Bowling Pro-Shop Customer Inventory Manager

By

Sean Brady

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 2004

Bowling Pro-Shop Customer Inventory Manager

by

Sean Brady

Submitted
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

_____	_____
Sean Brady	Date
_____	_____
Professor Tamisra Sanyal, Faculty Advisor	Date
_____	_____
James F. Sullivan, Department Head	Date

Acknowledgements/Dedication

I would like to extend a special thank you to Bob Katzler who helped me devise many of the ideas for creating this project. I had many conversations including the one that is outlined later in this paper that helped me to create a user-friendly product for the bowling pro-shop business. I would also like to thank everyone that helped test my project to ensure the project would be successful on the market.

Table of Contents

List of Figures

Abstract

The *Bowling Pro-Shop Customer Inventory Manager* consists of an interface that allows pro-shop operators to keep track of customer information, store inventory, and sales. The project allows for the user to print multiple reports and receipts for customers and the pro-shop itself. The project contains a Windows GUI interface with a relational database on the backend. The Bowling Pro-Shop CIM was designed using one of the latest programming languages, Visual Basic .NET. The database backend of the project is a Microsoft Access 2000 database. The project is easy to install because it is designed for use on Microsoft Windows 98/2000/XP operating systems. Thus, if the operator has Windows it will be no problem to install the project. Overall, the project allows the user to more effectively run a pro-shop business.

1. Statement of Problem:

One of the largest participation sports in the United States is bowling. Literally thousands of bowling pro-shops have opened to accommodate bowlers' needs. Bowling pro-shops differ in the products and services they provide. However, most pro-shops offer a variety of bowling balls, shoes, accessories, and ball drilling.

The use of a general inventory database would be helpful for many bowling pro-shops. However, bowling pro-shops offer unique services, such as drilling a bowling ball, which would be better supported by a bowling-specific inventory database model.

After discussions with Bob Katzler, owner of Strikeline Pro-Shop, in January of 2003 I had the idea to create a bowling-specific inventory and drilling database. Mr. Katzler has been an independent pro-shop operator for 31 years and has a great insight into the pro-shop business. The features he felt were essential to the success of a bowling specific inventory and drilling database included:

- The program should be capable of printing receipts and reports for customers and company records
- The program should be easy to use. Since, those that will be using the program will most likely not be computer experts, it is essential the system be simple.
- The program should contain a form which will allow the pro-shop operator to input custom customer measurements without the use of paper.
- The program should also be able to contain customer purchase history, which will allow the pro-shop to send out information that would be relevant to customers.

While pro-shops around the United States range in size from single person owned shops to multiple location franchises, they all have similar characteristics that would make it easy to design a bowling specific inventory and drilling database. The similar characteristics include:

- Employees with a lack of computer skills
- A large range of product inventory
- Small employee staffs

For marketability, the Bowling Pro-Shop Customer Inventory Manager should be easy to install. The program should allow multiple licenses so pro-shop owners with more than one shop could use the program at all of their locations.

2. Description of Solution:

The solution to the problem was to create a program that would allow pro-shop operators to keep track of customer information, store inventory, and sales. The program, called the Bowling Pro-Shop Customer Inventory Manager, also allows the user to print multiple reports and receipts for customers and the pro-shop itself.

One of the major requirements that the program needed was that it be easy to learn and operate. Some key sources that were useful in creating the program include:

- Discussions with local pro-shop owners
- My knowledge of the industry as an employee in a pro-shop for 5 years
- Testing of the program with local pro-shop owners throughout development
- Looking at competitors software to locate features that could be added to this project such as Ebonite's Bowling Pro Shop Coordinator

The program is easy to install because it is designed for use on Microsoft Windows 98/2000/XP operating systems. Thus, if the operator has Windows it will be no problem to install the program. The program contains a Windows GUI interface with a database on the backend. The program allows for a variety of reports, including receipts, reports on customer purchase history, drilling sheets, and mailing lists.

2.1 User Profile:

Pro-shop workers consist of those who are computer savvy to those who lack computer skills. To cater to users of various skill levels the *Bowling Pro-Shop CIM* is designed to be a user friendly product that allows anyone with very basic knowledge of the Windows operating system to use the program. The only knowledge the user needs is to know how to open a program and how to point and click within a GUI interface.

