3.2.2 Staff Interface	6
3.2.3 Student Interface	7
3.2.4 Database Design	8
4. Objectives of the Project	15
5. Design and Development	15
6. Proof of Design	15
6.1 Student Interface	15
6.2 Staff Interface	20
6.3 Data Management System	32
7. Conclusions and Recommendations	33
Appendix A - Timeline	34
Appendix B - Budget	35
Appendix C - Active Server Pages Code	36
Appendix D - Visual Basic Code	49
Appendix E – Stored Procedures	111
Works Cited	130

## List of Figures

Figure 1. Design Requirements for Company Interface.	5
Figure 2. Sample of Company Interface.	5
Figure 3. Design Requirements for Staff Interface.	6
Figure 4. Staff Interface Sample.	6
Figure 5. Design requirements for Student Interface.	7
Figure 6. Student Interface Sample.	7
Figure 7. tblStudents Design.	9
Figure 8. tblEmployers Design.	10
Figure 9. tblJobDetails Design.	10
Figure 10. tblInterviewDetails Design.	10
Figure 11. tblQuarters Design.	11
Figure 12. tblQuarterDetails Design.	11
Figure 13. tblCoopAdv Design.	11
Figure 14. tblMajors Design.	11
Figure 15. Table Relationships.	12
Figure 16. User Roles.	13
Figure 17. Student Interface Login Page.	16
Figure 18. Student Login Feedback.	17
Figure 19. Student Personal Information Page.	18
Figure 20. Password Change Form.	19
Figure 21. Update Degree Plan or Resume.	20
Figure 22. Staff Welcome Page.	21
Figure 23. Staff Login Screen.	22
Figure 24. Create New Account.	23
Figure 25. Delete Account.	24
Figure 26. Change Password.	25
Figure 27. Add or Update Advisor Information.	26
Figure 28. Add or Update Company (Employer) Information.	27
Figure 29. Add or Update Job Detail Information.	28
Figure 30. Add or Update Major Information.	29
Figure 31. Add or Update Quarter Information.	30
Figure 32. Add or Update Student's Quarter Status Information.	31
Figure 33. View Student Degree Plan or Resume.	32

## **Abstract**

The Information Management System is a multi-faceted project that incorporated data management and user interface design. The purpose of the project was to build a custom software solution for the OCAS Professional Practice and Career Placement Office. The solution had to include data storage and user interaction with the data. SQL Server 7.0 was used as the data management back-end with user interfaces implemented in Active Server Pages and Visual Basic 6.0.

## **Abstract**

The Information Management System is a multi faceted project that incorporated data management and user interface design. The purpose of the project was to build a custom software solution for the OCAS Professional Practice and Career Placement Office. The solution had to include data storage and user interaction with the data. SQL Server 7.0 was used as the data management back-end with user interfaces implemented in Active Server Pages and Visual Basic 6.0.

Doug Troxell  
Joe Bartels

# **Information Management System for the OCAS Professional Practice and Career Placement Office**

## **1. Statement of the Problem**

The increasing enrollment and lack of sufficient staffing has caused the Professional Practice and Career Placement Office (PPCPO) at the Ohio College of Applied Science (OCAS) to look for more efficient processes to accomplish its mission. The main shortcoming in the workflow process of the PPCPO is the outdated data management software they use. The staff needs a more flexible and customizable software solution.

### **1.1 Definition of the Need**

The software used by the OCAS PPCPO is a proprietary package and is not customizable. Under the current system, the staff is required to manually update an average of 950 records per academic quarter. Queries and reports generated by the data management software do not function properly and cannot be sorted correctly because of the Year 2000 bug. To combat the deficiencies in the tools they use, the staff has developed many non-standard operations to get the results from the software. This has created a situation where training of new staff has become a daunting task.

Aside from the problems the software causes for the PPCPO staff, it is very troublesome for students to use. The learning curve for the software is very steep causing the time commitment for the student just to learn to use it to be unacceptable. There have been numerous problems reported with the floppy disks the software is distributed on. If the disk becomes corrupted, the student may lose hours of work because the backup

utility supplied with the software is unreliable and difficult to use. They are also unable to print their résumé and degree plan without help from the PPCPO.

## **2. Review of the Literature**

Several possible solutions would satisfy the needs of the PPCPO. The system used now was created by Academic Software© and they have developed several upgrades to the version in place. These upgrades have been evaluated by the staff and do not meet their requirements (17). The cost of the upgrades and lack of functionality make them unfeasible options. Other commercially available software products including Gooney and Metaphase are cost and hardware requirement inhibitive (1, pp 67-70).

The functional specifications used to determine the best possible solution and how to implement it were divided into four categories: 1) RDBMS choice, 2) RDBMS implementation, 3) GUI design and 4) GUI programming.

The RDBMS was chosen using the following criteria: 1) Purchase Cost, 2) Scalability, 3) Maintenance Cost, 4) Security and 5) Reliability. The clear choice for the RDBMS was Microsoft® SQL Server™. SQL Server offered the lowest purchase price because of the Microsoft Enterprise Licensing Agreement with the University of Cincinnati (agreement can be found at <http://mscontract.uc.edu/>). SQL Server fits into the existing network and operating system directives already in place in the PPCPO. The maintenance costs of SQL Server 7.0 will be much lower than any of the other RDBMS considered because of the staff's familiarity with Microsoft products and the interoperability of Microsoft Office products with SQL Server 7.0. Security is a major issue for any organization that stored sensitive data. SQL Server 7.0 allows for a wide and comprehensive security plan. Security was implemented using SQL accounts, stored

procedures and views. Reliability as a factor in a decision is hard to quantify, but SQL Server has a good reputation in the professional world for handling the amount of traffic the PPCPO will create.

The term "User Interface" refers to the methods and devices that are used to accommodate interaction between machines and the human beings who use them (users). User interfaces can take on many forms, but always accomplish two fundamental tasks: communicating information from the machine to the user, and communicating information from the user to the machine. Visual Basic 6.0 and Active Server Pages were chosen as the development tools for the GUIs needed for this project. This project did not require portability, so Visual Basic 6.0 was a natural extension of the SQL Server 7.0 development environment. Visual Basic 6.0 has built in tools to help in designing interfaces to SQL Server databases that made the development easier and quicker than if another programming package were used. Active Server Pages are platform independent and do not require the user to be sitting at any specific machine configuration. This allowed the flexibility and functionality that the students needed. Active Server Pages and Visual Basic 6.0 allow a great amount flexibility and functionality in design choices.

### **3. Description of the Solution**

The building of a data management system from ground level up with Microsoft compatible technologies is the best solution based on the specific needs. To fulfill the identified requirements, the solution has three distinct components: 1) Relational Database Management System (RDBMS) comprised of a custom built SQL Server 7.0 solution 2) User Interface Solutions comprised of a database

management software built in Visual Basic 6.0 and user interaction capabilities built with Active Server Pages 3) Web Capabilities consisting of full user interaction capabilities for students and search and query capabilities for potential employers. The solution will be cost effective to develop and maintain. The desired functionality will be identified and implemented. The highest degree of practical security will be implemented at every point of contention. The building of a new Management System makes the most sense because it allows the PPCPO to own the underlying structure and to control the development elements with the lowest possible cost for development and future expansion.

### **3.1 User Profiles**

There are three distinct groups of users who will interact with this software solution. These groups are the staff of the Professional Practices Office, the students required to co-op in order to fulfill their degree and potential employers looking to hire students from OCAS. The staff of the Professional Practices Office is expected to be familiar with Windows, Microsoft Word, Microsoft Excel, and how these Microsoft technologies can interact with each other. The students are expected to be familiar with Windows, Microsoft Word, Microsoft Excel and Web Browsers. The potential employers are expected to be familiar with Web Browsers.

### **3.2 Design Protocols**

This project consists of three independent user interfaces; 1) staff interface 2) employer interface 3) student interface.

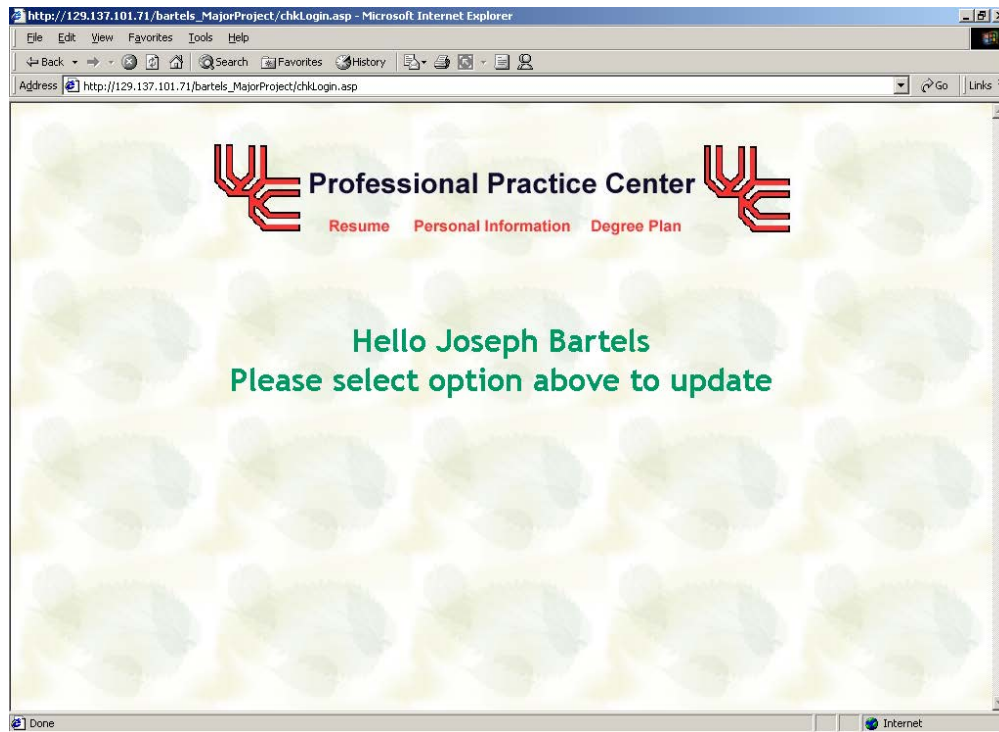
#### **3.2.1 Company Interface**

The company interface is designed for ease of navigation and user comfort as the main emphasis. The interface will provide an easy to use search facility for potential employers to find matches for their particular job needs. The main navigation tool for the company interface is an image map. The user is able to navigate to every destination and

return to the main page in two mouse clicks. Help is available through a link to a help page.

<b>Font</b>	<b>Headings</b> <b>H1</b> <b>Sub – Headings</b> <b>H2</b> <b>Text</b> <b>Normal</b> <b>Color</b> <b>Forest Green</b>
<b>Background</b>	<b>See Example</b>
<b>Menus</b>	<b>Image Map</b> <b>Resumés – Degree Plans – Help</b> <b>See Example</b>
<b>Icons</b>	<b>None</b>
<b>Development Tool</b>	<b>Microsoft Visual Interdev 6.0. Microsoft Front Page 2000. Paint Shop Pro 6.0. NotePad.</b>

**Figure 1. Design Requirements for Company Interface.**



**Figure 2. Sample of Company Interface.**

### 3.2.2\_Staff Interface

The staff interface is designed with ease of use and repeatability of common tasks as the main emphasis. The interface will provide robust data manipulation functionality along with client side error checking to ensure the integrity of the data. The interface will provide feedback to the user on the status of the program. The main navigation tool for the staff interface is drop down menus. Help is available from a drop down menu and tool tips.

<b>Font</b>	<b>Headings</b> <b>Sub – Headings</b> <b>Text</b> <b>Color</b>	<b>Sans Serif 18 pt.</b> <b>Sans Serif 14 pt.</b> <b>Sans Serif 10 pt.</b> <b>White</b>
<b>Background</b>	<b>Blue</b>	
<b>Menus</b>	<b>Main Navigation Process</b>	
<b>Icons</b>	<b>Standard Microsoft Icons</b>	
<b>Development Tools</b>	<b>Microsoft Visual Basic 6.0. Paint Shop Pro 6.0</b> <b>Icon Shop 2.3.</b>	

**Figure 3. Design Requirements for Staff Interface.**

OCAS CoOperative Education Office - [Create New SQL Server Account]

Accounts Database Maintenance

Solutions

Select Group for new Account  
Students

Enter User ID

Enter Password  
password

Create Account Dismiss

Status Area

Database Provider: Microsoft SQL Server  
Database Name: Coop\_Production1  
The Students group provides the necessary permissions for student users of this software.

**Figure 4. Staff Interface Sample 1.**

### 3.2.3 Student Interface

The student interface is designed for ease of navigation and user comfort as the main emphasis. The interface will provide easy to use data input and retrieval capabilities along with client side error checking to ensure the integrity of the data. The main navigation tool for the company interface is an image map. The user is able to navigate to every destination and return to the main page in two mouse clicks. Help is available in the form of mouse-overs and a link to a help page.

<b>Font</b>	<b>Headings</b> <b>Sub – Headings</b> <b>Text</b> <b>Color</b>	<b>H1</b> <b>H2</b> <b>Normal</b> <b>Forest Green</b>
<b>Background</b>	<b>See Example</b>	
<b>Menus</b>	<b>Image Map</b> <b>Resumés – Degree Plans – Help</b> <b>See Example</b>	
<b>Icons</b>	<b>None</b>	
<b>Development Tool</b>	<b>Microsoft Visual Interdev 6.0. Microsoft Front Page 2000. Paint Shop Pro 6.0. NotePad.</b>	

**Figure 5. Design Requirements for Student Interface.**

The screenshot shows a web browser window with the URL `http://129.137.101.71/bartels_MajorProject/NewStudent.htm`. The page features the logo of the Professional Practice Center and a form titled "Personal Information". The form includes the following fields:

- Social Security #
- First Name
- Middle Name
- Last Name
- Email Address
- Local Information: Street, City, State, Zip, Phone #
- Permanent Information: Street, City, State, Zip, Phone #
- A checkbox labeled "Same as local" next to the Permanent Information section.
- A "Click to Submit" button at the bottom.

**Figure 6. Student Interface Sample 1.**

### **3.2.4 Database Design**

The database will be designed and implemented using the rules of normalization. Normalization is a process of evaluating table structure and reorganizing them as needed to produce a set of stable, well structured relations. The three rules of normalization are: 1) the table does not contain any column that represents multiple values (atomic), 2) each row in the table must be uniquely identified (primary key) and 3) the table contains no redundant non-key information that relies on non-key information in another table (no redundant data).

<b>tblStudents</b>		
<b>Field Name</b>	<b>Data Type</b>	<b>Mandatory</b>
StudID	nchar(9) (PK)	Yes
StudLastName	nvarchar(25)	Yes
StudFirstName	nvarchar(20)	Yes
StudMidName	nvarchar(20)	No
StudSAStreet	nvarchar(25)	Yes
StudSACity	nvarchar(20)	Yes
StudSAState	nchar(2)	Yes
StudSAZip	nvarchar(9)	Yes
StudSAPhone	nvarchar(14)	No
StudPermStreet	nvarchar(25)	Yes
StudPermCity	nvarchar(20)	Yes
StudPermState	nchar(2)	Yes
StudPermZip	nvarchar(9)	Yes
StudPermPhone	nvarchar(14)	No
StudStartQuarter	nchar(5)	Yes
StudGradQuarter	nchar(5)	Yes
StudCoopAdv	smallint (FK)	Yes
StudCurMajor	nvarchar(4)	Yes
StudPrevMajor	nvarchar(4)	No
StudEmail	nvarchar(30)	Yes
StudMajChgDate	datetime	No
StudNotes	nvarchar(4000)	No
StudResume	image	Yes
StudDegreePlan	image	Yes
StudApproved	nchar(1)	Yes
MajorID	nvarchar(4) (FK)	Yes
StudCmpSkill1	nvarchar(25)	No
StudCmpSkill2	nvarchar(25)	No
StudCmpSkill3	nvarchar(25)	No
StudForeignLang1	nvarchar(25)	No
StudEmpDesired1	nvarchar(25)	No
StudLocPref1	nvarchar(25)	No
StudLocPref2	nvarchar(25)	No
StudExpLevel1	nvarchar(25)	No
StudDegreeAttained1	nvarchar(25)	No
StudDegreeAttained2	nvarchar(25)	No

**Figure 7. tblStudents Design.**

<b>tblEmployers</b>		
<b>Field Name</b>	<b>Data Type</b>	<b>Mandatory</b>
CompanyID	Identity (smallint) (PK)	Yes
CompanyName	nvarchar(25)	Yes
CompanyStreet	nvarchar(25)	Yes
CompanyCity	nvarchar(20)	Yes
CompanyState	nchar(2)	Yes
CompanyZip	nvarchar(9)	Yes
CompanyPhone	nvarchar(14)	Yes
CompanyWebPage	nvarchar(50)	No
CompanyRepLastName	nvarchar(25)	Yes
CompanyRepFirstName	nvarchar(20)	Yes
CompanyRepPhone	nvarchar(14)	Yes
CompanyRepFax	nvarchar(14)	No
CompanyRepEmail	nvarchar(35)	No

**Figure 8. tblEmployers Design.**

<b>tblJobDetails</b>		
<b>Field Name</b>	<b>Data Type</b>	<b>Mandatory</b>
JobID (PK)	Identity int	Yes
CompanyID (FK)	smallint	Yes
JobDescription	ntext	Yes
Major1	nvarchar(4)	Yes
Major2	nvarchar(4)	No
Major3	nvarchar(4)	No

**Figure 9. tblJobDetails Design.**

<b>tblInterviewDetails</b>		
<b>Field Name</b>	<b>Data Type</b>	<b>Mandatory</b>
JobID (PK) (FK)	Identity int	Yes
StudID (PK) (FK)	Nchar(9)	Yes
InterviewDate	datetime	Yes
InterviewTime	datetime	Yes
InterviewerLastName	nvarchar(25)	Yes
InterviewerFirstName	nvarchar(20)	Yes
InterviewerPhone	nvarchar(14)	Yes
InterviewerEmail	nvarchar(35)	No
InterviewStreet	nvarchar(25)	Yes
InterviewCity	nvarchar(20)	Yes
InterviewState	nchar(2)	Yes
InterviewCompleted	nchar(1)	Yes
InterviewNotes	ntext	No
JobAccepted	nchar(1)	Yes

**Figure 10. tblInterviewDetails Design.**

<b>tblQuarters</b>		
<b>Field Name</b>	<b>Data Type</b>	<b>Mandatory</b>
QuarterID (PK)	nchar(5)	Yes
QStartDate	Datetime	Yes
QEndDate	Datetime	Yes

**Figure 11. tblQuarters Design.**

<b>tblQuarterDetails</b>		
<b>Field Name</b>	<b>Data Type</b>	<b>Mandatory</b>
QuarterID (PK) (FK)	nchar(5)	Yes
StudID (PK) (FK)	nvarchar(9)	Yes
StudStatus	nchar(1)	Yes
JobID (FK)	smallint	No

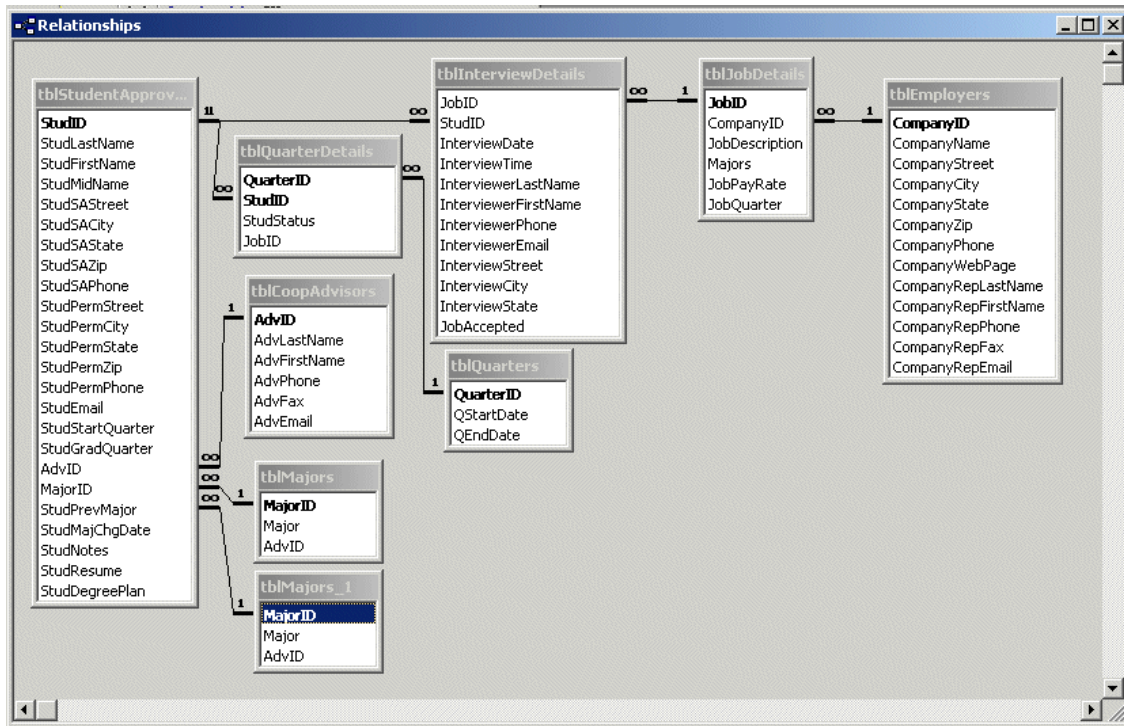
**Figure 12. tblQuarterDetails Design.**

<b>tblCoopAdv</b>		
<b>Field Name</b>	<b>Data Type</b>	<b>Mandatory</b>
AdvID (PK)	smallint Identity	Yes
AdvLastName	nvarchar(25)	Yes
AdvFirstName	nvarchar(20)	Yes
AdvPhone	nvarchar(14)	Yes
AdvFax	nvarchar(14)	Yes
AdvEmail	nvarchar(30)	Yes

**Figure 13. tblCoopAdv Design.**

<b>tblMajors</b>		
<b>Field Name</b>	<b>Data Type</b>	<b>Mandatory</b>
MajorID	nvarchar(4)	Yes
Major	nvarchar(50)	Yes
AdvID	smallint	Yes

**Figure 14. tblMajors Design.**



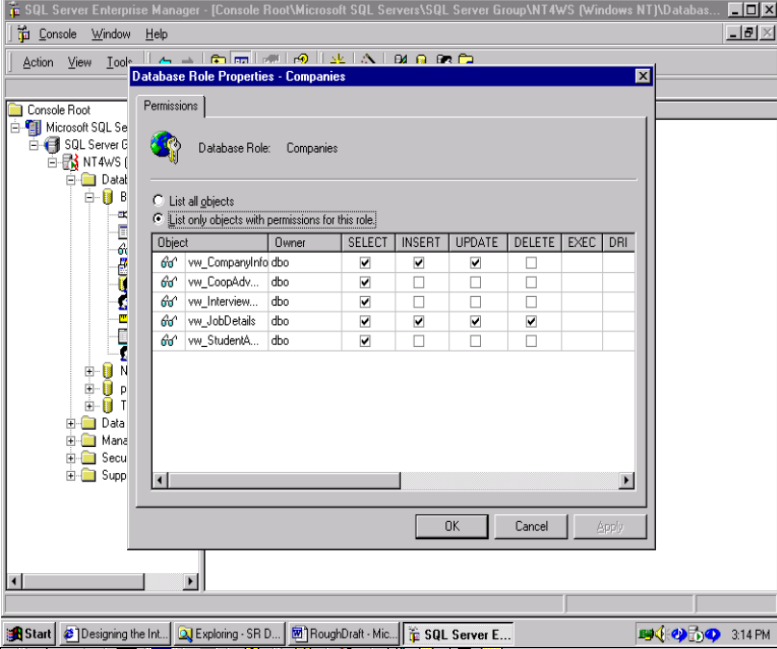
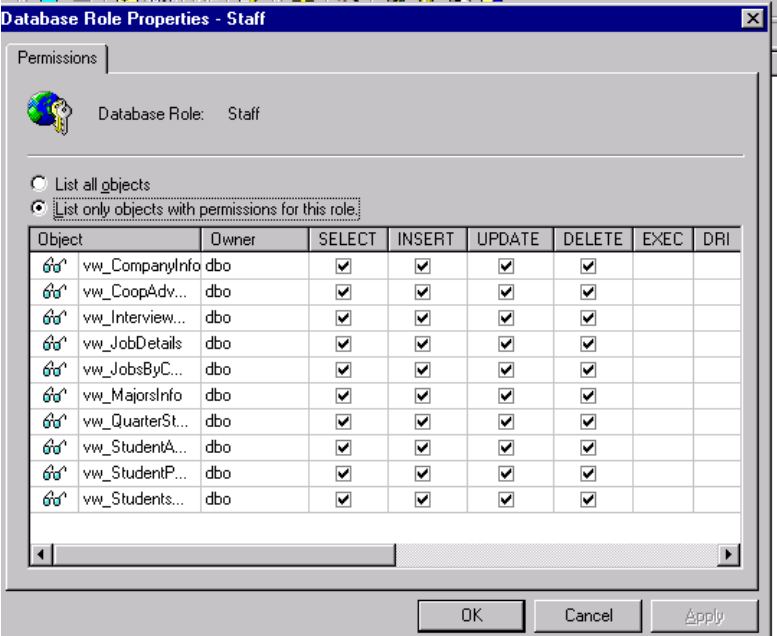
**Figure 15. Table Relationships.**

One of the functions of a database is to protect the data by preventing certain users from seeing or changing highly sensitive data and preventing all users from making costly mistakes. The security system SQL Server 7.0 controls which users can work with what data and which users can perform what activities in the database.

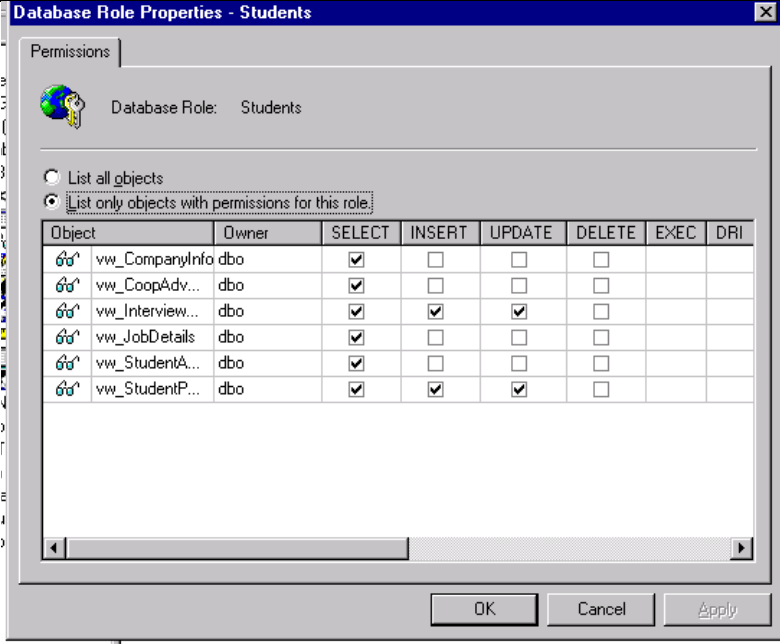
Each user needs to gain access to the SQL Server through a login account that establishes the ability to connect (authentication). This login then has to be mapped to a SQL Server user account used to control activities performed in the database (permissions validation). Therefore, a single login is mapped to one user account created in each database the login has to access. If no user account exists in a database, the user cannot access the database even though the user may be able to connect to SQL Server.

Roles are a powerful tool that allows database administrators to collect users into a single unit against which they can apply permissions. Permissions granted to, denied to, or revoked from a role also apply to any members of the role. Administrators can establish a role that represents a job performed by a

class of workers and grant the appropriate permissions to that role. As workers rotate into the job, administrators simply add them as a member of the role; as they rotate out of the job, remove them from the role. Administrators do not have to repeatedly grant, deny, and revoke permissions to or from each person as they accept or leave the job. The permissions are applied automatically when the users become members of the role.

Role Name	Definition	Members																																																																																								
<b>DBAdmin</b>	<b>Have full control over the database objects.</b>	<b>Database Administrator</b>																																																																																								
<b>Companies</b>	 <table border="1" data-bbox="602 842 1084 961"> <thead> <tr> <th>Object</th> <th>Owner</th> <th>SELECT</th> <th>INSERT</th> <th>UPDATE</th> <th>DELETE</th> <th>EXEC</th> <th>DR</th> </tr> </thead> <tbody> <tr> <td>vw_CompanyInfo</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_CoopAdv...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_Interview...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_JobDetails</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_StudentA...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Object	Owner	SELECT	INSERT	UPDATE	DELETE	EXEC	DR	vw_CompanyInfo	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_CoopAdv...	dbo	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_Interview...	dbo	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_JobDetails	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_StudentA...	dbo	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<b>Potential co-op Employers</b>																																								
Object	Owner	SELECT	INSERT	UPDATE	DELETE	EXEC	DR																																																																																			
vw_CompanyInfo	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_CoopAdv...	dbo	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_Interview...	dbo	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_JobDetails	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_StudentA...	dbo	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
<b>Staff</b>	 <table border="1" data-bbox="477 1486 1187 1772"> <thead> <tr> <th>Object</th> <th>Owner</th> <th>SELECT</th> <th>INSERT</th> <th>UPDATE</th> <th>DELETE</th> <th>EXEC</th> <th>DR</th> </tr> </thead> <tbody> <tr> <td>vw_CompanyInfo</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_CoopAdv...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_Interview...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_JobDetails</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_JobsByC...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_MajorsInfo</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_QuarterSt...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_StudentA...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_StudentP...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>vw_Students...</td> <td>dbo</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Object	Owner	SELECT	INSERT	UPDATE	DELETE	EXEC	DR	vw_CompanyInfo	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_CoopAdv...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_Interview...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_JobDetails	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_JobsByC...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_MajorsInfo	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_QuarterSt...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_StudentA...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_StudentP...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	vw_Students...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<b>Co-op office Staff</b>
Object	Owner	SELECT	INSERT	UPDATE	DELETE	EXEC	DR																																																																																			
vw_CompanyInfo	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_CoopAdv...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_Interview...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_JobDetails	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_JobsByC...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_MajorsInfo	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_QuarterSt...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_StudentA...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_StudentP...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			
vw_Students...	dbo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																			

**Students**



**Students who have taken the Professional Development course.**

**Figure 16. User Roles.**

#### **4. Objectives of the Project**

The delivered product will consist of a fully designed and implemented RDBMS developed in SQL Server 7.0 and appropriate user interfaces for each user group. The database will adhere to the principles of normalization. All database users will be members of the appropriate group so the SQL Server and database security can be easily managed. All interactions with the database will be done through stored procedures or views. No one except the database administrator will have access to the data tables. The user interfaces will be developed with the needs of each group as the major concern. All interfaces will be intuitive and easy to understand and use.

