Optometry Data System for The Sight Shop

By

Amy Hansel
And
Scot Rosenhoffer

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

June 2008

Optometry Data System for The Sight Shop

By

Amy Hansel
And
Scot Rosenhoffer

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

_____	_____
Amy Hansel	Date
_____	_____
Scot Rosenhoffer	Date
_____	_____
Hazem Said, PhD, Faculty Advisor	Date
_____	_____
Hazem Said, PhD, Department Head	Date

Table of Content

Table of Figures

ABSTRACT

The Optometry Data System for The Sight Shop is a software application built to warehouse both patient and business information. The application has been created to the specifications of the owner of *The Sight Shop*, Dr. Tim O'Leary. This is due to the lack of robust functionality of their current software, and lack of commercial software alternative on the market today for an Optometrist office. The software automates many manual processes performed in the office today by Dr. O'Leary and his staff, as well as incorporates aspects of their current software that they would like to keep using. The primary functionality is to maintain patient, exam, contact, billing, order and insurance information.

The software is built on Microsoft's .NET platform version 2.0. The primary software language is C#, and the database is Microsoft SQL 2005 Express. The application has been written in a modular form. This is to allow for easy updates and maintenance. The software is made for ease of use, and to provide an economical means to manage the functions of the optometry office.

Product Description and Intended Use

The Sight Shop is an optometry office owned by Dr. Tim O'Leary. The previous software system utilized by the optometry office is called MaximEyes and did not fulfill all the needs of the business.

There were several manual processes that Dr. O'Leary and his staff performed using their old system. Processes such as scheduling appointments and tracking patient orders, were done by hand, and were not handled by their old software. The process caused mistakes and confusion between the staff and their patients. Other manual processes utilized by the staff in the past include exporting data to their accounting software, QuickBooks and their third party insurance billing service, Zirmed. The processes to move data into these separate software packages were a time-consuming cut-and-paste process between the interfaces of the separate software.

Dr. O'Leary wanted the software that he and his staff use to be able to perform the existing functionality of MaximEyes and automate many of these manual processes that they are performing currently.

There are not many quality and affordable choices of software on the market to be utilized specifically by an optometrist office. Dr. O'Leary's old software MaximEyes is one of the few choices available, but its functionality is limited to medical and patient information. Upgrading the software is expensive and switching to new software runs the risk of creating further mistakes and incompatibility issues between systems.

As a result of these limitations we have created custom-built software, which fulfills the office's medical and business needs. The application can incorporate the

functionality of the business' old software as well as automate many of the old manual processes, saving both time and money for this office.

This application is written in C# on Microsoft's .NET 2.0 version platforms through Visual Studio. The database is contained in Microsoft SQL Server Express. The software implements the current functionality of MaximEyes, which includes eye exam data entry, patient profile information like medical history and contact information, and insurance information.

User Profiles

Each employee can have a user ID profile unique to that employee. Due to the small size of the business, and the fact that each employee performs a multitude of job functions, there is no division as to the authority each employee has to the system. However, there is an administrator function to add, delete and change users and provide for passwords for a system login. A user without administrative authority they cannot access these functions.

The receptionist's role is primarily to manage the patient appointments and update patient contact information. The optometrist office employees' roles include maintaining patient medical and insurance information, performing insurance and patient billing, as well as maintaining patient orders, vendor orders, and the associated vendor information. The doctor's primary role is performing the medical examinations, recording and measuring the optical medical data, as well as recording all patient related prescriptions and diagnoses.

Shown in figure 1 are the various job duties of the Sight Shop employees.

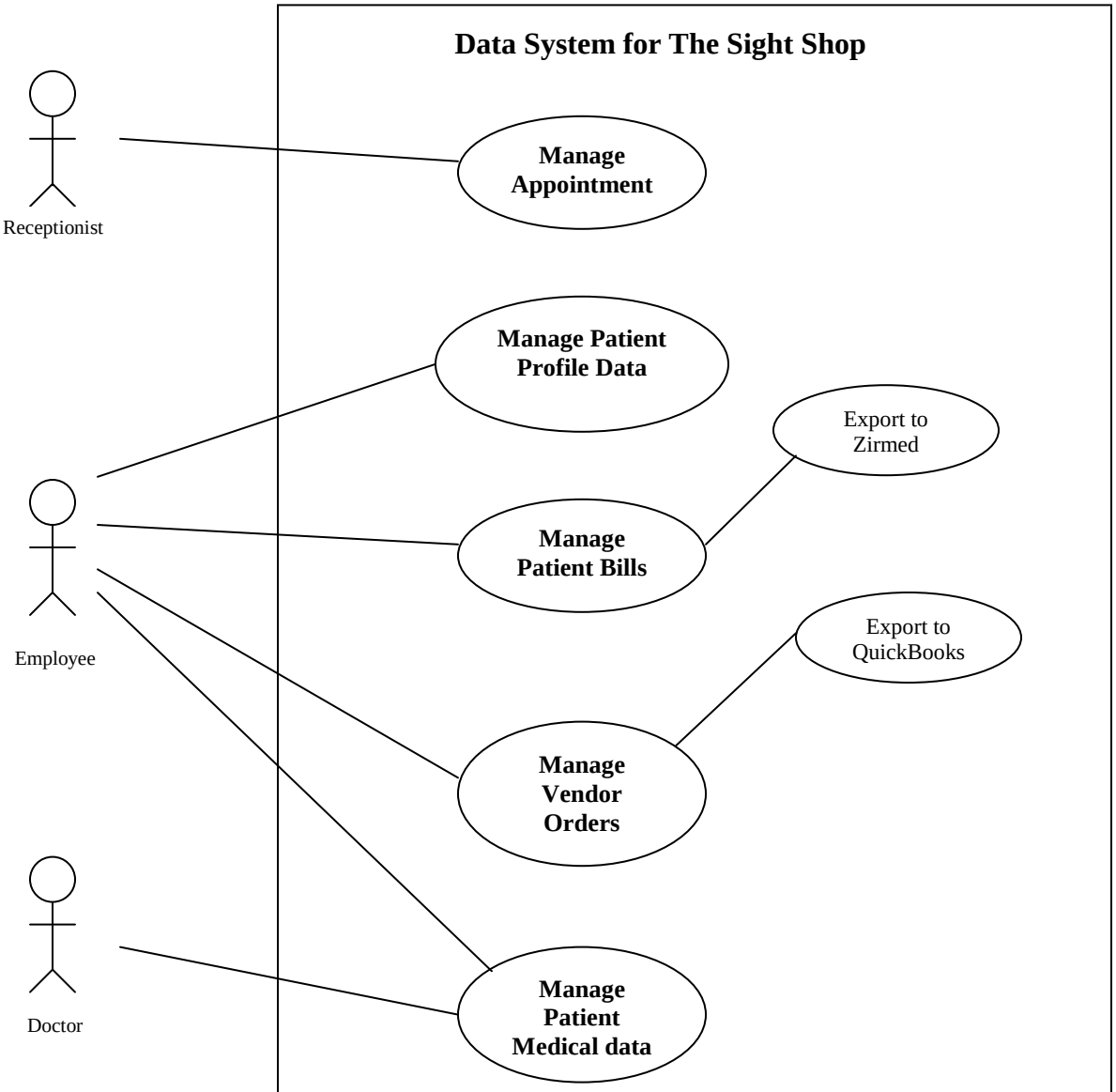


Figure 1 Use Case Diagram

This Use-Case diagram shows the interface between the employee users and the function of the computerized system.

Design Protocol

The system is constructed on Microsoft's .NET platform using C# as the primary programming language. The database is in Microsoft SQL 2005 Express. This is a stable software and database for creating a data entry software application.

Software

Three primary namespaces are used by the software:

1. A connection namespace that contains all the classes to connect, get, and update the data.
2. A business namespace that contains all the classes to interface with the connection classes and the presentation classes. The business namespace provides additional methods that can be used by all the classes such as field validation.
3. The presentation namespace that presents the forms classes that the user will see to interact with the system.

Class abstraction is used to simplify construction and debugging. Figure 2 shows a graphical representation of how the namespace configuration interact with each other and the database.

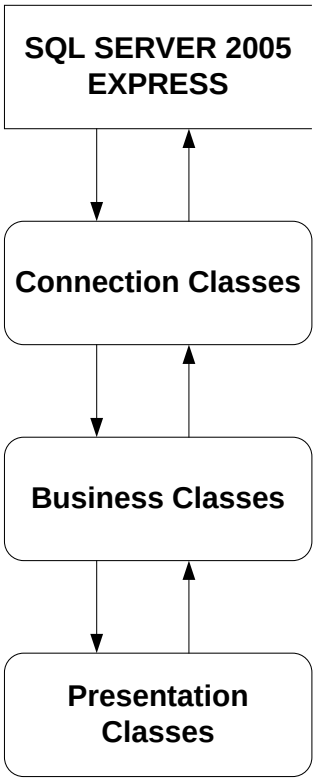


Figure 2 - Basic Diagram of the Software

Database

The tables are all linked to the patient table with the exception of the inventory and vendor tables. The database is able to contain data on the patient such as contact and address information, insurance information, exam information, billing information, prescription drug information, diseases, and other pertinent information. Figure 3 is Microsoft SQL Server 2005 Express, displaying the Optical database and the tables included which the Optometry Data System utilizes. Figure 3 shows the tables that are used in the Optical database.

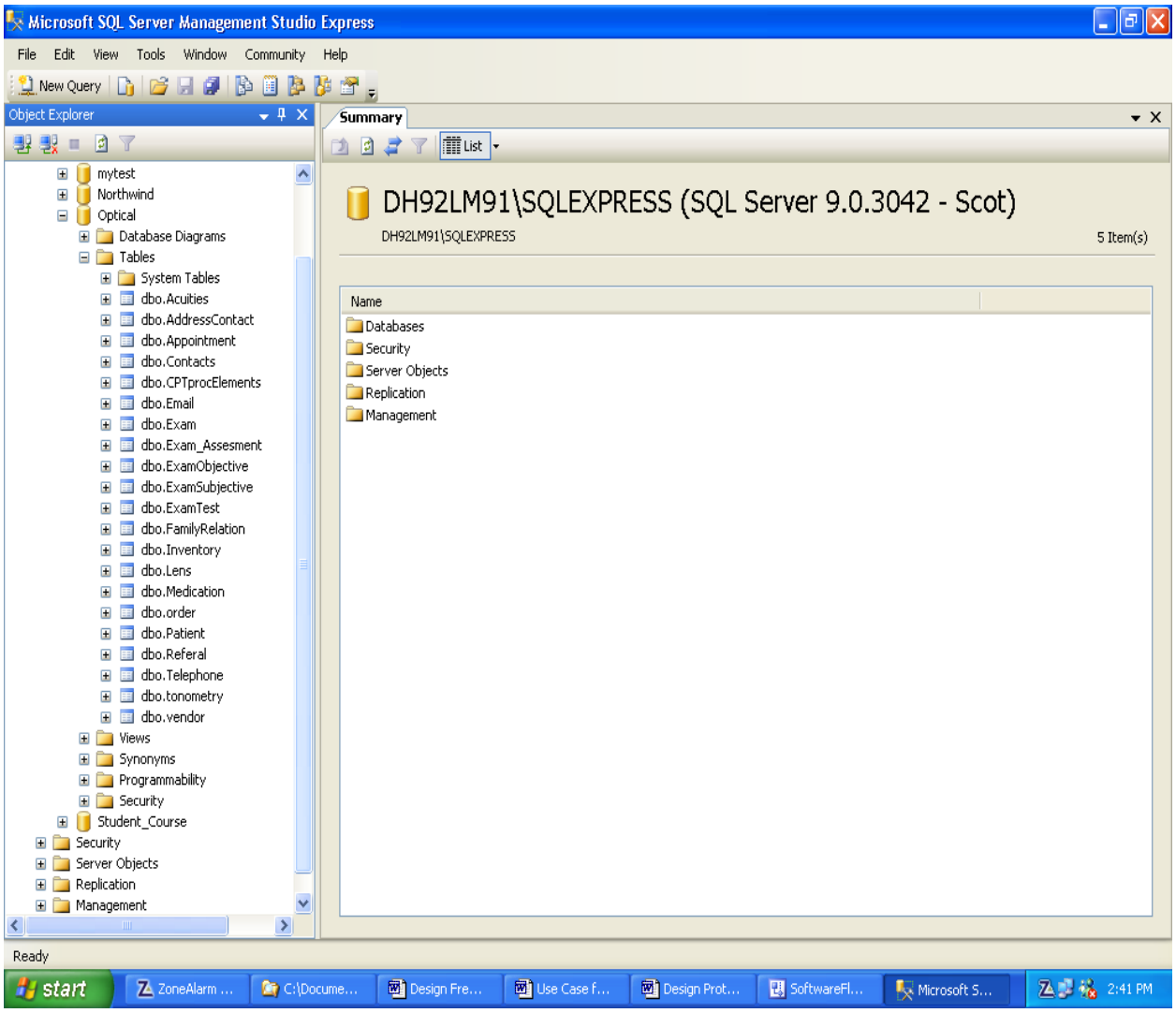


Figure 3 – Database in Microsoft SQL 2005 Express

Proof of Concept

The Optometry Data System for The Sight Shop is written in C# as a Windows desktop application. Sight Shop employees can utilize one software package to handle both their medical and business data needs.