2.2 Design Protocols

The resources that were needed to create the *Bowling Pro-Shop CIM* included software, hardware, money, and time. The software resources that were used are Microsoft Access and Visual Basic .Net.

Originally the plan was to use SQL Server 2000 for the database backend of the *Bowling Pro-Shop CIM.* However, after weighing the costs and benefits associated with SQL Server 2000 I found it was unnecessary to use the software. Instead Access was used for the project. Microsoft Access provided enough power and options for the database backend of the project. Using Access also cut down on production costs. This is the software responsible for developing the database that contains all of the customer, inventory, and drilling information. A screenshot of the relationships as well as the design of the finished tables in the database can be seen on the next couple pages.

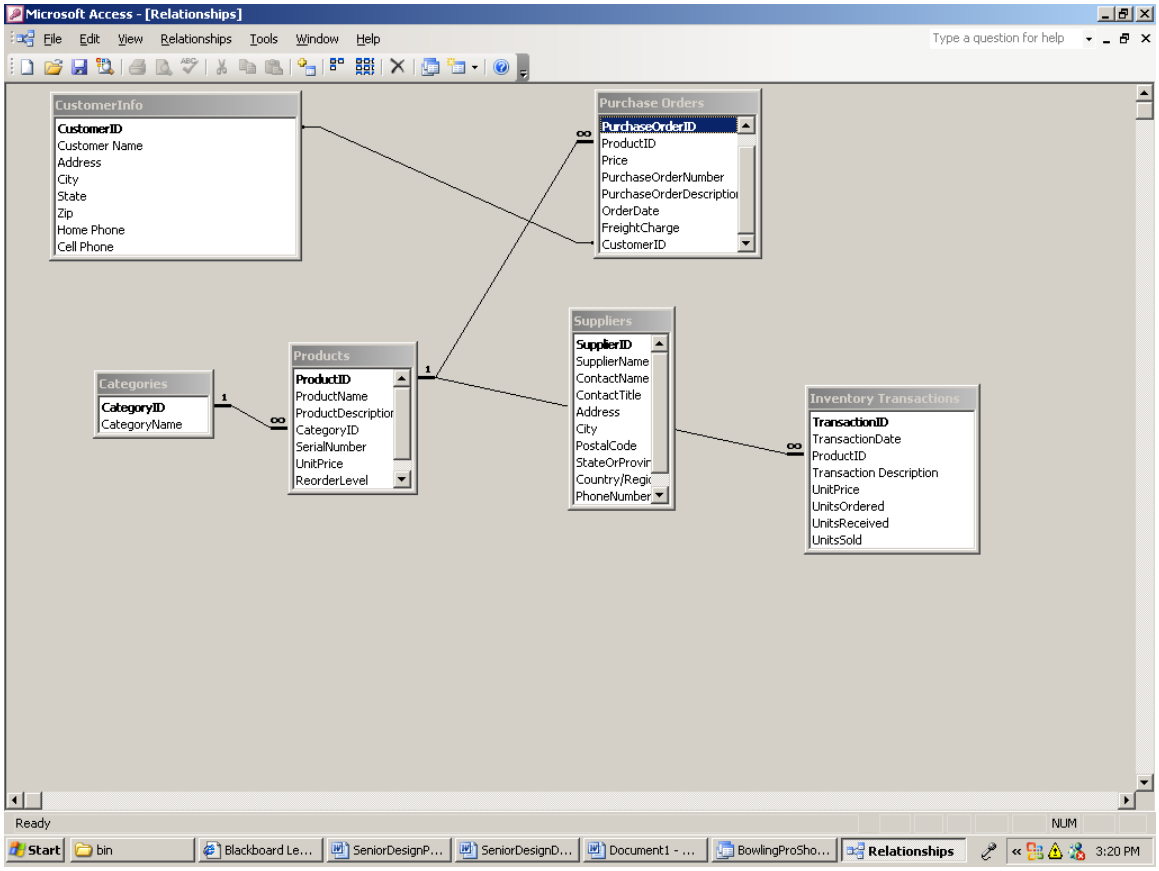

Figure 1. Bowling CIM relationships

Table Design in Microsoft Access for Bowling Pro-Shop CIM

Field Name	Data Type
CategoryID	AutoNumber
CategoryName	Text

Field Name	Data Type
EmployeeID	AutoNumber
FirstName	Text
LastName	Text
Address	Text
Extension	Text
PhoneNo	Text