#### **5. Design and Development**

Timeline – Appendix A

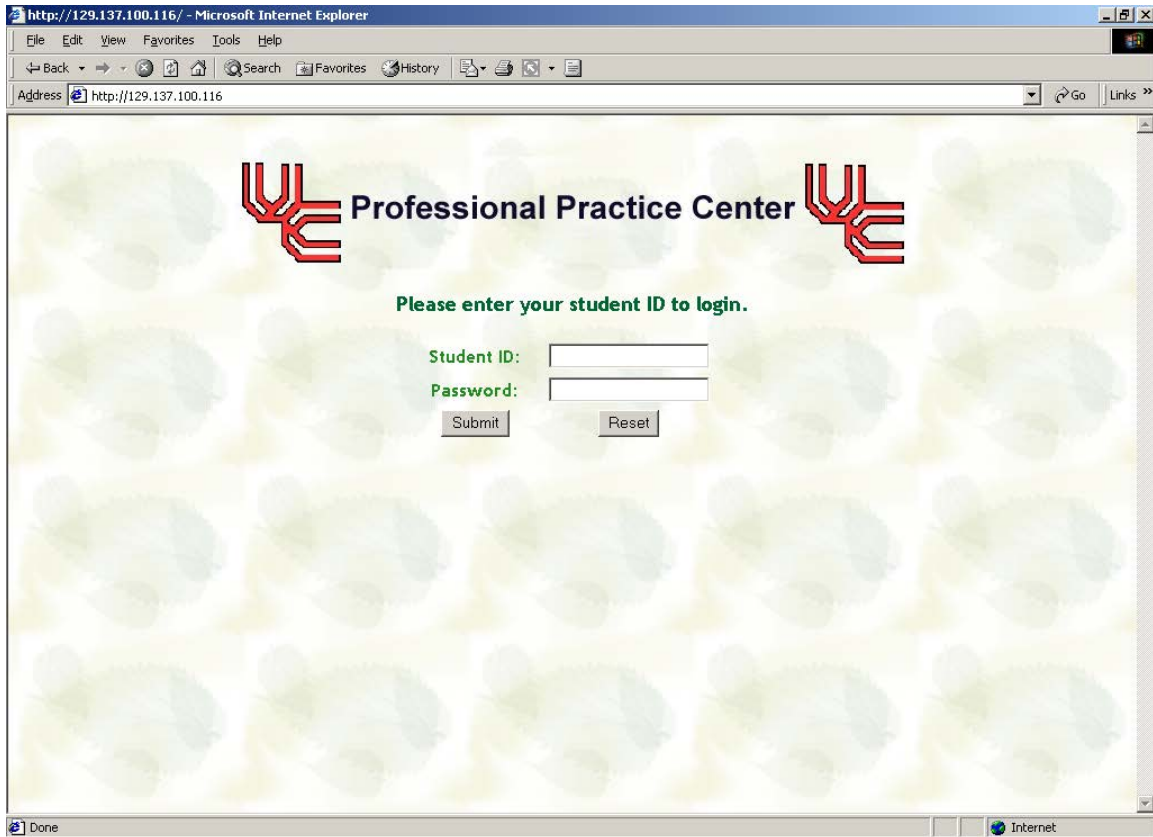
Budget – Appendix B

#### **6. Proof of Design**

##### **6.1 Student Interface**

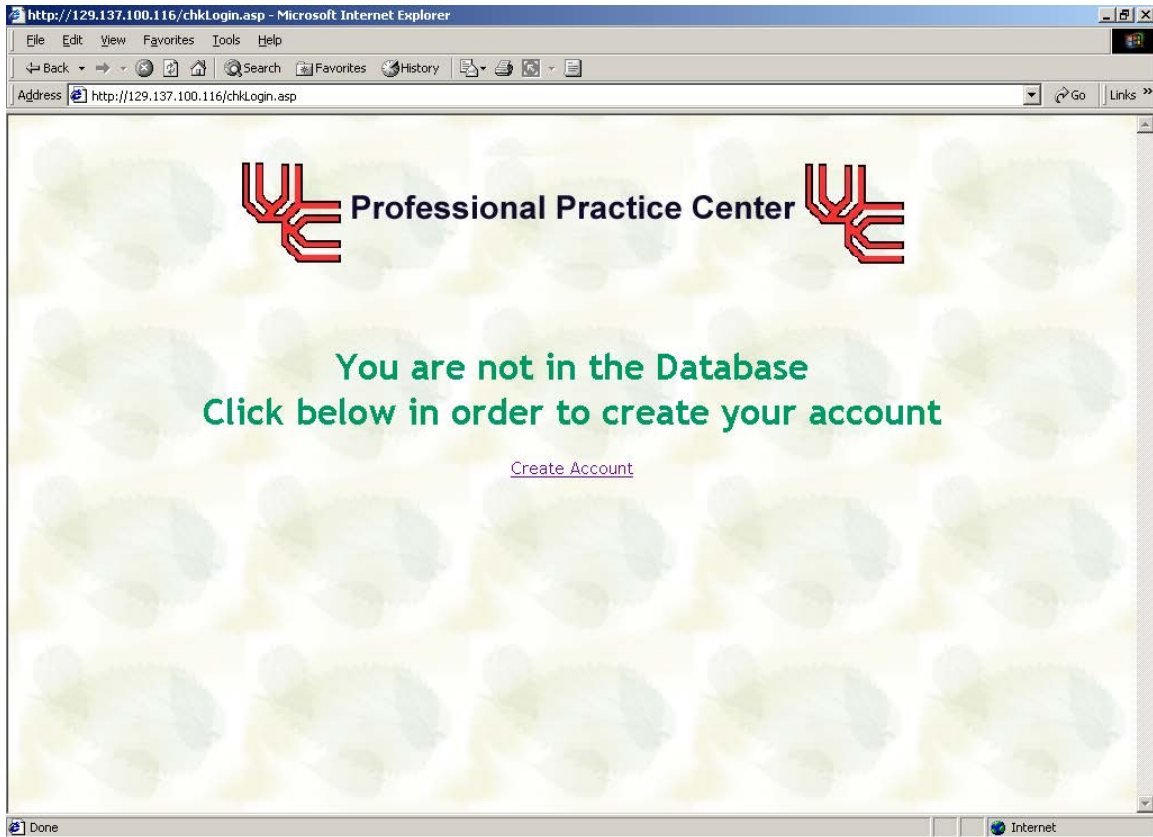
The student interface was created with Visual Interdev and is accessible with Internet Explorer Version 4.0 or higher. The initial page presented to the student is the login page. In order for the student to receive a login ID and password, they must contact the Professional Practice and Career Placement Office. Once they receive a login, they will be able to enter their student information into the database.

The background and font colors are all a part of the nature theme in Visual Interdev. All of the colors are different shades of green. The banner was created using Paint Shop Pro 7 using the same background from the nature theme and the University of Cincinnati colors for the fonts.



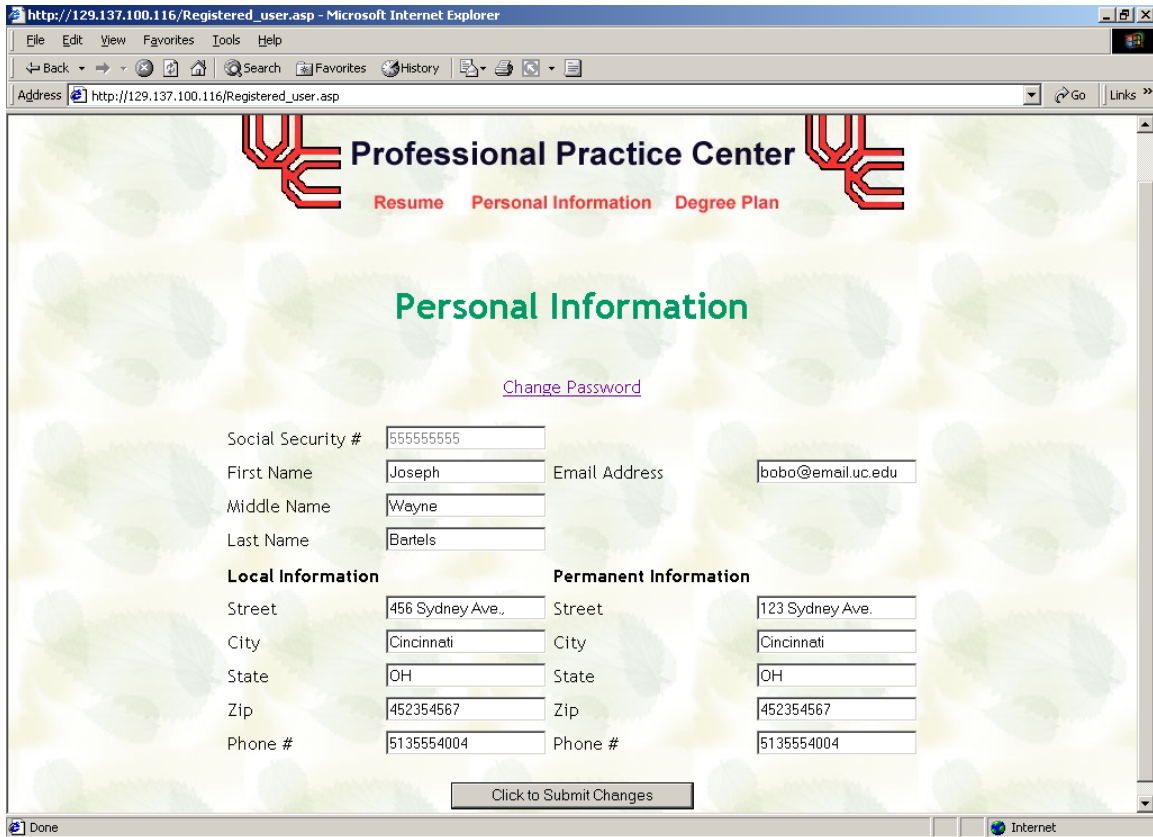
**Figure 17. Student Interface Login Page.**

When the user enters their student ID and password, the information is passed to an Active Server Page (ASP) where it is checked to see if the user has entered the proper information. If this is the first time that the student has accessed the database, or they have entered their information already, the ASP page will recognize this and display it on the screen and display the appropriate links. If the student's information is in the database, the student will be able to edit it. If not, then they are required to input their personal information before being able to access the résumé and degree plan options of the program.



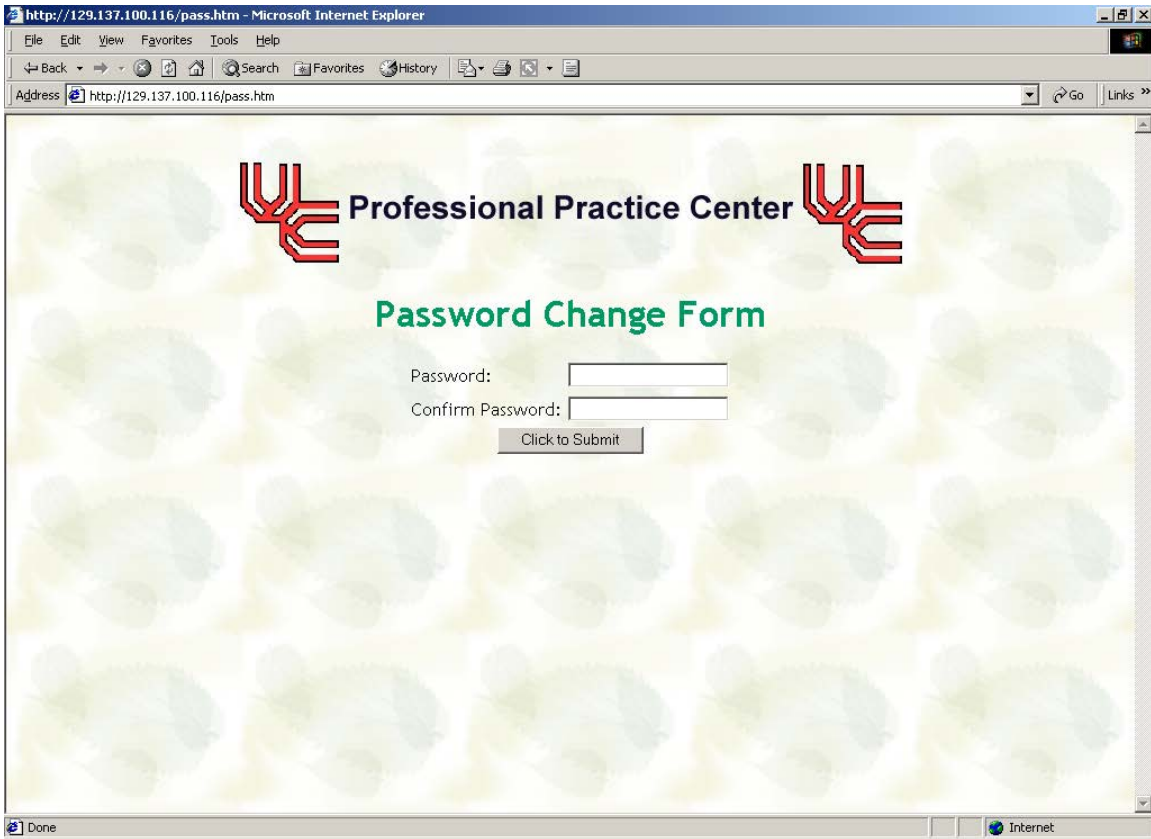
**Figure 18. Student Login Feedback.**

If the student has already entered information into the database, the link will pass them to the personal information page and display the user's personal information from the database. The user will be able to update or change any information by inserting the new data into the appropriate textbox. In order for the information to be updated, the user must click on the button at the bottom of the page.

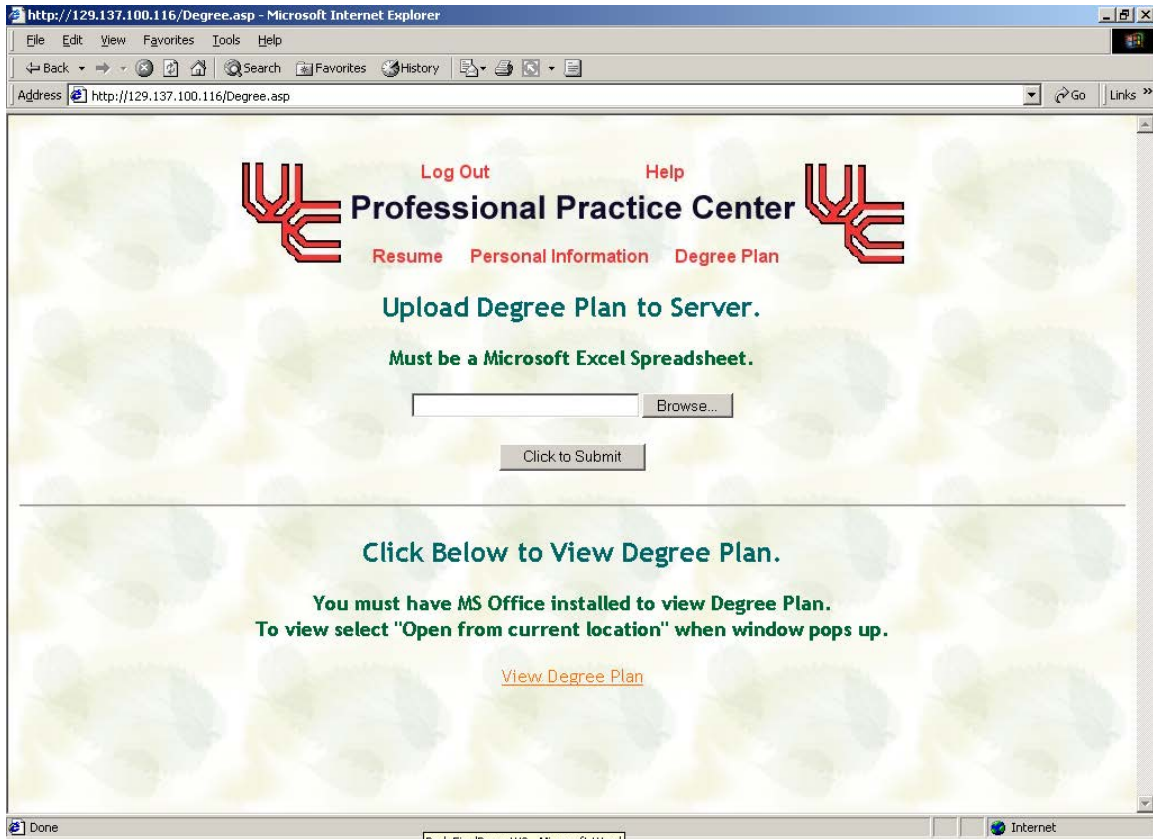


**Figure 19. Student Personal Information Page.**

If the user wants to change their login password, they just click on the link in the center of the page. This will direct them to another Web page to change their password. After the password is entered twice, the user must click on the button below for the changes to take effect.



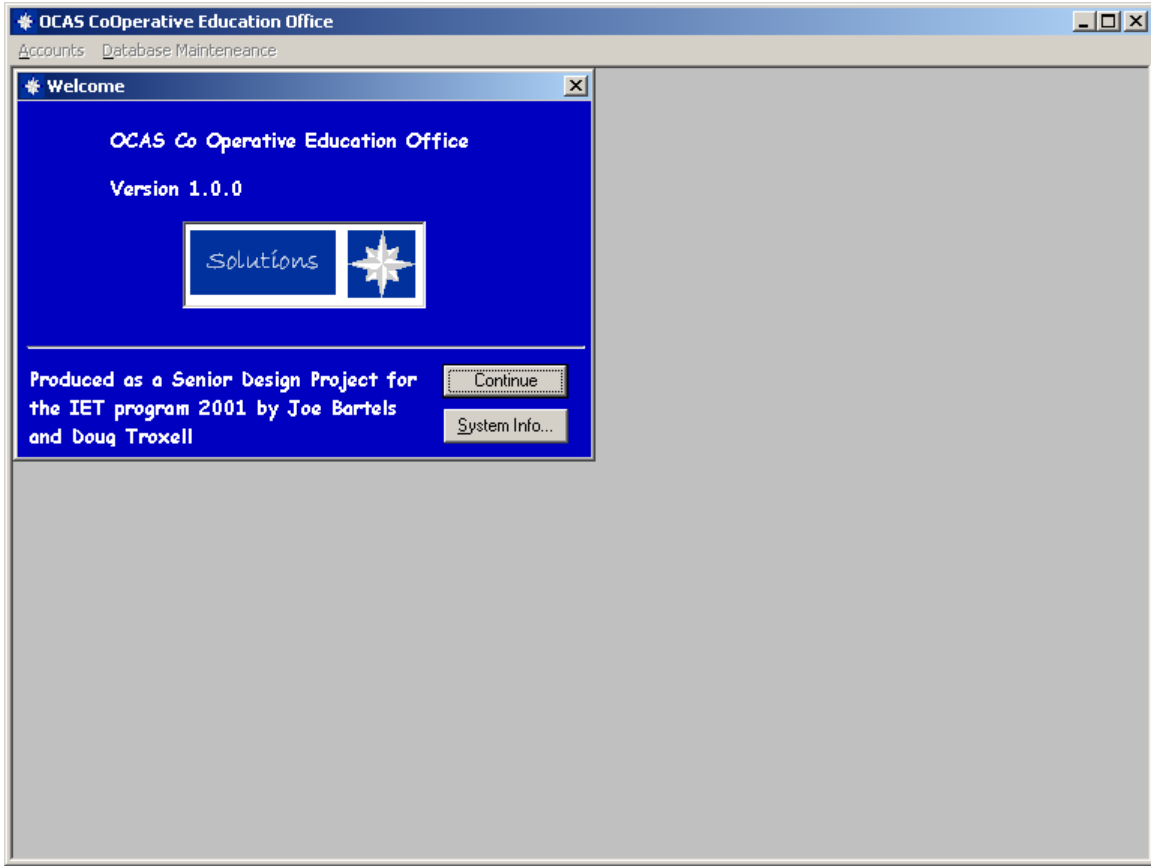
**Figure 20. Password Change Form.**



**Figure 21. Update Resume or Degree Plan.**

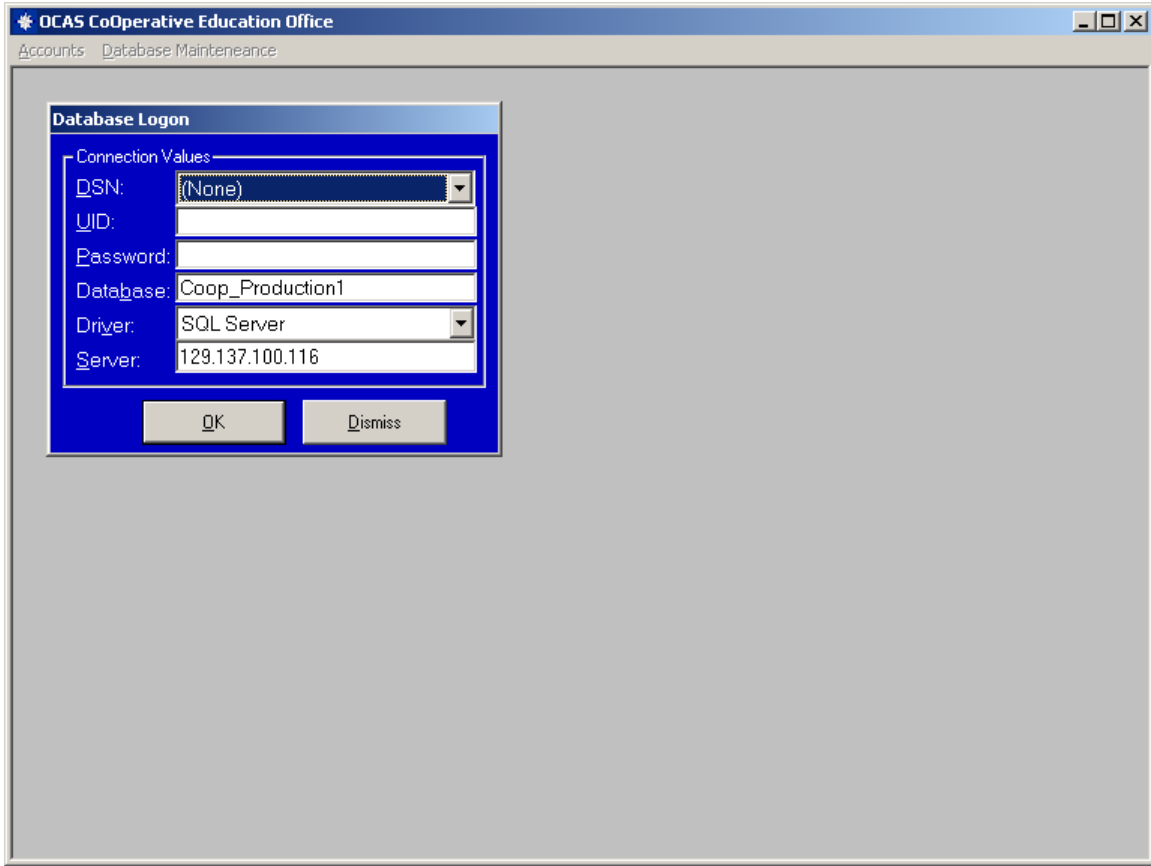
## 6.2 Staff Interface

The staff interface was created using Visual Basic 6.0 as an executable program that runs from a Windows environment. The color scheme is composed of a dark blue background with a white font. When a staff member runs the program, a general welcome is displayed stating the creators of the program. From this screen they are able to continue further to login to the database or to display information about their computer.



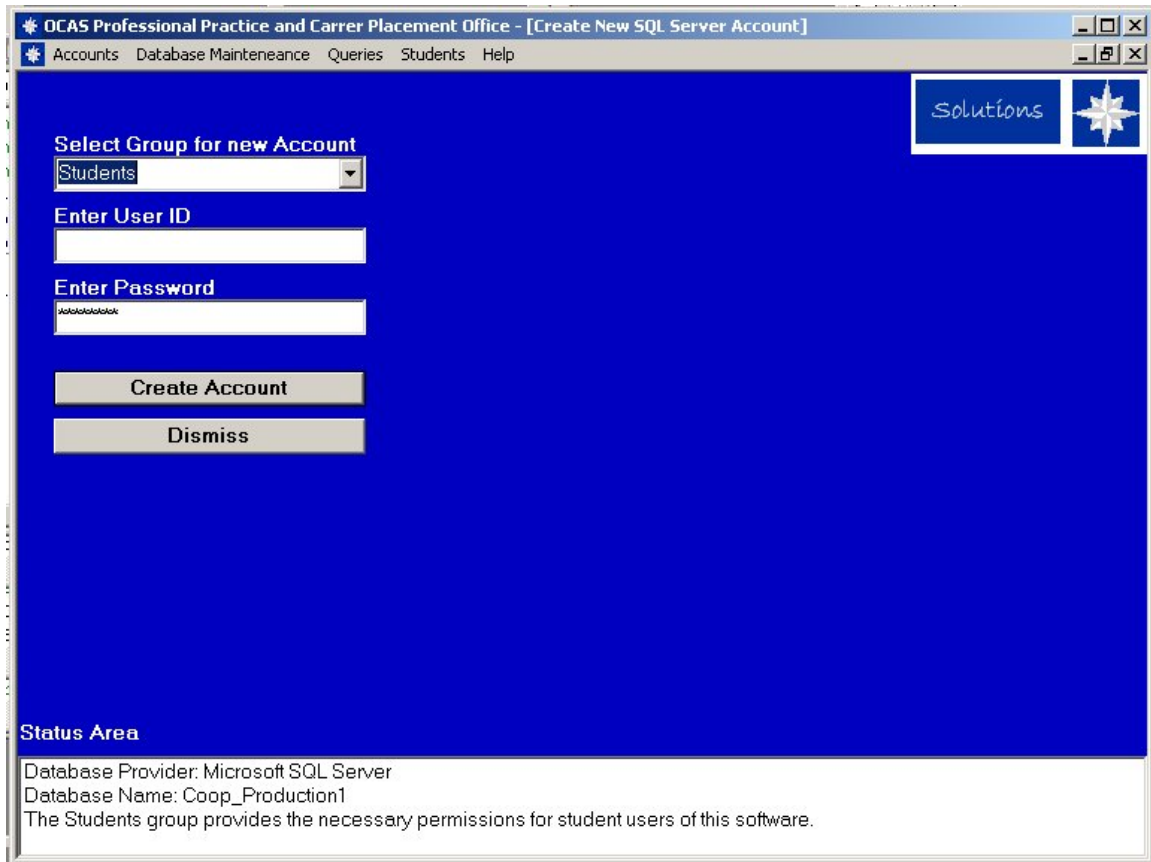
**Figure 22. Staff Welcome Page.**

When the user clicks onto the continue button, a login screen pops up. Here the user decides how they want to connect to the database. They can either connect using OLE DB or they can enter the name of an ODBC connection that is setup on their computer. If they started the program by accident, they can click on the Dismiss Button to exit the program or click on the OK button to login.



**Figure 23. Staff Login Screen.**

When the user logs in, the program displays a message box letting them know that they logged in successfully. Once a login connection has been established, the staff member may perform common administrative tasks to add and update data in the database.



OCAS Professional Practice and Career Placement Office - [Create New SQL Server Account]

Accounts Database Maintenance Queries Students Help

Solutions

Select Group for new Account

Students

Enter User ID

Enter Password

Create Account

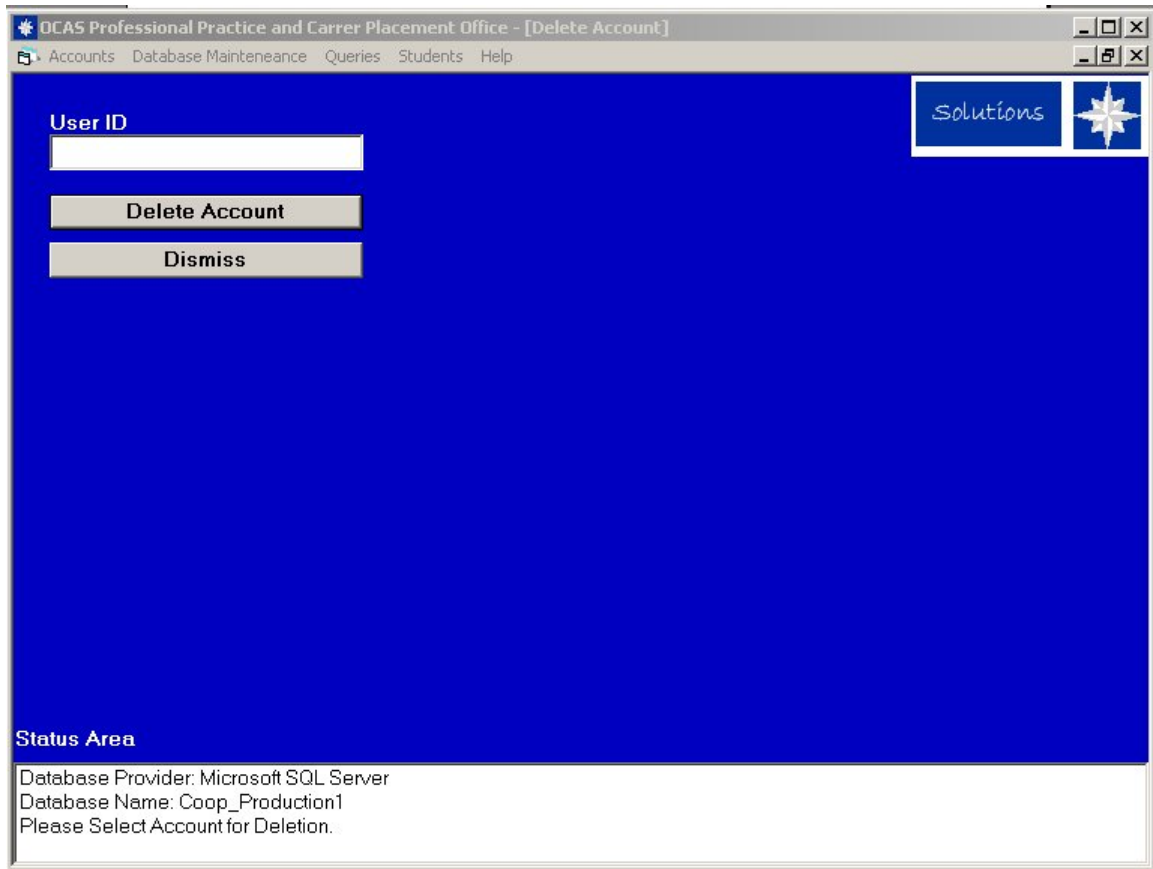
Dismiss

Status Area

Database Provider: Microsoft SQL Server  
Database Name: Coop\_Production1  
The Students group provides the necessary permissions for student users of this software.

**Figure 24. Create New Account.**

**This screen allows the staff to add new accounts to the SQL Server. The account is created and the account is assigned to the appropriate role within the database. This is accomplished with stored procedures.**




**Figure 25. Delete Account.**

**This screen allows the staff to delete a user account from the SQL Server. This will stop the individual from being able to access the database.**

OCAS Professional Practice and Career Placement Office - [Change Password]

Accounts Database Maintenance Queries Students Help

Solutions 

User ID

New Password


Re-Enter New Password

Status Area  
Database Provider: Microsoft SQL Server  
Database Name: Coop\_Production1  
Please Change Password.