Initial Login

The first screen the user sees is the login screen. Figure 4 shows this screen.

Figure 4 Initial Login Screen

The login users and passwords are contained in the database. The passwords are encrypted using a hash code supplied by the .NET libraries System.Security.Cryptography.

When the user first logs into the software the user is taken to the primary user interface. Once there, the user can navigate to the multiple functions within the application.

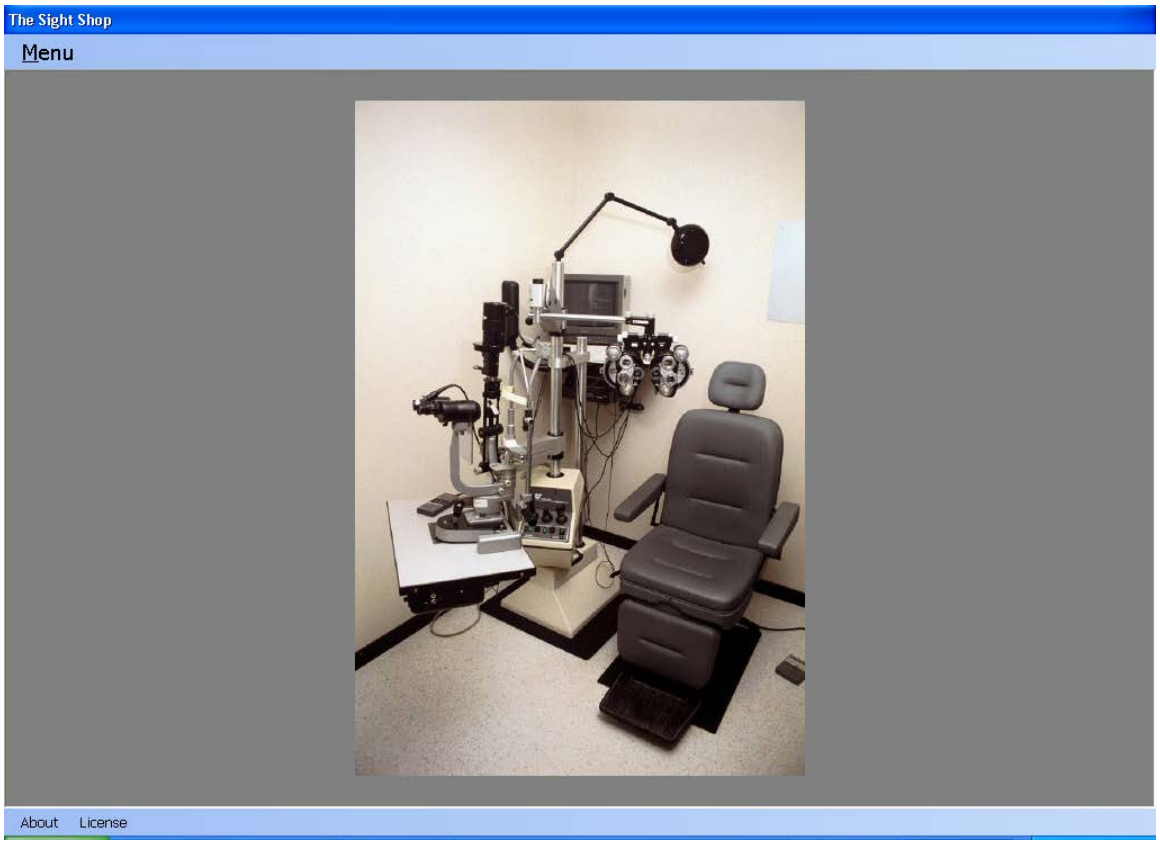


Figure 5 – Patient User Interface

Patient Profile

Figure 5 shows the first screen when the user logs onto the computer system. When the menu is clicked, the various functions scroll down so the user can pick the needed function. Figure 6 shows the menu scroll down for the user to select. Figure 7 shows the patient profile function in use.

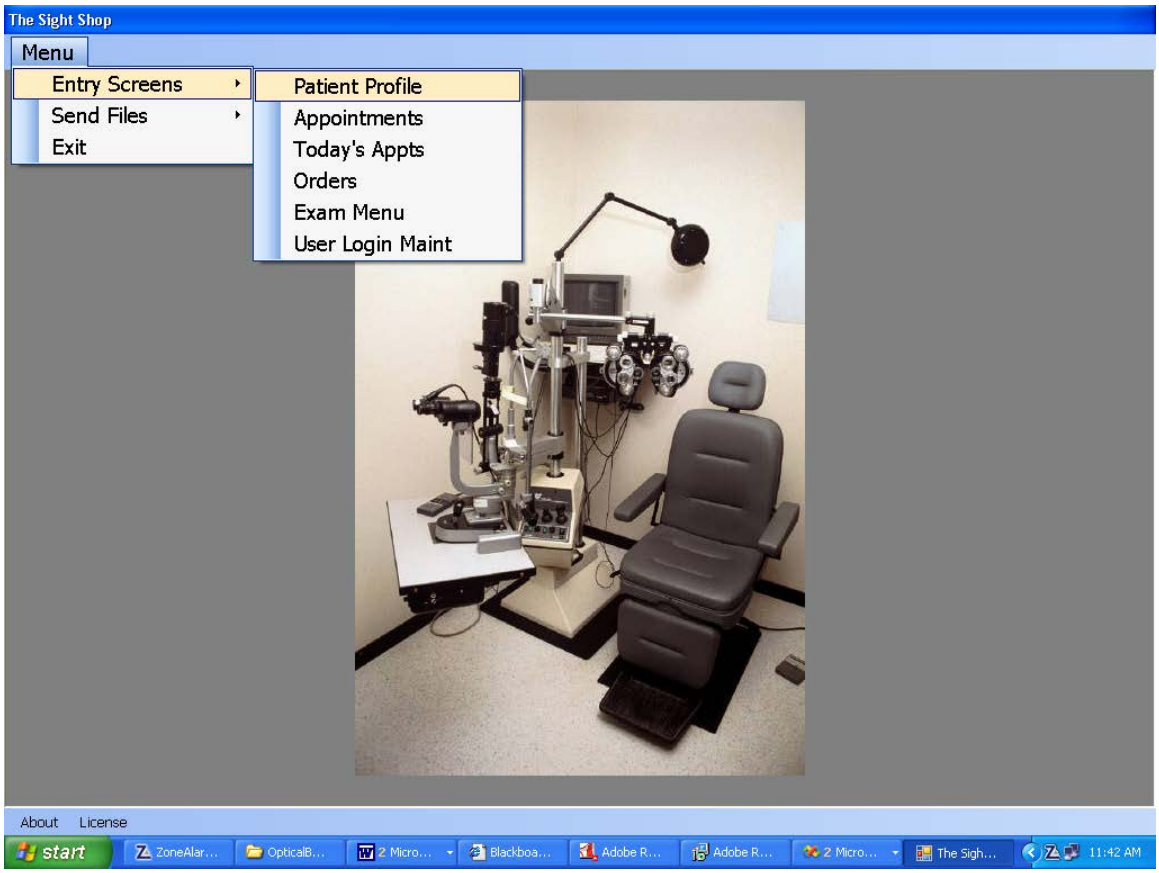


Figure 6 – Application Menu

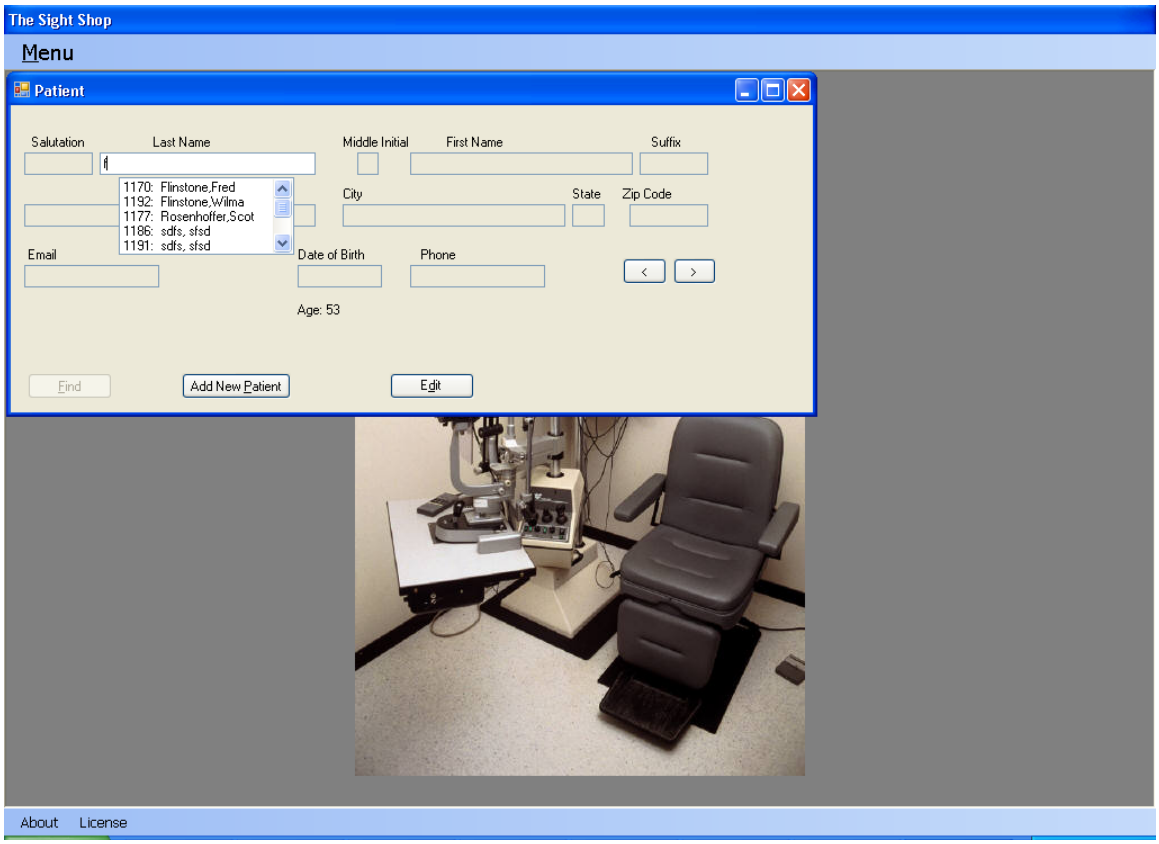


Figure 7 – Patient Profile User Interface

The user profile module allows the user to add, change, find users by last name, and scroll through all patients one at a time.

Figure 8 shows a user adding a patient.