Field Name	Data Type
TransactionID	AutoNumber
TransactionDate	Date/Time
ProductID	Number
TransactionDescription	Text
UnitPrice	Currency
UnitsOrdered	Number
UnitsReceived	Number
UnitsSold	Number
UnitsShrinkage	Number

Field Name	Data Type
ProductID	AutoNumber
ProductName	Text
ProductDescription	Text
CategoryID	Number
SerialNumber	Text
UnitPrice	Currency
ReorderLevel	Number
LeadTime	Text

Field Name	Data Type
Customer Name	Text
PurchaseOrderID	AutoNumber
ProductID	Number
Price	Currency
PurchaseOrderNumber	Text
PurchaseOrderDescription	Text
SupplierID	Number
EmployeeID	Number
OrderDate	Date/Time
DateRequired	Date/Time
DatePromised	Date/Time
ShipDate	Date/Time
ShippingMethodID	Number
FreightCharge	Currency

Field Name	Data Type
ShippingMethodID	AutoNumber
ShippingMethod	Text

Field Name	Data Type
SupplierID	AutoNumber
SupplierName	Text
ContactName	Text
ContactTitle	Text
Address	Text
City	Text
PostalCode	Text
StateOrProvince	Text
Country/Region	Text
PhoneNumber	Text
FaxNumber	Text

Figure 2. Table Design

3. Deliverables:

1. A GUI interface that will allow pro-shop operators to keep track of customer information, store inventory, and sales.
2. An interface that is easy to navigate even for users with basic knowledge of Windows.
3. Allow users to print receipts and reports for customers and company records.
4. Will contain a form to input customers' individual drilling measurements.
5. Will connect to the database to view customer purchase history.
6. A backend relational database that will contain all customer and store information.

4. Design and Development

The next section describes the budget, timeline, and the software/hardware used in the project.

4.1 Budget:

The costs for these products are shown in figure 2.

Software	***Hardware***	***Developer***	***Price***
Access		Microsoft	$499.00
VB. Net		Microsoft	$1079.00
	Pentium IV 2.0 GHz Dimension 2350 Series	Dell	$1007.00
Total			**$2585.00**

Figure 3. Price of software and hardware required for project

4.2 Timeline:

Task	Start Date	End Date	Completed
Research Project	1/6/2003	1/30/2003	Yes
Project Approval	1/20/2003	1/24/2003	Yes
Continue Research	1/24/2003	2/22/2003	Yes
First Draft of Proposal	2/12/2003	2/26/2003	Yes
Revise Proposal	3/1/2003	3/8/2003	Yes
Prepare Presentation	3/3/2003	3/11/2003	Yes
Presentation	3/12/2003	3/12/2003	Yes
Develop Database	6/23/2003	7/28/2003	Yes
Test Database	7/28/2003	7/28/2003	Yes
Develop GUI Interface	6/23/2003	8/12/2003	Yes
Test GUI Interface	8/13/2003	8/13/2003	Yes
Connect Database to GUI	8/14/2003	8/14/2003	Yes
Test Connection	8/14/2003	8/15/2003	Yes
First Draft of Design Freeze	7/28/2003	7/28/2003	Yes
Revise Design Freeze	8/4/2003	8/15/2003	Yes
Prepare Presentation	8/10/2003	8/18/2003	Yes
Presentation	8/18/2003	8/18/2003	Yes
Tweak/Polish Prototype	4/1/2004	5/9/2004	Yes
Build Final Project	5/9/2004	5/22/2004	Yes
Test Final Project	5/22/2004	5/26/2004	Yes
Prepare Final Report	4/20/2004	5/06/2004	Yes
Revise Final Report	5/13/2004	5/26/2004	Yes
Prepare Presentation	5/20/2004	5/27/2004	Yes
Presentation	5/27/2004	5/27/2004	Yes