**Figure 26. Change Password.**

OCAS Professional Practice and Career Placement Office - [Add Co-op Advisor to Database]

Accounts Database Maintenance Queries Students Help

Solutions 

Advisor Last Name  Advisor First Name

Phone Number  Fax Number

( ) - ( ) -

Email Address

Status Area

Database Provider: Microsoft SQL Server  
Database Name: Coop\_Production1

**Figure 27. Add or Update Advisor Information.**

OCAS Professional Practice and Career Placement Office - [Add Company to Database]

Accounts Database Maintenance Queries Students Help

**Company Name**

**Street Address**

**City** **State** **Zip Code**


**Phone** **Company Web Page**  
( ) -

**Representative's Last Name** **Representative's First Name**

**Representative's Phone** **Representative's Fax**  
( ) -  ( ) -

**Representative's Email**


**Status Area**  
Database Provider: Microsoft SQL Server  
Database Name: Coop\_Production1

Solutions 

**Figure 28. Add or Update Company (Employer) Information.**

OCAS Professional Practice and Career Placement Office - [Add Job Details to Database]

Accounts Database Maintenance Queries Students Help

Solutions 

**Company**

ABC Computer Solutions  
Around the Horn  
Bottom-Dollar Software  
Cactus Chemical  
Carol Computers

**Job Description**

**Major**  
AET

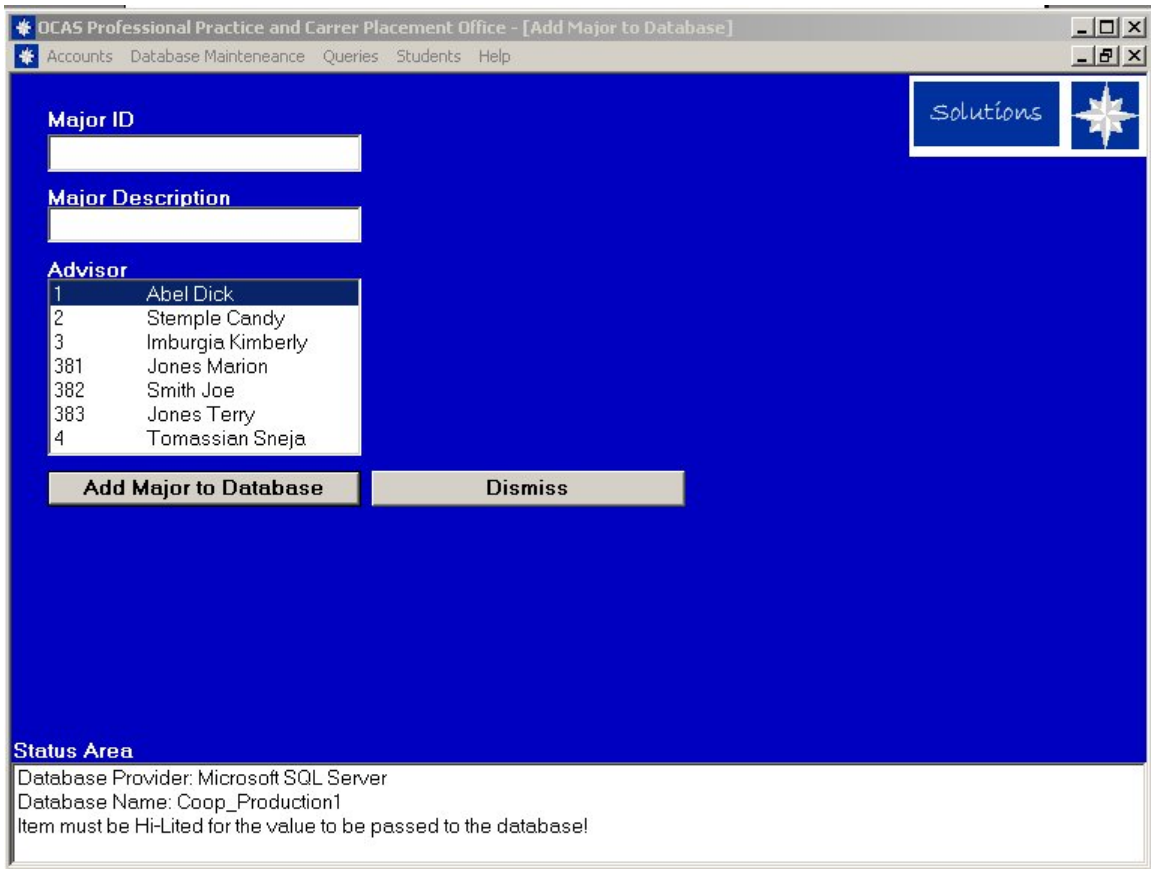
**Major**  
AET

**Major**  
AET

**Add Job to Database** **Dismiss**

**Status Area**  
Database Provider: Microsoft SQL Server  
Database Name: Coop\_Production1  
Item must be Hi-Lited for the value to be passed to the database!


**Figure 29. Add or Update Job Detail Information.**



**Figure 30. Add or Update Major Information.**

OCAS Professional Practice and Career Placement Office - [Add Quarter to Database]

Accounts Database Maintenance Queries Students Help

Solutions 

Quarter ID

Quarter Start Date  
###-###-####


Quarter End Date  
###-###-####

Status Area  
Database Provider: Microsoft SQL Server  
Database Name: Coop\_Production1

**Figure 31. Add or Update Quarter Information.**

OCAS Professional Practice and Career Placement Office - [Add Student Status to Database]

Accounts Database Maintenance Queries Students Help

Solutions 

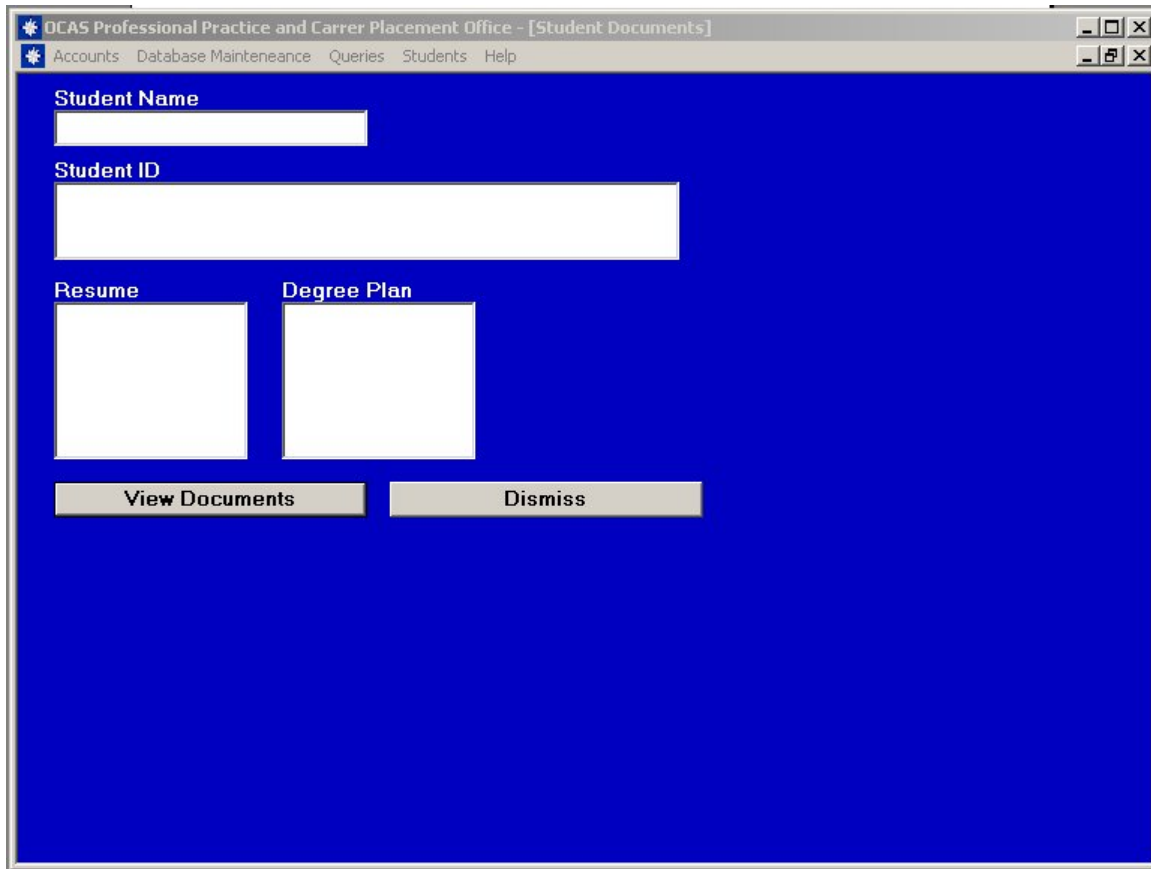
Student ID

Quarter Status  
A2000 Coop

Add Quarter Status Dismiss

Status Area  
Database Provider: Microsoft SQL Server  
Database Name: Coop\_Production1  
Item must be Hi-Lited for the value to be passed to the database!

**Figure 32. Add or Update Student's Quarter Status Information.**



**Figure 33. View Student Degree Plan and Resume Information.**

## **6.4 Data Management System**

The data management system is based on client/server architecture. This architecture was chosen over the multi-tier approach because the needs and requirements of the OCAS PPCPO do not change rapidly enough to warrant the need of separating the business logic from the user interface and RDBMS components. This approach requires less hardware overhead and future maintenance costs for the PPCPO.

The client portion of the project includes all the required user capabilities plus business rules such as data validation and logic. An example of this is the error checking capabilities built into the forms used for data input. The server side includes all the functionality of the RDBMS plus implements the business logic needed by the PPCPO. An example of this is the use of triggers to set flags when student data is changed. Once

the flag is changed, a query can be run against the flag to identify all students who need to have their data verified by the PPCPO advisor.

## **7. Conclusions and Recommendations**

The project accomplished its primary goal in that we learned a great deal about project management, software development and customer support. The project management aspect grew out of scope for the abilities and time that we had to devote to this project. We had to overcome the steeper than expected learning curves for the technologies we were using and the constraints of working for a customer.

It is recommended that the users of the Active Server Pages use Internet Explorer 5.0 or higher and that the users of the Visual Basic program have their monitor resolution set to 800 x 600 or higher. For future improvements to the project, it is recommended that the fax and email capabilities of sending multiple resumes be implemented.

## Works Cited

1. Ashenfelter, John. *Choosing a Database for Your Web Site*. Somerset, New Jersey: Wiley Computer Publishing, 1998.
2. Brickley, Ron. Database Administrator, Structural Dynamics Research Corporation. Personal Interview. April 11, 2000.
3. Desai, Anil. *SQL Server 7 backup& recover*. New York: McGraw-Hill, 2000.
4. Dyck, Timothy. "DB2 7.1 Covers the Spectrum." [HTTP://www.zdnet.com/eweek/stories/general/0,11011,2550466,00.html](http://www.zdnet.com/eweek/stories/general/0,11011,2550466,00.html). April 17, 2000.
5. Dyck, Timothy. "Oracle 8i: Polished for Web". *PC Week*. March 6, 2000. 45 –53.
6. Feibus, Andy. "SQL Server 7.0: Room to Grow". *Informationweek*. February 1, 1999. 112 – 123.
7. Guengerich, Steve and Patrick Smith. *Client/Server Computing, Second Edition*. Indianapolis, Indiana: Sams, Macmillan Computer Publishing. 1994.
8. Hart, Ron. Adjunct Professor, Information Technology Program, University of Cincinnati. Personal Interview. April 11, 2000.
9. Imburgia, Kimberly. Program Coordinator, OCAS Professional Practice and Placement Office. Personal Interview. April 12, 2000.
10. Johnson, Amy Helen. "Web Application Development Tools". *Government Computer News*. September 13, 1999. 23 – 27.
11. Maheri, Anil. "Database Management – Database Security for the Web". *Information Systems Management*. Volume 16, Number 2 1999. 85.
12. Sinclair, Ian R. *Essentials of Computer Security*. London: Bernard Babani, 1997.
13. Son, S H. "Issues and Approaches to Supporting Timeliness and Security in Real-Time Database Systems". *Journal of Systems Architecture*. Volume 46, Number 4 2000. 397.
14. Soto, Carlos A. "FileMaker Pro is logical database pick". *Government Computer News*. April 24, 2000. 45 – 47.
15. Thuraingham, Bhavani. *Handbook of Data Management*. Boca Raton, Florida: Auerbach Publications, CRC Press LLC. 1998.
16. Wood, Larry E. *User Interface Design: Bridging the Gap from User Requirements to Design*. Boca Raton, Florida: CRC Press, CRC Press LLC. 1997.
17. Young, Gerry. Database Technician, OCAS Professional Practice and Placement Office. Personal Interview. April 26, 2000.

## Appendix A. Timeline

<b>Task</b>	<b>Start Date</b>	<b>End Date</b>	<b>Completed</b>
Research Possible Project	3/29/00	4/18/00	Yes
Project Approval	3/29/00	4/20/00	Yes
Research Project	3/29/00	5/7/00	Yes
Write First Draft of Proposal	4/30/00	5/10/00	Yes
Revise Proposal	5/24/00	5/30/00	Yes
Prepare Presentation	5/24/00	5/31/00	Yes
Documentation	4/20/00	5/31/00	
<b>Present Proposal</b>	5/30/00	5/31/00	Yes
Develop Final Project Definition	4/19/00	6/26/00	Yes
Develop Prototype 1	6/5/00	10/20/00	Yes
Test Prototype 1	10/21/00	11/7/00	Yes
Research Deficiencies	10/21/00	11/14/00	Yes
Revise Prototype 1	11/14/00	11/20/00	Yes
Test Prototype 2	11/20/00	11/27/00	Yes
Revise Prototype 2	11/20/00	11/27/00	Yes
Final Testing of Prototype	11/27/00	11/29/00	Yes
Final Revision of Prototype	11/27/00	11/29/00	Yes
Prepare Presentation	11/27/00	11/29/00	Yes
Documentation	6/1/00	11/29/00	
<b>Present Completed Prototype</b>	12/3/00	12/4/00	Yes
Build Beta Project	12/4/00	1/1/01	Yes
Test Beta Project	1/3/01	2/1/01	Yes
Revise Beta Project	2/1/01	2/7/01	Yes
Build Final Project	2/7/01	2/12/01	Yes
Test Final Project	2/12/01	2/15/01	
Revise Final Project	2/16/01	2/23/01	
Prepare Presentation	2/24/01	3/2/01	
Documentation	12/4/00	2/24/01	
<b>Present Completed Project</b>	3/3/01	3/4/01	

## Appendix B. Budget

Hardware	Description	Cost
Development Server	Waiting	\$1795.00
Development Workstations (2)	In Place	N/A
ZIP Disks (5)	Quick Exchange	50.00
Test Workstations (24)	Testing	N/A
Software		
SQL Server 7.0	Production RDBMS (Unlimited licenses)	8500.00
Access 2000	Development RDBMS (Included in Office Suite)	N/A
Dream Weaver 3.0	User Interface Development	285.00
Visual Studio 6	User Interface Production	450.00
MS Office Suite	Resumé and Degree Plan	430.00
Miscellaneous		300.00
TOTAL		\$11810.00

## Appendix C. Active Server Page Code

### Login Page

```
<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">

<link REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/THEME.CSS"
      VI6.0THEME="Nature">
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/GRAPH0.CSS"
      VI6.0THEME="Nature">
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/COLOR0.CSS"
      VI6.0THEME="Nature">
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/CUSTOM.CSS"
      VI6.0THEME="Nature">
</head>
<body>
<form action="chkLogin.asp" id="FORM1" method="post" name="FORM1">
<center>&nbsp;  </center>
<CENTER><IMG height=100 src="images/Intro_Image.jpg" width=600></CENTER>
<h3 align="center">Please enter your student ID to login.</h3>
<p align="center">
<table border="0" cellPadding="2" cellSpacing="2" style="HEIGHT: 59px; WIDTH:
      276px">
<tr>
  <td><center><font color="forestgreen"><strong>Student ID:</strong></font> </center> </td>
  <td><center><input id="chkSocial" name="chkSocial" maxLength="9"></center> </td></tr>
<tr>
  <td><CENTER><font color="forestgreen"><strong>Password:</strong>
    </font></CENTER></td>
  <td><CENTER><INPUT type="password" id=chkPin name=chkPin
    maxLength="20"></CENTER></td></tr>
<tr>
  <td><center><input id="submit1" name="submit1" type="submit"
    value="Submit"></center></td>
  <td><center><input id="reset1" name="reset1" type="reset"
    value="Reset"></center></td></tr>
</table></p>
</form>
</body>
</html>
```

## ChkLogin.asp Page

```
<%@ Language=VBScript %>
<%Session("User") = Request.Form("chkSocial")
   Session("Pin") = Request.Form("chkPin")
%>
<HTML>
<HEAD>
<META name=VI60_defaultClientScript content=VBScript>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">

<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/THEME.CSS"
   VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/GRAPH0.CSS"
   VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/COLOR0.CSS"
   VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/CUSTOM.CSS"
   VI6.0THEME="Nature"><BR>
<center></center>

</HEAD>
<BODY>
<%
Dim strStudent
Dim strPin
Dim Correct
Dim dbconn
Dim recset

Set dbconn = server.CreateObject ("adodb.connection")
Set recset = server.CreateObject ("adodb.recordset")

dbconn.Open "DSN=ConnectMe",Session("User"),Session("Pin")

recset.Open "Select * from tblStudentApproved where StudID='" & Session("User") &
   """,dbconn,2,3
Correct = True

Do While(Not Recset.EOF and Correct = True)
   If txtSocial.value = Session("User") then
   If Recset.Fields("StudID") = Session("User") then
       Correct = False
   else
       Recset.moveNext
   end if
Loop

If Correct = False then
   Response.Write "<BR><BR><BR><CENTER><H1>Hello " & recset.Fields("StudFirstName") &
   " " & recset.Fields("StudLastName") & "<BR>"
   Response.Write "Please click below to continue</H1></CENTER>"
   Response.Write "<CENTER><A HREF=
       'http://129.137.100.116/Registered_user.asp'>Continue</a></CENTER>"

Else
```

```

Response.Write "<BR><BR><BR><CENTER><H1>You are not in the
                Database<BR>"
Response.Write "Click below in order to create your account</H1>"
Response.Write "<a href='New_Student.htm'>Create Account</A></CENTER>"
End If

reset.Close
dbconn.Close
%></P>
</BODY>
</HTML>

```

### New\_Student.asp

```

<HTML>
<HEAD>
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/THEME.CSS"
        VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/GRAPH0.CSS"
        VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/COLOR0.CSS"
        VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/CUSTOM.CSS"
        VI6.0THEME="Nature"></HEAD>

<BODY>
<FORM method="POST" name="FORM1">
<center>&nbsp;&nbsp;&nbsp;</center>
<CENTER><IMG height=100 src="images/Intro_Image.jpg" width=600></CENTER>
<CENTER>
<H1 align=center>New Student Information</H1>
<P align=center></P></CENTER>
<h4><IMG src="Bullet.ico">&nbsp;&nbsp;&nbsp; Required Information</h4>
<DIV align=center>

<TABLE cellPadding=2 cellSpacing=2 cols=2 id=TABLE1 border=0>
<TR>
<TD><IMG src="Bullet.ico"></TD>
<TD>First Quarter at OCAS</TD>
<TD><SELECT name=startdate>
<%
    Dim temp
    Dim counter
    temp = Year(Now())
    counter = temp - 1
    While counter <= temp
        Response.Write "<OPTION>Winter " & counter & "</OPTION>"
        Response.Write "<OPTION>Spring " & counter & "</OPTION>"
        Response.Write "<OPTION>Summer " & counter & "</OPTION>"
        Response.Write "<OPTION>Fall " & counter & "</OPTION>"
        counter =counter +1
    WEnd
%>
</SELECT>
<TD><IMG src="Bullet.ico"></TD>

```

```

<TD>Expected Graduation</TD>
<TD><SELECT name=graddate>
<% Dim temp1
Dim counter1
temp1 = Year(Now())
counter1 = temp1 + 1
temp1 = temp1 + 6
While counter1 <= temp1
Response.Write "<OPTION>Winter " & counter1 & "</OPTION>"
Response.Write "<OPTION>Spring " & counter1 & "</OPTION>"
Response.Write "<OPTION>Summer " & counter1 & "</OPTION>"
Response.Write "<OPTION>Fall " & counter1 & "</OPTION>"
counter1 =counter1 +1
WEnd
%>
</SELECT></TD>
</TR>
<TR> <TD></TD>
<TD>Social Security # </TD>
<TD><INPUT id=txtSocial name=txtSocial value="<% =Session("User")%>" disabled></TD>
<TD></TD><TD></TD>
<TR>
<TD><IMG src="Bullet.ico"></TD>
<TD>First Name </TD>
<TD><INPUT id=txtfname name=txtfname>
</TD><TD></TD>
<TD>Email Address</TD>
<TD><INPUT id=txtemail name=txtemail>
</TD>
<TR>
<TD></TD> <TD>Middle Name </TD>
<TD><INPUT id=txtmname name=txtmname></TD>
</TD></TR>
<TR>
<TD><IMG src="Bullet.ico"></TD>
<TD>Last Name </TD>
<TD><INPUT id=txtlname name=txtlname></TD>
<TD></TD>
<TR>
<TD></TD><TD></TD><TD></TD><TD></TD>
<TR>
<TD></TD>
<TD><B>Local Information</B></TD>
<TD></TD>
<TD></TD>
<TD><B>Permanent Information</B></TD>
<TD> <INPUT name="chksame" type=checkbox>&nbsp;&nbsp;&nbsp;Same as Local </TD>
</TR>
<TR>
<TD><IMG src="Bullet.ico"></TD>
<TD>Street</TD>
<TD><INPUT id=txtLstreet name=txtLstreet></TD>
<TD><IMG src="Bullet.ico"></TD>
<TD>Street
<TD><INPUT id=txtPstreet name=txtPstreet></TD>
<TR>

```

```

<TD><IMG src="Bullet.ico"></TD>
<TD>City</TD>
<TD><INPUT id=txtLcity name=txtLcity></TD>
<TD><IMG src="Bullet.ico"></TD>
<TD>City</TD>
<TD><INPUT id=txtPcity name=txtPcity></TD>
<TR>
<TD><IMG src="Bullet.ico"></TD>
<TD>State</TD>
<TD><INPUT id=txtLstate name=txtLstate maxLength="2"></TD>
<TD><IMG src="Bullet.ico"></TD>
<TD>State
<TD><INPUT id=txtPstate name=txtPstate maxLength="2"></TD>
<TR>
<TD><IMG src="Bullet.ico"></TD>
<TD>Zip </TD>
<TD><INPUT id=txtLzip name=txtLzip></TD>
<TD><IMG src="Bullet.ico"></TD>
<TD>Zip </TD>
<TD><INPUT id=txtPzip name=txtPzip></TD></TR>
<TR>
<TD>Phone #</TD>
<TD><INPUT id=txtLphone name=txtLphone></TD><TD></TD>
<TD>Phone #
<TD><INPUT id=txtPphone name=txtPphone></TD></TR>
</TABLE></DIV>
<BR>&nbsp;

<P align=center><INPUT type=button name="Submit" value="Click to Submit"
onClick="Submit_OnClick"><BR></P>
</FORM>
<SCRIPT LANGUAGE="VBScript">
<!--
Dim CheckText
Sub Submit_OnClick
CheckText = True

Call CheckInfo(FORM1.txtfname.Value, "Please specify a first name.")
Call CheckInfo(FORM1.txtlname.Value, "Please specify a last name.")
Call CheckInfo(FORM1.txtlstreet.Value, "Please specify a local street.")
Call CheckInfo(FORM1.txtlcity.Value, "Please specify a local city.")
Call CheckInfo(FORM1.txtlstate.Value, "Please specify a local state.")
Call CheckInfo(FORM1.txtlzip.Value, "Please specify a local zip code.")

If Not FORM1.chksame.checked then
Call CheckInfo(FORM1.txtpstreet.Value, "Please specify a permanent
street.")
Call CheckInfo(FORM1.txtpcity.Value, "Please specify a permanent
city.")
Call CheckInfo(FORM1.txtpstate.Value, "Please specify a permanent
state.")
Call CheckInfo(FORM1.txtpzip.Value, "Please specify a permanent
zip code.")
End If

If CheckText Then

```

```

        window.navigate "New_Stud.asp"
    End If
End Sub

Sub CheckInfo(ByVal strFieldValue, ByVal strMsg)
    If strFieldValue = "" And CheckText Then
        MsgBox strMsg, 0, "Attention"
        CheckText = False
    End If
End Sub
-->
</SCRIPT>
</BODY>
</HTML>

```

### **New\_Stud.asp**

```

<HTML>
<HEAD>
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/THEME.CSS"
    VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/GRAPH0.CSS"
    VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/COLOR0.CSS"
    VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/CUSTOM.CSS"
    VI6.0THEME="Nature">
</HEAD>
<BODY>
</BODY>
</HTML>
<%
Dim dbconn
Dim recset

Set dbconn = server.CreateObject ("adodb.connection")
Set recset = server.CreateObject ("adodb.recordset")

If Session("User") <> "" then
    dbconn.Open "DSN=ConnectMe",Session("User"),Session("Pin")
    dbconn.Open "DSN=ConnectMe","sa","joecamel"
    recset.Open "Select * from tblStudentApproved",dbconn,2,3

    recset.AddNew
    recset.Fields("StudID") = Session("User")
    recset.Fields("StudFirstName") = Request.Form("txtfname")
    recset.Fields("StudMidName") = Request.Form("txtmname")
    recset.Fields("StudLastName") = Request.Form("txtlname")
    recset.Fields("StudSASstreet") = Request.Form("txtlstreet")
    recset.Fields("StudSACity") = Request.Form("txtlcity")
    recset.Fields("StudSAState") = Request.Form("txtlstate")
    recset.Fields("StudSAZip") = Request.Form("txtlzip")
    recset.Fields("StudSAPhone") = Request.Form("txtlphone")
    recset.Fields("StudEmail") = Request.Form("txtemail")
    recset.Fields("StudPermStreet") = Request.Form("txtpstreet")
    recset.Fields("StudPermCity") = Request.Form("txtpcity")

```

```

recset.Fields("StudPermState") = Request.Form("txtpstate")
recset.Fields("StudPermZip") = Request.Form("txtpzip")
recset.Fields("StudPermPhone") = Request.Form("txtpphone")

recset.Update
recset.Close
dbconn.Close

Response.Write "<CENTER><h1>Your information has been added to the
database."
Response.Write "You must login again in order to update any
information.</H1></CENTER>"

Session("User")=""
Session("Pin")=""
Else
Response.Write "You are not logged in. Please try to login again."
End if
%>

```

### **Image\_Map.asp**

```

<HTML>
<HEAD>
<map NAME="header">&nbsp;
<area SHAPE="RECT" COORDS="120,80,186,96" HREF="resume.asp"
ALT="Resume">
<area SHAPE="RECT" COORDS="208,80,372,96" HREF="Registered_user.asp"
ALT="Personal Info">
<area SHAPE="RECT" COORDS="390,80,485,96" HREF="Degree.asp" ALT="Degree
Plan">
</map>
<center></center>
</HEAD>
</HTML>

```

## Logout.asp

```
<%  
Session("User") = ""  
Session("Pin") = ""  
Response.redirect "http://129.137.100.116"  
%>
```

## Pass.htm

```
<HTML>  
<HEAD>  
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">  
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/THEME.CSS"  
VI6.0THEME="Nature">  
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/GRAPH0.CSS"  
VI6.0THEME="Nature">  
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/COLOR0.CSS"  
VI6.0THEME="Nature">  
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/CUSTOM.CSS"  
VI6.0THEME="Nature"></HEAD>  
<BODY>  
<FORM action="ChangePass.asp" id=form1 method=post name=form1>  
<center>&nbsp;  </center>  
<CENTER><IMG height=100 src="images/Intro_Image.jpg" width=600></CENTER>  
<CENTER><H1>Password Change Form</H1>  
<TABLE cellPadding=2 cellSpacing=2 cols=2 id=TABLE1 border=0>  
  
<TR>  
  <TD>Password:</TD>  
  <TD><INPUT type=password id=txtpass1 name=txtpass1 maxLength="20"></TD>  
<TR>  
  <TD>Confirm Password: </TD>  
  <TD><INPUT type=password id=txtpass2 name=txtpass2 maxLength="20">  
</TD></TR>  
</TABLE>  
<INPUT type=submit name=button1 value="Click to Submit">  
</CENTER>  
</FORM>  
</BODY>  
</HTML>
```

## ChangePass.asp

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<!-- #include FILE=Image_map.asp -->
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/THEME.CSS"
      VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/GRAPH0.CSS"
      VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/COLOR0.CSS"
      VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/CUSTOM.CSS"
      VI6.0THEME="Nature"></HEAD>
</HTML>
<%
Dim strTest, strTest2
Dim Correct
Dim dbconn
Dim recset
Dim reset
Dim temp1, temp2, temp3
Dim tempuser

pass1 = request.form("txtPass1")
pass2 = request.form("txtPass2")

If pass1 = pass2 then
    response.write "<CENTER><H1>Your password has been
                    updated</H1></CENTER>"

    'Create database connection and recordset
    Set dbconn = server.CreateObject("ADODB.Connection")
    Set recset = server.CreateObject("ADODB.Recordset")
    dbconn.Open "DSN=ConnectMe",Session("User"),Session("Pin")
    Correct = True
    strTest = Session("Pin")
    strTest2 = Session("User")
    temp1 = "sp_password " & strTest & ", " & pass1 & ""
    'Change login password by using stored procedure
    dbconn.execute(temp1)
    dbconn.Close

Else
    response.write "<CENTER><H1>Your password was typed incorrectly<BR>Click back to try
again</H1></CENTER>"
End If
```

## Registered\_User.asp

```
<%@ Language=VBScript %>
<%
Dim strTest
Dim Correct
Dim dbconn
Dim recset
Dim tempstr
Dim tempuser
'Create database connection and recordset
Set dbconn = server.CreateObject("ADODB.Connection")
Set recset = server.CreateObject("ADODB.Recordset")
dbconn.Open "DSN=ConnectMe",Session("User"),Session("Pin")
Correct = True
strTest = Session("User")
'Setup SQL string
tempstr = "Select * from tblStudentApproved where StudID=" & strTest & ""
tempstr = "UserInfo " & Session("User") & ""
recset.Open tempstr,dbconn,2,3
'Set recset = dbconn.Execute (tempstr)

Do While(Not recset.EOF and Correct = True)
    tempuser = recset.Fields("StudID").Value
    If tempuser = strTest then
        Correct = False
    else
        recset.MoveNext
    end if
Loop
%>
<HTML>
<HEAD>
<META name=VI60_defaultClientScript content=VBScript>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/THEME.CSS"
VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/GRAPH0.CSS"
VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/COLOR0.CSS"
VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/CUSTOM.CSS"
VI6.0THEME="Nature">
<!-- #include FILE=Image_map.asp -->
</HEAD>
<BODY>
<FORM action="UpdateInfo.asp" id=registered method=post name=registered>&nbsp;

<H1 align=center>Personal Information</H1>
<BR><CENTER><A HREF = "http://129.137.100.116/pass.htm">Change Password</a></CENTER>
<DIV align=center>
<P align=center>
<TABLE cellPadding=2 cellSpacing=2 cols=2 id=TABLE1 border=0>