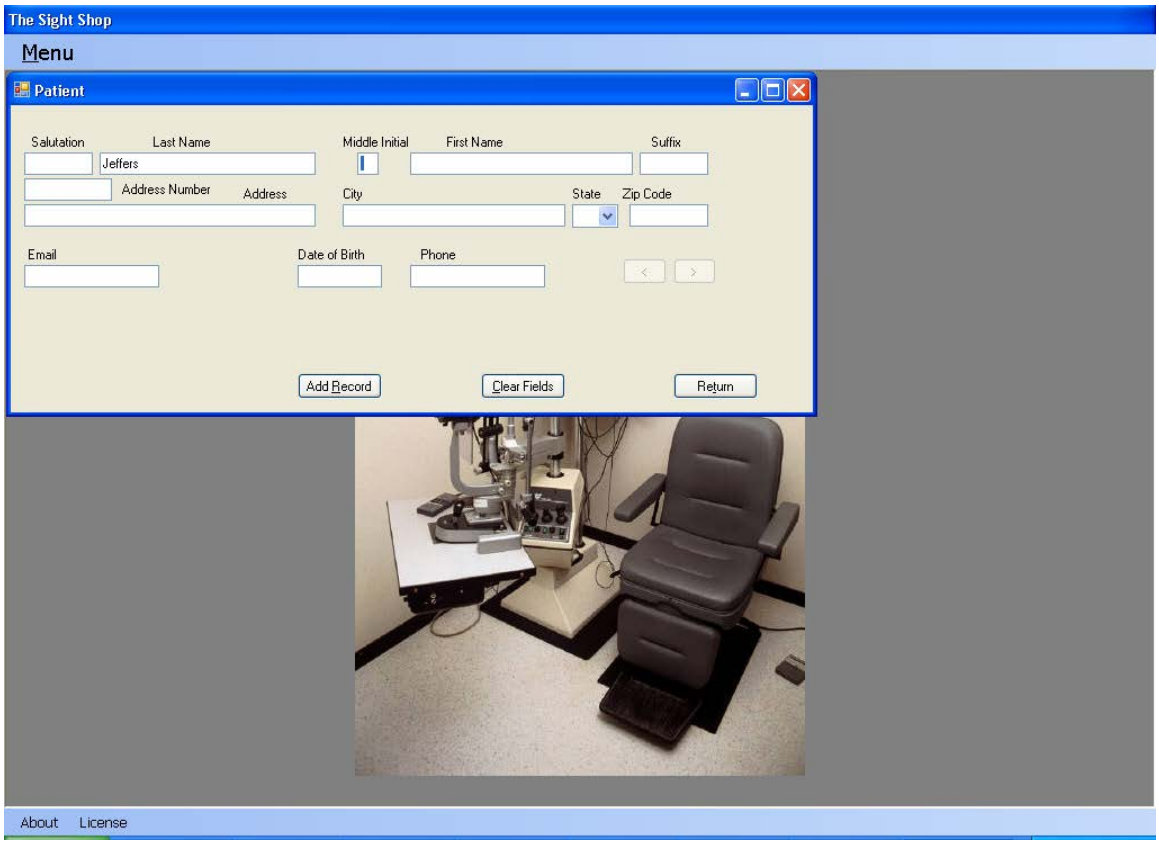


Figure 8 adding a patient

There are several fields that are required in this process. The fields include first and last name, address number and address, city, state, zip code, date of birth, and telephone number. If any required fields are not filled or are filled improperly a message box is shown and the cursor is put in the field that needs the required information.

Editing patient information is performed in much the same way as adding a new patient. The same requirement fields and field error checking are performed.

There is no function to remove a patient because patient record removal is not consistent with common medical practices and government regulations.

Appointments

The appointment module brings up the appointment screen and the patient profile. The appointment screen is tied to the patient screen which is used to search for the patient who needs the appointment. The patient profile screen is enabled with the appointment screen in order to allow patient edit and patient addition. The appointment screen is tied to the patient profile screen. Therefore, the patient that is shown on the profile screen is the patient on the appointment screen. The appointment screen is free from the confines of the main screen. This is to allow the use of a dual monitor system which is used on all of the Sight Shop's workstations..

Figure 9 shows the appointment screen. If there are any pending appointments they will be under the patient's name. The module also checks for any other current appointments at the same time the current appointment is being scheduled. This prevents two or more patients being scheduled at the same time.

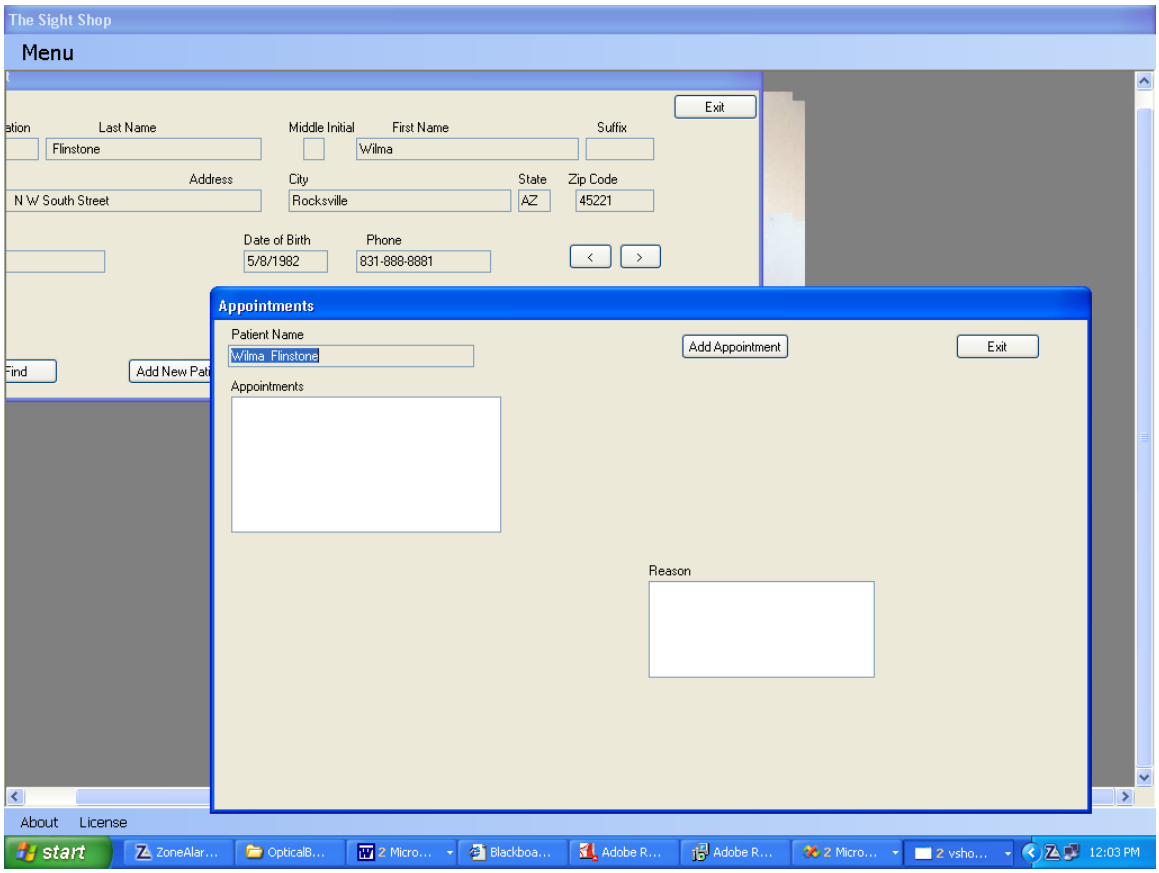


Figure 9 adding an appointment

The appointment module also includes a screen that shows all the appointments scheduled for any selected day or month and all appointments in the system. Appointments can be deleted. Figure 10 shows this screen.

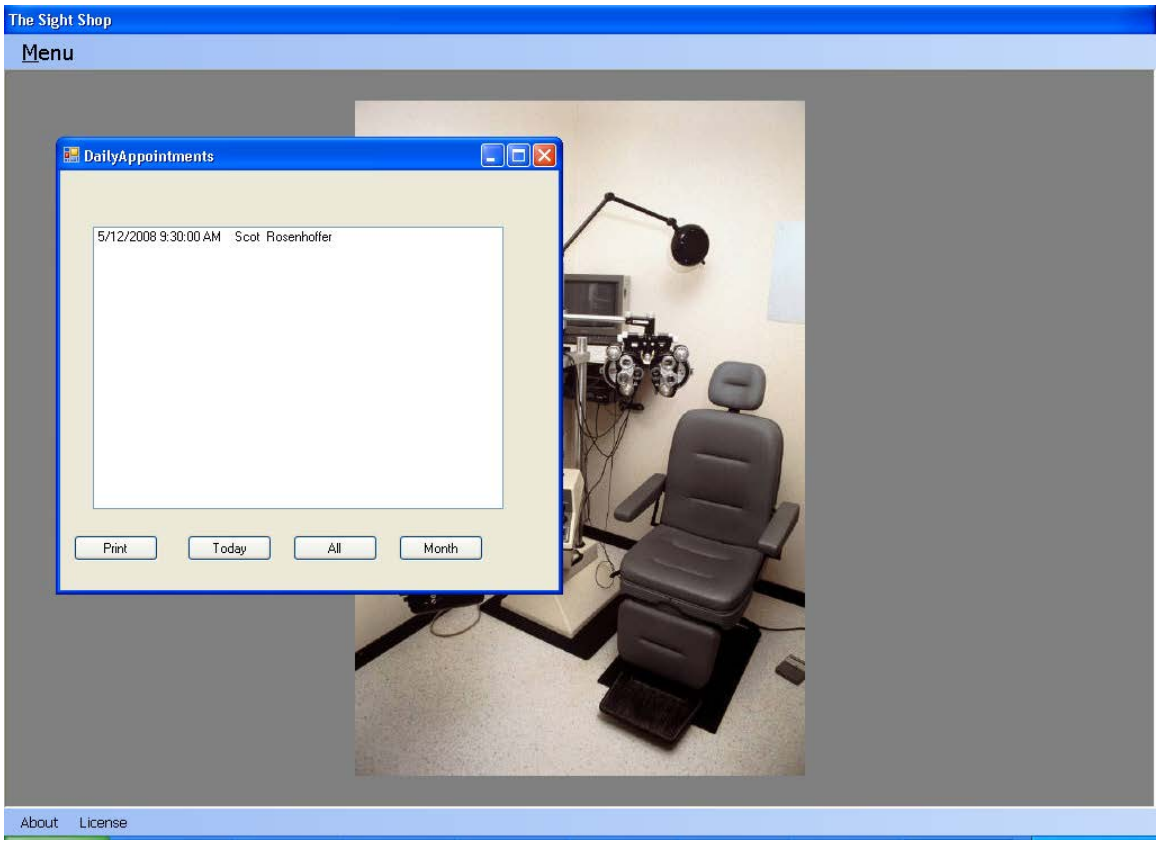


Figure 10 screen showing appointments

Eye Examination

The eye examination module consists of four parts. In each module the data entered can be committed to the database. Until the information is committed to the database it is held in memory.

There are no required fields on the entry forms due to the variance in the needs for each exam. Therefore, what is added to the patient record is entirely at the discretion of the doctor. The module also allows the return of the last previous record in order to avoid repetition of data that has not changed since the last exam.

Figure 11 shows the first screen in the exam module. The screen requires a selection of a patient name before continuing with the exam. The patient is found by last name search function.

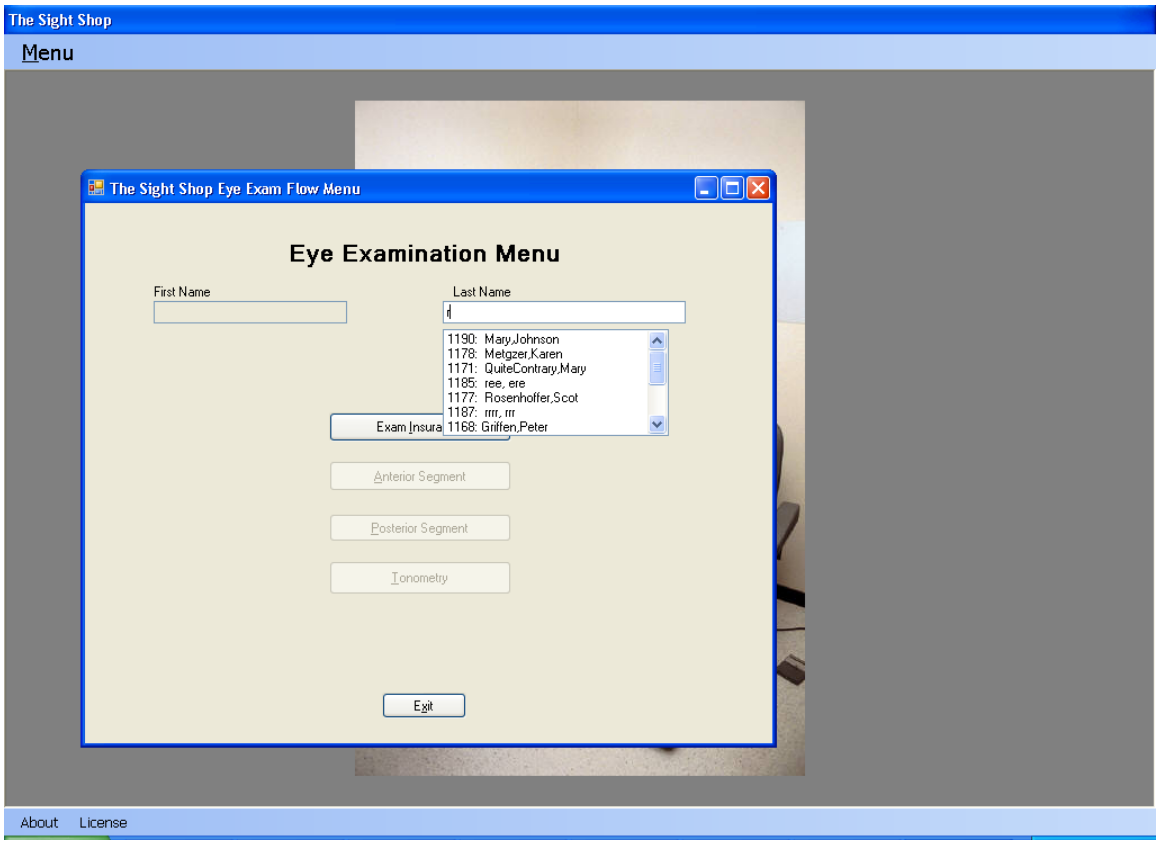


Figure 11 getting patient for exam

Once a patient is entered, the exam can continue. Each exam form must be complete before the next exam form is shown. Figure 12 shows the gathering of insurance information as part of the exam.