4.3 Software

Visual Basic .Net was the software used for the Windows based GUI interface that allowed mouse point and click functionality. The reason for the choice of Visual Basic .Net was less about the user of the program and more about the actual design of the program. Visual Basic .Net is designer friendly, allowing "you to create rich applications for Microsoft Windows® in less time, incorporate data access from a wider range of database scenarios, create components with minimal code, and build Web-based

applications using the skills you already have" (Microsoft). Since, the drilling interface will be a major part of the program, it is necessary to have a software package that can produce a professional looking application. Visual Basic .Net offers the ability for the designer to build robust Windows applications with "new features such as automatic control resizing eliminate the need for complex resize code. New controls such as the in-place menu editor deliver visual authoring of menus directly within the Windows Forms Designer. Combined with greater application responsiveness, as well as simplified localization and accessibility, these new features in Windows Forms make Visual Basic .NET the choice for today's Visual Basic developers" (Microsoft).

4.4 Hardware:

Due to the requirements of the software, the hardware needed for the project was a Dell 2.0 GHz Pentium IV Dimension 2350 series. The system has Windows XP, 512 MB DDR RAM, 60 GB hard drive, 40 X/10 X/40 X CD-RW drives, and a 17 inch monitor. This system was more than sufficient for this project.

5. Proof of Design

The next section describes in detail how deliverables of the project were fulfilled and what challenges I encountered.

5.1 GUI Interface

One of the major features that the program needed was to have a GUI interface that would allow pro-shop operators to keep track of customer information, store inventory, and sales. As you can see below, the GUI interface allows point and click functionality for the user to do this.