<TR>
<TD>Social Security # </TD>
```

```

        <TD><INPUT id=txtSocial name=txtSocial maxLength="9" value=
            "<%Response.write recset.Fields("StudID")%">" disabled></TD>
<TR>
    <TD>First Name </TD>
    <TD><INPUT id=txtfname name=txtfname value="<%Response.write
        recset.Fields("StudFirstName")%">"></TD>
    <TD>Email Address
    <TD><INPUT id=txtemail name=txtemail value="<%Response.write
        recset.Fields("StudEmail")%">"></TD>
<TR>
    <TD>Middle Name </TD>
    <TD><INPUT id=txtmname name=txtmname value="<%Response.write
        recset.Fields("StudMidName")%">"></TD>
</TR>
<TR>
    <TD>Last Name </TD>
    <TD><INPUT id=txtlname name=txtlname value="<%Response.write
        recset.Fields("StudLastName")%">"></TD>
<TR>
    <TD></TD>
    <TD></TD>
<TR>
    <TD><P><B>Local Information</B> </P></TD>
    <TD></TD>
    <TD><B>Permanent Information</B></TD>
    <TD><P><B> </B>&nbsp;</P></TD>
</TR>
<TR>
    <TD>Street</TD>
    <TD><INPUT id=txtLstreet name=txtLstreet value="<%Response.write
        recset.Fields("StudSAStreet")%">"></TD>
    <TD>Street
    <TD><INPUT id=txtPstreet name=txtPstreet value="<%Response.write
        recset.Fields("StudPermStreet")%">"></TD>
<TR>
    <TD>City</TD>
    <TD><INPUT id=txtLcity name=txtLcity value="<%Response.write
        recset.Fields("StudSACity")%">"></TD>
    <TD>City
    <TD><INPUT id=txtPcity name=txtPcity value="<%Response.write
        recset.Fields("StudPermCity")%">"></TD>
<TR>
    <TD>State</TD>
    <TD><INPUT id=txtLstate name=txtLstate maxLength="2"
        value="<%Response.write recset.Fields("StudSAState")%">"></TD>
    <TD>State
    <TD><INPUT id=txtPstate name=txtPstate maxLength="2"
        value="<%Response.write recset.Fields("StudPermState")%">"></TD>
<TR>
    <TD>Zip </TD>
    <TD><INPUT id=txtLzip name=txtLzip value="<%Response.write
        recset.Fields("StudSAZip")%">"></TD>
    <TD>Zip </TD>
    <TD><INPUT id=txtPzip name=txtPzip value="<%Response.write
        recset.Fields("StudPermZip")%">"></TD></TR>
<TR>

```

```
<TD>Phone #</TD>
<TD><INPUT id=txtLphone name=txtLphone value="<%Response.write
recset.Fields("StudSAPhone")%">"></TD>
<TD>Phone #
<TD><INPUT id=txtPphone name=txtPphone value="<%Response.write
recset.Fields("StudPermPhone")%">"></TD></TR>
</TABLE></P>

<CENTER>
<INPUT type=submit value="Click to Submit Changes" id=button1 name=button1>
</CENTER>
</DIV></FORM>
<%
recset.Close
dbconn.Close
%>
</BODY>
</HTML>
```

## UpdateInfo.asp

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<!-- #include FILE=Image_map.asp -->
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/THEME.CSS"
VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/GRAPH0.CSS"
VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/COLOR0.CSS"
VI6.0THEME="Nature">
<LINK REL="stylesheet" TYPE="text/css" HREF="_Themes/nature/CUSTOM.CSS"
VI6.0THEME="Nature"></HEAD>
</HTML>
<%
Dim strTest
Dim Correct
Dim dbconn
Dim recset
Dim tempstr
Dim tempuser
'Create database connection and recordset
Set dbconn = server.CreateObject("ADODB.Connection")
Set recset = server.CreateObject("ADODB.Recordset")
dbconn.Open "DSN=ConnectMe",Session("User"),Session("Pin")
Correct = True
strTest = Session("User")
'Setup SQL string
tempstr = "Select * from tblStudentApproved where StudID=" & strTest & ""
recset.Open tempstr,dbconn,2,3

recset.Fields("StudFirstName") = Request.Form("txtfname")
recset.Fields("StudLastName") = Request.Form("txtlname")
recset.Fields("StudMidName") = Request.Form("txtmname")
recset.Fields("StudSASstreet") = Request.Form("txtLstreet")
recset.Fields("StudSACity") = Request.Form("txtLcity")
recset.Fields("StudSASstate") = Request.Form("txtLstate")
recset.Fields("StudSAZip") = Request.Form("txtLzip")
recset.Fields("StudSAPhone") = Request.Form("txtLPhone")
recset.Fields("StudPermStreet") = Request.Form("txtPstreet")
recset.Fields("StudPermState") = Request.Form("txtPstate")
recset.Fields("StudPermCity") = Request.Form("txtPcity")
recset.Fields("StudPermZip") = Request.Form("txtPzip")
recset.Fields("StudPermPhone") = Request.Form("txtPphone")
recset.Fields("StudEmail") = Request.Form("txtemail")

recset.Update
recset.Close
dbconn.Close
Response.Write "<CENTER><H1>Your information has been updated</H1></CENTER>"
%>
```

## Appendix D. Visual Basic Code

### MDIForm\_Production

```
Rem Code Written by Doug Troxell
Rem Senior Design Project
Rem OCAS 2000 - 2001
Option Explicit
Public DocColl As New Collection
Public CollKeyIndex As Integer
```

```
Private Sub MDIForm_Load()
    Rem Size and position form.
    Me.Width = Screen.Width * 0.75
    Me.Height = Screen.Height * 0.75
    Me.Left = (Screen.Width - Me.Width) * 0.5
    Me.Top = (Screen.Height - Me.Height) * 0.5

    Rem Disable menus until appropriate to use.
    mnuAccounts.Enabled = False
    mnuDBMaintenance.Enabled = False
    mnuStudents.Enabled = False
    mnuQueries.Enabled = False

    Rem Hide Toolbar.
    Me.Toolbar.Visible = False
    Me.Toolbar.Enabled = False

    Rem Setup forms to show cascade
    MDIForm_Production.Arrange vbCascade
End Sub
```

```
Private Sub MDIForm_Terminate()
    Rem Free memory.
    Set DocColl = Nothing
End Sub
```

```
Private Sub MDIForm_Unload(Cancel As Integer)
    Rem Destroy connection object on close.
    Set gConnection = Nothing
End Sub
```

```
Private Sub mnuAccountDelete_Click()
    Rem Show form to delete account.
    frmDeleteAccount.Show
    frmDeleteAccount.SetFocus
End Sub
```

```
Private Sub mnuAccountMaintain_Click()
    Rem Show form to change password.
    frmChangePWD.Show
    frmChangePWD.SetFocus
End Sub
```

```
Private Sub mnuAccountNew_Click()
```

```
    frmNewAccount.Show  
    frmNewAccount.SetFocus
```

```
End Sub
```

```
Public Sub mnuDBMaintenanceAddAdvisors_Click()
```

```
    Rem Load frmAddStaff in Add Mode.  
    frmAddStaff.Caption = "Add Co-op Advisor to Database"  
    frmAddStaff.cmdAddAdv.Visible = True  
    frmAddStaff.cmdAddAdv.Enabled = True  
    frmAddStaff.cmdAddAdv.Default = True  
    frmAddStaff.cmdUpdateAdv.Visible = False  
    frmAddStaff.cmdUpdateAdv.Enabled = False  
    frmAddStaff.txtID.Visible = False  
    frmAddStaff.lblID.Visible = False
```

```
    Rem Clear all the test boxes.  
    Call subClearFormAdv  
    frmAddStaff.Show  
    frmAddStaff.SetFocus
```

```
End Sub
```

```
Public Sub mnuDBMaintenanceAddCompanies_Click()
```

```
    Rem Load frmAddcompany in Add mode.  
    frmAddCompany.Caption = "Add Company to Database"  
    frmAddCompany.cmdAddCo.Visible = True  
    frmAddCompany.cmdAddCo.Enabled = True  
    frmAddCompany.cmdAddCo.Default = True  
    frmAddCompany.cmdUpdateCo.Visible = False  
    frmAddCompany.cmdUpdateCo.Enabled = False
```

```
    Rem Clear all the text boxes.  
    Call subClearFormCompany  
    frmAddCompany.Show  
    frmAddCompany.SetFocus
```

```
End Sub
```

```
Public Sub mnuDBMaintenanceAddJobDetail_Click()
```

```
    Rem Show form to add Quarter Info.  
    'Call subClearFormJobDetail  
    frmAddJobDetail.Caption = "Add Job Details to Database"  
    frmAddJobDetail.cmdAddJobDetail.Visible = True  
    frmAddJobDetail.cmdAddJobDetail.Enabled = True  
    frmAddJobDetail.cmdAddJobDetail.Default = True  
    frmAddJobDetail.cmdUpdateJobDetail.Visible = False  
    frmAddJobDetail.cmdUpdateJobDetail.Enabled = False  
    frmAddJobDetail.lblJobID.Visible = False  
    frmAddJobDetail.txtJobID.Visible = False
```

```
    frmAddJobDetail.Show  
    frmAddJobDetail.SetFocus
```

```
End Sub
```

```
Public Sub mnuDBMaintenanceAddMajor_Click()
```

```
    frmAddMajor.Caption = "Add Major to Database"  
    frmAddMajor.cmdAddMajor.Visible = True
```

```
frmAddMajor.cmdAddMajor.Enabled = True
frmAddMajor.cmdAddMajor.Default = True
frmAddMajor.cmdUpdateMajor.Visible = False
frmAddMajor.cmdUpdateMajor.Enabled = False
```

```
Rem Show form to add Quarter Info.
Call subClearFormMajor
frmAddMajor.Show
frmAddMajor.SetFocus
End Sub
```

```
Public Sub mnuDBMaintenanceAddQuarter_Click()
    frmAddQuarter.Caption = "Add Quarter to Database"
    frmAddQuarter.cmdAddQuarter.Visible = True
    frmAddQuarter.cmdAddQuarter.Enabled = True
    frmAddQuarter.cmdAddQuarter.Default = True
    frmAddQuarter.cmdUpdateQuarter.Visible = False
    frmAddQuarter.cmdUpdateQuarter.Enabled = False
```

```
Rem Show form to add Quarter Info.
Call subClearFormQuarter
frmAddQuarter.Show
frmAddQuarter.SetFocus
End Sub
```

```
Private Sub mnuDBMaintenanceAddStudentsStatus_Click()
    frmQUpdateStudent.Caption = "Add Student Status to Database"
    frmQUpdateStudent.cmdAddQuarterStatus.Visible = True
    frmQUpdateStudent.cmdAddQuarterStatus.Enabled = True
    frmQUpdateStudent.cmdAddQuarterStatus.Default = True
    frmQUpdateStudent.cmdUpdateQuarterStatus.Visible = False
    frmQUpdateStudent.cmdUpdateQuarterStatus.Enabled = False
    frmQUpdateStudent.Show
    frmQUpdateStudent.SetFocus
End Sub
```

```
Public Sub mnuDBMaintenanceUpdateAdvisor_Click()
    Rem Load frmSelect in Advisor Mode.
    SelectState = 1 'Advisors
    frmSelect.Caption = "Select Advisor"
    Call subPopulateAdvisors(frmSelect.ListBox1)
    frmSelect.Show vbModal
End Sub
```

```
Public Sub mnuDBMaintenanceUpdateCompany_Click()
    Rem Load frmSelect in Company Mode.
    SelectState = 2 'Company
    frmSelect.Caption = "Select Company"
    Call subPopulateCompany(frmSelect.ListBox1)
    frmSelect.Show vbModal
End Sub
```

```
Public Sub mnuDBMaintenanceUpdateJobDetail_Click()
    Rem Load frmSelect in Company Mode.
    SelectState = 5 'JobDetail
    frmSelect.Caption = "Select Job"
```

```

    Call subPopulateJobDetails(frmSelect.ListBox1)
    frmSelect.Show vbModal
End Sub

Public Sub mnuDBMaintenanceUpdateMajor_Click()
    Rem Load frmSelect in Company Mode.
    SelectState = 4 'Major
    frmSelect.Caption = "Select Major"
    Call subPopulateMajors(frmSelect.ListBox1)
    frmSelect.Show vbModal
End Sub

Public Sub mnuDBMaintenanceUpdateQuarter_Click()
    Rem Load frmSelect in Company Mode.
    SelectState = 3 'Quarter
    frmSelect.Caption = "Select Quarter"
    Call subPopulateQuarter(frmSelect.ListBox1)
    frmSelect.Show vbModal
End Sub

Private Sub mnuDBMaintenanceUpdateStudentsStatus_Click()
    frmQUpdateStudent.Caption = "Update Student Status"
    frmQUpdateStudent.cmdAddQuarterStatus.Visible = False
    frmQUpdateStudent.cmdAddQuarterStatus.Enabled = False
    frmQUpdateStudent.cmdUpdateQuarterStatus.Visible = True
    frmQUpdateStudent.cmdUpdateQuarterStatus.Enabled = True
    frmQUpdateStudent.cmdUpdateQuarterStatus.Default = True
    frmQUpdateStudent.cmdUpdateQuarterStatus.Left = 360
    frmQUpdateStudent.cmdUpdateQuarterStatus.Top = 2520

    Rem Show form to add Quarter Info.
    Call subClearFormQuarterStatus
    frmQUpdateStudent.Show
    frmQUpdateStudent.SetFocus
End Sub

Private Sub mnuHelpAbout_Click()
    frmAbout.Show
End Sub

Private Sub mnuHelpError_Click()
    frmErrors.Show
End Sub

Private Sub mnuQueriesStudentByAdvisor_Click()
    Call subQueryStudentAdvisor
End Sub

Private Sub mnuStudentsDocuments_Click()
    Rem Show form to add Quarter Info.
    Call subClearFormStudentDocs
    frmStudentDocs.Show
    frmStudentDocs.SetFocus
End Sub

Private Sub Toolbar_ButtonClick(ByVal Button As ComctlLib.Button)

```

```

Select Case Button.Key
Case "OpenDoc"
    Call subOpenDoc
Case "NewDoc"
    Call AddForm
Case "SaveDoc"
    Call subSaveDocument
Case "PrintDoc"
    Call subPrintDoc
Case "EditCut"
    Call subCut
Case "EditCopy"
    Call subCopy
Case "EditPaste"
    Call subPaste
Case "EditColor"
    Call subColor
Case "EditFont"
    Call subFont
Case "Find"
    frmFind.Show vbModal
End Select
End Sub
frmAbout
Option Explicit

Private Sub cmdSysInfo_Click()
Call StartSysInfo
End Sub

Private Sub cmdOK_Click()
Unload Me
Rem Show frmODBCLogon modeless.
frmODBCLogon.Show (vbModeless)
End Sub
frmAddCompany
Option Explicit

Private Sub cmdAddCo_Click()
Dim cmdAddCompany As ADODB.Command
Set cmdAddCompany = New ADODB.Command
Rem Clear Error Collection.
gConnection.Errors.Clear

On Error GoTo ErrorHandler
With cmdAddCompany
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_AddCompany"
    .Parameters.Refresh
    .Parameters("@CoName") = Trim(frmAddCompany.txtCoName.Text)
    .Parameters("@CoStreet") = Trim(frmAddCompany.txtCoStreet.Text)
    .Parameters("@CoCity") = Trim(frmAddCompany.txtCoCity.Text)
    .Parameters("@CoState")
Trim(UCCase(frmAddCompany.txtCoState.ClipText))

```

=

```

.Parameters("@CoZip") = Trim(frmAddCompany.txtCoZip.ClipText)
.Parameters("@CoPhone") = Trim(frmAddCompany.txtCoPhone.ClipText)
.Parameters("@CoWebPage") = Trim(frmAddCompany.txtCoWebPage.Text)
.Parameters("@CoRepLName") = Trim(frmAddCompany.txtCoRepLName.Text)
.Parameters("@CoRepFName") = Trim(frmAddCompany.txtCoRepFName.Text)
.Parameters("@CoRepPhone") = Trim(frmAddCompany.txtCoRepPhone.ClipText)
.Parameters("@CoRepFax") = Trim(frmAddCompany.txtCoRepFax.ClipText)
.Parameters("@CoRepEmail") = Trim(frmAddCompany.txtCoRepEmail.Text)
End With
cmdAddCompany.Execute
Rem Successful Completion of Insert into Advisor Table.
If cmdAddCompany.Parameters(0) = 0 Then
    Msg = frmAddCompany.txtCoName.Text & _
        " has successfully been added to the Databse." ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Company Added" ' Define title.
    R = MsgBox(Msg, Style, Title)

    Rem Add Completion to status area.
    frmAddCompany.txtboxInfo.AddItem (Msg)

    Rem Destroy Command Objects.
    Set cmdAddCompany = Nothing

    Rem Ask if they want to add another Company.
    Msg = " Do you wish to add another Company? " ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Add Another Company?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem Clear Form.
        Call subClearFormCompany
    Else
        Unload frmAddCompany
    End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub cmdCloseAdd_Click()
    Rem Remove form from memory.
    Unload frmAddCompany
End Sub

Private Sub cmdUpdateCo_Click()
    Dim cmdUpdateCompany As ADODB.Command
    Set cmdUpdateCompany = New ADODB.Command
    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdUpdateCompany
        .ActiveConnection = gConnection
    End With
End Sub

```

```

.CommandTimeout = 15
.CommandType = adCmdStoredProc
.CommandText = "stp_UpdateCompany"
.Parameters.Refresh
.Parameters("@CoName") = Trim(frmAddCompany.txtCoName.Text)
.Parameters("@CoStreet") = Trim(frmAddCompany.txtCoStreet.Text)
.Parameters("@CoCity") = Trim(frmAddCompany.txtCoCity.Text)
.Parameters("@CoState") = Trim(UCCase(frmAddCompany.txtCoState.ClipText))
.Parameters("@CoZip") = Trim(frmAddCompany.txtCoZip.ClipText)
.Parameters("@CoPhone") = Trim(frmAddCompany.txtCoPhone.ClipText)
.Parameters("@CoWebPage") = Trim(frmAddCompany.txtCoWebPage.Text)
.Parameters("@CoRepLName") = Trim(frmAddCompany.txtCoRepLName.Text)
.Parameters("@CoRepFName") = Trim(frmAddCompany.txtCoRepFName.Text)
.Parameters("@CoRepPhone") = Trim(frmAddCompany.txtCoRepPhone.ClipText)
.Parameters("@CoRepFax") = Trim(frmAddCompany.txtCoRepFax.ClipText)
.Parameters("@CoRepEmail") = Trim(frmAddCompany.txtCoRepEmail.Text)
End With
cmdUpdateCompany.Execute
Rem Successful Completion of Insert into Advisor Table.
If cmdUpdateCompany.Parameters(0) = 0 Then
    Msg = frmAddCompany.txtCoName.Text & _
    " has successfully been updated in the Database." ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Company Updated" ' Define title.
    R = MsgBox(Msg, Style, Title)

    Rem Add Completion to status area.
    frmAddCompany.txtboxInfo.AddItem (Msg)

    Rem Destroy Command Objects.
    Set cmdUpdateCompany = Nothing

    Rem Ask if they want to update another Company.
    Msg = " Do you wish to update another Company? " ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Update Another Company?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem Clear Form.
        Call subClearFormCompany
    Else
        Unload frmAddCompany
    End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub Form_Load()
    Rem Set up initial messages for the user.
    frmAddCompany.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS
Name"))
    frmAddCompany.txtboxInfo.AddItem ("Database Name: " & sDB)

```

```
frmAddCompany.WindowState = 2 'Maximized
End Sub
```

```
Private Sub Form_Resize()
    Rem Place Info box on form.
    Call subPlaceInfoBox(frmAddCompany.txtboxInfo, frmAddCompany.lblStatus, frmAddCompany)
```

```
    Rem Place Logo on Screen.
    Call subPlacePicBox(frmAddCompany.Picture1, frmAddCompany)
End Sub
```

### **frmAddJobDetail**

Option Explicit

```
Private Sub cmdAddJobDetail_Click()
    Dim cmdAddJobDetail As ADODB.Command
    Set cmdAddJobDetail = New ADODB.Command
    Rem Clear Error Collection.
    gConnection.Errors.Clear
```

On Error GoTo ErrorHandler

With cmdAddJobDetail

```
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_AddJobDetail"
    .Parameters.Refresh
    .Parameters("@CompanyName") = Trim(frmAddJobDetail.lstCompany.Text)
    .Parameters("@JobDescription") = Trim(frmAddJobDetail.txtDescription.Text)
    .Parameters("@Major1") = Trim(frmAddJobDetail.lstMajor1.Text)
    .Parameters("@Major2") = Trim(frmAddJobDetail.lstMajor2.Text)
    .Parameters("@Major3") = Trim(frmAddJobDetail.lstMajor3.Text)
```

End With

cmdAddJobDetail.Execute

Rem Successful Completion of Insert into Advisor Table.

If cmdAddJobDetail.Parameters(0) = 0 Then

```
    Msg = "Job for " & frmAddJobDetail.lstCompany.Text & _
    " has successfully been added to the Database." ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Job Added" ' Define title.
    R = MsgBox(Msg, Style, Title)
```

Rem Clear Form.

Call subClearFormJobDetail

Rem Add Completion to status area.

frmAddJobDetail.txtboxInfo.AddItem (Msg)

Rem destroy Command Objects.

Set cmdAddJobDetail = Nothing

Rem Ask if they want to add another Job.

Msg = " Do you wish to add another Job? " ' Define message.

Style = vbYesNo + vbQuestion

Title = "Add Another Job?" ' Define title.

R = MsgBox(Msg, Style, Title)

Rem If yes was selected msgbox returns 6.

If R = 6 Then

```

        Rem call company update routine.
        Call MDIForm_Production.mnuDBMaintenanceAddJobDetail_Click
    Else
        Unload Me
    End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub cmdClose_Click()
    Rem Free memory.
    Unload frmAddJobDetail
End Sub

Private Sub cmdUpdateJobDetail_Click()
    Dim cmdUpdateJobDetail As ADODB.Command
    Set cmdUpdateJobDetail = New ADODB.Command
    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdUpdateJobDetail
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_UpdateJobDetail"
        .Parameters.Refresh
        .Parameters("@JobID") = Trim(frmAddJobDetail.txtJobID.Text)
        .Parameters("@JobDescription") = Trim(frmAddJobDetail.txtDescription.Text)
        .Parameters("@Major1") = Trim(frmAddJobDetail.lstMajor1.Text)
        .Parameters("@Major2") = Trim(frmAddJobDetail.lstMajor2.Text)
        .Parameters("@Major3") = Trim(frmAddJobDetail.lstMajor3.Text)
    End With
    cmdUpdateJobDetail.Execute
    Rem Successful Completion of Insert into Advisor Table.
    If cmdUpdateJobDetail.Parameters(0) = 0 Then
        Msg = "Job ID: " & frmAddJobDetail.txtJobID.Text & _
            " has been successfully updated in the Database." ' Define message.
        Style = vbOKOnly + vbInformation
        Title = "Job Details Updated" ' Define title.
        R = MsgBox(Msg, Style, Title)

        Rem Add Completion to status area.
        frmAddJobDetail.txtboxInfo.AddItem (Msg)

        Rem destroy Command Objects.
        Set cmdUpdateJobDetail = Nothing

        Rem Ask if they want to update another Job.
        Msg = " Do you wish to update another Job? " ' Define message.
        Style = vbYesNo + vbQuestion
        Title = "Update Another Job?" ' Define title.
        R = MsgBox(Msg, Style, Title)
        Rem If yes was selected msgbox returns 6.

```

```

    If R = 6 Then
        Rem call company update routine.
        Call MDIForm_Production.mnuDBMaintenanceUpdateJobDetail_Click
    Else
        Unload Me
    End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub Form_Load()
    Rem Set up initial messages for the user.
    frmAddJobDetail.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS
Name"))
    frmAddJobDetail.txtboxInfo.AddItem ("Database Name: " & sDB)
    frmAddJobDetail.txtboxInfo.AddItem ("Item must be Hi-Lited for the value to be passed to the
database!")
    frmAddJobDetail.WindowState = 2 'Maximized
    Call frmAddJobDetailsSetup
End Sub

Private Sub Form_Resize()
    Rem Place Info box on form.
    Call subPlaceInfoBox(frmAddJobDetail.txtboxInfo, frmAddJobDetail.lblStatus, frmAddJobDetail)

    Rem Place Logo on Screen.
    Call subPlacePicBox(frmAddJobDetail.Picture1, frmAddJobDetail)
End Sub

frmAddMajor
Option Explicit

Private Sub cmdAddMajor_Click()
    Dim cmdAddMajor As ADODB.Command
    Set cmdAddMajor = New ADODB.Command
    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdAddMajor
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_AddMajor"
        .Parameters.Refresh
        .Parameters("@MajorID") = Trim(frmAddMajor.txtMajorID.ClipText)
        .Parameters("@Major") = Trim(frmAddMajor.txtMajor.Text)
        .Parameters("@AdvID") = Trim((Left(frmAddMajor.lstAdvisor.Text,
InStr(frmAddMajor.lstAdvisor.Text, " "))))
    End With
    cmdAddMajor.Execute
    Rem Successful Completion of Insert into Advisor Table.
    If cmdAddMajor.Parameters(0) = 0 Then
        Msg = frmAddMajor.txtMajorID.ClipText & _

```

```

" has successfully been added to the Database." ' Define message.
Style = vbOKOnly + vbInformation
Title = "Major Added" ' Define title.
R = MsgBox(Msg, Style, Title)

Rem Clear Form.
Call subClearFormMajor
Rem Add Completion to status area.
frmAddMajor.txtboxInfo.AddItem (Msg)

Rem destroy Command Objects.
Set cmdAddMajor = Nothing

Rem Ask if they want to Add another Major.
Msg = " Do you wish to add another Major?" ' Define message.
Style = vbYesNo + vbQuestion
Title = "Add Another Major?" ' Define title.
R = MsgBox(Msg, Style, Title)
Rem If yes was selected msgbox returns 6.
If R = 6 Then
    Rem call company update routine.
    Call MDIForm_Production.mnuDBMaintenanceAddMajor_Click
Else
    Unload Me
End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub cmdUpdateMajor_Click()
    Dim cmdUpdateMajor As ADODB.Command
    Set cmdUpdateMajor = New ADODB.Command
    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdUpdateMajor
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_UpdateMajor"
        .Parameters.Refresh
        .Parameters("@MajorID") = Trim(frmAddMajor.txtMajorID.ClipText)
        .Parameters("@Major") = Trim(frmAddMajor.txtMajor.Text)
        .Parameters("@AdvID") = Trim((Left(frmAddMajor.lstAdvisor.Text,
InStr(frmAddMajor.lstAdvisor.Text, " "))))
    End With
    cmdUpdateMajor.Execute
    Rem Successful Completion of Insert into Advisor Table.
    If cmdUpdateMajor.Parameters(0) = 0 Then
        Msg = frmAddMajor.txtMajorID.ClipText & _
        " has been successfully updated in the Database." ' Define message.
        Style = vbOKOnly + vbInformation
        Title = "Major Updated" ' Define title.

```

```

R = MsgBox(Msg, Style, Title)

Rem Add Completion to status area.
frmAddMajor.txtboxInfo.AddItem (Msg)

Rem destroy Command Objects.
Set cmdUpdateMajor = Nothing

Rem Ask if they want to update another Major.
Msg = " Do you wish to update another Major? " ' Define message.
Style = vbYesNo + vbQuestion
Title = "Update Another Major?" ' Define title.
R = MsgBox(Msg, Style, Title)
Rem If yes was selected msgbox returns 6.
If R = 6 Then
    Call subClearFormMajor
    Rem call company update routine.
    Call MDIForm_Production.mnuDBMaintenanceUpdateMajor_Click
Else
    Unload Me
End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub cmdClose_Click()
    Rem remove from memory.
    Unload frmAddMajor
End Sub

Private Sub Form_Load()
    Rem Set up initial messages for the user.
    frmAddMajor.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS Name"))
    frmAddMajor.txtboxInfo.AddItem ("Database Name: " & sDB)
    frmAddMajor.txtboxInfo.AddItem ("Item must be Hi-Lited for the value to be passed to the database!")
    frmAddMajor.WindowState = 2 'Maximized
    Call subPopulateAdvisors(frmAddMajor.lstAdvisor)
    frmAddMajor.lstAdvisor.Text = frmAddMajor.lstAdvisor.List(0)
End Sub

Private Sub Form_Resize()
    Rem Place Info box on form.
    Call subPlaceInfoBox(frmAddMajor.txtboxInfo, frmAddMajor.lblStatus, frmAddMajor)

    Rem Place Logo on Screen.
    Call subPlacePicBox(frmAddMajor.Picture1, frmAddMajor)
End Sub
frmAddQuarter
Option Explicit

Private Sub cmdAddQuarter_Click()
    Dim cmdAddQuarter As ADODB.Command
    Set cmdAddQuarter = New ADODB.Command
    Rem Clear Error Collection.

```

```

gConnection.Errors.Clear

On Error GoTo ErrorHandler
With cmdAddQuarter
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_AddQuarter"
    .Parameters.Refresh
    .Parameters("@QuarterID") = Trim(frmAddQuarter.txtQuarterID.ClipText)
    .Parameters("@QStart") = Trim(frmAddQuarter.txtStart.Text)
    .Parameters("@QEnd") = Trim(frmAddQuarter.txtEnd.Text)
End With
cmdAddQuarter.Execute
Rem Successful Completion of Insert into Advisor Table.
If cmdAddQuarter.Parameters(0) = 0 Then
    Msg = frmAddQuarter.txtQuarterID.Text & _
    " has successfully been added to the Database." ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Quarter Added" ' Define title.
    R = MsgBox(Msg, Style, Title)

    Rem Clear Form.
    Call subClearFormQuarter
    Rem Add Completion to status area.
    frmAddQuarter.txtboxInfo.AddItem (Msg)

    Rem destroy Command Objects.
    Set cmdAddQuarter = Nothing

    Rem Ask if they want to add another Quarter.
    Msg = " Do you wish to add another Quarter? " ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Add Another quarter?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem call company update routine.
        Call MDIForm_Production.mnuDBMaintenanceAddQuarter_Click
    Else
        Unload Me
    End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub cmdCloseQuarter_Click()
    Unload frmAddQuarter
End Sub

Private Sub cmdUpdateQuarter_Click()
    Dim cmdUpdateQuarter As ADODB.Command
    Set cmdUpdateQuarter = New ADODB.Command
    Rem Clear Error Collection.