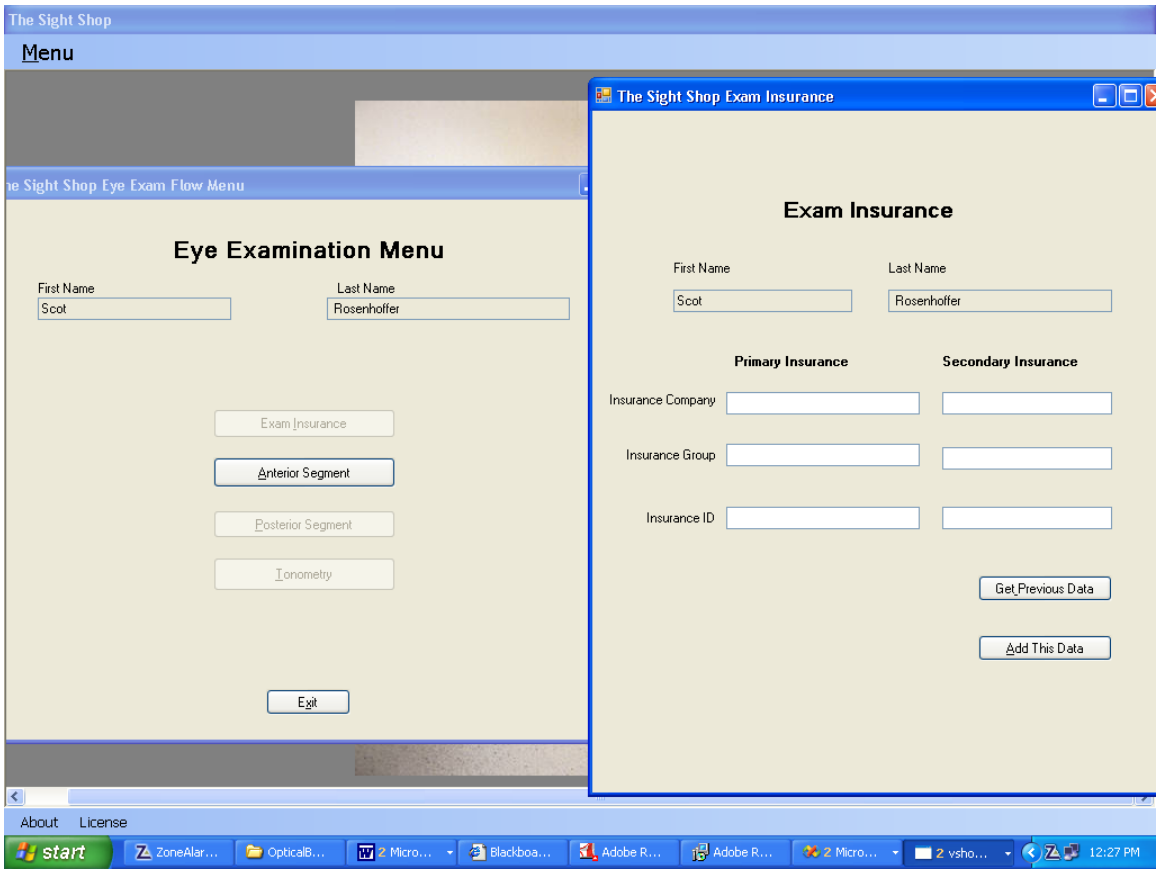


Figure 12 Insurance Information

Figure 13 shows the anterior segment form.

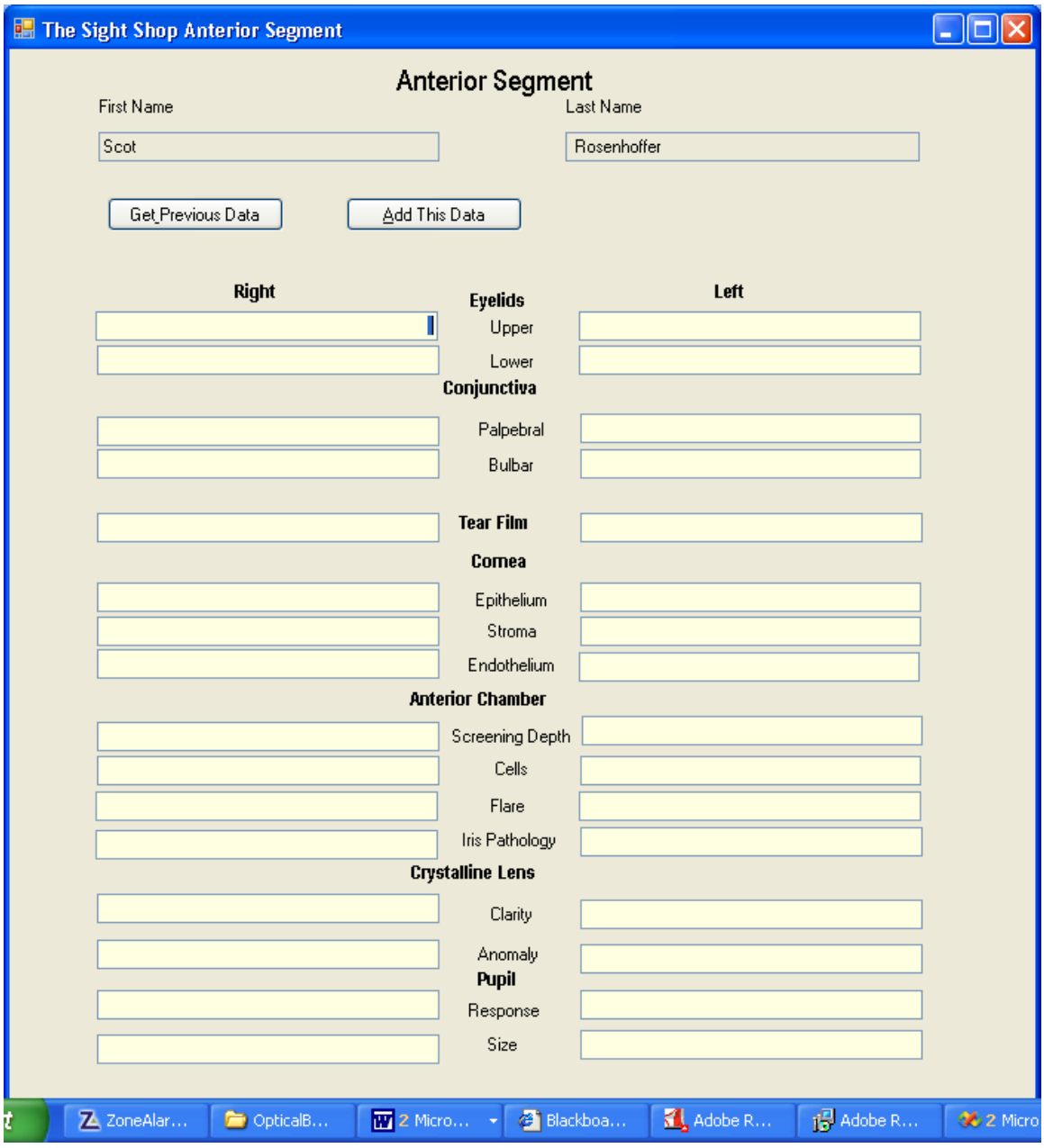


Figure 13 Anterior Segment

Figure 14 shows the posterior form of the exam.

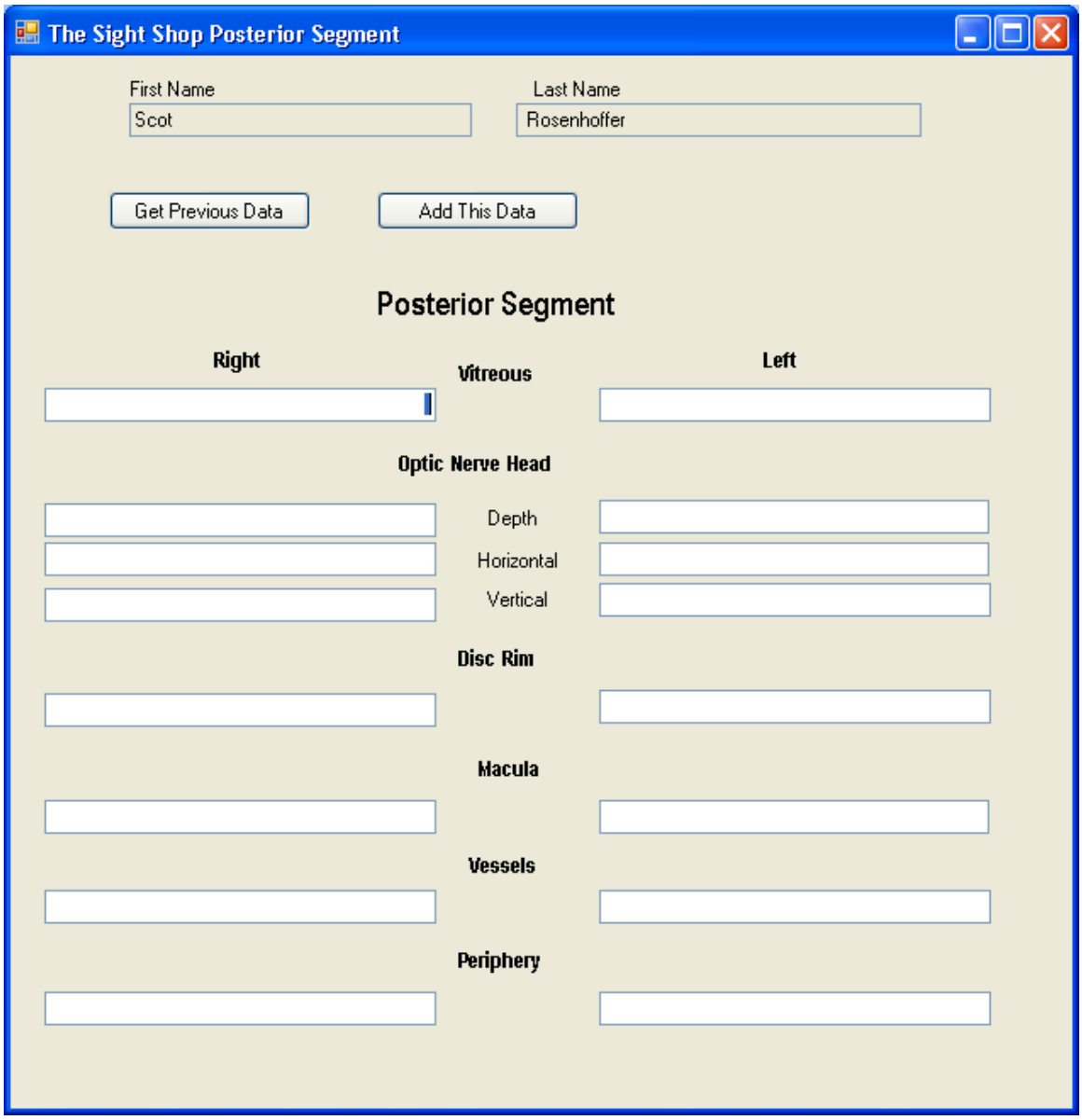


Figure 14 posterior segment

Figure 15 shows the tonometry form.