```

```

gConnection.Errors.Clear

On Error GoTo ErrorHandler
With cmdUpdateQuarter
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_UpdateQuarter"
    .Parameters.Refresh
    .Parameters("@QuarterID") = Trim(frmAddQuarter.txtQuarterID.ClipText)
    .Parameters("@QStart") = Trim(frmAddQuarter.txtStart.Text)
    .Parameters("@QEnd") = Trim(frmAddQuarter.txtEnd.Text)
End With
cmdUpdateQuarter.Execute
Rem Successful Completion of Insert into quarters Table.
If cmdUpdateQuarter.Parameters(0) = 0 Then
    Msg = frmAddQuarter.txtQuarterID.Text & _
        " has successfully been updated in the Database." ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Advisor Info Updated" ' Define title.
    R = MsgBox(Msg, Style, Title)

    Rem Clear Form.
    Call subClearFormQuarter
    Rem Add Completion to status area.
    frmAddQuarter.txtboxInfo.AddItem (Msg)

    Rem destroy Command Objects.
    Set cmdUpdateQuarter = Nothing

    Rem Ask if they want to update another Company.
    Msg = " Do you wish to update another Quarter?" ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Update Another quarter?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem call company update routine.
        Call MDIForm_Production.mnuDBMaintenanceUpdateQuarter_Click
    Else
        Unload Me
    End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub Form_Load()
    Rem Set up initial messages for the user.
    frmAddQuarter.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS Name"))
    frmAddQuarter.txtboxInfo.AddItem ("Database Name: " & sDB)
    frmAddQuarter.WindowState = 2 'Maximized
End Sub

Private Sub Form_Resize()

```

```
Rem Place Info box on form.
Call subPlaceInfoBox(frmAddQuarter.txtboxInfo, frmAddQuarter.lblStatus, frmAddQuarter)
```

```
Rem Place Logo on Screen.
Call subPlacePictureBox(frmAddQuarter.Picture1, frmAddQuarter)
```

```
End Sub
```

### **frmAddStaff**

```
Option Explicit
```

```
Private Sub cmdAddAdv_Click()
```

```
Dim cmdAddAdv As ADODB.Command
Set cmdAddAdv = New ADODB.Command
Rem Clear Error Collection.
gConnection.Errors.Clear
```

```
On Error GoTo ErrorHandler
```

```
With cmdAddAdv
```

```
.ActiveConnection = gConnection
.CommandTimeout = 15
.CommandType = adCmdStoredProc
.CommandText = "stp_AddAdv"
.Parameters.Refresh
.Parameters("@AdvLName") = Trim(frmAddStaff.txtLName.Text)
.Parameters("@AdvFName") = Trim(frmAddStaff.txtFName.Text)
.Parameters("@AdvPhone") = Trim(frmAddStaff.txtPhone.ClipText)
.Parameters("@AdvFax") = Trim(frmAddStaff.txtFax.ClipText)
.Parameters("@AdvEmail") = Trim(frmAddStaff.txtEmail.Text)
```

```
End With
```

```
cmdAddAdv.Execute
```

```
Rem Successful Completion of Insert into Advisor Table.
```

```
If cmdAddAdv.Parameters(0) = 0 Then
```

```
Msg = frmAddStaff.txtFName.Text & " " & frmAddStaff.txtLName.Text & _
" has successfully been added to the Database." ' Define message.
Style = vbOKOnly + vbInformation
Title = "Advisor Added" ' Define title.
R = MsgBox(Msg, Style, Title)
```

```
Rem Add Completion to status area.
frmAddStaff.txtboxInfo.AddItem (Msg)
```

```
Rem destroy Command Objects.
```

```
Set cmdAddAdv = Nothing
```

```
Rem ask if this affects Majors.
```

```
Msg = " Do you wish to Update Major Details? " ' Define message.
```

```
Style = vbYesNo + vbQuestion
```

```
Title = "Update Major Details?" ' Define title.
```

```
R = MsgBox(Msg, Style, Title)
```

```
Rem If yes was selected msgbox returns 6.
```

```
If R = 6 Then
```

```
Rem Update Major Info.
```

```
Rem Load frmSelect in Company Mode.
```

```
SelectState = 4 'Major
```

```
frmSelect.Caption = "Select Major"
```

```
Call subPopulateMajors(frmSelect.ListBox1)
```

```
frmSelect.Show vbModal
```

```

End If
Else
    Rem Ask if they want to add another Company.
    Msg = " Do you wish to add another Advisor?" ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Add Another Advisor?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem Clear Form.
        Call subClearFormAdv
    Else
        Unload frmAddStaff
    End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub cmdAddClose_Click()
    Rem Remove form from memory.
    Unload frmAddStaff
End Sub

Private Sub cmdUpdateAdv_Click()
    Dim cmdUpdateAdv As ADODB.Command
    Set cmdUpdateAdv = New ADODB.Command
    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdUpdateAdv
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_UpdateAdv"
        .Parameters.Refresh
        .Parameters("@AdvID") = Trim(frmAddStaff.txtID.Text)
        .Parameters("@AdvLName") = Trim(frmAddStaff.txtLName.Text)
        .Parameters("@AdvFName") = Trim(frmAddStaff.txtFName.Text)
        .Parameters("@AdvPhone") = Trim(frmAddStaff.txtPhone.ClipText)
        .Parameters("@AdvFax") = Trim(frmAddStaff.txtFax.ClipText)
        .Parameters("@AdvEmail") = Trim(frmAddStaff.txtEmail.Text)
    End With
    cmdUpdateAdv.Execute
    Rem Successful Completion of Insert into Advisor Table.
    If cmdUpdateAdv.Parameters(0) = 0 Then
        Msg = frmAddStaff.txtFName.Text & " " & frmAddStaff.txtLName.Text & _
        " has successfully been updated in the Database." ' Define message.
        Style = vbOKOnly + vbInformation
        Title = "Advisor Info Updated" ' Define title.
        R = MsgBox(Msg, Style, Title)
        Rem Add Completion to status area.
        frmAddStaff.txtboxInfo.AddItem (Msg)
    End If
End Sub

```

```

Rem destroy Command Objects.
Set cmdUpdateAdv = Nothing

Rem Ask if they want to update another Company.
Msg = " Do you wish to update another Advisor?" ' Define message.
Style = vbYesNo + vbQuestion
Title = "Update Another Advisor?" ' Define title.
R = MsgBox(Msg, Style, Title)
Rem If yes was selected msgbox returns 6.
If R = 6 Then
    Rem call company update routine.
    Call MDIForm_Production.mnuDBMaintenanceUpdateAdvisor_Click
Else
    Unload Me
End If
End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Private Sub Form_Load()
    Rem Set up initial messages for the user.
    frmAddStaff.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS Name"))
    frmAddStaff.txtboxInfo.AddItem ("Database Name: " & sDB)

    frmAddStaff.WindowState = 2 'Maximized
End Sub

Private Sub Form_Resize()
    Rem Place Info box on form.
    Call subPlaceInfoBox(frmAddStaff.txtboxInfo, frmAddStaff.lblStatus, frmAddStaff)

    Rem Place Logo on Screen.
    Call subPlacePicBox(frmAddStaff.Picture1, frmAddStaff)
End Sub
frmChangePWD
Option Explicit

Private Sub cmdClose_Click()
    Rem Remove form from memory.
    Unload frmChangePWD
End Sub

Private Sub cmdChangePWD_Click()
    Rem Test that both Text boxes have info.
    If frmChangePWD.txtEnterPWD.Text <> "" And frmChangePWD.txtReEnterPWD.Text <> "" And _
    frmChangePWD.txtEnterPWD.Text = frmChangePWD.txtReEnterPWD.Text Then
        Rem Clear error collection.
        gConnection.Errors.Clear
        On Error GoTo ErrorHandler

        Rem Change default databse to master
        gConnection.DefaultDatabase = "master"

        Dim cmdChangePWD As ADODB.Command

```

```

Set cmdChangePWD = New ADODB.Command
With cmdChangePWD
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "sp_password"
    .Parameters.Refresh
    Rem Set parameter @old to NULL sp old password isn't necessary.
    .Parameters("@old") = Null
    .Parameters("@new") = Trim(frmChangePWD.txtEnterPWD.Text)
    .Parameters("@loginame") = Trim(frmChangePWD.txtUID.Text)
End With
cmdChangePWD.Execute
Rem Successful password change.
If cmdChangePWD.Parameters(0) = 0 Then
    Msg = " The Password has been changed for " & _
        frmChangePWD.txtUID.Text & " ." ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Password Changed" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Add message to status area.
    frmChangePWD.txtboxInfo.AddItem (Msg)

    Rem Destroy command object
    Set cmdChangePWD = Nothing

    Rem Ask if they want to change another password.
    Msg = "Do you want to change another password?" ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Change Another Password?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem Set text fields to empty.
        frmChangePWD.txtEnterPWD.Text = ""
        frmChangePWD.txtReEnterPWD.Text = ""
        frmChangePWD.txtUID.Text = ""
        Rem Set focus back to password input field.
        frmChangePWD.txtEnterPWD.SetFocus
    Else
        Unload frmChangePWD
    End If
End If
Else
    Msg = " The Password Text Fields are either empty or do not match. " & _
        " Please correct the values." ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "Password Change Failed" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Add message to status area.
    frmChangePWD.txtboxInfo.AddItem (Msg)
    Rem Set text fields to empty.
    frmChangePWD.txtEnterPWD.Text = ""
    frmChangePWD.txtReEnterPWD.Text = ""
    Rem Set focus back to password input field.
    frmChangePWD.txtEnterPWD.SetFocus

```

```

        End If

Exit Sub
ErrorHandler:
    Call ErrHandle(gConnection)
    Resume Next
End Sub

Private Sub Form_Load()
    Rem Set up initial messages for the user.
    frmChangePWD.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS
Name"))
    frmChangePWD.txtboxInfo.AddItem ("Database Name: " & sDB)
    frmChangePWD.txtboxInfo.AddItem ("Please Change Password.")
    frmChangePWD.WindowState = 2 'Maximized
End Sub

Private Sub Form_Resize()
    Rem Place Info box on form.
    Call subPlaceInfoBox(frmChangePWD.txtboxInfo, frmChangePWD.lblStatus, frmChangePWD)

    Rem Place Logo on Screen.
    Call subPlacePicBox(frmChangePWD.Picture1, frmChangePWD)
End Sub
frmDeleteAccount
Option Explicit

Private Sub cmdClose_Click()
    Rem Remove frmDeleteAccount from memory.
    Unload frmDeleteAccount
End Sub

Private Sub cmdDelete_Click()
    Rem Confirm Delete Request.
    Msg = " Do you wish to delete this account? " ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Delete Account?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem Clear error collection.
        gConnection.Errors.Clear
        On Error GoTo ErrorHandler
        gConnection.DefaultDatabase = "master"
        Dim cmdDropDB As ADODB.Command
        Set cmdDropDB = New ADODB.Command
        With cmdDropDB
            .ActiveConnection = gConnection
            .CommandTimeout = 15
            .CommandType = adCmdStoredProc
            .CommandText = "sp_dropuser"
            .Parameters.Refresh
            .Parameters("@name_in_db") = Trim(frmDeleteAccount.txtUID.Text)
        End With
        gConnection.DefaultDatabase = "Coop_Production1"
        cmdDropDB.Execute
    End If

```

```

Rem Test to see if current login has permission
If cmdDropDB.Parameters(0) = 1 Then 'Improper role for user.
    frmDeleteAccount.txtboxInfo.AddItem (Msg)
End If

Rem Test to see if user was deleted from database.
If cmdDropDB.Parameters(0) = 0 Then
    Rem Change default database to master
    gConnection.DefaultDatabase = "master"
    Dim cmdDeleteLogin As ADODB.Command
    Set cmdDeleteLogin = New ADODB.Command
    With cmdDeleteLogin
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "sp_droplogin"
        .Parameters.Refresh
        .Parameters("@loginame") = Trim(frmDeleteAccount.txtUID.Text)
    End With
    cmdDeleteLogin.Execute
    If cmdDeleteLogin.Parameters(0) = 0 Then 'Everything completed successfully
        Msg = " The SQL account has been deleted for " & _
            frmDeleteAccount.txtUID.Text & " ." ' Define message.
        Style = vbOKOnly + vbInformation
        Title = "Account Deleted" ' Define title.
        R = MsgBox(Msg, Style, Title)
        Rem Add message to status area.
        frmDeleteAccount.txtboxInfo.AddItem (Msg)
        Rem Set text fields to empty.
        frmDeleteAccount.txtUID.Text = ""
        Rem Set focus back to password input field.
        frmDeleteAccount.txtUID.SetFocus

        Rem Destroy command object
        Set cmdDeleteLogin = Nothing
        Set cmdDropDB = Nothing
        Rem Ask if they want to delete another account.
        Msg = "Do you want to delete another account?" ' Define message.
        Style = vbYesNo + vbQuestion
        Title = "Delete Another Account?" ' Define title.
        R = MsgBox(Msg, Style, Title)
        Rem If yes was selected msgbox returns 6.
        If R = 6 Then
            Rem Clear all the test boxes.
            frmDeleteAccount.txtUID = ""
            frmDeleteAccount.SetFocus
        Else
            Unload frmDeleteAccount
        End If
    End If
End If
Else
    frmDeleteAccount.txtboxInfo.AddItem ("Delete Request Cancelled!")
End If
Exit Sub

```

```
ErrorHandler:  
Call ErrHandle(gConnection)  
Resume Next  
End Sub
```

```
Private Sub Form_Load()  
    Rem Set up initial messages for the user.  
    frmDeleteAccount.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS  
Name"))  
    frmDeleteAccount.txtboxInfo.AddItem ("Database Name: " & sDB)  
    frmDeleteAccount.txtboxInfo.AddItem ("Please Select Account for Deletion.")  
    frmDeleteAccount.WindowState = 2 'Maximized  
End Sub
```

```
Private Sub Form_Resize()  
    Rem Place Info box on form.  
    Call subPlaceInfoBox(frmDeleteAccount.txtboxInfo, frmDeleteAccount.lblStatus, frmDeleteAccount)
```

```
    Rem Place Logo on Screen.  
    Call subPlacePicBox(frmDeleteAccount.Picture1, frmDeleteAccount)  
End Sub
```

### **frmDocument**

Option Explicit

Public objDocument As clsDocument

Dim CollKey As Integer

```
Private Sub Form_Activate()  
    Me.WindowState = 0 'Normal  
End Sub
```

```
Private Sub Form_Load()  
    Set objDocument = New clsDocument  
    MDIForm_Production.DocColl.Add objDocument, CStr(MDIForm_Production.CollKeyIndex)  
    CollKey = MDIForm_Production.CollKeyIndex  
    MDIForm_Production.CollKeyIndex = MDIForm_Production.CollKeyIndex + 1  
    Set objDocument.DocumentForm = Me  
    objDocument.DocName = Me.Caption  
    Me.WindowState = 0 'Normal
```

```
    Rem on initial form creation set up toolbar.  
    If MDIForm_Production.CollKeyIndex = 1 Then  
        MDIForm_Production.Toolbar.Visible = True  
        MDIForm_Production.Toolbar.Enabled = True  
    End If  
End Sub
```

```
Private Sub Form_Resize()  
    Me.txtDocument.Top = 0  
    Me.txtDocument.Left = 0  
    Me.txtDocument.Width = Me.ScaleWidth  
    Me.txtDocument.Height = Me.ScaleHeight  
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)  
    If objDocument.Changed Then  
        Select Case MsgBox("The text in the " & objDocument.DocName & _
```

```

        " file has changed." & Chr(13) & Chr(13) & _
        "Do you want to save the changes ?", _
        vbExclamation Or vbYesNoCancel, MDIForm_Production.Caption)
    Case vbYes
        Call subSaveDocument
    Case vbNo
        'Allow form to close
    Case vbCancel
        'Cancel close form
        Cancel = True
    End Select
End If
If Cancel = False Then
    objDocument.Closing = True
End If
If Cancel = False Then
    MDIForm_Production.DocColl.Remove CStr(CollKey)
    Set objDocument = Nothing
    Rem Hide Toolbar if no documents open.
    If CollKey = 0 Then
        MDIForm_Production.Toolbar.Visible = False
        MDIForm_Production.Toolbar.Enabled = False
        Rem reset control variable.
        MDIForm_Production.CollKeyIndex = 0
    End If
End If
End Sub

```

```

Private Sub txtDocument_KeyPress(KeyAscii As Integer)
    objDocument.Changed = True
End Sub

```

```

Private Sub txtDocument_SelChange()
    If Len(txtDocument.Text) > 0 Then
        objDocument.Changed = True
    End If
    Rem Setup active Toolbar.
    Call SetupToolBar
End Sub

```

### **frmErrors**

```

Option Explicit

```

```

Private Sub Form_Load()
    Rem Size and position form.
    Me.Width = Screen.Width * 0.5
    Me.Height = Screen.Height * 0.5
    Me.Left = (Screen.Width - Me.Width) * 0.5
    Me.Top = (Screen.Height - Me.Height) * 0.5
End Sub

```

```

Private Sub Form_Resize()
    Rem Resize form OLE.
    frmErrors.txtErrors.Width = frmErrors.ScaleWidth
    frmErrors.txtErrors.Height = frmErrors.ScaleHeight
    frmErrors.txtErrors.Left = frmErrors.ScaleLeft
    frmErrors.txtErrors.Top = frmErrors.ScaleTop

```

End Sub

**frmNewAccount**

Option Explicit

Dim sRole1 As String

Dim sRole2 As String

Dim sRole3 As String

Dim sRole1Tip As String

Dim sRole2Tip As String

Dim sRole3Tip As String

Private Sub cboRoles\_Click()

Rem Check for which group is selected so appropriate message can be displayed.

Select Case frmNewAccount.cboRoles.Text

Case sRole1

frmNewAccount.cboRoles.ToolTipText = sRole1Tip

Rem Set the Mask for students.

frmNewAccount.txtUID.Mask = "#####"

frmNewAccount.txtUID.MaxLength = 9

frmNewAccount.txtUID.PromptChar = "#"

frmNewAccount.txtUID.ToolTipText = "The student user ID is the Student's Social Security Number minus" & \_  
" the dashes."

Case sRole2

frmNewAccount.cboRoles.ToolTipText = sRole2Tip

Rem Set the Mask for companies.

frmNewAccount.txtUID.Mask = "CCCCCCCCCCCCCCCCCCCCCCCCCCCC"

frmNewAccount.txtUID.MaxLength = 25

frmNewAccount.txtUID.PromptChar = "\_"

frmNewAccount.txtUID.ToolTipText = "The Company user ID can be any combination of up to 25 characters."

Case sRole3

frmNewAccount.cboRoles.ToolTipText = sRole3Tip

Rem Set the Mask for staff.

frmNewAccount.txtUID.Mask = "CCCCCCCCCCCCCCCCCCCCCCCCCCCC"

frmNewAccount.txtUID.MaxLength = 25

frmNewAccount.txtUID.PromptChar = "\_"

frmNewAccount.txtUID.ToolTipText = "The staff user ID can be any combination of up to 25 characters."

Case Else

frmNewAccount.cboRoles.ToolTipText = "No Appropriate Group Chosen!"

End Select

Rem Set status area message!

frmNewAccount.txtboxInfo.AddItem (frmNewAccount.cboRoles.ToolTipText)

Rem Set focus to User ID.

frmNewAccount.txtUID.SetFocus

End Sub

Private Sub cmdClose\_Click()

Rem remove From Memory

Unload frmNewAccount

End Sub

Private Sub cmdCreate\_Click()

```
Rem Clear error collection.
gConnection.Errors.Clear
On Error GoTo ErrorHandler
```

```
Rem Change default database to master
gConnection.DefaultDatabase = "master"
```

```
Dim cmdCreateLogin As ADODB.Command
Set cmdCreateLogin = New ADODB.Command
With cmdCreateLogin
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "sp_addlogin"
    .Parameters.Refresh
    .Parameters("@loginame") = Trim(frmNewAccount.txtUID.Text)
    .Parameters("@passwd") = Trim(frmNewAccount.txtPWD.Text)
    .Parameters("@defdb") = sDB
End With
cmdCreateLogin.Execute
```

```
Rem Return value is stored in cmdCreateLogin.Parameters(0).
If cmdCreateLogin.Parameters(0) = 0 Then
    Dim cmdAddRole As ADODB.Command
    Set cmdAddRole = New ADODB.Command
    Rem Use select case to decide how to create database roles.
    Select Case cboRoles.Text
        Case sRole1 'Students
            With cmdAddRole
                .ActiveConnection = gConnection
                .CommandTimeout = 15
                .CommandType = adCmdStoredProc
                .CommandText = "sp_adduser"
                .Parameters.Refresh
                .Parameters("@loginame") = Trim(frmNewAccount.txtUID.Text)
                .Parameters("@name_in_db") = Trim(frmNewAccount.txtUID.Text)
                .Parameters("@grpname") = cboRoles.Text
            End With
            gConnection.DefaultDatabase = "Coop_Production1"
            cmdAddRole.Execute
```

```
        If cmdAddRole.Parameters(0) = 0 Then
            Msg = " User Account Created successfully for " & frmNewAccount.txtUID.Text _
                & " in the " & frmNewAccount.cboRoles.Text & " group." ' Define message.
            Style = vbOKOnly + vbInformation
            Title = "Account Created" ' Define title.
            R = MsgBox(Msg, Style, Title)
            Rem Add message to status area.
            frmNewAccount.txtboxInfo.AddItem (Msg)
            frmNewAccount.txtUID.Text = ""
            frmNewAccount.txtUID.SetFocus
            Rem Ask if they want to add another account.
            Call subAskQuestion
        Else
            Msg = " A problem occurred with the account creation for " & frmNewAccount.txtUID.Text _
                & " in the " & frmNewAccount.cboRoles.Text & " group." _
```

```

    & " Please contact the database administrator.' Define message."
    Style = vbOKOnly + vbExclamation
    Title = "Account Creation Failed" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Add message to status area.
    frmNewAccount.txtboxInfo.AddItem (Msg)
End If

```

```
Case sRole2 'Companies
```

```

    With cmdAddRole
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "sp_adduser"
        .Parameters.Refresh
        .Parameters("@loginame") = Trim(frmNewAccount.txtUID.Text)
        .Parameters("@name_in_db") = Trim(frmNewAccount.txtUID.Text)
        .Parameters("@grpname") = cboRoles.Text
    End With

```

```

gConnection.DefaultDatabase = "Coop_Production1"
cmdAddRole.Execute

```

```

If cmdAddRole.Parameters(0) = 0 Then
    Msg = " User Account Created successfully for " & frmNewAccount.txtUID.Text _
    & " in the " & frmNewAccount.cboRoles.Text & " group." ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Account Created" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Add message to status area.
    frmNewAccount.txtboxInfo.AddItem (Msg)
    Rem Clear Textboxes for reuse.
    frmNewAccount.txtUID.Text = ""
    Rem Set focus back to User ID for another user account.
    frmNewAccount.txtUID.SetFocus

```

```

    Rem Set up to add new Advisor to the database.
    Msg = " Do you wish to add a new Company to the Company database? " ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Add Advisor?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem Load frmAddCompany in Add mode.
        frmAddCompany.Caption = "Add Company to Database"
        frmAddCompany.cmdAddCo.Visible = True
        frmAddCompany.cmdAddCo.Enabled = True
        frmAddCompany.cmdUpdateCo.Visible = False
        frmAddCompany.Show
        frmAddCompany.SetFocus
    Else
        Rem Ask if they want to add another account.
        Call subAskQuestion
    End If

```

```

Else
    Msg = " A problem occurred with the account creation for " & frmNewAccount.txtUID.Text _
    & " in the " & frmNewAccount.cboRoles.Text & " group." _

```

```

& " Please contact the database administrator. ' Define message."
Style = vbOKOnly + vbExclamation
Title = "Account Creation Failed" ' Define title.
R = MsgBox(Msg, Style, Title)
Rem Add message to status area.
frmNewAccount.txtboxInfo.AddItem (Msg)
End If

```

Case sRole3 'Staff

```

Rem Staff need to have server roles beyond the other groups for admin work.
Dim cmdAddSrvRole As ADODB.Command
Set cmdAddSrvRole = New ADODB.Command

```

With cmdAddSrvRole

```

.ActiveConnection = gConnection
.CommandTimeout = 15
.CommandType = adCmdStoredProc
.CommandText = "sp_addsrvrolemember"
.Parameters.Refresh
.Parameters("@loginame") = Trim(frmNewAccount.txtUID.Text)
Rem Set server role to sysadmin for Staff.
.Parameters("@rolename") = "sysadmin"

```

End With

cmdAddSrvRole.Execute

With cmdAddRole

```

.ActiveConnection = gConnection
.CommandTimeout = 15
.CommandType = adCmdStoredProc
.CommandText = "sp_adduser"
.Parameters.Refresh
.Parameters("@loginame") = Trim(frmNewAccount.txtUID.Text)
.Parameters("@name_in_db") = Trim(frmNewAccount.txtUID.Text)
Rem Set staff to db_owner group because it is needed to create new logins.
.Parameters("@grpname") = "db_owner"

```

End With

gConnection.DefaultDatabase = "Coop\_Production1"

cmdAddRole.Execute

```

If cmdAddRole.Parameters(0) = 0 And cmdAddSrvRole.Parameters(0) = 0 Then
Msg = " User Account Created successfully for " & frmNewAccount.txtUID.Text _
& " in the " & frmNewAccount.cboRoles.Text & " group. ' Define message.

```

```

Style = vbOKOnly + vbInformation
Title = "Account Created" ' Define title.
R = MsgBox(Msg, Style, Title)
Rem Add message to status area.
frmNewAccount.txtboxInfo.AddItem (Msg)
Rem Clear Textboxes for reuse.
frmNewAccount.txtUID.Text = ""

```

Rem Set up to add new Advisor to the database.

```

Msg = " Do you wish to add " & frmNewAccount.txtUID.Text _
& " to the co-op advisor database? " ' Define message.

```

```

Style = vbYesNo + vbQuestion
Title = "Add Advisor?" ' Define title.

```

```

R = MsgBox(Msg, Style, Title)
Rem If yes was selected msgbox returns 6.

```

```

    If R = 6 Then
        Rem Load frmAddStaff in Add Mode.
        frmAddStaff.Caption = "Add Advisor to the Database"
        frmAddStaff.cmdAddAdv.Visible = True
        frmAddStaff.cmdAddAdv.Enabled = True
        frmAddStaff.cmdUpdateAdv.Visible = False

        Rem Clear all the test boxes.
        Call subClearFormAdv
        frmAddStaff.Show
        frmAddStaff.SetFocus
    Else
        Rem Ask if they want to add another account.
        Call subAskQuestion
    End If
Else
    Msg = " A problem occurred with the account creation for " & frmNewAccount.txtUID.Text _
    & " in the " & frmNewAccount.cboRoles.Text & " group." _
    & " Please contact the database administrator." ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "Account Creation Failed" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Add message to status area.
    frmNewAccount.txtboxInfo.AddItem (Msg)
End If
End Select
End If
Rem Destroy Command object.
Set cmdCreateLogin = Nothing
Set cmdAddRole = Nothing
Set cmdAddSrvRole = Nothing

Exit Sub
ErrorHandler:
    Call ErrHandle(gConnection)
    Resume Next
End Sub

Private Sub Form_Load()
    sRole1 = "Students"
    sRole2 = "Companies"
    sRole3 = "Staff"

    Rem Define Messages!
    sRole1Tip = "The Students group provides the necessary " & _
        "permissions for student users of this software."
    sRole2Tip = "The Companies group provides the necessary " & _
        "permissions for Employers who may want to use this software."
    sRole3Tip = "The Staff group provides the necessary " & _
        "permissions for office users of this software."

    Rem Get Role information from module definitions.
    frmNewAccount.cboRoles.AddItem (sRole1)
    frmNewAccount.cboRoles.AddItem (sRole2)
    frmNewAccount.cboRoles.AddItem (sRole3)

```

```

Rem set initial value for combo box.
frmNewAccount.cboRoles.Text = sRole1
frmNewAccount.cboRoles.ToolTipText = sRole1Tip

Rem Set up initial messages for the user.
frmNewAccount.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS
Name"))
frmNewAccount.txtboxInfo.AddItem ("Database Name: " & sDB)
frmNewAccount.txtboxInfo.AddItem (frmNewAccount.cboRoles.ToolTipText)

frmNewAccount.WindowState = 2 'Maximized

Rem Set the Mask for students.
frmNewAccount.txtUID.Mask = "#####"
frmNewAccount.txtUID.MaxLength = 9
frmNewAccount.txtUID.PromptChar = "#"
frmNewAccount.txtUID.ToolTipText = "The student user ID is the Student's Social Security Number
minus" & _
" the dashes."
End Sub

Private Sub Form_Resize()
Rem Place Info box on form.
Call subPlaceInfoBox(frmNewAccount.txtboxInfo, frmNewAccount.lblStatus, frmNewAccount)

Rem Place Logo on Screen.
Call subPlacePicBox(frmNewAccount.Picture1, frmNewAccount)
End Sub
frmODBCLogon
Option Explicit
Rem Declare variables for connection object.
Dim sConnect As String
Dim sADODConnect As String
Dim sDAODConnect As String
Dim sDSN As String

Private Sub cmdCancel_Click()
Unload Me
End
End Sub

Private Sub cmdOK_Click()
Rem Populate DSN list from GetDSNsAndDrivers.
If cboDSNList.ListIndex > 0 Then
sDSN = "DSN=" & cboDSNList.Text & ";"
sConnect = sConnect & "Server=" & txtServer.Text & ";"
Else
sConnect = sConnect & "Driver=" & cboDrivers.Text & ";"
sConnect = sConnect & "Server=" & txtServer.Text & ";"
End If

Rem Set up connection string.
sConnect = sConnect & "UID=" & txtUID.Text & ";"
sConnect = sConnect & "PWD=" & txtPWD.Text & ";"

```

```

Rem Check to see if database name is given if so add to connection string.
If Len(txtDatabase.Text) > 0 Then
    sConnect = sConnect & "Database=" & txtDatabase.Text & ";"
End If

Rem Complete connection string.
sADODConnect = "PROVIDER=SQLOLEDB;" & sDSN & sConnect

On Error GoTo ErrorHandler

Set gConnection = New ADODB.Connection
gConnection.Open sADODConnect

Rem Clear errors out of errors collection.
gConnection.Errors.Clear

Rem Test the status of the connection.
If gConnection.State = 1 Then
    Msg = "Connection Status: Connected" ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Connection Status" ' Define title.
    R = MsgBox(Msg, Style, Title)

    Rem Assign values to global variables.
    sDB = txtDatabase.Text
    sServer = txtServer.Text

    Rem Free memory used by login form.
    Unload frmODBCLogon

    Rem Activate Menus.
    MDIForm_Production.mnuAccounts.Enabled = True
    MDIForm_Production.mnuDBMaintenance.Enabled = True
    MDIForm_Production.mnuStudents.Enabled = True
    MDIForm_Production.mnuQueries.Enabled = True
End If
Exit Sub

ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub

Private Sub Form_Load()
    GetDSNsAndDrivers
End Sub

Private Sub cboDSNList_Click()
    On Error Resume Next
    If frmODBCLogon.cboDSNList.Text = "(None)" Then
        frmODBCLogon.txtServer.Enabled = True
        frmODBCLogon.cboDrivers.Enabled = True
        frmODBCLogon.cboDrivers.Text = "SQL Server"
    Else
        frmODBCLogon.txtServer.Enabled = False
        frmODBCLogon.cboDrivers.Enabled = False
    End If
End Sub

```

```

    End If
End Sub
frmQUpdateStudent
Option Explicit

Private Sub cmdAddQuarterStatus_Click()
    Dim cmdAddQuarterStatus As ADODB.Command
    Set cmdAddQuarterStatus = New ADODB.Command
    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdAddQuarterStatus
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_AddQuarterStatus"
        .Parameters.Refresh
        .Parameters("@STID") = Trim(frmQUpdateStudent.txtSTID.ClipText)
        .Parameters("@QuarterID") = Trim(frmQUpdateStudent.lstQuarter.Text)
        .Parameters("@Status") = Trim(frmQUpdateStudent.lstStatus.Text)
        .Parameters("@CompanyName") = Trim(frmQUpdateStudent.lstCompany.Text)
    End With
    cmdAddQuarterStatus.Execute
    Rem Successful Completion of Insert into Advisor Table.
    If cmdAddQuarterStatus.Parameters(0) = 0 Then
        Msg = frmAddMajor.txtMajorID.ClipText & _
        " has successfully been added to the Database." ' Define message.
        Style = vbOKOnly + vbInformation
        Title = "Major Added" ' Define title.
        R = MsgBox(Msg, Style, Title)

        Rem Clear Form.
        Call subClearFormMajor
        Rem Add Completion to status area.
        frmAddMajor.txtboxInfo.AddItem (Msg)

        Rem destroy Command Objects.
        Set cmdAddQuarterStatus = Nothing

        Rem Ask if they want to Add another Major.
        Msg = " Do you wish to add another Major? " ' Define message.
        Style = vbYesNo + vbQuestion
        Title = "Add Another Major?" ' Define title.
        R = MsgBox(Msg, Style, Title)
        Rem If yes was selected msgbox returns 6.
        If R = 6 Then
            Rem call company update routine.
            Call MDIForm_Production.mnuDBMaintenanceAddMajor_Click
        Else
            Unload Me
        End If
    End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)

```

End Sub

```
Private Sub cmdClose_Click()  
    Rem Remove from memory.  
    Unload frmQUpdateStudent  
End Sub
```

```
Private Sub Form_Load()  
    Rem Set up initial messages for the user.  
    frmQUpdateStudent.txtboxInfo.AddItem ("Database Provider: " & gConnection.Properties("DBMS  
Name"))  
    frmQUpdateStudent.txtboxInfo.AddItem ("Database Name: " & sDB)  
    frmQUpdateStudent.txtboxInfo.AddItem ("Item must be Hi-Lited for the value to be passed to the  
database!")  
    frmQUpdateStudent.WindowState = 2 'Maximized  
  
    Rem Populate the boxes.  
    Call subPopulateQuarter(frmQUpdateStudent.lstQuarter)  
    Call subPopulateCompany(frmQUpdateStudent.lstCompany)  
End Sub
```

```
Private Sub Form_Resize()  
    Rem Place Info box on form.  
    Call subPlaceInfoBox(frmQUpdateStudent.txtboxInfo, frmQUpdateStudent.lblStatus,  
frmQUpdateStudent)  
  
    Rem Place Logo on Screen.  
    Call subPlacePicBox(frmQUpdateStudent.Picture1, frmQUpdateStudent)  
End Sub
```

```
Private Sub lstStatus_Click()  
    Rem If coop is chosen make company info viisble.  
    If frmQUpdateStudent.lstStatus.Text = "Coop" Then  
        frmQUpdateStudent.lstCompany.Visible = True  
        frmQUpdateStudent.lblCompany.Visible = True  
    Else  
        frmQUpdateStudent.lstCompany.Visible = False  
        frmQUpdateStudent.lblCompany.Visible = False  
    End If  
End Sub
```

**frmSelect**  
Option Explicit

```
Private Sub cmdClose_Click()  
    Rem Remove frmselect from memory.  
    Unload frmSelect  
End Sub
```

```
Private Sub Form_Load()  
    Rem Place form on the screen.  
    frmSelect.Left = (Screen.Width - frmSelect.Width) * 0.5  
    frmSelect.Top = (Screen.Height - frmSelect.Height) * 0.5  
End Sub
```

```
Private Sub Form_Resize()  
    Rem Place List Box on form.
```

```

frmSelect.ListBox1.Width = frmSelect.ScaleWidth
frmSelect.ListBox1.Height = frmSelect.ScaleHeight * 0.75
frmSelect.ListBox1.Left = frmSelect.ScaleWidth - ListBox1.Width
frmSelect.ListBox1.Top = 0
Rem Place Dismiss button on the form.
frmSelect.cmdClose.Width = frmSelect.ScaleWidth * 0.75
frmSelect.cmdClose.Height = frmSelect.ScaleHeight * 0.15
frmSelect.cmdClose.Left = (frmSelect.ScaleWidth - frmSelect.cmdClose.Width) * 0.5
frmSelect.cmdClose.Top = frmSelect.ScaleHeight - frmSelect.cmdClose.Height
End Sub

```

```

Private Sub ListBox1_Click()
Rem Call Right Routine.
Select Case SelectState
Case 1 'Advisor Select
Call subPopulateAdvInfo
Case 2 'Company Select
Call subPopulateCompanyInfo
Case 3 'Quarters
Call subPopulateQuarterInfo
Case 4 'Majors
Call subPopulateMajorInfo
Case 5
Call subPopulateJobDetailInfo
End Select
End Sub

```

### **frmstudentDocs**

Option Explicit

```

Private Sub cmdClose_Click()
Rem Remove from memory.
Unload frmStudentDocs
End Sub

```

```

Private Sub cmdView_Click()
Dim cmdQueryStudents As ADODB.Command
Set cmdQueryStudents = New ADODB.Command
Dim rsQueryStudents As ADODB.Recordset
Set rsQueryStudents = New ADODB.Recordset

Rem clear form.
Call subClearFormStudentDocs

Rem Clear Error Collection.
gConnection.Errors.Clear

On Error GoTo ErrorHandler
With cmdQueryStudents
.ActiveConnection = gConnection
.CommandTimeout = 15
.CommandType = adCmdStoredProc
.CommandText = "stp_StudentsByLastName"
.Parameters.Refresh
.Parameters("@LName") = Trim(frmStudentDocs.txtSTName.Text)
End With
Set rsQueryStudents = cmdQueryStudents.Execute

```

```

Rem Successful Completion of Insert into Advisor Table.
If cmdQueryStudents.Parameters(0) = 0 Then
    Msg = "Please Click on the Student ID!" ' Define message.
    Style = vbOKOnly + vbInformation
    Title = "Select Student" ' Define title.
    R = MsgBox(Msg, Style, Title)
    If rsQueryStudents.EOF Then
        Msg = "There are no matches for your query!" & vbCrLf & _
            "Please submit another query." ' Define message.
        Style = vbOKOnly + vbInformation
        Title = "No Matches" ' Define title.
        R = MsgBox(Msg, Style, Title)
    Else
        While Not rsQueryStudents.EOF
            frmStudentDocs.lstStudent.AddItem (rsQueryStudents.Fields("StudID") & " " & vbCrLf &
rsQueryStudents.Fields("StudFirstName") & _
            vbCrLf & rsQueryStudents.Fields("StudLastName"))
            rsQueryStudents.MoveNext
        Wend
    End If
End If

```

```

Rem destroy command objects
Set cmdQueryStudents = Nothing
Set rsQueryStudents = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

```

```

Private Sub Form_Load()
    frmStudentDocs.WindowState = 2 'maximized
    Rem Set up tabs for listbox.
    Call SetListTabStops(frmStudentDocs.lstStudent.hWnd, 36, 116)
End Sub

```

```

Private Sub lstStudent_Click()
    Dim cmdQueryStudentDocs As ADODB.Command
    Set cmdQueryStudentDocs = New ADODB.Command
    Dim rsQueryStudentDocs As ADODB.Recordset
    Set rsQueryStudentDocs = New ADODB.Recordset

```

```

Rem Clear Error Collection.
gConnection.Errors.Clear

```

```

On Error GoTo ErrorHandler
With cmdQueryStudentDocs
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_StudentDoc"
    .Parameters.Refresh
    .Parameters("@STID") = Trim(Left(frmStudentDocs.lstStudent.Text,
InStr(frmStudentDocs.lstStudent.Text, " ")))
End With

```

```

Set rsQueryStudentDocs = cmdQueryStudentDocs.Execute
Rem Successful Completion of Insert into Advisor Table.
If cmdQueryStudentDocs.Parameters(0) = 0 Then
    Dim strQueryStudentResume As ADODB.Stream
    Set strQueryStudentResume = New ADODB.Stream
    strQueryStudentResume.Type = adTypeBinary
    Dim strQueryStudentPlan As ADODB.Stream
    Set strQueryStudentPlan = New ADODB.Stream
    strQueryStudentPlan.Type = adTypeBinary
    Dim ResumePath As String
    Dim PlanPath As String
    ResumePath = GetTmpPath() & _
    Trim(Left(frmStudentDocs.lstStudent.Text, InStr(frmStudentDocs.lstStudent.Text, " "))) & ".doc"
    PlanPath = GetTmpPath() & _
    Trim(Left(frmStudentDocs.lstStudent.Text, InStr(frmStudentDocs.lstStudent.Text, " "))) & ".xls"
    strQueryStudentResume.Open
    strQueryStudentResume.Write rsQueryStudentDocs.Fields("StudResume")
    strQueryStudentPlan.Open
    strQueryStudentPlan.Write rsQueryStudentDocs.Fields("StudDegreePlan")
    Rem Save file to Temp directory.
    strQueryStudentResume.SaveToFile ResumePath, adSaveCreateOverWrite
    strQueryStudentPlan.SaveToFile PlanPath, adSaveCreateOverWrite
    frmStudentDocs.OLEResume.CreateLink ResumePath
    frmStudentDocs.OLEPlan.CreateLink PlanPath
End If

```

```

Rem destroy command objects
Set cmdQueryStudentDocs = Nothing
Set rsQueryStudentDocs = Nothing
Set strQueryStudentResume = Nothing
Set strQueryStudentPlan = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

```

## Modules

Option Explicit

```

Public Sub subPopulateAdvisors(ListBox1 As ListBox)
    Rem clear contents of ListBox1.
    ListBox1.Clear

    Rem Set up tabs for listbox.
    Call SetListTabStops(frmStudentDocs.lstStudent.hWnd, 16)

    Rem Populate List Box.
    Dim cmdQuery As ADODB.Command
    Set cmdQuery = New ADODB.Command
    Dim rsQuery As ADODB.Recordset
    Set rsQuery = New ADODB.Recordset

    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler

```

```

With cmdQuery
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_Advisors"
End With
Set rsQuery = cmdQuery.Execute
While Not rsQuery.EOF
    ListBox1.AddItem (rsQuery.Fields("AdvID") & " " & vbTab & rsQuery.Fields("AdvLastName") & "
" & rsQuery.Fields("AdvFirstName"))
    rsQuery.MoveNext
Wend
Rem destroy Record Set and Command Object.
Set cmdQuery = Nothing
Set rsQuery = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub

```

```

Public Sub subClearFormAdv()
    frmAddStaff.txtID = ""
    frmAddStaff.txtLName.Text = ""
    frmAddStaff.txtFName.Text = ""
    frmAddStaff.txtPhone.Mask = ""
    frmAddStaff.txtPhone.Text = ""
    frmAddStaff.txtFax.Mask = ""
    frmAddStaff.txtFax.Text = ""
    frmAddStaff.txtEmail.Text = ""

    Rem Reset Masks.
    frmAddStaff.txtPhone.Mask = "(###)-###-####"
    frmAddStaff.txtFax.Mask = "(###)-###-####"
End Sub

```

```

Public Sub subPopulateAdvInfo()
    Rem Load frmAddStaff in Update Mode.
    frmAddStaff.Caption = "Update Advisor Details"
    frmAddStaff.cmdAddAdv.Visible = False
    frmAddStaff.cmdAddAdv.Enabled = False
    frmAddStaff.cmdUpdateAdv.Enabled = True
    frmAddStaff.cmdUpdateAdv.Left = 360
    frmAddStaff.cmdUpdateAdv.Top = 3360
    frmAddStaff.cmdUpdateAdv.Visible = True
    frmAddStaff.cmdUpdateAdv.Default = True
    frmAddStaff.txtID.Visible = True
    frmAddStaff.lblID.Visible = True

    Dim cmdUpdateQuery As ADODB.Command
    Set cmdUpdateQuery = New ADODB.Command

    Dim rsUpdateQuery As ADODB.Recordset
    Set rsUpdateQuery = New ADODB.Recordset

    Rem Clear Error Collection.

```

```
gConnection.Errors.Clear
```

```
On Error GoTo ErrorHandler
```

```
With cmdUpdateQuery
```

```
    .ActiveConnection = gConnection
```

```
    .CommandTimeout = 15
```

```
    .CommandType = adCmdStoredProc
```

```
    .CommandText = "stp_AdvisorByID"
```

```
    .Parameters.Refresh
```

```
    Rem Complicated Get the Left characters before the space between Last Name and first Name.
```

```
    .Parameters("@AdvID") = Trim(Left(frmSelect.ListBox1.Text, InStr(frmSelect.ListBox1.Text, " ")))
```

```
End With
```

```
Rem Close frmSelectcompany because it is vbModal.
```

```
frmSelect.Hide
```

```
Set rsUpdateQuery = cmdUpdateQuery.Execute
```

```
Rem Stop making calls to error handler because things are not formatted correctly in the DB.
```

```
On Error Resume Next
```

```
frmAddStaff.txtID.Text = rsUpdateQuery.Fields("AdvID")
```

```
frmAddStaff.txtLName.Text = rsUpdateQuery.Fields("AdvLastName")
```

```
frmAddStaff.txtFName.Text = rsUpdateQuery.Fields("AdvFirstName")
```

```
frmAddStaff.txtPhone.SelText = rsUpdateQuery.Fields("AdvPhone")
```

```
frmAddStaff.txtFax.SelText = rsUpdateQuery.Fields("AdvFax")
```

```
frmAddStaff.txtEmail.SelText = rsUpdateQuery.Fields("AdvEmail")
```

```
frmAddStaff.Show
```

```
frmAddStaff.SetFocus
```

```
Rem destroy Record Set and Command Object.
```

```
Set cmdUpdateQuery = Nothing
```

```
Set rsUpdateQuery = Nothing
```

```
Exit Sub
```

```
ErrorHandler:
```

```
Call ErrHandle(gConnection)
```

```
Resume Next
```

```
End Sub
```

```
Option Explicit
```

```
Public Sub subClearFormCompany()
```

```
    frmAddCompany.txtCoName.Text = ""
```

```
    frmAddCompany.txtCoStreet.Text = ""
```

```
    frmAddCompany.txtCoCity.Text = ""
```

```
    Rem Clear MaskedEdit Control.
```

```
    frmAddCompany.txtCoState.Mask = ""
```

```
    frmAddCompany.txtCoState.Text = ""
```

```
    frmAddCompany.txtCoZip.Mask = ""
```

```
    frmAddCompany.txtCoZip.Text = ""
```

```
    frmAddCompany.txtCoPhone.Mask = ""
```

```
    frmAddCompany.txtCoPhone.Text = ""
```

```
    frmAddCompany.txtCoWebPage.Text = ""
```

```
    frmAddCompany.txtCoRepLName.Text = ""
```

```
    frmAddCompany.txtCoRepFName.Text = ""
```

```
    frmAddCompany.txtCoRepPhone.Mask = ""
```

```
    frmAddCompany.txtCoRepPhone.Text = ""
```

```
    frmAddCompany.txtCoRepFax.Mask = ""
```

```
    frmAddCompany.txtCoRepFax.Text = ""
```

```

frmAddCompany.txtCoRepEmail.Text = ""

Rem Reset Masks.
frmAddCompany.txtCoState.Mask = ">??"
frmAddCompany.txtCoZip.Mask = "#####-9999"
frmAddCompany.txtCoPhone.Mask = "(###)-###-####"
frmAddCompany.txtCoRepPhone.Mask = "(###)-###-####"
frmAddCompany.txtCoRepFax.Mask = "(###)-###-####"
End Sub

Public Sub subPopulateCompany(ListBox1 As ListBox)
    Rem clear contents of ListBox1.
    ListBox1.Clear

    Rem Populate List Box.
    Dim cmdQuery As ADODB.Command
    Set cmdQuery = New ADODB.Command

    Dim rsQuery As ADODB.Recordset
    Set rsQuery = New ADODB.Recordset

    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdQuery
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_Employers"
    End With
    Set rsQuery = cmdQuery.Execute
    While Not rsQuery.EOF
        ListBox1.AddItem rsQuery.Fields("CompanyName")
        rsQuery.MoveNext
    Wend
    Rem destroy Record Set and Command Object.
    Set cmdQuery = Nothing
    Set rsQuery = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub

Public Sub subPopulateCompanyInfo()
    Rem Load frmAddCompany in Update Mode mode.
    frmAddCompany.Caption = "Update Company Details"
    frmAddCompany.cmdAddCo.Visible = False
    frmAddCompany.cmdAddCo.Enabled = False
    frmAddCompany.cmdUpdateCo.Left = 360
    frmAddCompany.cmdUpdateCo.Top = 4320
    frmAddCompany.cmdUpdateCo.Visible = True
    frmAddCompany.cmdUpdateCo.Enabled = True
    frmAddCompany.cmdUpdateCo.Default = True

```

```

Dim cmdUpdateQuery As ADODB.Command
Set cmdUpdateQuery = New ADODB.Command

Dim rsUpdateQuery As ADODB.Recordset
Set rsUpdateQuery = New ADODB.Recordset

Rem Clear Error Collection.
gConnection.Errors.Clear

On Error GoTo ErrorHandler
With cmdUpdateQuery
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_EmployersBycompanyName"
    .Parameters.Refresh
    .Parameters("@CompName") = Trim(frmSelect.ListBox1.Text)
End With
Rem Close frmSelectcompany because it is vbModal.
frmSelect.Hide

Set rsUpdateQuery = cmdUpdateQuery.Execute
Rem Stop making calls to error handler because things are not formatted correctly in the DB.
On Error Resume Next
frmAddCompany.txtCoName.Text = rsUpdateQuery.Fields("companyName")
frmAddCompany.txtCoStreet.Text = rsUpdateQuery.Fields("companyStreet")
frmAddCompany.txtCoCity.Text = rsUpdateQuery.Fields("companyCity")
frmAddCompany.txtCoState.SelText = rsUpdateQuery.Fields("companyState")
frmAddCompany.txtCoZip.SelText = rsUpdateQuery.Fields("companyZip")
frmAddCompany.txtCoPhone.SelText = rsUpdateQuery.Fields("companyPhone")
frmAddCompany.txtCoWebPage.Text = rsUpdateQuery.Fields("companyWebPage")
frmAddCompany.txtCoRepLName.Text = rsUpdateQuery.Fields("companyRepLastName")
frmAddCompany.txtCoRepFName.Text = rsUpdateQuery.Fields("companyRepFirstName")
frmAddCompany.txtCoRepPhone.SelText = rsUpdateQuery.Fields("companyRepPhone")
frmAddCompany.txtCoRepFax.SelText = rsUpdateQuery.Fields("companyRepFax")
frmAddCompany.txtCoRepEmail.Text = rsUpdateQuery.Fields("companyRepEmail")
frmAddCompany.Show
frmAddCompany.SetFocus

Rem destroy Record Set and Command Object.
Set cmdUpdateQuery = Nothing
Set rsUpdateQuery = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub
Option Explicit

Public Sub subClearFormJobDetail()
    frmAddJobDetail.txtDescription.Text = ""
    frmAddJobDetail.lstCompany.Clear
    frmAddJobDetail.lstMajor1.Clear
    frmAddJobDetail.lstMajor2.Clear
    frmAddJobDetail.lstMajor3.Clear
End Sub

```

```

Public Sub subPopulateJobDetails(ListBox1 As ListBox)
    Rem clear contents of ListBox1.
    ListBox1.Clear

    Rem Set up tabs for listbox.
    Call SetListTabStops(ListBox1.hWnd, 16)

    Rem Populate List Box.
    Dim cmdQuery As ADODB.Command
    Set cmdQuery = New ADODB.Command

    Dim rsQuery As ADODB.Recordset
    Set rsQuery = New ADODB.Recordset

    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdQuery
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_JobDetails"
    End With
    Set rsQuery = cmdQuery.Execute
    While Not rsQuery.EOF
        ListBox1.AddItem (rsQuery.Fields("JobID") & " " & vbTab & rsQuery.Fields("CompanyName"))
        rsQuery.MoveNext
    Wend
    Rem destroy Record Set and Command Object.
    Set cmdQuery = Nothing
    Set rsQuery = Nothing
Exit Sub
ErrorHandler:
    Call ErrHandle(gConnection)
    Resume Next
End Sub

Public Sub subPopulateJobDetailInfo()
    Rem Load frmAddMajor in Update Mode.
    frmAddJobDetail.Caption = "Update Job Details"
    frmAddJobDetail.cmdAddJobDetail.Visible = False
    frmAddJobDetail.cmdAddJobDetail.Enabled = False

    frmAddJobDetail.cmdUpdateJobDetail.Left = 360
    frmAddJobDetail.cmdUpdateJobDetail.Top = 4230
    frmAddJobDetail.cmdUpdateJobDetail.Visible = True
    frmAddJobDetail.cmdUpdateJobDetail.Default = True
    frmAddJobDetail.cmdUpdateJobDetail.Enabled = True
    frmAddJobDetail.lblJobID.Visible = True
    frmAddJobDetail.txtJobID.Visible = True

    Call subClearFormJobDetail

    Dim cmdUpdateQuery As ADODB.Command

```

```

Set cmdUpdateQuery = New ADODB.Command

Dim rsUpdateQuery As ADODB.Recordset
Set rsUpdateQuery = New ADODB.Recordset

Rem Clear Error Collection.
gConnection.Errors.Clear

On Error GoTo ErrorHandler
With cmdUpdateQuery
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_JobDetailByID"
    .Parameters.Refresh
    Rem Complicated Get the Left characters before the space between Last Name and first Name.
    .Parameters("@JobID") = Trim(Left(frmSelect.ListBox1.Text, InStr(frmSelect.ListBox1.Text, " ")))
End With
Rem Close frm Select because it is vbModal.
frmSelect.Hide

Set rsUpdateQuery = cmdUpdateQuery.Execute
Rem Stop making calls to error handler because things are not formatted correctly in the DB.
On Error Resume Next
Rem Test to see if lists have already been populated.
If frmAddJobDetail.lstCompany.ListCount = 0 Then
    Call frmAddJobDetailsSetup
End If
frmAddJobDetail.lstCompany.Text = (rsUpdateQuery.Fields("CompanyName"))
frmAddJobDetail.lstMajor1.Text = (rsUpdateQuery.Fields("Major1"))
frmAddJobDetail.lstMajor2.Text = (rsUpdateQuery.Fields("Major2"))
frmAddJobDetail.lstMajor3.Text = (rsUpdateQuery.Fields("Major3"))
frmAddJobDetail.txtJobID.Text = (rsUpdateQuery.Fields("JobID"))
frmAddJobDetail.txtDescription.Text = (rsUpdateQuery.Fields("JobDescription"))

frmAddJobDetail.Show
frmAddJobDetail.SetFocus

Rem destroy Record Set and Command Object.
Set cmdUpdateQuery = Nothing
Set rsUpdateQuery = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub

Public Sub frmAddJobDetailsSetup()
    Call subPopulateCompany(frmAddJobDetail.lstCompany)
    Call subPopulateMajors(frmAddJobDetail.lstMajor1)
    Dim i As Integer
    For i = 0 To frmAddJobDetail.lstMajor1.ListCount - 1
        frmAddJobDetail.lstMajor2.AddItem (frmAddJobDetail.lstMajor1.List(i))
        frmAddJobDetail.lstMajor3.AddItem (frmAddJobDetail.lstMajor1.List(i))
    Next i
    frmAddJobDetail.lstMajor1.Text = frmAddJobDetail.lstMajor1.List(0)

```

```
End Sub
Option Explicit
```

```
Public Sub subClearFormMajor()
    frmAddMajor.txtMajorID.Mask = ""
    frmAddMajor.txtMajorID.Text = ""
    frmAddMajor.txtMajor.Text = ""
    frmAddMajor.txtMajorID.Mask = ">??CCC"
End Sub
```

```
Public Sub subPopulateMajors(ListBox1 As ListBox)
    Rem clear contents of ListBox1.
    ListBox1.Clear

    Rem Populate List Box.
    Dim cmdQuery As ADODB.Command
    Set cmdQuery = New ADODB.Command

    Dim rsQuery As ADODB.Recordset
    Set rsQuery = New ADODB.Recordset

    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdQuery
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_Majors"
    End With
    Set rsQuery = cmdQuery.Execute
    While Not rsQuery.EOF
        ListBox1.AddItem (rsQuery.Fields("MajorID"))
        rsQuery.MoveNext
    Wend
    Rem destroy Record Set and Command Object.
    Set cmdQuery = Nothing
    Set rsQuery = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub
```

```
Public Sub subPopulateMajorInfo()
    Rem Load frmaddmajor in Update Mode.
    frmAddMajor.Caption = "Update Major Details"
    frmAddMajor.cmdAddMajor.Visible = False
    frmAddMajor.cmdAddMajor.Enabled = False
    frmAddMajor.cmdUpdateMajor.Enabled = True
    frmAddMajor.cmdUpdateMajor.Left = 360
    frmAddMajor.cmdUpdateMajor.Top = 3960
    frmAddMajor.cmdUpdateMajor.Visible = True
    frmAddMajor.cmdUpdateMajor.Default = True
```

```

Dim cmdUpdateQuery As ADODB.Command
Set cmdUpdateQuery = New ADODB.Command

Dim rsUpdateQuery As ADODB.Recordset
Set rsUpdateQuery = New ADODB.Recordset

Rem Clear Error Collection.
gConnection.Errors.Clear

On Error GoTo ErrorHandler
With cmdUpdateQuery
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_MajorByID"
    .Parameters.Refresh
    Rem Complicated Get the Left characters before the space between Last Name and first Name.
    .Parameters("@MajorID") = Trim(frmSelect.ListBox1.Text)
End With
Rem Close frmSelectcompany because it is vbModal.
frmSelect.Hide

Set rsUpdateQuery = cmdUpdateQuery.Execute
Rem Stop making calls to error handler because things are not formatted correctly in the DB.
On Error Resume Next
frmAddMajor.txtMajorID.SelText = rsUpdateQuery.Fields("MajorID")
frmAddMajor.txtMajor.Text = rsUpdateQuery.Fields("Major")
Rem Populate Advisor List Box.
Call subPopulateAdvisors(frmAddMajor.lstAdvisor)

frmAddMajor.Show
frmAddMajor.SetFocus

Rem destroy Record Set and Command Object.
Set cmdUpdateQuery = Nothing
Set rsUpdateQuery = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub

Public Sub subClearFormQuarter()
    frmAddQuarter.txtQuarterID.Mask = ""
    frmAddQuarter.txtQuarterID.Text = ""
    frmAddQuarter.txtStart.Mask = ""
    frmAddQuarter.txtStart.Text = ""
    frmAddQuarter.txtEnd.Mask = ""
    frmAddQuarter.txtEnd.Text = ""

    Rem Reset Masks.
    frmAddQuarter.txtQuarterID.Mask = ">?####"
    frmAddQuarter.txtStart.Mask = "##-##-####"
    frmAddQuarter.txtEnd.Mask = "##-##-####"
End Sub

Public Sub subPopulateQuarter(ListBox1 As ListBox)

```

```

Rem clear contents of ListBox1.
ListBox1.Clear

Rem Populate List Box.
Dim cmdQuery As ADODB.Command
Set cmdQuery = New ADODB.Command

Dim rsQuery As ADODB.Recordset
Set rsQuery = New ADODB.Recordset

Rem Clear Error Collection.
gConnection.Errors.Clear

On Error GoTo ErrorHandler
With cmdQuery
    .ActiveConnection = gConnection
    .CommandTimeout = 15
    .CommandType = adCmdStoredProc
    .CommandText = "stp_Quarters"
End With
Set rsQuery = cmdQuery.Execute
While Not rsQuery.EOF
    ListBox1.AddItem rsQuery.Fields("QuarterID")
    rsQuery.MoveNext
Wend
Rem destroy Record Set and Command Object.
Set cmdQuery = Nothing
Set rsQuery = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub

Public Sub subPopulateQuarterInfo()
    Rem set up Form for Update Mode.
    frmAddQuarter.Caption = "Update Quarter Details"
    frmAddQuarter.cmdAddQuarter.Visible = False
    frmAddQuarter.cmdAddQuarter.Enabled = False
    frmAddQuarter.cmdUpdateQuarter.Left = 360
    frmAddQuarter.cmdUpdateQuarter.Top = 2880
    frmAddQuarter.cmdUpdateQuarter.Visible = True
    frmAddQuarter.cmdUpdateQuarter.Enabled = True
    frmAddQuarter.cmdUpdateQuarter.Default = True

    Dim cmdUpdateQuery As ADODB.Command
    Set cmdUpdateQuery = New ADODB.Command

    Dim rsUpdateQuery As ADODB.Recordset
    Set rsUpdateQuery = New ADODB.Recordset

    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdUpdateQuery

```

```

.ActiveConnection = gConnection
.CommandTimeout = 15
.CommandType = adCmdStoredProc
.CommandText = "stp_QuarterByID"
.Parameters.Refresh
Rem Complicated Get the Left characters before the space between Last Name and first Name.
.Parameters("@QuarterID") = Trim(frmSelect.ListBox1.Text)
End With
Rem Close frmSelectcompany because it is vbModal.
frmSelect.Hide

Set rsUpdateQuery = cmdUpdateQuery.Execute
Rem Stop making calls to error handler because things are not formatted correctly in the DB.
On Error Resume Next
frmAddQuarter.txtQuarterID.SetText = rsUpdateQuery.Fields("QuarterID")
frmAddQuarter.txtStart.SetText = rsUpdateQuery.Fields("QStartDate")
frmAddQuarter.txtEnd.SetText = rsUpdateQuery.Fields("QEndDate")

frmAddQuarter.Show
frmAddQuarter.SetFocus

Rem destroy Record Set and Command Object.
Set cmdUpdateQuery = Nothing
Set rsUpdateQuery = Nothing
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
Resume Next
End Sub
Public Sub subClearFormQuarterStatus()
    frmQUpdateStudent.txtSTID.Mask = ""
    frmQUpdateStudent.txtSTID.Text = ""
    frmQUpdateStudent.lstCompany.Clear
    frmQUpdateStudent.lstQuarter.Clear
    Rem Reset Masks.
    frmQUpdateStudent.txtSTID.Mask = "#####"
End Sub
Option Explicit

Public Sub subClearFormStudentDocs()
    frmStudentDocs.txtSTName = ""
    frmStudentDocs.lstStudent.Clear
End Sub
Option Explicit
Private Declare Function SQLDataSources Lib "ODBC32.DLL" (ByVal henv&, ByVal fDirection%,
ByVal szDSN$, ByVal cbDSNMax%, pcbDSN%, ByVal szDescription$, ByVal cbDescriptionMax%,
pcbDescription%) As Integer
Private Declare Function SQLAllocEnv% Lib "ODBC32.DLL" (env&)
Const SQL_SUCCESS As Long = 0
Const SQL_FETCH_NEXT As Long = 1

Sub GetDSNsAndDrivers()
    Dim i As Integer
    Dim sDSNItem As String * 1024
    Dim sDRVItem As String * 1024
    Dim sDSN As String

```

```

Dim sDRV As String
Dim iDSNLen As Integer
Dim iDRVLen As Integer
Dim lHenv As Long      'handle to the environment

On Error Resume Next
frmODBCLogon.cboDSNList.AddItem "(None)"

'get the DSNs
If SQLAllocEnv(lHenv) <> -1 Then
    Do Until i <> SQL_SUCCESS
        sDSNItem = Space$(1024)
        sDRVItem = Space$(1024)
        i = SQLDataSources(lHenv, SQL_FETCH_NEXT, sDSNItem, 1024, iDSNLen, sDRVItem, 1024,
iDRVLen)
        sDSN = Left$(sDSNItem, iDSNLen)
        sDRV = Left$(sDRVItem, iDRVLen)

        If sDSN <> Space(iDSNLen) Then
            frmODBCLogon.cboDSNList.AddItem sDSN
            frmODBCLogon.cboDrivers.AddItem sDRV
        End If
    Loop
End If
'remove the dupes
If frmODBCLogon.cboDSNList.ListCount > 0 Then
    With frmODBCLogon.cboDrivers
        If .ListCount > 1 Then
            i = 0
            While i < .ListCount
                If .List(i) = .List(i + 1) Then
                    .RemoveItem (i)
                Else
                    i = i + 1
                End If
            Wend
        End If
    End With
End If
frmODBCLogon.cboDSNList.ListIndex = 0
End Sub
Option Explicit

Public Function AddForm() As frmDocument
    Set AddForm = New frmDocument
    AddForm.Show vbModeless
End Function

Public Sub subOpenDoc()
    Rem Create a new instance of frmDocument.
    Call AddForm
    Dim Cancel As Boolean
    On Error GoTo ErrorHandler
    Cancel = False
    MDIForm_Production.CommonDialog1.Filter = "Text files (*.txt)|*.txt" & _
        "Rich Text files (*.rtf)|*.rtf |All files (*.*)|*.*"