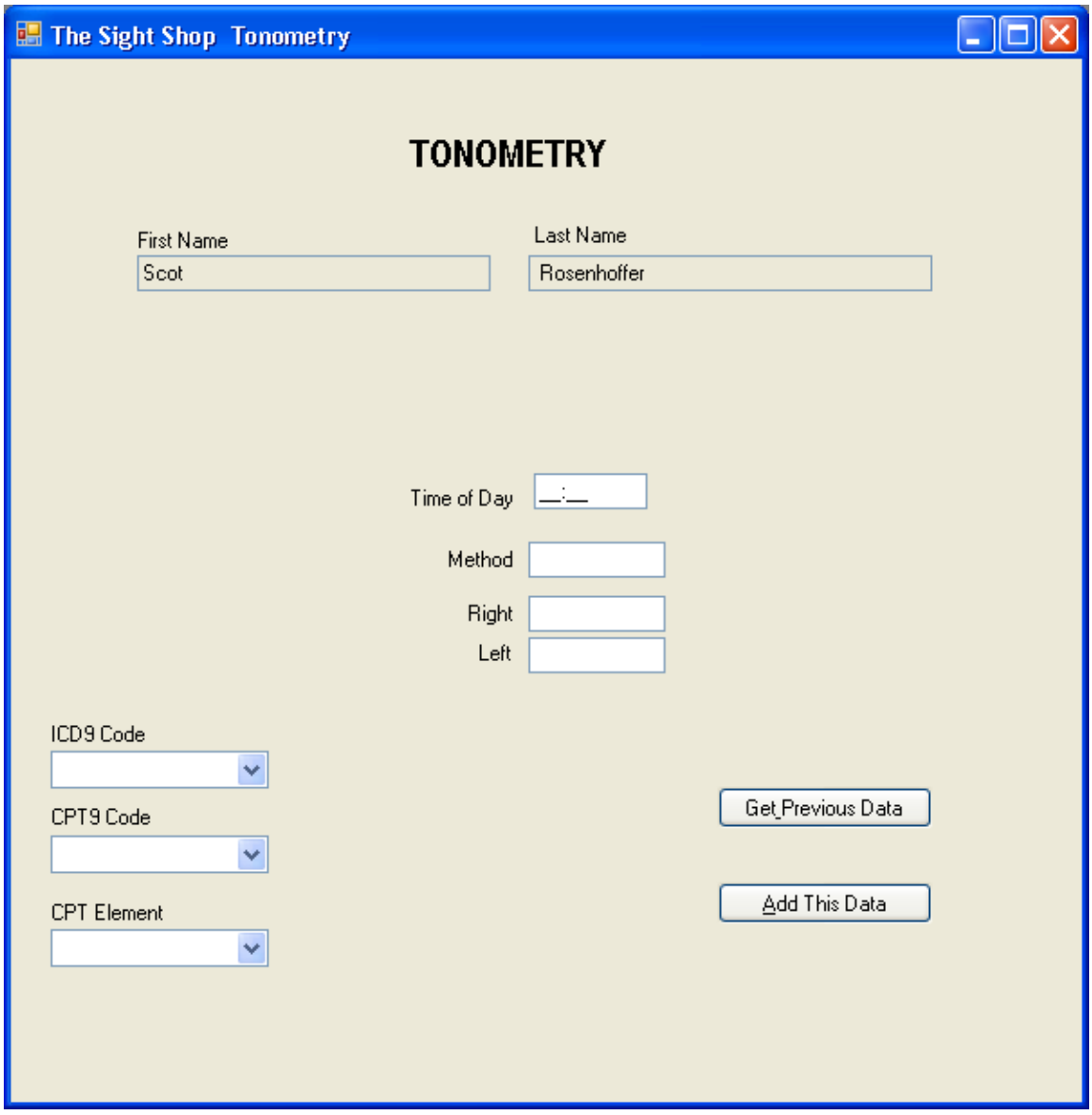


Figure 15 tonometry

Each form has the ability to get previous data. However, when the "Get Previous Data" button is pushed on any form in the sequence all forms are updated with the previous data. The "Add This Data" button commits data to the database. If the button has been used in a form previously in the sequence the button will update the data in the database. Therefore, the correct exam number is retained.

System User Login Maintenance

The screens to add, delete, and change user passwords are available only to users who have the administrator designation. The default designation is ‘N’ on user creation and it can be changed by an administrator on user maintenance.

Figure 16 shows the user maintenance screen. From this screen the administrator can add, delete, or change a user, including their password. The password cannot be retrieved due to the type of cryptography used to encrypt the password.

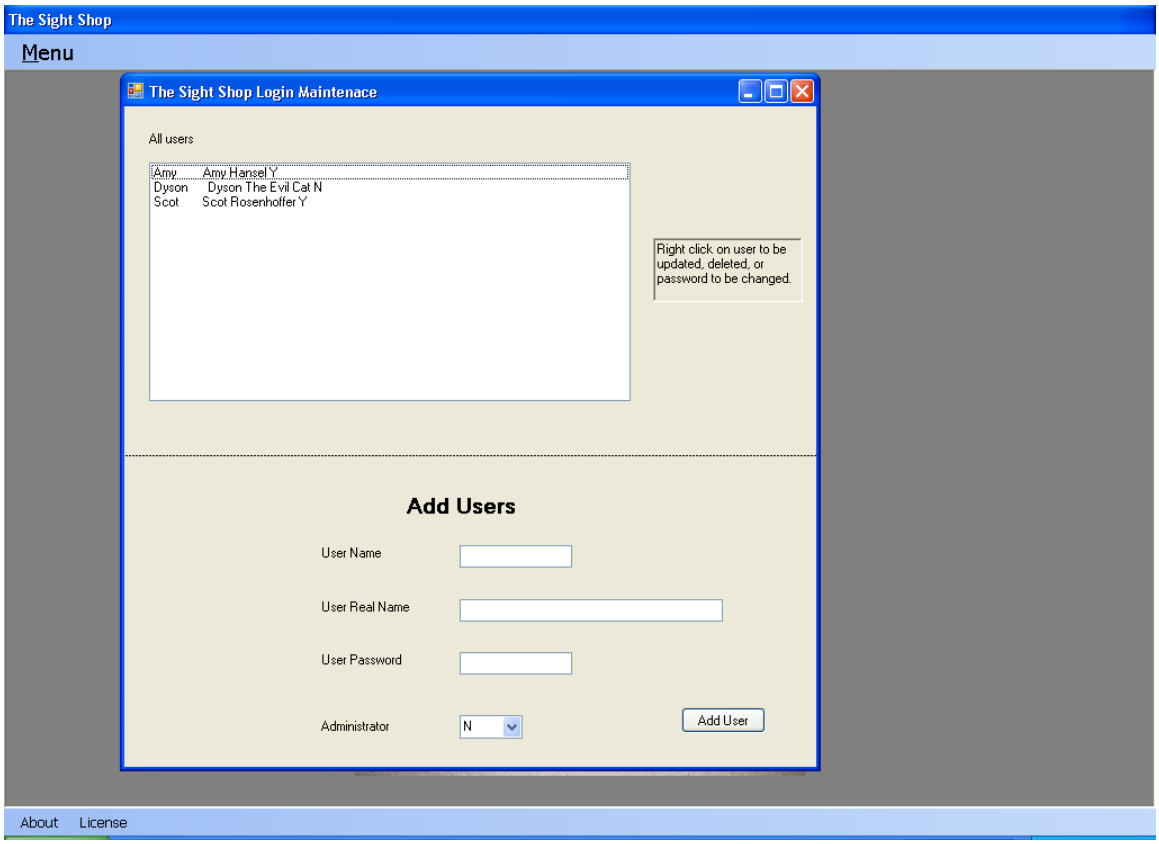


Figure 16 user maintenance

All the users of the system are in the text box at the top of the form. By right clicking the user the administrator can edit, delete, or show the user logs.

The user logs are saved in a table that holds the name of each user and their time logged on and logged off. Each time a user logs onto the system the user's real name and the date and time are recorded. Each time the user logs off the same information are recorded. Figure 17 shows the screen.

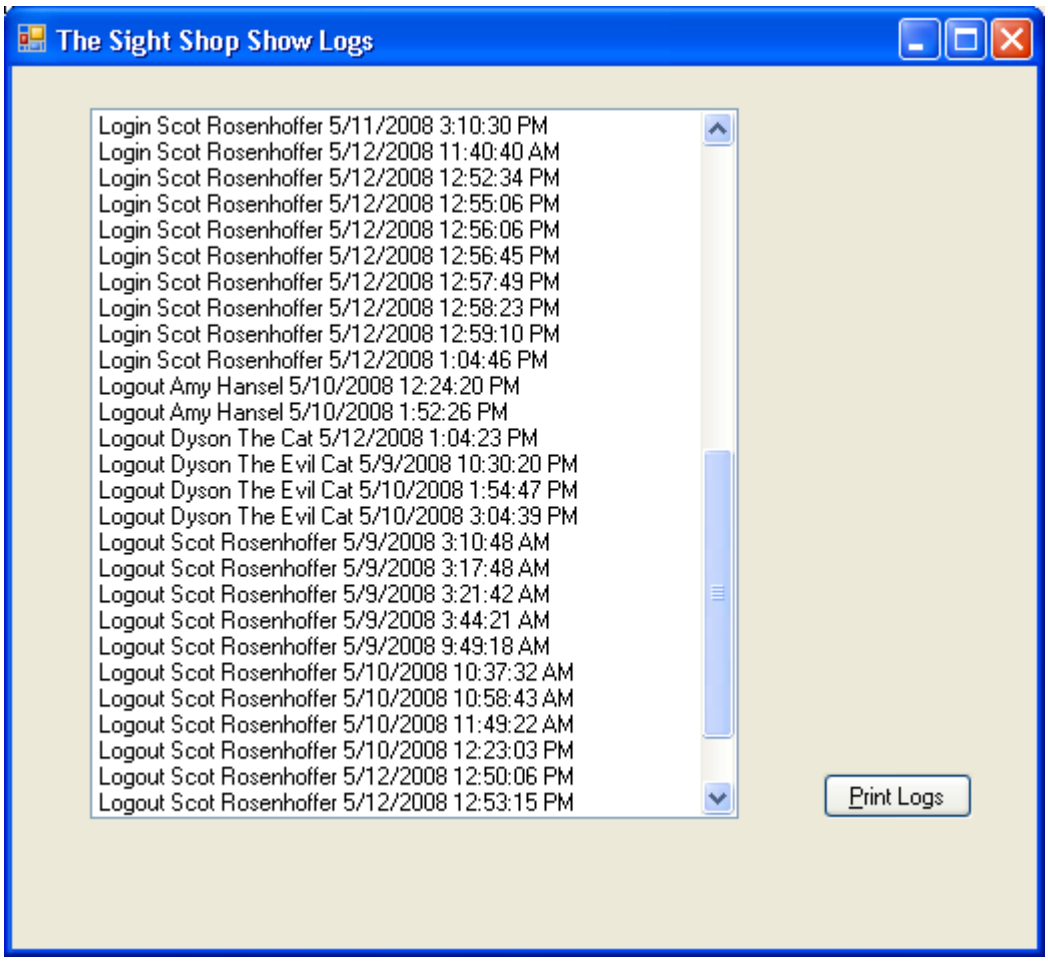


Figure 17 logs

Figure 18 shows the user edit screen. The password is changed at this screen as well.

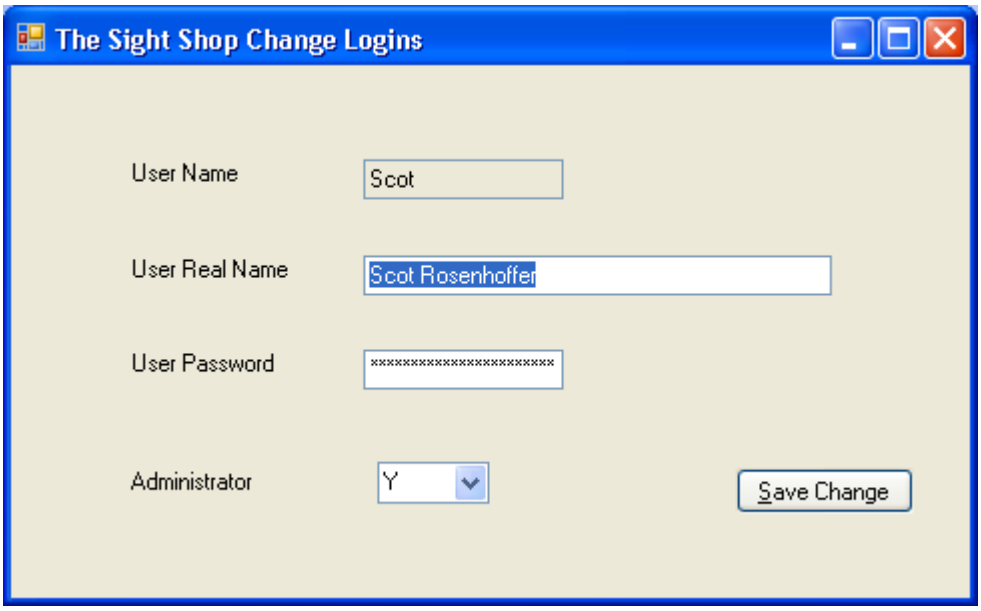


Figure 18 edit a user

From the main screen there is also a menu item to create a file to send to Zirmed for insurance billing. This file is an ASCII text file that is comma delimited. The file is uploaded to Zirmed, using Zirmed's secure upload service, via their web site.