```

```

MDIForm_Production.CommonDialog1.CancelError = True
MDIForm_Production.CommonDialog1.Flags = cdlOFNHideReadOnly Or _
    cdlOFNFileMustExist
MDIForm_Production.CommonDialog1.ShowOpen
If Not Cancel Then
    If UCase(Right(MDIForm_Production.CommonDialog1.FileName, 3)) = "RTF" Then
        MDIForm_Production.ActiveForm.txtDocument.LoadFile
MDIForm_Production.CommonDialog1.FileName, rtfRTF
    Else
        MDIForm_Production.ActiveForm.txtDocument.LoadFile
MDIForm_Production.CommonDialog1.FileName, rtfText
    End If
    MDIForm_Production.ActiveForm.objDocument.DocName =
MDIForm_Production.CommonDialog1.FileName
    MDIForm_Production.ActiveForm.objDocument.Changed = False
End If
Exit Sub

```

```

ErrorHandler:
If Err.Number = cdlCancel Then
    Cancel = True
    Resume Next
End If
End
End Sub

```

```

Public Sub subSaveDocument()
    Rem Use commondialog control for save operations.
    Dim Cancel As Boolean
    On Error GoTo ErrorHandler
    Cancel = False
    MDIForm_Production.CommonDialog1.DefaultExt = ".txt"
    MDIForm_Production.CommonDialog1.Filter = "Text files (*.txt)|*.txt|" & _
        "Rich Text files (*.rtf)|*.rtf|All files (*.*)|*.*"
    MDIForm_Production.CommonDialog1.CancelError = True
    MDIForm_Production.CommonDialog1.Flags = cdlOFNHideReadOnly Or _
        cdlOFNOverwritePrompt
    MDIForm_Production.CommonDialog1.ShowSave
    If Not Cancel Then
        With MDIForm_Production.ActiveForm
            If UCase(Right(MDIForm_Production.CommonDialog1.FileName, 3)) = "RTF" Then
                .txtDocument.SaveFile MDIForm_Production.CommonDialog1.FileName, rtfRTF
            Else
                .txtDocument.SaveFile MDIForm_Production.CommonDialog1.FileName, rtfText
            End If
            .objDocument.DocName = MDIForm_Production.CommonDialog1.FileName
            .objDocument.Changed = False
        End With
    End If
Exit Sub

```

```

ErrorHandler:
If Err.Number = cdlCancel Then
    Cancel = True
    Resume Next
End If

```

End Sub

Public Sub subPrintDoc()

```
    Dim BeginPage, EndPage, NumCopies, i
    Rem Set Cancel to True
    MDIForm_Production.CommonDialog1.CancelError = True
    Rem set Printerdefault to true.
    MDIForm_Production.CommonDialog1.PrinterDefault = True
    On Error GoTo ErrHandler
    Rem Display the Print dialog box
    MDIForm_Production.CommonDialog1.ShowPrinter
    Rem Get user-selected values from the dialog box
    BeginPage = MDIForm_Production.CommonDialog1.FromPage
    EndPage = MDIForm_Production.CommonDialog1.ToPage
    NumCopies = MDIForm_Production.CommonDialog1.Copies
    For i = 1 To NumCopies
        Printer.Print MDIForm_Production.ActiveForm
    Next i
    Exit Sub
```

ErrHandler:

```
    ' User pressed the Cancel button
    Exit Sub
End Sub
```

Public Sub subCut()

```
    Rem Copy text to clipboard.
    Call subCopy
    Rem Delete text.
    Call DeleteSelectedText
```

End Sub

Public Sub subCopy()

```
    Rem Copy selected text to clipboard.
    Clipboard.SetText MDIForm_Production.ActiveForm.txtDocument.SelText
```

End Sub

Public Sub subPaste()

```
    Dim Text As String
    Dim ClipboardText As String
    Dim SelStart As Long
    If Clipboard.GetFormat(vbCFText) Then
        If MDIForm_Production.ActiveForm.txtDocument.SelLength > 0 Then
            DeleteSelectedText
        End If
        Text = MDIForm_Production.ActiveForm.txtDocument.Text
        SelStart = MDIForm_Production.ActiveForm.txtDocument.SelStart
        ClipboardText = Clipboard.GetText
        MDIForm_Production.ActiveForm.txtDocument.Text = Left(Text, SelStart) & ClipboardText &
        Right(Text, Len(Text) - SelStart)
        MDIForm_Production.ActiveForm.txtDocument.SelStart = SelStart
    End If
```

End Sub

Public Sub subColor()

```
    MDIForm_Production.CommonDialog1.Flags = cdlCCFullOpen
    MDIForm_Production.CommonDialog1.ShowColor
```

```
MDIForm_Production.ActiveForm.txtDocument.SelColor =  
MDIForm_Production.CommonDialog1.Color  
End Sub
```

```
Public Sub subFont()  
MDIForm_Production.CommonDialog1.Flags = cdlCFBoth Or cdlCFEffects  
MDIForm_Production.CommonDialog1.ShowFont  
With MDIForm_Production.ActiveForm.txtDocument  
.SelFontName = MDIForm_Production.CommonDialog1.FontName  
.SelFontSize = MDIForm_Production.CommonDialog1.FontSize  
.SelBold = MDIForm_Production.CommonDialog1.FontBold  
.SelItalic = MDIForm_Production.CommonDialog1.FontItalic  
.SelStrikeThru = MDIForm_Production.CommonDialog1.FontStrikethru  
.SelUnderline = MDIForm_Production.CommonDialog1.FontUnderline  
.SelColor = MDIForm_Production.CommonDialog1.Color  
End With  
End Sub
```

```
Public Sub DeleteSelectedText()  
Dim Text As String  
Dim SelStart As Long  
Dim SelLength As Long  
Text = MDIForm_Production.ActiveForm.txtDocument.Text  
SelStart = MDIForm_Production.ActiveForm.txtDocument.SelStart  
SelLength = MDIForm_Production.ActiveForm.txtDocument.SelLength  
MDIForm_Production.ActiveForm.txtDocument.Text = Left(Text, SelStart) & Right(Text, Len(Text) -  
(SelStart + SelLength))  
MDIForm_Production.ActiveForm.txtDocument.SelStart = SelStart  
End Sub
```

```
Public Sub SetupToolBar()  
Dim Found As Boolean  
Dim i As Integer  
For i = 0 To Forms.Count - 1  
If Forms(i).Name = "frmDocument" Then  
If Forms(i).objDocument.Closing = False Then  
Found = True  
Exit For  
End If  
End If  
Next i  
MDIForm_Production.Toolbar.Buttons("SaveDoc").Enabled = Found  
MDIForm_Production.Toolbar.Buttons("PrintDoc").Enabled = Found  
MDIForm_Production.Toolbar.Buttons("Find").Enabled = Found  
  
If Found = False Then  
MDIForm_Production.Toolbar.Buttons("EditCopy").Enabled = False  
MDIForm_Production.Toolbar.Buttons("EditCut").Enabled = False  
MDIForm_Production.Toolbar.Buttons("EditPaste").Enabled = False  
MDIForm_Production.Toolbar.Buttons("EditColor").Enabled = False  
MDIForm_Production.Toolbar.Buttons("EditFont").Enabled = False  
Else  
If MDIForm_Production.ActiveForm.txtDocument.SelLength > 0 Then  
MDIForm_Production.Toolbar.Buttons("EditCopy").Enabled = True  
MDIForm_Production.Toolbar.Buttons("EditCut").Enabled = True  
MDIForm_Production.Toolbar.Buttons("EditColor").Enabled = True
```

```

    MDIForm_Production.Toolbar.Buttons("EditFont").Enabled = True
Else
    MDIForm_Production.Toolbar.Buttons("EditCopy").Enabled = False
    MDIForm_Production.Toolbar.Buttons("EditCut").Enabled = False
    MDIForm_Production.Toolbar.Buttons("EditColor").Enabled = False
    MDIForm_Production.Toolbar.Buttons("EditFont").Enabled = False
End If
If Clipboard.GetFormat(vbCFText) Then
    MDIForm_Production.Toolbar.Buttons("EditPaste").Enabled = True
Else
    MDIForm_Production.Toolbar.Buttons("EditPaste").Enabled = False
End If
End If
End Sub
Option Explicit

Public Sub ErrHandle(errCollection As ADODB.Connection)
    ' Enumerate Errors collection and display
    ' properties of each Error object.
    For Each errLoop In errCollection.Errors
        Rem frmChangePWD Errors.
        Rem NativeError = 15007 Wrong User ID.
        If errLoop.NativeError = 15007 Then
            Msg = errLoop.Description & _
            " Please verify user ID is correct." ' Define message.
            Style = vbOKOnly + vbExclamation
            Title = "SQL Error Code: " & errLoop.NativeError & " Invalid User ID" ' Define title.
            R = MsgBox(Msg, Style, Title)
            Rem Set focus back to User ID.
            frmChangePWD.txtUID.Text = ""
            frmChangePWD.txtUID.SetFocus
            Rem Add message to status area.
            frmChangePWD.txtboxInfo.AddItem (Msg)
        Rem NativeError = 15210 User logged in has wrong role.
        ElseIf errLoop.NativeError = 15210 Then
            Msg = errLoop.Description & _
            " Please verify user ID is correct." ' Define message.
            Style = vbOKOnly + vbExclamation
            Title = "SQL Error Code: " & errLoop.NativeError & " Invalid Role to Perform Operation" ' Define
            title.
            R = MsgBox(Msg, Style, Title)
            Rem Set focus back to User ID.
            frmChangePWD.txtUID.Text = ""
            frmChangePWD.txtUID.SetFocus
            Rem Add message to status area.
            frmChangePWD.txtboxInfo.AddItem (Msg)

            Rem frmODBCLogon Errors.
            Rem NativeError = 18456 is a login error.
            ElseIf errLoop.NativeError = 18456 Then
                Msg = errLoop.Description & vbCr & _
                " Please Check User ID and Password for Accuracy!" ' Define message.
                Style = vbOKOnly + vbExclamation
                Title = "SQL Error Code: " & errLoop.NativeError & " Login Failure" ' Define title.
                R = MsgBox(Msg, Style, Title)
                frmODBCLogon.txtPWD.Text = ""

```

```

    frmODBCLogon.txtUID.SetFocus
Rem Error handling for not being able to connect to the server.
ElseIf errLoop.NativeError = 6 Then
    Msg = errLoop.Description & vbCr & _
    " Please Try to Log in again!" ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " SQL Server Not Available" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmODBCLogon.cmdOK.SetFocus
ElseIf errLoop.NativeError = 1326 Then
    Msg = errLoop.Description & vbCr & _
    " Please contact your database administrator!" ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " SQL Server Not Available" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmODBCLogon.cmdOK.SetFocus

    Rem frmNewAccount Errors.
Rem NativeError = 15025 login already exists.
ElseIf errLoop.NativeError = 15025 Then
    Msg = errLoop.Description & _
    " User ID already exists. Please use a unique ID!" ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Login Already Exists" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmNewAccount.txtUID.Text = ""
    frmNewAccount.txtUID.SetFocus
    Rem Add message to status area.
    frmNewAccount.txtboxInfo.AddItem (Msg)
Rem NativeError = 15006 invalid used ID.
ElseIf errLoop.NativeError = 15006 Then
    Msg = errLoop.Description & _
    " User ID does conform to SQL standards. Please use a different ID!" ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Invalid User ID" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmNewAccount.txtUID.SetFocus
    Rem Add message to status area.
    frmNewAccount.txtboxInfo.AddItem (Msg)
Rem NativeError = 15003 must be DBAdmin to create login accounts.
ElseIf errLoop.NativeError = 15003 Or errLoop.NativeError = 15000 Or _
    errLoop.NativeError = 15403 Then
    Msg = errLoop.Description & _
    " Please contact your Database Administrator!" ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Invalid User ID" ' Define title.
    R = MsgBox(Msg, Style, Title)
Rem NativeError = 15488 or 15341 Successful operations.
ElseIf errLoop.NativeError = 15488 Or errLoop.NativeError = 15341 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Invalid User ID" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Add message to status area.

```

```

frmNewAccount.txtboxInfo.AddItem (Msg)
Rem Clear UID field and set focus.
frmNewAccount.txtUID.SetFocus

Rem frmAddStaff Errors.
Rem NativeError 60000 Advisor Last Name Missing.
ElseIf errLoop.NativeError = 60000 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Advisor Last Name is Required" ' Define
title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddStaff.txtLName.SetFocus
Rem NativeError 60001 Advisor First Name Missing.
ElseIf errLoop.NativeError = 60001 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Advisor First Name is Required" ' Define
title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddStaff.txtFName.SetFocus
Rem NativeError 60002 Advisor Phone Missing.
ElseIf errLoop.NativeError = 60002 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Advisor Phone Number is Required" '
Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddStaff.txtPhone.SetFocus
Rem NativeError 60003 Advisor Fax Number Missing.
ElseIf errLoop.NativeError = 60003 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Advisor Fax Number is Required" ' Define
title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddStaff.txtFax.SetFocus
Rem NativeError 60004 Advisor Email Address Missing.
ElseIf errLoop.NativeError = 60004 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Advisor Email Address is Required" '
Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddStaff.txtEmail.SetFocus
ElseIf errLoop.NativeError = 60003 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Advisor Fax Number is Required" ' Define
title.
    R = MsgBox(Msg, Style, Title)

```

```

Rem NativeError 60004 Advisor Email Address Missing.
ElseIf errLoop.NativeError = 60004 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Advisor Email Address is Required" '
Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddStaff.txtEmail.SetFocus
    Rem Set focus back to User ID.
    frmAddStaff.txtFax.SetFocus
Rem NativeError 70003 Improper Format Email Address.
ElseIf errLoop.NativeError = 70003 Then
    Msg = errLoop.Description & _
    " Please Verify all Phone Numbers are in the correct format."
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Improper Format for Email Address" '
Define title.
    R = MsgBox(Msg, Style, Title)
Rem NativeError 70005 Improper Format for Advisor Phone Number.
ElseIf errLoop.NativeError = 70005 Then
    Msg = errLoop.Description & _
    " Please Verify Advisor Phone Number is in the correct format."
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Improper Format for Advisor Phone
Number" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmAddStaff.txtPhone.SetFocus

    Rem frmAddCompany Errors.
Rem NativeError 70006 Improper Format for Fax Number.
ElseIf errLoop.NativeError = 70006 Then
    Msg = errLoop.Description & _
    " Please Verify Advisor Phone Number is in the correct format."
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Improper Format for Advisor Fax Number"
' Define title.
    R = MsgBox(Msg, Style, Title)
    frmAddStaff.txtPhone.SetFocus

    Rem frmAddCompany Errors.
Rem NativeError 60010 Company Name Missing.
ElseIf errLoop.NativeError = 60010 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Company Name is Required" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddCompany.txtCoName.SetFocus
Rem NativeError 60011 Company Street Address Missing.
ElseIf errLoop.NativeError = 60011 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Company Street Address is Required" '
Define title.
    R = MsgBox(Msg, Style, Title)

```

```

Rem Set focus back to User ID.
frmAddCompany.txtCoStreet.SetFocus
Rem NativeError 60012 Company City Missing.
ElseIf errLoop.NativeError = 60012 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Company City is Required" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddCompany.txtCoCity.SetFocus
Rem NativeError 60013 Company State Missing.
ElseIf errLoop.NativeError = 60013 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Company State is Required" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddCompany.txtCoState.SetFocus
Rem NativeError 60014 Company Zip Missing.
ElseIf errLoop.NativeError = 60014 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Company Zip Code is Required" ' Define
title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddCompany.txtCoZip.SetFocus
Rem NativeError 70002 Improper Format for Company Phone Number.
ElseIf errLoop.NativeError = 70002 Then
    Msg = errLoop.Description & _
    " Please Verify Company Phone Number is in the correct format."
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Improper Format for Company Phone
Number" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmAddCompany.txtCoPhone.SetFocus
Rem NativeError 70014 Company Name Missing.
ElseIf errLoop.NativeError = 70014 Then
    Msg = errLoop.Description
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Wrong Format for Zip Code" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem Set focus back to User ID.
    frmAddCompany.txtCoZip.SetFocus

Rem frmDeleteAccount Errors.
Rem NativeError = 15008 user does not exist in the database.
ElseIf errLoop.NativeError = 15008 Then
    Msg = errLoop.Description & vbCr & _
    " Please Check User ID for Accuracy!" ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " User does not Exist" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmDeleteAccount.txtUID.Text = ""
    frmDeleteAccount.txtUID.SetFocus
    frmDeleteAccount.txtboxInfo.AddItem (Msg)

```

```

Rem frmAddQuarter Errors.
Rem The QuarterID is incorrect.
ElseIf errLoop.NativeError = 60026 Then
  Msg = errLoop.Description ' Define message.
  Style = vbOKOnly + vbExclamation
  Title = "SQL Error Code: " & errLoop.NativeError & " Quarter ID is Required" ' Define title.
  R = MsgBox(Msg, Style, Title)
  Rem clear QuarterID field.
  frmAddQuarter.txtQuarterID.Mask = ""
  frmAddQuarter.txtQuarterID.Text = ""
  frmAddQuarter.txtQuarterID.Mask = ">?#####"
  frmAddQuarter.txtQuarterID.SetFocus
  frmAddQuarter.txtboxInfo.AddItem (Msg)
Rem The QuarterID is incorrect.
ElseIf errLoop.NativeError = 70020 Then
  Msg = errLoop.Description & vbCr & _
  " Please check Quarter ID for Accuracy!" ' Define message.
  Style = vbOKOnly + vbExclamation
  Title = "SQL Error Code: " & errLoop.NativeError & " Wrong format for Quarter ID" ' Define
title.
  R = MsgBox(Msg, Style, Title)
  Rem clear QuarterID field.
  frmAddQuarter.txtQuarterID.Mask = ""
  frmAddQuarter.txtQuarterID.Text = ""
  frmAddQuarter.txtQuarterID.Mask = ">?#####"
  frmAddQuarter.txtQuarterID.SetFocus
  frmAddQuarter.txtboxInfo.AddItem (Msg)
Rem The QuarterID is incorrect.
ElseIf errLoop.NativeError = 2627 Then
  Msg = errLoop.Description & vbCr & "Primary Key already Exists in the Database." & vbCr & _
  "Please verify you entered the correct data." ' Define message.
  Style = vbOKOnly + vbExclamation
  Title = "SQL Error Code: " & errLoop.NativeError & " Major Already Exists" ' Define title.
  R = MsgBox(Msg, Style, Title)

Rem frmAddMajor errors.
Rem The MajorID is missing.
ElseIf errLoop.NativeError = 60030 Then
  Msg = errLoop.Description ' Define message.
  Style = vbOKOnly + vbExclamation
  Title = "SQL Error Code: " & errLoop.NativeError & " Major ID is Required" ' Define title.
  R = MsgBox(Msg, Style, Title)
  Rem clear QuarterID field.
  frmAddMajor.txtMajorID.SetFocus
  frmAddMajor.txtboxInfo.AddItem (Msg)
Rem MajorID is incorrect.
ElseIf errLoop.NativeError = 70030 Then
  Msg = errLoop.Description ' Define message.
  Style = vbOKOnly + vbExclamation
  Title = "SQL Error Code: " & errLoop.NativeError & " Major ID is in Wrong Format" ' Define
title.
  R = MsgBox(Msg, Style, Title)
  Rem clear QuarterID field.
  frmAddMajor.txtMajorID.SetFocus
  frmAddMajor.txtboxInfo.AddItem (Msg)

```

```

Rem The Major Desc. is missing.
ElseIf errLoop.NativeError = 60031 Then
    Msg = errLoop.Description ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Major Description is Required" ' Define
title.
    R = MsgBox(Msg, Style, Title)
    Rem clear QuarterID field.
    frmAddMajor.txtMajor.SetFocus
    frmAddMajor.txtboxInfo.AddItem (Msg)
Rem The AdvisorID is missing.
ElseIf errLoop.NativeError = 60032 Then
    Msg = errLoop.Description ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Advisor ID is Required" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem clear QuarterID field.
    frmAddMajor.lstAdvisor.SetFocus
    frmAddMajor.txtboxInfo.AddItem (Msg)

    Rem frmAddJobDetails Errors.
Rem The Company Name is required.
ElseIf errLoop.NativeError = 60040 Then
    Msg = errLoop.Description ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Company Name is Required" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmAddJobDetail.lstCompany.SetFocus
Rem The QuarterID is incorrect.
ElseIf errLoop.NativeError = 60042 Then
    Msg = errLoop.Description ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Job Description is Required" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmAddJobDetail.txtDescription.SetFocus

    Rem frmQUpdateStudent errors.
Rem The StudentID is missing.
ElseIf errLoop.NativeError = 60060 Then
    Msg = errLoop.Description ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Student ID Missing" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmQUpdateStudent.txtSTID.SetFocus
Rem The StudentID is incorrect.
ElseIf errLoop.NativeError = 60060 Then
    Msg = errLoop.Description ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "SQL Error Code: " & errLoop.NativeError & " Student ID Incorrect" ' Define title.
    R = MsgBox(Msg, Style, Title)
    frmQUpdateStudent.txtSTID.SetFocus
End If
Next
Rem frmStudentDocs Errors.
Rem The resume or degree plan is not in the database.
Rem Capture a Visual Basic Error.

```

```

If Err.Number = 3001 Then
    Msg = "The student has not uploaded their Resume or Degree Plan!" ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "Visual Basic Error Code: " & Err.Number & " Resume or Degree Plan Missing" ' Define
title.
    R = MsgBox(Msg, Style, Title)
End If

Rem frmAddQuarter Errors.
Rem The data in the date fields is incorrect.
Rem Capture a Visual Basic Error.
If Err.Number = 3421 Then
    Msg = Err.Description & vbCr & _
    "Please Check Dates for Accuracy!" & vbCr & _
    "The Dates should use the mm-dd-yy format for input." ' Define message.
    Style = vbOKOnly + vbExclamation
    Title = "Visual Basic Error Code: " & Err.Number & " Bad Input" ' Define title.
    R = MsgBox(Msg, Style, Title)
End If
End Sub
Option Explicit

Public gConnection As ADODB.Connection
Rem Variable to hold the default DB.
Public sDB As String
Rem Variable to hold default Server.
Public sServer As String

Rem Set up for error handling.
Public errLoop As ADODB.Error
Public strError As String

Rem Set up variables to be used for message boxes.
Public Msg, Style, Title As String
Public R As Long

Rem Control variables to determine state forms should be shown in.
Public SelectState As Integer

Public SelectAdvAddUpdate As Integer

Public SelectCoAddUpdate As Integer
Option Explicit
Rem Set up tabs for listboxes.
Public Declare Function SendMessage Lib _
"user32" Alias "SendMessageA" (ByVal hWnd As _
Long, ByVal wParam As Long, ByVal lParam As _
Long, lParam As Any) As Long
Public Const LB_SETTABSTOPS = &H192

Rem Get path to Temp directory.
Declare Function GetTempPath Lib "kernel32" Alias _
"GetTempPathA" (ByVal nBufferLength As Long, ByVal _
lpBuffer As String) As Long
Public Const MAX_PATH = 260

```

```

Public Sub subPlacePictureBox(PictureBox As PictureBox, Form1 As Form)
    Rem Place Logo on Screen.
    PictureBox.Left = Form1.ScaleWidth - PictureBox.Width
    PictureBox.Top = 10
End Sub

Public Sub subPlaceInfoBox(InfoBox As ListBox, Label1 As Label, Form1 As Form)
    Rem On resize event of the form resize info box.
    Rem Scalewidth gets the inside dimensions of the parent form.
    InfoBox.Width = Form1.ScaleWidth
    InfoBox.Height = Form1.ScaleHeight * 0.15
    InfoBox.Top = Form1.ScaleHeight - InfoBox.Height
    InfoBox.Left = Form1.ScaleWidth - InfoBox.Width
    Label1.Left = Form1.txtboxInfo.Left + 10
    Label1.Top = Form1.txtboxInfo.Top - Label1.Height
End Sub

Public Sub subAskQuestion()
    Rem Ask if they want to add another account.
    Msg = "Do you want to add another account?" ' Define message.
    Style = vbYesNo + vbQuestion
    Title = "Add Another Account?" ' Define title.
    R = MsgBox(Msg, Style, Title)
    Rem If yes was selected msgbox returns 6.
    If R = 6 Then
        Rem Clear all the test boxes.
        frmNewAccount.txtUID = ""
        frmNewAccount.txtPWD = "password"
        frmNewAccount.cboRoles.SetFocus
    Else
        Unload frmNewAccount
    End If
End Sub

Public Sub SetListTabStops(ListHandle As Long, _
    ParamArray ParmList() As Variant)
    Dim i As Long
    Dim ListTabs() As Long
    Dim NumColumns As Long

    ReDim ListTabs(UBound(ParmList))
    For i = 0 To UBound(ParmList)
        ListTabs(i) = ParmList(i)
    Next i
    NumColumns = UBound(ParmList) + 1

    Call SendMessage(ListHandle, LB_SETTABSTOPS, _
        NumColumns, ListTabs(0))
End Sub

Public Function GetTmpPath()
    Dim strFolder As String
    Dim lngResult As Long
    strFolder = String(MAX_PATH, 0)
    lngResult = GetTempPath(MAX_PATH, strFolder)
    If lngResult <> 0 Then

```

```

    GetTmpPath = Left(strFolder, InStr(strFolder, Chr(0)) - 1)
Else
    GetTmpPath = ""
End If
End Function
Option Explicit

Public Sub subQueryStudentAdvisor()
    Rem Create form.
    Call AddForm

    Dim cmdStudentAdvisor As ADODB.Command
    Set cmdStudentAdvisor = New ADODB.Command
    Dim rsStudentAdvisor As ADODB.Recordset
    Set rsStudentAdvisor = New ADODB.Recordset

    Rem Set up Header Information.
    Call subAddHeader(MDIForm_Production.ActiveForm.txtDocument)

    Rem Clear Error Collection.
    gConnection.Errors.Clear

    On Error GoTo ErrorHandler
    With cmdStudentAdvisor
        .ActiveConnection = gConnection
        .CommandTimeout = 15
        .CommandType = adCmdStoredProc
        .CommandText = "stp_QueryStudentAdvisor"
    End With
    Set rsStudentAdvisor = cmdStudentAdvisor.Execute
    Rem records exist.
    If Not rsStudentAdvisor.EOF Then
        Rem Set to left justify.
        'MDIForm_Production.ActiveForm.txtDocument.SelAlignment = rtfLeft
        While Not rsStudentAdvisor.EOF
            MDIForm_Production.ActiveForm.txtDocument.Text =
MDIForm_Production.ActiveForm.txtDocument.Text & _
                vbTab & rsStudentAdvisor.Fields("StudID") & " " & _
                rsStudentAdvisor.Fields("StudLastName") & " " & _
                rsStudentAdvisor.Fields("StudFirstName") & vbCrLf
            rsStudentAdvisor.MoveNext
        Wend
    End If
Exit Sub
ErrorHandler:
Call ErrHandle(gConnection)
End Sub

Public Sub subAddHeader(RichTextBox1 As RichTextBox)
    Dim sHeader As String
    sHeader = "OMI College of Applied Science" & vbCrLf & _
"Professional Practice and Career Placement Office" & vbCrLf & _
"2220 Victory Parkway" & vbCrLf & _
"Cincinnati, Ohio 45206-2822" & vbCrLf & vbCrLf
    Rem Select all the text for formatting.

```

```

RichTextBox1.Text = sHeader
RichTextBox1.SelStart = 0
RichTextBox1.SelLength = Len(sHeader)
RichTextBox1.SelFontSize = 18
'richtextbox1.Bold
RichTextBox1.SelAlignment = rtfCenter
End Sub
Option Explicit
Rem System Info set up.
' Reg Key Security Options...
Const READ_CONTROL = &H20000
Const KEY_QUERY_VALUE = &H1
Const KEY_SET_VALUE = &H2
Const KEY_CREATE_SUB_KEY = &H4
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10
Const KEY_CREATE_LINK = &H20
Const KEY_ALL_ACCESS = KEY_QUERY_VALUE + KEY_SET_VALUE + _
    KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + _
    KEY_NOTIFY + KEY_CREATE_LINK + READ_CONTROL

' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1          ' Unicode nul terminated string
Const REG_DWORD = 4      ' 32-bit number

Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"

Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey As
Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, ByRef
phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA" (ByVal hKey As
Long, ByVal lpValueName As String, ByVal lpReserved As Long, ByRef lpType As Long, ByVal lpData
As String, ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long
Public Sub StartSysInfo()
    On Error GoTo SysInfoErr

    Dim rc As Long
    Dim SysInfoPath As String

    ' Try To Get System Info Program Path\Name From Registry...
    If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO,
SysInfoPath) Then
        ' Try To Get System Info Program Path Only From Registry...
        ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC,
gREGVALSYSINFOLOC, SysInfoPath) Then
            ' Validate Existance Of Known 32 Bit File Version
            If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
                SysInfoPath = SysInfoPath & "\MSINFO32.EXE"

            ' Error - File Can Not Be Found...

```

```

Else
    GoTo SysInfoErr
End If
' Error - Registry Entry Can Not Be Found...
Else
    GoTo SysInfoErr
End If

Call Shell(SysInfoPath, vbNormalFocus)

Exit Sub
SysInfoErr:
    MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub

Public Function GetKeyValue(KeyRoot As Long, KeyName As String, SubKeyRef As String, ByRef
KeyVal As String) As Boolean
    Dim i As Long                ' Loop Counter
    Dim rc As Long               ' Return Code
    Dim hKey As Long             ' Handle To An Open Registry Key
    Dim hDepth As Long          '
    Dim KeyValType As Long       ' Data Type Of A Registry Key
    Dim tmpVal As String         ' Tempory Storage For A Registry Key Value
    Dim KeyValSize As Long       ' Size Of Registry Key Variable
    '-----
    ' Open RegKey Under KeyRoot {HKEY_LOCAL_MACHINE...}
    '-----
    rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Open Registry Key

    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError    ' Handle Error...

    tmpVal = String$(1024, 0)                ' Allocate Variable Space
    KeyValSize = 1024                        ' Mark Variable Size

    '-----
    ' Retrieve Registry Key Value...
    '-----
    rc = RegQueryValueEx(hKey, SubKeyRef, 0, _
        KeyValType, tmpVal, KeyValSize) ' Get/Create Key Value

    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError    ' Handle Errors

    If (Asc(Mid(tmpVal, KeyValSize, 1)) = 0) Then    ' Win95 Adds Null Terminated String...
        tmpVal = Left(tmpVal, KeyValSize - 1)      ' Null Found, Extract From String
    Else                                            ' WinNT Does NOT Null Terminate String...
        tmpVal = Left(tmpVal, KeyValSize)          ' Null Not Found, Extract String Only
    End If

    '-----
    ' Determine Key Value Type For Conversion...
    '-----
    Select Case KeyValType                        ' Search Data Types...
    Case REG_SZ                                  ' String Registry Key Data Type
        KeyVal = tmpVal                          ' Copy String Value
    Case REG_DWORD                              ' Double Word Registry Key Data Type
        For i = Len(tmpVal) To 1 Step -1        ' Convert Each Bit
            KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1))) ' Build Value Char. By Char.
        Next i
    End Select

```

```

    Next
    KeyVal = Format("&h" + KeyVal)           ' Convert Double Word To String
End Select

GetKeyValue = True                         ' Return Success
rc = RegCloseKey(hKey)                    ' Close Registry Key
Exit Function                              ' Exit

GetKeyError: ' Cleanup After An Error Has Occured...
    KeyVal = ""                            ' Set Return Val To Empty String
    GetKeyValue = False                    ' Return Failure
    rc = RegCloseKey(hKey)                ' Close Registry Key
End Function

Class Module
Private DocChanged As Boolean
Public Closing As Boolean
Public DocumentForm As frmDocument
Public DocName As String

Public Property Get Changed() As Boolean
    Changed = DocChanged
End Property

Public Property Let Changed(NewValue As Boolean)
DocChanged = NewValue
If DocumentChanged = True Then
    If Right(DocumentForm.Caption, 2) <> " *" Then
        DocumentForm.Caption = DocName & " *"
    End If
Else
    If Right(DocumentForm.Caption, 2) = " *" Then
        DocumentForm.Caption = DocName
    End If
End If
End Property

```

## Appendix E. SQL Stored Procedures

### stp\_AddAdv

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Add an Advisor to the Database.*/
CREATE PROCEDURE stp_AddAdv @AdvLName nvarchar(25), @AdvFName nvarchar(20),
@AdvPhone nvarchar(15), @AdvFax nvarchar(15), @AdvEmail nvarchar(50) AS
IF @AdvLName IS NULL OR @AdvLName=""
BEGIN
    RAISERROR (60000,16,1)
    RETURN (1)
END
IF @AdvFName IS NULL OR @AdvFName=""
BEGIN
    RAISERROR( 60001,16,1)
    RETURN(2)
END
IF @AdvPhone IS NULL OR @AdvPhone=""
BEGIN
    RAISERROR(60002,16,1)
    RETURN (3)
END
IF @AdvPhone NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70005,16,1)
    RETURN (33)
END
IF @AdvFax IS NULL OR @AdvFax=""
BEGIN
    RAISERROR(60003,16,1)
    RETURN(4)
END
IF @AdvFax NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70006,16,2)
    RETURN(44)
END
IF @AdvEmail IS NULL OR @AdvEmail=""
BEGIN
    RAISERROR(60004,16,1)
    RETURN(4)
END
IF @AdvEmail NOT LIKE '%'+ '@'+ '%'
BEGIN
    RAISERROR(70003,16,1)
    RETURN(44)
END
INSERT INTO tblCoopAdvisors (AdvLastName, AdvFirstName, AdvPhone, AdvFax, AdvEmail)
VALUES (@AdvLName, @AdvFName, @AdvPhone, @AdvFax, @AdvEmail)
RETURN(0)
```

A. stp\_AddCompany

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Add Employer to the Database.*/
CREATE PROCEDURE stp_AddCompany @CoName nvarchar(25), @CoStreet nvarchar(25), @CoCity
nvarchar(20), @CoState nchar(2), @CoZip nvarchar(12),
@CoPhone nvarchar(15), @CoWebPage nvarchar(50), @CoRepLName nvarchar(25), @CoRepFName
nvarchar(20), @CoRepPhone nvarchar(15), @CoRepFax nvarchar(15),
@CoRepEmail nvarchar(50) AS
IF @CoName IS NULL OR @CoName=""
BEGIN
    RAISERROR (60010,16,1)
    RETURN (1)
END
IF @CoStreet IS NULL OR @CoStreet=""
BEGIN
    RAISERROR( 60011,16,1)
    RETURN(2)
END
IF @CoCity IS NULL OR @CoCity=""
BEGIN
    RAISERROR(60012,16,1)
    RETURN (3)
END
IF @CoState IS NULL OR @CoState=""
BEGIN
    RAISERROR(60013,16,1)
    RETURN (4)
END
IF @CoState NOT LIKE '[A-Z][A-Z]'
BEGIN
    RAISERROR(70013,16,1)
    RETURN (44)
END
IF @CoZip IS NULL OR @CoZip=""
BEGIN
    RAISERROR(60014,16,1)
    RETURN (5)
END
IF @CoZip NOT LIKE '[0-9][0-9][0-9][0-9]' AND @CoZip NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70014,16,1)
    RETURN (55)
END
IF @CoPhone IS NULL OR @CoPhone=""
BEGIN
    RAISERROR(60015,16,1)
    RETURN(6)
END
IF @CoPhone NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70002,16,2)
    RETURN(66)
END
```

```

INSERT INTO tblEmployers (CompanyName, CompanyStreet, CompanyCity, CompanyState,
CompanyZip, CompanyPhone, CompanyWebPage, CompanyRepLastName,
CompanyRepFirstName, CompanyRepPhone, CompanyRepFax,
CompanyRepEmail)
VALUES (@CoName, @CoStreet, @CoCity, UPPER(@CoState), @CoZip, @CoPhone, @CoWebPage,
@CoRepLName, @CoRepFName, @CoRepPhone, @CoRepFax, @CoRepEmail)
RETURN(0)

```

### **stp\_AddJobDetail**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Add Job Detail to the Database.*/
CREATE PROCEDURE stp_AddJobDetail @CompanyName nvarchar(25), @JobDescription
nvarchar(4000), @Major1 nvarchar(5), @Major2 nvarchar(5),
@Major3 nvarchar(5) AS
DECLARE @CompanyID int
IF @CompanyName IS NULL OR @CompanyName=""
BEGIN
    RAISERROR (60040,16,1)
    RETURN (1)
END
IF @JobDescription IS NULL OR @JobDescription=""
BEGIN
    RAISERROR(60042,16,1)
    RETURN (3)
END
IF @Major1 IS NULL OR @Major1=""
BEGIN
    RAISERROR( 60041,16,1)
    RETURN(2)
END
IF @Major1 NOT LIKE '[A-Z][A-Z]' AND @Major1 NOT LIKE '[A-Z][A-Z][A-Z]' AND @Major1 NOT
LIKE '[A-Z][A-Z][A-Z][A-Z]' AND @Major1 NOT LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z]'
BEGIN
    RAISERROR(70041,16,1)
    RETURN (11)
END
IF @Major2 NOT LIKE '[A-Z][A-Z]' AND @Major2 IS NOT NULL AND @Major2 <> "
BEGIN
    IF @Major2 NOT LIKE '[A-Z][A-Z]' AND @Major2 NOT LIKE '[A-Z][A-Z][A-Z]' AND
@Major2 NOT LIKE '[A-Z][A-Z][A-Z][A-Z]' AND @Major2 NOT LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z]'
    BEGIN
        RAISERROR(70041,16,1)
        RETURN (12)
    END
END
IF @Major3 IS NOT NULL AND @Major3 <> "
BEGIN
    IF @Major3 NOT LIKE '[A-Z][A-Z]' AND @Major3 NOT LIKE '[A-Z][A-Z][A-Z]' AND
@Major3 NOT LIKE '[A-Z][A-Z][A-Z][A-Z]' AND @Major3 NOT LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z]'
    BEGIN
        RAISERROR(70041,16,1)
        RETURN (13)
    END
END

```

```
END
SET @CompanyID = (SELECT CompanyID FROM tblEmployers WHERE CompanyName =
@CompanyName)
INSERT INTO tblJobDetails (CompanyID, JobDescription, Major1, Major2, Major3)
VALUES (@CompanyID, @JobDescription, @Major1, @Major2, @Major3)
RETURN(0)
```

### **stp\_AddMajor**

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Add Major to the Database.*/
CREATE PROCEDURE stp_AddMajor @MajorID nvarchar(5), @Major nvarchar(50), @AdvID int AS
IF @MajorID IS NULL OR @MajorID=""
BEGIN
    RAISERROR (60030,16,1)
    RETURN (1)
END
IF @MajorID NOT LIKE '[A-Z][A-Z]' AND @MajorID NOT LIKE '[A-Z][A-Z][A-Z]' AND @MajorID
NOT LIKE '[A-Z][A-Z][A-Z][A-Z]' AND @MajorID NOT LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z]'
BEGIN
    RAISERROR(70030,16,1)
    RETURN (11)
END
IF @Major IS NULL OR @Major=""
BEGIN
    RAISERROR( 60031,16,1)
    RETURN(2)
END
IF @AdvID IS NULL OR @AdvID=""
BEGIN
    RAISERROR(60032,16,1)
    RETURN (3)
END
INSERT INTO tblMajors (MajorID, Major, AdvID)
VALUES (@MajorID, @Major, @AdvID)
RETURN(0)
```

### **stp\_AddNewStudent**

```
/*Stored Procedure Written by Joe Bartels*/
/*Solutions Software for OCAS Coop Office*/
/*query Students by Last Name*/
CREATE PROCEDURE stp_AddNewStudent
    @StudID nvarchar(9) , @fname nvarchar(25), @mname nvarchar(20),
    @lname nvarchar(25), @lstreet nvarchar(25), @lcity nvarchar(20),
    @lstate nvarchar(2), @lzip nvarchar(12), @lphone nvarchar(14),
    @pstreet nvarchar(25), @pcity nvarchar(20),
    @pstate nvarchar(2), @pzip nvarchar(12), @pphone nvarchar(14),
    @email nvarchar(35), @startdate nvarchar(5), @graddate nvarchar(5),
    @majorid nvarchar(4),@approve nchar(1)
AS
IF @lstate NOT LIKE '[A-Z][A-Z]'
Begin
    RAISERROR(65000,16,1)
    RETURN(1)
End
IF @pstate NOT LIKE '[A-Z][A-Z]'
Begin
    RAISERROR(65000,16,1)
    RETURN(2)
End
```

```

IF @lzip NOT LIKE '[0-9][0-9][0-9][0-9][0-9]' AND @lzip NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(65001,16,1)
    RETURN (55)
END
IF @pzip NOT LIKE '[0-9][0-9][0-9][0-9][0-9]' AND @pzip NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(65001,16,1)
    RETURN (55)
END
IF @lphone NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]' AND @lphone NOT LIKE
'([0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]) AND @lphone !='
BEGIN
    RAISERROR(65002,16,1)
    RETURN(66)
END
IF @pphone NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]' AND @pphone NOT LIKE
'([0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]) AND @pphone !='
BEGIN
    RAISERROR(65002,16,1)
    RETURN(66)
END
INSERT INTO tblStudents
    (StudID,StudFirstName,StudLastName,StudMidName,
    StudSAStreet,StudSACity,StudSAState,StudSAZip,
    StudSAPhone,StudPermStreet,StudPermCity,
    StudPermState,StudPermZip,StudPermPhone,
    StudEmail,StudStartQuarter,StudGradQuarter,
    MajorID,StudApproved)
VALUES (@StudID,@fname,@lname,@mname,@lstreet,@lcity,UPPER(@lstate),@lzip,@lphone,
    @pstreet,@pcity,UPPER(@pstate),@pzip,@pphone,@email,@startdate,
    @graddate,@majorid,@approve)

```

### **stp\_AddQuarter**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Add an Quarter to the Database.*/
CREATE PROCEDURE stp_AddQuarter @QuarterID nchar(5), @QStart datetime, @QEnd datetime AS
IF @QuarterID IS NULL OR @QuarterID=""
BEGIN
    RAISERROR (60026,16,1)
    RETURN (1)
END
IF @QuarterID NOT LIKE '[A,W,S,U][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70020,16,1)
    RETURN (11)
END
IF @QStart IS NULL OR @QStart=""
BEGIN
    RAISERROR( 60021,16,1)
    RETURN(2)
END

```

```

IF @QEnd IS NULL OR @QEnd="
BEGIN
    RAISERROR(60022,16,1)
    RETURN (3)
END
INSERT INTO tblQuarters (QuarterID, QStartDate, QEndDate)
VALUES (@QuarterID, @QStart, @QEnd)
RETURN(0)

```

### **stp\_AddQuarterStatus**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Add Quarter Status to the Database.*/
CREATE PROCEDURE stp_AddQuarterStatus @STID nchar(9), @QuarterID nchar(5), @Status
nvarchar(7), @CompanyName nvarchar(25) AS
IF @STID IS NULL OR @STID="
BEGIN
    RAISERROR (60060,16,1)
    RETURN (1)
END
IF @STID NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70060,16,1)
    RETURN (11)
END
IF @QuarterID IS NULL OR @QuarterID="
BEGIN
    RAISERROR( 60061,16,1)
    RETURN(2)
END
IF @Status IS NULL OR @Status="
BEGIN
    RAISERROR(60062,16,1)
    RETURN (3)
END
INSERT INTO tblQuarterDetails (StudID, QuarterID, StudStatus, CompanyName)
VALUES (@StID, @QuarterID, @Status, @CompanyName)
RETURN(0)

```

### **stp\_AdvisorByID**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Employers by Name*/
CREATE PROCEDURE stp_AdvisorByID @AdvID int AS
IF @AdvID IS NULL OR @AdvID="
BEGIN
    RAISERROR (60016,16,1)
    RETURN (1)
END
SELECT * FROM tblCoopAdvisors
WHERE AdvID = @AdvID
RETURN (0)

```

### **stp\_AdvisorByLastName**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Employers by Name*/
CREATE PROCEDURE stp_AdvisorByLastName @AdvLName nvarchar(25) AS
IF @AdvLName IS NULL OR @AdvLName=""
BEGIN
    RAISERROR (60010,16,1)
    RETURN (1)
END
SELECT * FROM tblCoopAdvisors
WHERE AdvLastName LIKE @AdvLName
ORDER BY AdvLastName
RETURN (0)

```

### **stp\_Advisors**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Advisors.*/
CREATE PROCEDURE stp_Advisors AS
SELECT AdvID, AdvLastName, AdvFirstName FROM tblCoopAdvisors
ORDER BY AdvLastName

```

### **stp\_CheckStudent**

```

/*Stored Procedure Written by Joe Bartels*/
/*Solutions Software for OCAS Coop Office*/
/*query Students by Last Name*/
CREATE PROCEDURE stp_CheckStudent @StudID nvarchar(9) AS
Select StudID, StudFirstName, StudLastName FROM tblStudents
WHERE StudID = @StudID

```

### **stp\_Employers**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Employers.*/
CREATE PROCEDURE stp_Employers AS
SELECT CompanyID, CompanyName FROM tblEmployers
ORDER BY CompanyName

```

### **stp\_EmployersByCompanyName**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Employers by Name*/
CREATE PROCEDURE stp_EmployersByCompanyName @CompName nvarchar(25) AS
IF @CompName IS NULL OR @CompName=""
BEGIN
    RAISERROR (60010,16,1)
    RETURN (1)
END
SELECT * FROM tblEmployers
WHERE CompanyName LIKE @CompName
ORDER BY CompanyName

```

RETURN (0)

### **stp\_GetStudentDegree**

```
/*Stored Procedure Written by Joe Bartels*/  
/*Solutions Software for OCAS Coop Office*/  
/*query Students by Last Name*/  
CREATE PROCEDURE stp_GetStudentDegree @StudID nvarchar(9) AS  
Select StudID, StudDegreePlan FROM tblStudents  
WHERE StudID = @StudID
```

### **stp\_JobDetailByID**

```
/*Stored Procedure Written by Doug Troxell*/  
/*Solutions Software for OCAS Coop Office*/  
/*Query Job Details by ID*/  
CREATE PROCEDURE stp_JobDetailByID @JobID int AS  
IF @JobID IS NULL OR @JobID=""  
BEGIN  
    RAISERROR (60050,16,1)  
    RETURN (1)  
END  
SELECT tblJobDetails.JobID, tblJobDetails.JobDescription, tblJobDetails.Major1, tblJobDetails.Major2,  
tblJobDetails.Major3, tblEmployers.CompanyName  
FROM tblJobDetails INNER JOIN tblEmployers  
ON tblJobDetails.CompanyID = tblEmployers.CompanyID  
WHERE JobID = @JobID  
RETURN (0)
```

### **stp\_JobDetails**

```
/*Stored Procedure Written by Doug Troxell*/  
/*Solutions Software for OCAS Coop Office*/  
/*Query JobDetails*/  
CREATE PROCEDURE stp_JobDetails AS  
SELECT tblJobDetails.JobID, tblEmployers.CompanyName FROM tblJobDetails INNER JOIN  
tblEmployers  
ON tblJobDetails.CompanyID = tblEmployers.CompanyID  
ORDER BY tblEmployers.CompanyName
```

### **stp\_MajorByID**

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Employers by Name*/
CREATE PROCEDURE stp_MajorByID @MajorID nvarchar(5) AS
IF @MajorID IS NULL OR @MajorID=""
BEGIN
    RAISERROR (60030,16,1)
    RETURN (1)
END
SELECT * FROM tblMajors
WHERE MajorID = @MajorID
RETURN (0)
```

### **stp\_Majors**

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Advisors.*/
CREATE PROCEDURE stp_Majors AS
SELECT MajorID FROM tblMajors
ORDER BY MajorID
```

### **stp\_PopulateMajors**

```
/*Stored Procedure Written by Joe Bartels*/
/*Solutions Software for OCAS Coop Office*/
/*Query Students by Last Name*/
CREATE PROCEDURE stp_PopulateMajors AS
Select MajorID FROM tblMajors
```

### **stp\_QuarterByID**

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Employers by Name*/
CREATE PROCEDURE stp_QuarterByID @QuarterID nchar(5) AS
IF @QuarterID IS NULL OR @QuarterID=""
BEGIN
    RAISERROR (60026,16,1)
    RETURN (1)
END
SELECT * FROM tblQuarters
WHERE QuarterID = @QuarterID
RETURN (0)
```

### **stp\_Quarters**

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Advisors.*/
CREATE PROCEDURE stp_Quarters AS
SELECT QuarterID FROM tblQuarters
ORDER BY QuarterID DESC
```

### **stp\_QueryStudentAdvisor**

```

/*Stored Procedure Written by Joe Bartels*/
/*Solutions Software for OCAS Coop Office*/
/*query Students by Advisor*/
CREATE PROCEDURE stp_QueryStudentAdvisor AS
SELECT tblStudents.StudID, tblStudents.StudLastName, tblStudents.StudFirstName,
tblCoopAdvisors.AdvLastName, tblCoopAdvisors.advFirstName
FROM tblStudents INNER JOIN tblMajors
ON tblStudents.MajorID = tblMajors.MajorID INNER JOIN tblCoopAdvisors
ON tblMajors.AdvID = tblCoopAdvisors.AdvID
ORDER BY tblCoopAdvisors.AdvLastName

```

### **stp\_ShowStudentInfo**

```

/*Stored Procedure Written by Joe Bartels*/
/*Solutions Software for OCAS Coop Office*/
/*query Students by Last Name*/
CREATE PROCEDURE stp_ShowStudentInfo @StudID nchar(9) AS
Select * FROM tblStudents
WHERE StudID = @StudID

```

### **stp\_StudentDoc**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Students by StudID*/
CREATE PROCEDURE stp_StudentDoc @STID nchar(9) AS
SELECT StudResume, StudDegreePlan FROM tblStudents
WHERE StudID = @STID
ORDER BY StudLastName

```

### **stp\_StudentsByAdvisor**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Students by Advisor Name*/
CREATE PROCEDURE stp_StudentsByAdvisor @AdvID INT AS
SELECT StudLastName, StudFirstName, StudMidName FROM tblStudents
WHERE AdvID = @AdvID
ORDER BY StudLastName

```

### **stp\_StudentsByLastName**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*query Students by Last Name*/
CREATE PROCEDURE stp_StudentsByLastName @LName nvarchar(25) AS
Select StudID, StudLastName, StudFirstName, StudMidName FROM tblStudents
WHERE StudLastName LIKE '%' + @LName + '%'
ORDER BY StudLastName

```

### **stp\_StudentsByMajor**

```

/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Query Students by Major*/

```

```
CREATE PROCEDURE stp_StudentsByMajor @MajorID nvarchar (5) AS
SELECT StudLastName, StudFirstName, StudMidName FROM tblStudents
WHERE MajorID = @MajorID
ORDER BY StudLastName
```

### **stp\_UpdateAdv**

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Update Advisor Info in the Database.*/
CREATE PROCEDURE stp_UpdateAdv @AdvID int, @AdvLName nvarchar(25), @AdvFName
nvarchar(20), @AdvPhone nvarchar(15), @AdvFax nvarchar(15), @AdvEmail nvarchar(50) AS
IF @AdvID IS NULL OR @AdvLName=""
BEGIN
    RAISERROR (60016,16,1)
    RETURN (11)
END
IF @AdvLName IS NULL OR @AdvLName=""
BEGIN
    RAISERROR (60000,16,1)
    RETURN (1)
END
IF @AdvFName IS NULL OR @AdvFName=""
BEGIN
    RAISERROR( 60001,16,1)
    RETURN(2)
END
IF @AdvPhone IS NULL OR @AdvPhone=""
BEGIN
    RAISERROR(60002,16,1)
    RETURN (3)
END
IF @AdvPhone NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70005,16,1)
    RETURN (33)
END
IF @AdvFax IS NULL OR @AdvFax=""
BEGIN
    RAISERROR(60003,16,1)
    RETURN(4)
END
IF @AdvFax NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70006,16,2)
    RETURN(44)
END
IF @AdvEmail IS NULL OR @AdvEmail=""
BEGIN
    RAISERROR(60004,16,1)
    RETURN(4)
END
IF @AdvEmail NOT LIKE '%'+ '@'+ '%'
```

```
END
UPDATE tblCoopAdvisors
SET
    AdvLastName = @AdvLName,
    AdvFirstName = @AdvFName,
    AdvPhone = @AdvPhone,
    AdvFax = @AdvFax,
    AdvEmail = @AdvEmail
WHERE AdvID = @AdvID
RETURN(0)
```



```
UPDATE tblEmployers
```

```
Set CompanyStreet = @CoStreet,  
    CompanyCity = @CoCity,  
    CompanyState = UPPER(@CoState),  
    CompanyZip = @CoZip,  
    CompanyPhone = @CoPhone,  
    CompanyWebPage = @CoWebPage,  
    CompanyRepLastName = @CoRepLName,  
    CompanyRepFirstName = @CoRepFName,  
    CompanyRepPhone = @CoRepPhone,  
    CompanyRepFax = @CoRepPhone,  
    CompanyRepEmail = @CoRepEmail  
Where CompanyName = @CoName  
RETURN(0)
```

### **stp\_UpdateJobDetail**

```
/*Stored Procedure Written by Doug Troxell*/  
/*Solutions Software for OCAS Coop Office*/  
/*Update Job Detail to the Database.*/  
CREATE PROCEDURE stp_UpdateJobDetail @JobID nvarchar(25), @JobDescription nvarchar(4000),  
    @Major1 nvarchar(5), @Major2 nvarchar(5),  
    @Major3 nvarchar(5) AS  
DECLARE @CompanyID int  
IF @JobID IS NULL OR @JobID=""  
BEGIN  
    RAISERROR (60050,16,1)  
    RETURN (1)  
END  
IF @JobDescription IS NULL OR @JobDescription=""  
BEGIN  
    RAISERROR(60042,16,1)  
    RETURN (3)  
END  
IF @Major1 IS NULL OR @Major1=""  
BEGIN  
    RAISERROR( 60041,16,1)  
    RETURN(2)  
END  
IF @Major1 NOT LIKE '[A-Z][A-Z]' AND @Major1 NOT LIKE '[A-Z][A-Z][A-Z]' AND @Major1 NOT  
LIKE '[A-Z][A-Z][A-Z][A-Z]' AND @Major1 NOT LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z]'  
BEGIN  
    RAISERROR(70041,16,1)  
    RETURN (11)  
END  
IF @Major2 IS NOT NULL AND @Major2 <> "  
BEGIN  
    IF @Major2 NOT LIKE '[A-Z][A-Z]' AND @Major2 NOT LIKE '[A-Z][A-Z][A-Z]' AND  
@Major2 NOT LIKE '[A-Z][A-Z][A-Z][A-Z]' AND @Major2 NOT LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z]'  
    BEGIN  
        RAISERROR(70041,16,1)  
        RETURN (12)  
    END  
END  
END
```

```
IF @Major3 IS NOT NULL AND @Major3 <> "  
BEGIN  
    IF @Major3 NOT LIKE '[A-Z][A-Z]' AND @Major3 NOT LIKE '[A-Z][A-Z][A-Z]' AND  
@Major3 NOT LIKE '[A-Z][A-Z][A-Z][A-Z]' AND @Major3 NOT LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z]'  
    BEGIN  
        RAISERROR(70041,16,1)  
        RETURN (13)  
    END  
END  
UPDATE tblJobDetails  
SET  
    JobDescription = @JobDescription,  
    Major1 = @Major1,  
    Major2 = @Major2,  
    Major3 = @Major3  
WHERE JobID = @JobID  
RETURN(0)
```

### **stp\_UpdateMajor**

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Add Major to the Database.*/
CREATE PROCEDURE stp_UpdateMajor @MajorID nvarchar(5), @Major nvarchar(50), @AdvID int
AS
IF @MajorID IS NULL OR @MajorID=""
BEGIN
    RAISERROR (60030,16,1)
    RETURN (1)
END
IF @MajorID NOT LIKE '[A-Z][A-Z]' AND @MajorID NOT LIKE '[A-Z][A-Z][A-Z]' AND @MajorID
NOT LIKE '[A-Z][A-Z][A-Z][A-Z]' AND @MajorID NOT LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z]'
BEGIN
    RAISERROR(70030,16,1)
    RETURN (11)
END
IF @Major IS NULL OR @Major=""
BEGIN
    RAISERROR( 60031,16,1)
    RETURN(2)
END
IF @AdvID IS NULL OR @AdvID=""
BEGIN
    RAISERROR(60032,16,1)
    RETURN (3)
END
UPDATE tblMajors
SET
    MajorID = @MajorID,
    Major = @Major,
    AdvID = @AdvID
WHERE MajorID = @MajorID
RETURN(0)
```

### **stp\_UpdateQuarter**

```
/*Stored Procedure Written by Doug Troxell*/
/*Solutions Software for OCAS Coop Office*/
/*Update Quarter Info in the Database.*/
CREATE PROCEDURE stp_UpdateQuarter @QuarterID nchar(5), @QStart datetime, @QEnd datetime
AS
IF @QuarterID IS NULL OR @QuarterID=""
BEGIN
    RAISERROR (60026,16,1)
    RETURN (1)
END
IF @QuarterID NOT LIKE '[A,W,S,U][0-9][0-9][0-9]'
BEGIN
    RAISERROR(70020,16,1)
    RETURN (11)
END
IF @QStart IS NULL OR @QStart=""
BEGIN
    RAISERROR( 60021,16,1)
```

```

        RETURN(2)
    END
    IF @QEnd IS NULL OR @QEnd=""
    BEGIN
        RAISERROR(60022,16,1)
        RETURN (3)
    END
    UPDATE tblQuarters
    SET
        QuarterID = @QuarterID,
        QStartDate = @QStart,
        QEndDate = @QEnd
    WHERE QuarterID = @QuarterID
    RETURN(0)

```

### **stp\_UpdateStudentInfo**

```

/*Stored Procedure Written by Joe Bartels*/
/*Solutions Software for OCAS Coop Office*/
/*query Students by Last Name*/
CREATE PROCEDURE stp_UpdateStudentInfo
    @StudID nvarchar(9) , @fname nvarchar(25), @mname nvarchar(20),
    @lname nvarchar(25), @lstreet nvarchar(25), @lcity nvarchar(20),
    @lstate nvarchar(2), @lzip nvarchar(12), @lphone nvarchar(14),
    @pstreet nvarchar(25), @pcity nvarchar(20),
    @pstate nvarchar(2), @pzip nvarchar(12), @pphone nvarchar(14),
    @email nvarchar(35), @graddate nvarchar(5)
AS
IF @lstate NOT LIKE '[A-Z][A-Z]'
Begin
    RAISERROR(65000,16,1)
    RETURN(1) End
IF @pstate NOT LIKE '[A-Z][A-Z]'
Begin
    RAISERROR(65000,16,1)
    RETURN(2)
End
IF @lzip NOT LIKE '[0-9][0-9][0-9][0-9][0-9]' AND @lzip NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(65001,16,1)
    RETURN (55)
END
IF @pzip NOT LIKE '[0-9][0-9][0-9][0-9][0-9]' AND @pzip NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
BEGIN
    RAISERROR(65001,16,1)
    RETURN (55)
END
IF @lphone NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]' AND @lphone NOT LIKE
'([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'AND @lphone !="" AND @lphone IS NOT NULL
BEGIN
    RAISERROR(65002,16,1)
    RETURN(66)
END

```

```
IF @pphone NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]' AND @pphone NOT LIKE  
'([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'AND @pphone !=" AND @pphone IS NOT NULL  
BEGIN
```

```
    RAISERROR(65002,16,1)
```

```
    RETURN(66)
```

```
END
```

```
UPDATE tblStudents
```

```
    Set StudFirstName = @fname,
```

```
    StudLastName = @lname,
```

```
    StudMidName = @mname,
```

```
    StudSAStreet = @lstreet,
```

```
    StudSACity = @lcity,
```

```
    StudSAState = UPPER(@lstate),
```

```
    StudSAZip = @lzip,
```

```
    StudSAPhone = @lphone,
```

```
    StudPermStreet = @pstreet,
```

```
    StudPermCity = @pcity,
```

```
    StudPermState = UPPER(@pstate),
```

```
    StudPermZip = @pzip,
```

```
    StudPermPhone = @pphone,
```

```
    StudEmail = @email,
```

```
    StudGradQuarter = @graddate
```

```
WHERE StudID = @StudID
```

## Works Cited

18. Ashenfelter, John. *Choosing a Database for Your Web Site*. Somerset, New Jersey: Wiley Computer Publishing, 1998.
19. Brickley, Ron. Database Administrator, Structural Dynamics Research Corporation. Personal Interview. April 11, 2000.
20. Desai, Anil. *SQL Server 7 backup& recover*. New York: McGraw-Hill, 2000.
21. Dyck, Timothy. "DB2 7.1 Covers the Spectrum." [HTTP://www.zdnet.com/eweek/stories/general/0,11011,2550466,00.html](http://www.zdnet.com/eweek/stories/general/0,11011,2550466,00.html). April 17, 2000.
22. Dyck, Timothy. "Oracle 8i: Polished for Web". *PC Week*. March 6, 2000. 45 –53.
23. Feibus, Andy. "SQL Server 7.0: Room to Grow". *Informationweek*. February 1, 1999. 112 – 123.
24. Guengerich, Steve and Patrick Smith. *Client/Server Computing, Second Edition*. Indianapolis, Indiana: Sams, Macmillan Computer Publishing. 1994.
25. Hart, Ron. Adjunct Professor, Information Technology Program, University of Cincinnati. Personal Interview. April 11, 2000.
26. Imburgia, Kimberly. Program Coordinator, OCAS Professional Practice and Placement Office. Personal Interview. April 12, 2000.
27. Johnson, Amy Helen. "Web Application Development Tools". *Government Computer News*. September 13, 1999. 23 – 27.
28. Maheri, Anil. "Database Management – Database Security for the Web". *Information Systems Management*. Volume 16, Number 2 1999. 85.
29. Sinclair, Ian R. *Essentials of Computer Security*. London: Bernard Babani, 1997.
30. Son, S H. "Issues and Approaches to Supporting Timeliness and Security in Real-Time Database Systems". *Journal of Systems Architecture*. Volume 46, Number 4 2000. 397.
31. Soto, Carlos A. "FileMaker Pro is logical database pick". *Government Computer News*. April 24, 2000. 45 – 47.
32. Thuraisingham, Bhavani. *Handbook of Data Management*. Boca Raton, Florida: Auerbach Publications, CRC Press LLC. 1998.
33. Wood, Larry E. *User Interface Design: Bridging the Gap from User Requirements to Design*. Boca Raton, Florida: CRC Press, CRC Press LLC. 1997.
34. Young, Gerry. Database Technician, OCAS Professional Practice and Placement Office. Personal Interview. April 26, 2000.