Orders

The order system is used to order glasses and contacts. All frames on site are for the patient to find the best style and size frame to fit their face and are not sold to the patient. Each pair of glasses is ordered individually and made to the specifications of the patient and the doctor's prescription for the correction.

Figure 19 shows the beginning of the order process.

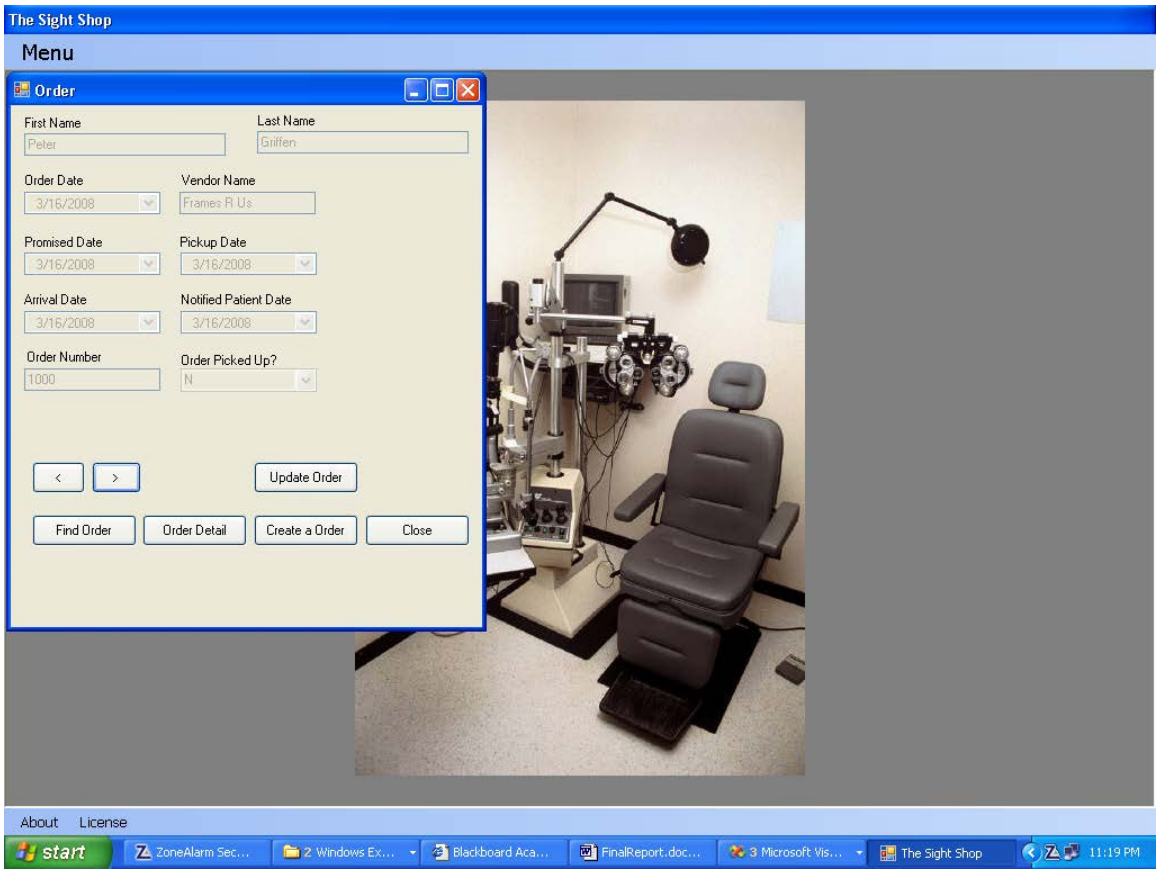


Figure 19 Orders

Inventory

The inventory system keeps the test frames, diagnostic contacts, and various sundries used by the office. The inventory does not contain items that are sold because items sold are ordered on an individual basis.

Figure 20 shows the inventory system screen.

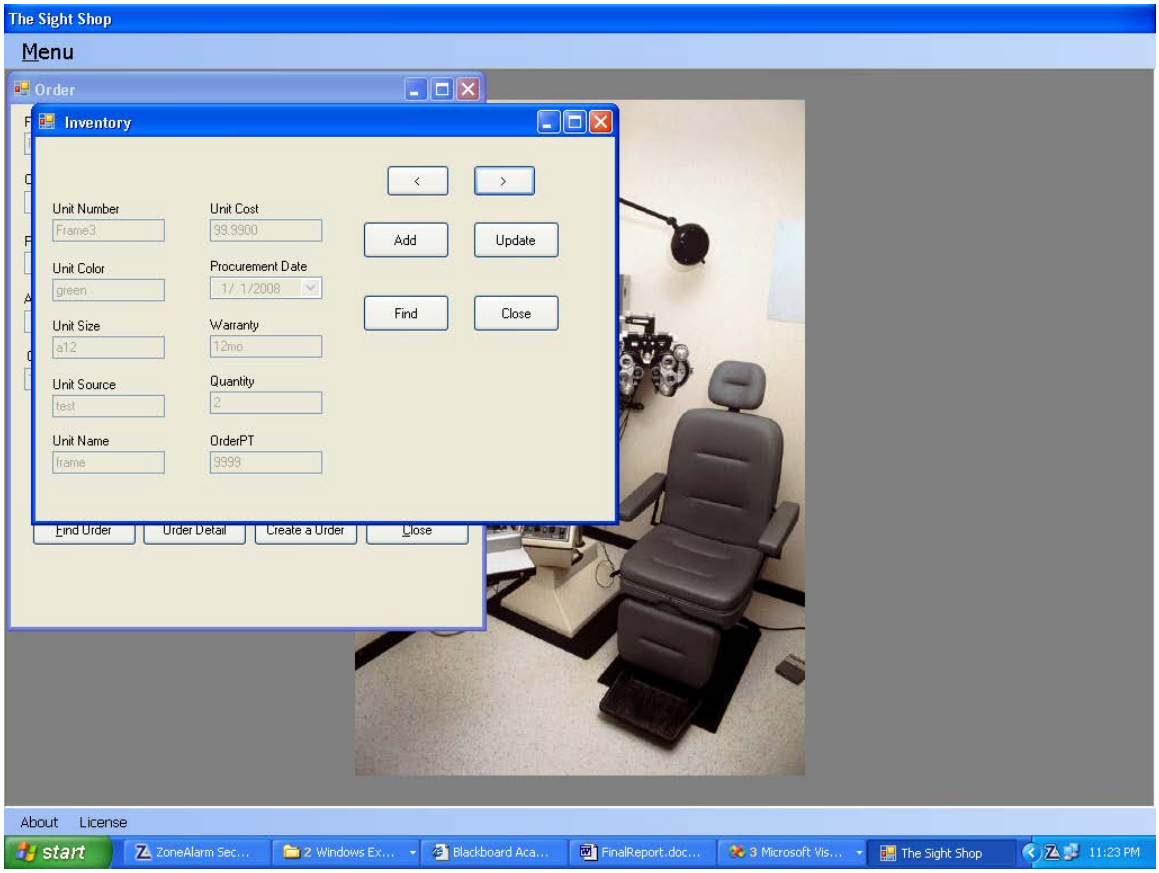


Figure 20 Inventory System

Vendor Contact

The vendor system shows the vendors from which items are obtained and from whom glasses and contacts are ordered.

Figure 21 shows the Vendor Contact screen.

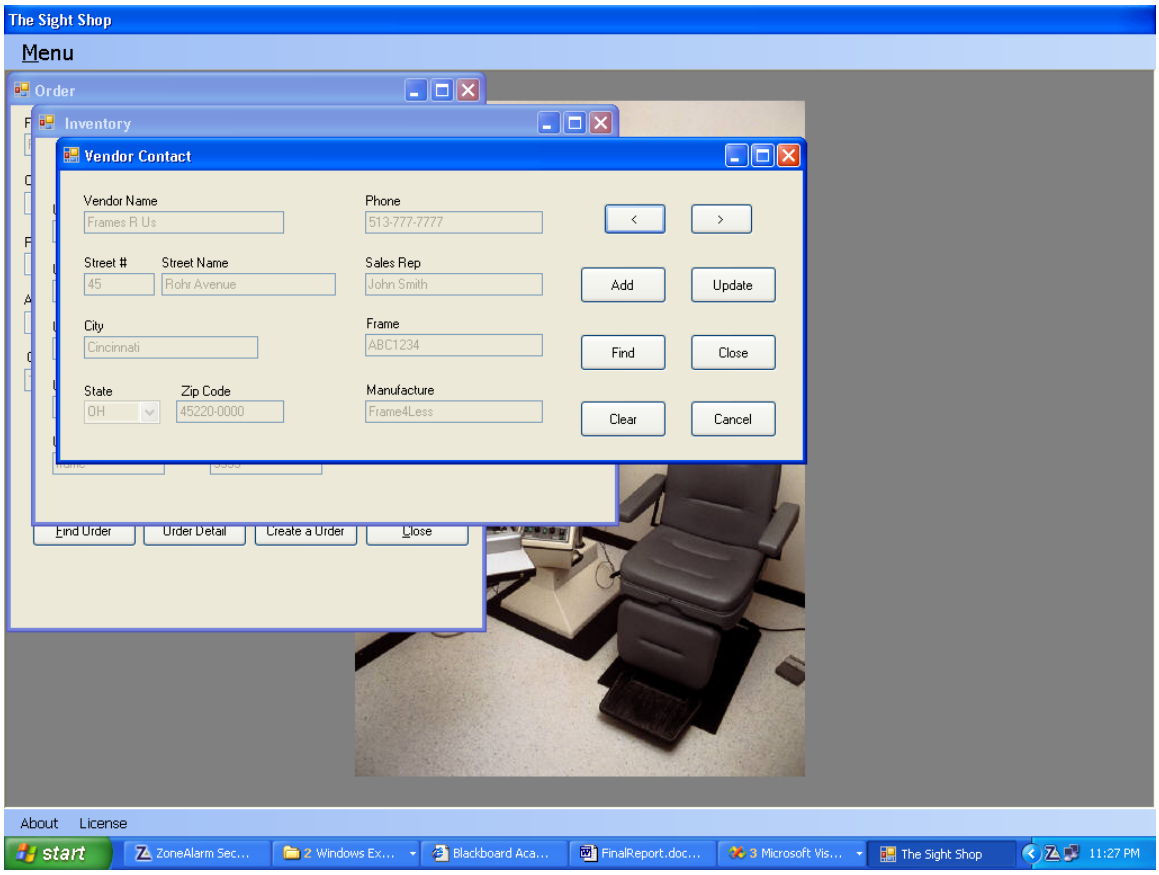


Figure 21 Vendor Contact

Testing Plan

Testing of the Optometry Data System has been ongoing throughout the life cycle of this project.

Unit testing for the Optometry Data System has been ongoing throughout system development. Software testing consisted of structured testing and consisted of functional and non-functional testing. The testing was conducted during and after development.

The functional testing included validation testing of the user interfaces which is to ensure that records can be inserted, updated, and deleted into and from the database through the user interface functionality. Testing determined that the same records can be selected from the database using the find functions and that the correct data is pulled and displayed to the user. Testing also included checking the field validation. Therefore, field size constraints and character verification is in place.

The nonfunctional testing included continuity checks. Continuity checks assure that all the interfaces follow the same naming conventions in both the code and the user interface. The look and feel aspects of the user interface were tested to ensure it is easy to use and understand from the user perspective.

Once development was complete user acceptance testing commenced. UAT (User Acceptance Testing) testing took place after the complete functional and nonfunctional testing. It included users of the Sight Shop test the application in comparison to their current product. This ensured that the expected results are received and the process-flow of the application functioned as anticipated and the application has a user-friendly aspect.

Testing is important to all applications, not only to ensure a quality product but that it is easy for the user to use.

Figure 22 is an example of the testing sheet for each module. As shown each function of the module has been tested, there is record of any error code identification numbers as well as the line in which the error occurs. The correct input and output have been checked and notes made as to the usability of the module and function.

Optometry Software Test

Software Functionality	Pass/Fail	Error ID	Error Line Numbers	Output Correct	Input Correct	Notes
Patient Profile						
Input Patients						
Find Patients						
Edit Patient Records						
Overall Functionality						

Figure 22 Testing

Figure 23 shows an example of the second test sheet used in our testing.

Step	Description	Expected Result	Problem with this Step?
	Login Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Patient Profile	Patient Profile window is displayed	
4	Click Find	type a letter	
5	Select a name from the list	the corresponding Patient and Patient info is displayed	
		Test Result	Pass

Figure 23 Testing

Testing Results

The results of testing created several changes to the software.

Orders

It was found that it was difficult to enter new orders into the order system due to the number of buttons displayed while using the system and caused confusion for the user as to what button to use. An example is the “Create Order” button, “Find Patient” button, and “Change Order” button were all showing at the same time. The buttons were made to be invisible at the appropriate time and visible only when needed. This made the process of entering an order much easier on the user.

Patient Profile Entry

During user testing it was found that the zip code was difficult to enter during new patient entry and patient update. This was due to the cursor not going completely to the left of the text box. A masked text box was added to the zip code entry and the focus() method was added to the text box after the state information was entered so that the cursor would automatically be in the correct zip code area without user input.

Exam Entry

During testing on a laptop with a different aspect ratio it was found several of the exam forms went off the screen and information at the bottom of the forms was not accessible. This problem was solved by creating a pane and adding it to the form. The scroll property of the pane was set to true and this added scroll bars on the form allowed the user to scroll down to see hidden information and entry boxes.

System User Entry Updates

It was found when updating a system user's information the information was not updated in the screen showing the information on the changed user. It remained the same. The problem was solved by refreshing the screen when information is changed on the user update.

Time Line

Figure 24 shows the schedule for completion of the project and figure 25 shows the last quarter schedule.

ID	Task Name	Start	Finish	Duration
1	Create DB Diagram	1/14/2008	1/23/2008	10d
2	Create Class Diagram	1/14/2008	1/23/2008	10d
3	Create Timeline	1/21/2008	1/25/2008	5d
4	Create Connection Class	1/21/2008	2/9/2008	20d
5	Create Appointment Module	1/28/2008	2/17/2008	21d
6	Create Patient Profile Module	2/2/2008	2/22/2008	21d
7	Create Patient Order Module	2/8/2008	2/29/2008	22d
8	Prepare Design Freeze Paper	2/19/2008	3/14/2008	25d
9	Prepare Design Freeze Presentation and Prototype	2/23/2008	3/2/2008	9d
10	Create Inventory Module	3/4/2008	3/17/2008	14d
11	Create Medical History Module	3/15/2008	3/28/2008	14d
12	Create Exam Module	3/18/2008	4/7/2008	21d
13	Create Assessment Module	3/29/2008	4/18/2008	21d
14	Create Vendor Contact Module	4/8/2008	4/21/2008	14d
15	Create Vendor Order Module	4/19/2008	5/2/2008	14d
16	Final Testing and Configuration	4/22/2008	5/9/2008	18d
17	Prepare Final Presentation	5/3/2008	5/11/2008	9d
18	Prepare Final Paper	5/5/2008	5/30/2008	26d
19	Prepare Project for Tech Expo	5/7/2008	5/20/2008	14d

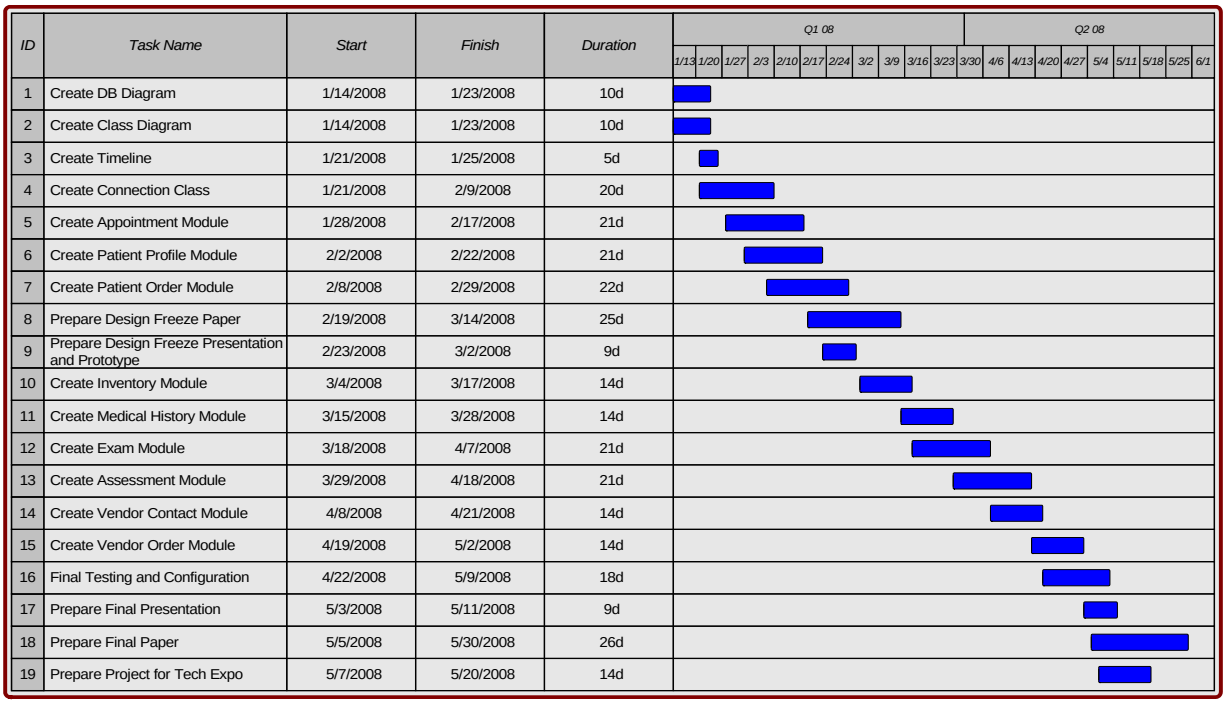
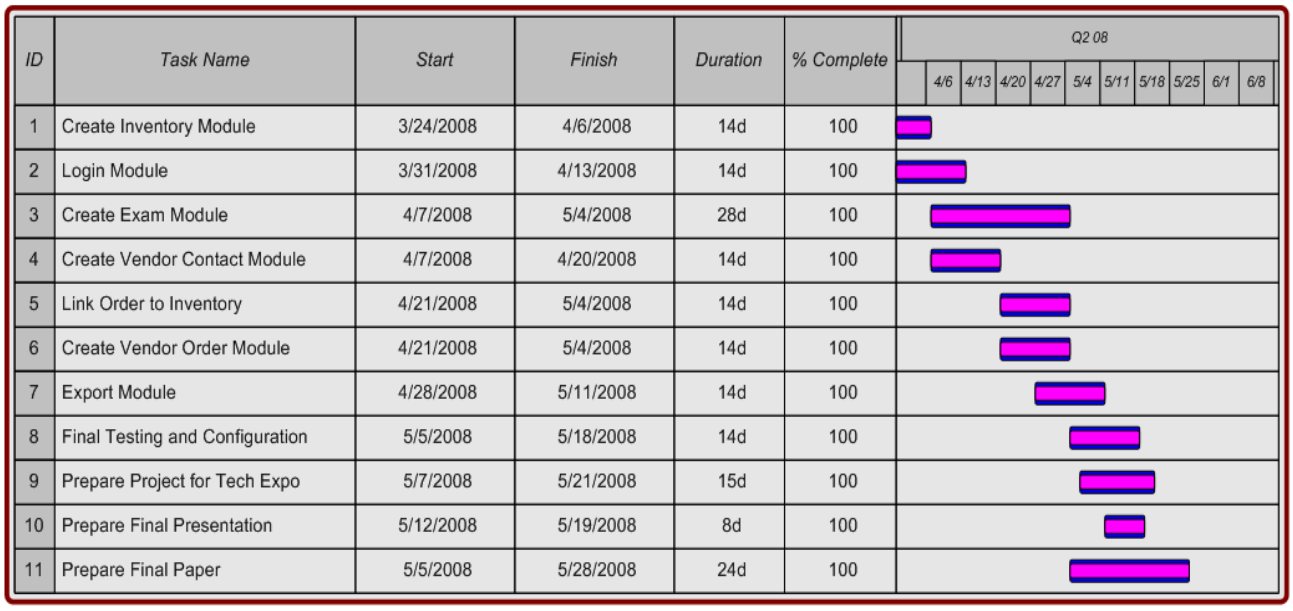


Figure 24 – Gantt Chart



ID	Task Name	Start	Finish	Duration	% Complete
1	Create Inventory Module	3/24/2008	4/6/2008	14d	100
2	Login Module	3/31/2008	4/13/2008	14d	100
3	Create Exam Module	4/7/2008	5/4/2008	28d	100
4	Create Vendor Contact Module	4/7/2008	4/20/2008	14d	100
5	Link Order to Inventory	4/21/2008	5/4/2008	14d	100
6	Create Vendor Order Module	4/21/2008	5/4/2008	14d	100
7	Export Module	4/28/2008	5/11/2008	14d	100
8	Final Testing and Configuration	5/5/2008	5/18/2008	14d	100
9	Prepare Project for Tech Expo	5/7/2008	5/21/2008	15d	100
10	Prepare Final Presentation	5/12/2008	5/19/2008	8d	100
11	Prepare Final Paper	5/5/2008	5/28/2008	24d	100

Figure 25 Last Quarter Schedule

Budget

The following describes the breakdown of the project costs. Figure 26 shows the breakdown of the costs of the Optometry Data System. We are using the office's existing Dell Optiplex server running Windows 2000 with Service Pack 4, which can adequately handle the system requirements of our software application and database. We are using Microsoft SQL Server 2005 Express for the database. This database is offered free of charge by Microsoft, provides up to 4 GB of data storage, which the office is intending to exceed. We are using Microsoft Visual Studio 2005 to write to software. The University of Cincinnati provides Visual Studio as part of their MSDN Alliance (). We estimate labor costs at $50.00 per programmer ($100.00 for both of us), as a common software developer consultant fee, and approximately 360 hours in system analysis, design, coding, and testing the project. We are providing our services free of charge.

ITEM	DESCRIPTION	RETAIL COST	ACTUAL COST	NEED
Server	Dell Poweredge 840	$749.00	$0.00	Has Server
Operating System	Win 2003 Server per server with 5 CAL*	$999.00	$0.00	Has 2000
Data Base	MS SQL 2005 Express	$0.00	$0.00	Free
Visual Studio	Visual Studio 2005	$799.00	$0.00	Have
Labor	360 Hours @ $100 hr	$36,000.00	$0.00	Donated
Ad Hoc Reporting	Crystal Reports 2008	$495.00	$495.00	Need
Classes for Crystal Reports (4)	Online @ $119 ed2go.com	$119.00	$119.00	Need
	Totals:	$39,161.00	$614.00	

Figure 26 - Project Costs (2) (3) (4) (5) (6)

Deliverables

The following is a list of the deliverable functions that have been completed.

1. A C#-based data entry software for The Sight Shop, in which both medical and business information can be maintained
2. Create a class to handle the database connections
3. Create a module to maintain patient contact and personal information
4. Create a module to maintain patient appointments
5. Create a module to maintain patient order information
6. Create a module to maintain medical exam data
7. Create a module to maintain vendor information
8. Create a module to maintain vendor orders
9. Create logins for each employee per HIPAA regulations *(1)*

Challenges

There were some challenges while creating this software. One of the greatest challenges was communication with the client. We found it difficult during meetings to keep the client on the subject at hand and with resulting delays in receiving the information needed to solve the problems he had with his office software needs. We solved this problem by creating an agenda for each meeting with the client. This allowed our meetings to be productive by having a clear goal for everyone.

Technical challenges included the creation of the database. There are 25 tables in the final database and all needed to be related to the patient. We found Churcher's (3) book *Beginning Database Design* invaluable, as well has help from Professor Said and Professor Schlemmer.

Conclusion

The problems experienced at the Sight Shop can be solved economically and effectively using software created on Microsoft's .NET platform. Microsoft SQL Server 2005 Express is used as the database because it is an effective, proven, and has worldwide support. Updates to the database are readily available free of charge over the Internet. Should the database become larger than four megabytes of data, upgrading to the next level of SQL involves nothing more than buying the larger package. None of the database will need to be changed to upgrade.

References

1. Business Objects Online Store. 2007. 12 October 2007 http://store.businessobjects.com

2. Churcher, Clare. Beginning Database Design. Berkeley, CA: Apress, 2007

3. Dell's New PowerEdge Servers. 2007. 17 October 2007 <http://www.dell.com/servers>

4. Ed2go online courses. 2007. 26 October 2007 <http://www.ed2go.com>

5. HEALTH INSURANCE PORTABILITY AND ACCOUNTABILITY ACT OF 1996. 1996. 7 November 2007 <http://aspe.hhs.gov/admnsimp/pl104191.htm>.

6. Jones, Allen. C# Programmer's Cookbook. Redmond, Washington: Microsoft Press, 2004

7. Microsoft SQL Server. 2007. 16 October 2007 <http://www.microsoft.com/sql>

8. Mullins, Craig. Database Administration: The Complete Guide to Practices and Procedures. Boston, MA: Pearson Education, 2002

9. SQL Server Developer Center. 2007. 2 October 2007

<http://msdn2.microsoft.com/en-us/sql/default.aspx>

APPENDIX A: CODE

The following are some of the examples of code we used to solve the problem.

Find Method

The patient find method, LastNameFind. This method takes a list box and a string sort name. It is located in the business base class and is available to all inherited classes. Figure 27 shows the find method code.

```
//*********************************************
        // Find a set of patient records based on string
        // Takes a listbox and the string to find
        //*********************************************
        public void LastNameFind(ListBox bx, string str)
        {

            bx.Items.Clear();
            string strSelection = str;
            string strSort = "LName";

            string strSel = "LName LIKE " + "'%" + strSelection + "%'";

            DataRow[] roSelected = tblPatient.Select(strSel, strSort,
DataViewRowState.CurrentRows);

            // load the listbox



            foreach(DataRow r in roSelected)
            {
                bx.Items.Add(r["PID"].ToString() + ": " +
r["LName"].ToString() + "," + r["FName"].ToString());

            }
        }
```

Figure 27 Find Method

Field Validation

The following methods are used for field validation. Desktop forms do not come with error validation as in ASP.NET 2.0. Therefore, we had to write our own

validation routines. Figure 28 shows the three methods for field validation: IsNumeric, IsZipCode, and IsPhoneNumber. All return a Boolean value.

```
//*****************************************************
        // field entry error handling
        // Checks that the given string is an equivalent
        // numeric value.  Returns a boolean.
        // *****************************************************
        public bool IsNumeric(string strCheck)
        {
            bool ok = true;
            try
            {
                int intTest = Convert.ToInt32(strCheck);
            }
            catch
            {

                ok = false;
            }
            return ok;
        }

        //************************************************
        // Field entry error handling
        // Checks that a telephone number has been
        // entered in 999-999-9999 format
        // Uses IsNumeric and returns a booliean value
        //************************************************
        public bool IsPhoneNumber(string strPhoneNumber)
        {
            // get rid of whitespace for true length
            string strPhoneNo = strPhoneNumber.Trim();
            bool isnumber = true;
            if (strPhoneNo.Length < 12 || strPhoneNo.Length > 12)
                isnumber = false;
            for (int i = 0; i < strPhoneNo.Length; i++)
            {
                if (isnumber == false)
                    break;

                switch (i)
                {
                    case 0:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 1:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 2:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 3:
```

```
                        if (strPhoneNo[i] != '-')
                            isnumber = false;
                        break;
                    case 4:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 5:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 6:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 7:
                        if (strPhoneNo[i] != '-')
                            isnumber = false;
                        break;
                    case 8:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 9:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 10:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                    case 11:
                        isnumber = IsNumeric(strPhoneNo[i].ToString());
                        break;
                }

            }
            return isnumber;

        }

        //*****************************************
        // Checks to see if a zipcode is correct
        // checks for both a 5 number zip and
        // the + 4 zip  99999  99999-9999
        // returns a bool
        //*****************************************
        public bool IsZipCode(string strZipCode)
        {

            string strZip = strZipCode.Trim();
            bool isnumber = true;
             // get rid of whitespace for true size
            if (strZip.Length < 5)
                isnumber = false;
            if (strZip.Length > 5 && strZip.Length != 10)
                isnumber = false;
            if (strZip.Length == 5)
            {
                for (int i = 0; i < strZip.Length; i++)
                {
                    Console.WriteLine(i.ToString() + " In 5 one " +
isnumber.ToString());
```

```
            if (isnumber == false)
                break;

            switch (i)
            {
                case 0:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 1:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 2:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 3:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;

                case 4:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;

            } // end switch

        }// end forloop
    } // endif string length 5

    if (strZip.Length == 10)
    {
        for (int i = 0; i < strZip.Length; i++)
        {
            if (isnumber == false)
                break;

            switch (i)
            {
                case 0:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 1:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 2:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 3:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;

                case 4:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 5:
                    if (strZip[i] != '-')
                    {
                        isnumber = false;
                    }
```

```
                    break;
                case 6:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 7:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 8:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;
                case 9:
                    isnumber = IsNumeric(strZip[i].ToString());
                    break;

            } // end switch

        }// end forloop
     } // end if strZip = 10
     return isnumber;
    } // end IsZip
```

Figure 28 Validation Methods

User Login Passwords

The login for users required a password to be generated and retained in the database. The problem of having a password out in the open in the database was solved by using a hash. Fortunately .NET has a library that contains the necessary algorythm.

The following is code to create the encrypted password. The password cannot be decrypted. Therefore, when checking the password when the user logs into the system, the password that the user enters must be encrypted and checked against the encrypted password in the database. The code used to check the password on login is right after the code to encrypt the password. Figure 29 shows the two hash methods for encrypting the passwords.

```
   //*****************************************************
        //  Add a new user to the system
        //*****************************************************
        public void AddUser(string User, string Pass, string RName,
char Adm)
        {
            cmdInsertUser = new SqlCommand(strSQLInsert, conConn);
            cmdInsertUser.Parameters.Add("@UserName", SqlDbType.NChar,
10,"UserName");
            cmdInsertUser.Parameters.Add("@Passwrd",
SqlDbType.NVarChar, 50, "Passwrd");
            cmdInsertUser.Parameters.Add("@RealName",
SqlDbType.NVarChar, 50, "RealName");
            cmdInsertUser.Parameters.Add("@Admin", SqlDbType.NChar,
1,"Admin");
            daLogin.InsertCommand = cmdInsertUser;
            // Create hash for password encryption
            HashAlgorithm hashAlgo = HashAlgorithm.Create("SHA1"); //
hash type
            // cvt password to byte array
            byte[] passWord = Encoding.Default.GetBytes(Pass);
            // hash it
            byte[] hashPass = hashAlgo.ComputeHash(passWord);
            // put it in a string for the db
            string strHashPass = Convert.ToBase64String(hashPass);

            // Create new datatable
            DataTable tblUp = new DataTable();
            tblUp = dtaSetLogin.Tables["LoginUsers"];
            DataRow rowUp = tblUp.NewRow();
            rowUp["UserName"] = User;
            rowUp["Passwrd"] = strHashPass;
            rowUp["RealName"] = RName;
            rowUp["Admin"] = Adm;
            tblUp.Rows.Add(rowUp);
            daLogin.Update(dtaSetLogin, "LoginUsers");
        }


//*********************************************************************
        // Check the user name and password for login
        // Returns a bool

//*********************************************************************
        public bool InitialLogin(string Userin, string Pass)
        {
            bool isOK = false;

                string UserPass = Pass.Trim();
                // convert incomming password to hash
                HashAlgorithm hashAlgo = HashAlgorithm.Create("SHA1");

                byte[] inPass = Encoding.Default.GetBytes(UserPass);
                byte[] inPassHash = hashAlgo.ComputeHash(inPass);
                // convert to 64 base string and comp to db password
```

```
                string strUserPass =
Convert.ToBase64String(inPassHash);

                viewLogin.Sort = "UserName";  // find the user
                int intFind = viewLogin.Find(Userin.Trim());


                if (intFind != -1) //  user found
                {
                    string strDBPass =
viewLogin[intFind][1].ToString().Trim();
                    if (strDBPass == strUserPass)
                        isOK = true;
                }




                if (isOK) // password and user found get the real name
for the log
                {
                    this.RealN = viewLogin[intFind][2].ToString();
                    this.Admin = Convert.ToChar(viewLogin[intFind][3]);
                }


            return isOK;

        }
```

Figure 29 Password Encryption

APPENDIX B: Test Forms

The following are the forms we used to test each module. We had the testers fill out the form based on the performance.

Step	Description	Expected Result	Problem with this Step?
	Inventory Navigation Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Inventory	Inventory window is displayed	
4	Click the Next '>' button	the next inventory item is displayed in all fields	
5	Click the Previous '<' button	the previous inventory item is displayed in all fields	
6	Click the 'Close' button	the Inventory window is closed	
		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Inventory Find Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Inventory	Inventory window is displayed	
4	Click the 'Find' button	a blank Unit Number text box is displayed	this would be easier if the list automatically appeared
5	Type a letter	a list of like items is displayed	
6	Select an item from the list	the corresponding item details are listed for the Unit selected	
7	Click the 'Close' button	the Inventory window is closed	
		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Inventory Update Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Inventory	Inventory window is displayed	
4	Click the 'Update' button	The Unit Name field is highlighted.	
5	Update the Unit Name, and hit the 'Update Unit' button	A message box asking 'Are you sure?' will be displayed	
6	Select 'Yes'	the message box will disappear and the Updated Unit Name field will be displayed	
7	Click the 'Close' button	the Inventory window is closed	
		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Inventory Update Cancel Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Inventory	Inventory window is displayed	
4	Click the 'Update' button	The Unit Name field is highlighted.	
5	Update the Unit Name, and hit the 'Cancel' button	the changes you made should change back to the original state (cancelling the change)	
6	Click the 'Close' button	the Inventory window is closed	
		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Inventory Add Unit Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Inventory	Inventory window is	

		displayed	
4	Click the 'Add' button	The Unit Name field is targeted	
5	Fill out all the fields, and click 'Add Unit'	message box appears, 'Unit Added to Inventory. Add Another?'	
6	Click on 'No'	The message box disappears, and the unit you just added should be displayed.	
7	Click the 'Close' button	the Inventory window is closed	

		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Order Navigation Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Orders	Orders window is displayed	
4	Click the Next '>' button	the next order values are displayed in all fields	
5	Click the Previous '<' button	the previous order values are displayed in all fields	
6	Click the 'Close' button	the order window is closed	

		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Order Find Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Orders	Orders window is displayed	
4	Click the 'Find' button	a blank Order Number text box is displayed	it would be easier if we had a list of order numbers to choose from

5	Enter an Order Number	the order values are displayed in all fields	
6	Click the 'Close' button	the order window is closed	

		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Order Add Order Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Orders	Orders window is displayed	
4	Click the 'Create a Order' button	The Find Patient field is targeted	
5	Type in a Patient Name and hit Find Patient	if the name is valid corresponding patients are displayed	this part is confusing, it would be easier if the list of names appeared automatically
6	Double Click on the Patient's name	The Patient is loaded into the form	
7	Click Create Order	The order date fields are enabled for change, as well as the Order Picked Up selection box.	
8	Update the Order Dates and if the Order was Picked Up	all enabled fields should allow change	
9	Click the 'Add' button	the order is added, all fields should be disabled	
10	Click the 'Close' button	the order window is closed	

		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Order Add Order Reset Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Orders	Orders window is displayed	
4	Click the 'Create a Order' button	The Find Patient field is targeted	
5	Type in a Patient Name and hit Find Patient	if the name is valid cooresponding patients are displayed	
6	Double Click on the Patient's name	The Patient is loaded into the form	

7	Click Create Order	The order date fields are enabled for change, as well as the Order Picked Up selection box.
8	Update the Order Dates and if the Order was Picked Up	all enabled fields should allow change
9	Click the 'Reset' button	the field values are reset to their original state
10	Click the 'Close' button	the order window is closed

		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Order Detail Navigation Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Orders	Orders window is displayed	
4	Click the 'Order Details' button	the Order Details window is displayed with the detailed order information	
5	Click the 'Close' button	the order detail window is closed	
6	Click the 'Close' button	the order window is closed	

		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Order Detail Find Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Orders	Orders window is displayed	

		the Order Details window is displayed with the detailed order information
4	Click the 'Order Details' button	a list of the inventory items contained in the order are displayed
5	Click the 'Find' button	the inventory item values are displayed
6	Select an item from the list	the order detail window is closed
7	Click the 'Close' button	the order window is closed
8	Click the 'Close' button	

		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Order Detail Add Unit to Order Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Orders	Orders window is displayed	
4	Click the 'Order Details' button	the Order Details window is displayed with the detailed order information	
5	Click the 'Add' button	a list of inventory items are displayed	
6	select an inventory item from the list	the cooresponding item detail information is displayed	
7	Click the 'Add' button	The item is added to the order	
8	Message Box is displayed to ask if you would like to add another order	Click 'No'	
9	Click the 'Close' button	the order detail window is closed	
10	Click the 'Close' button	the order window is closed	

		Test Result - Pass/Fail	Pass

Step	Description	Expected Result	Problem with this Step?
	Order Detail Remove Unit from Order Test Case		
1	Login	Successful Logon	
2	Click Menu	expanded Menu options	
3	Select Orders	Orders window is displayed	
4	Click the 'Find' button	a list of the inventory items contained in the order are displayed	
5	select an inventory item from the list	the corresponding item detail information is displayed	
6	Click the 'Remove' button	a message box is displayed to ask 'are you sure?'	
7	Click 'Yes' on the Message Box	The Message Box disappears and the item is removed from the order	
8	Click the 'Reset' button	the field values are reset to their original state	
9	Click the 'Close' button	the order window is closed	

		Test Result - Pass/Fail	Pass