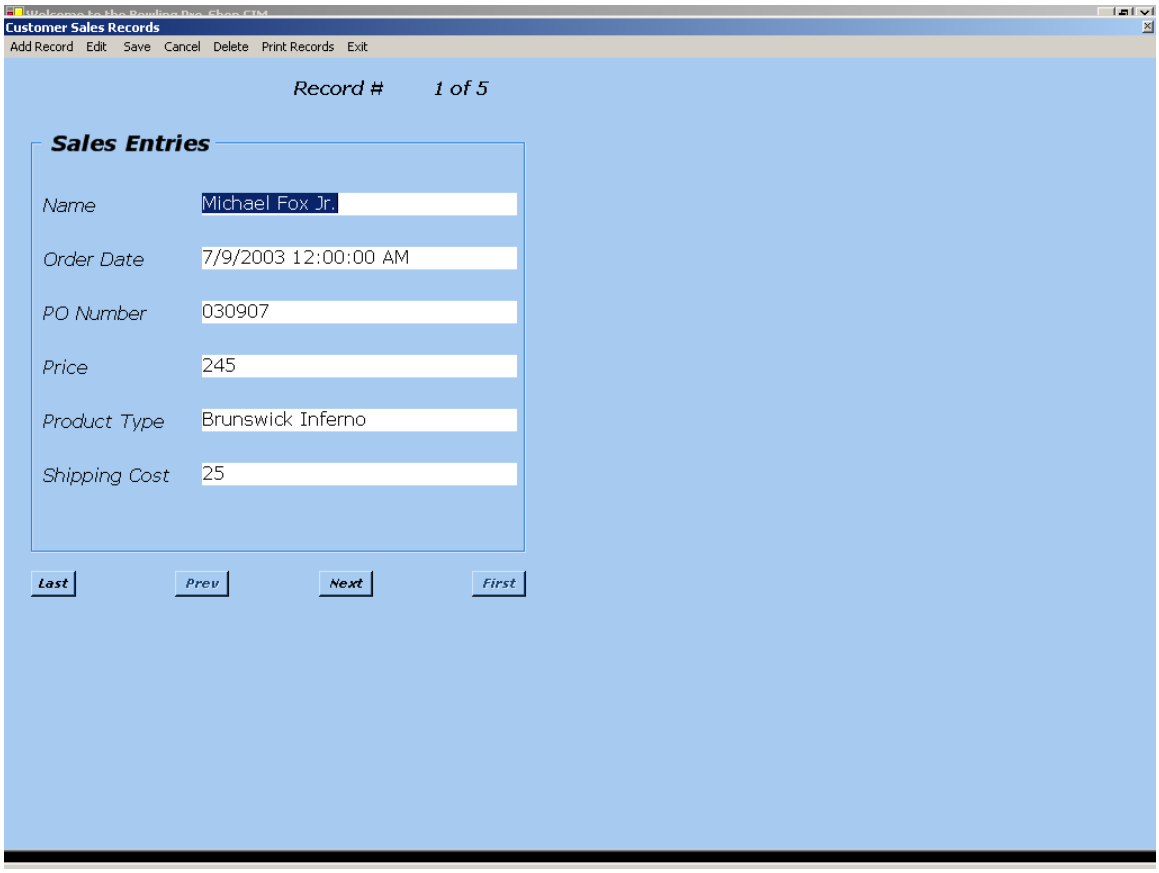

Figure 4. GUI Interface

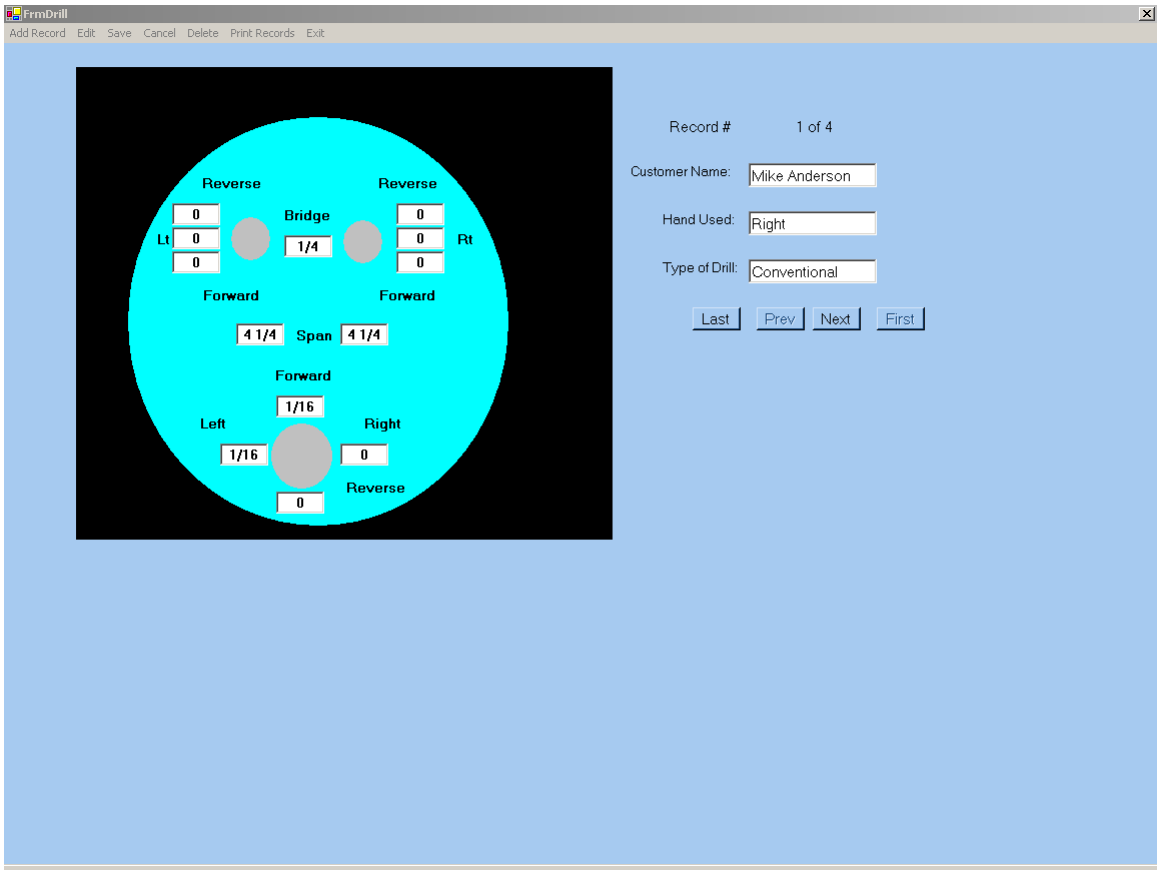

Figure 5. GUI Interface

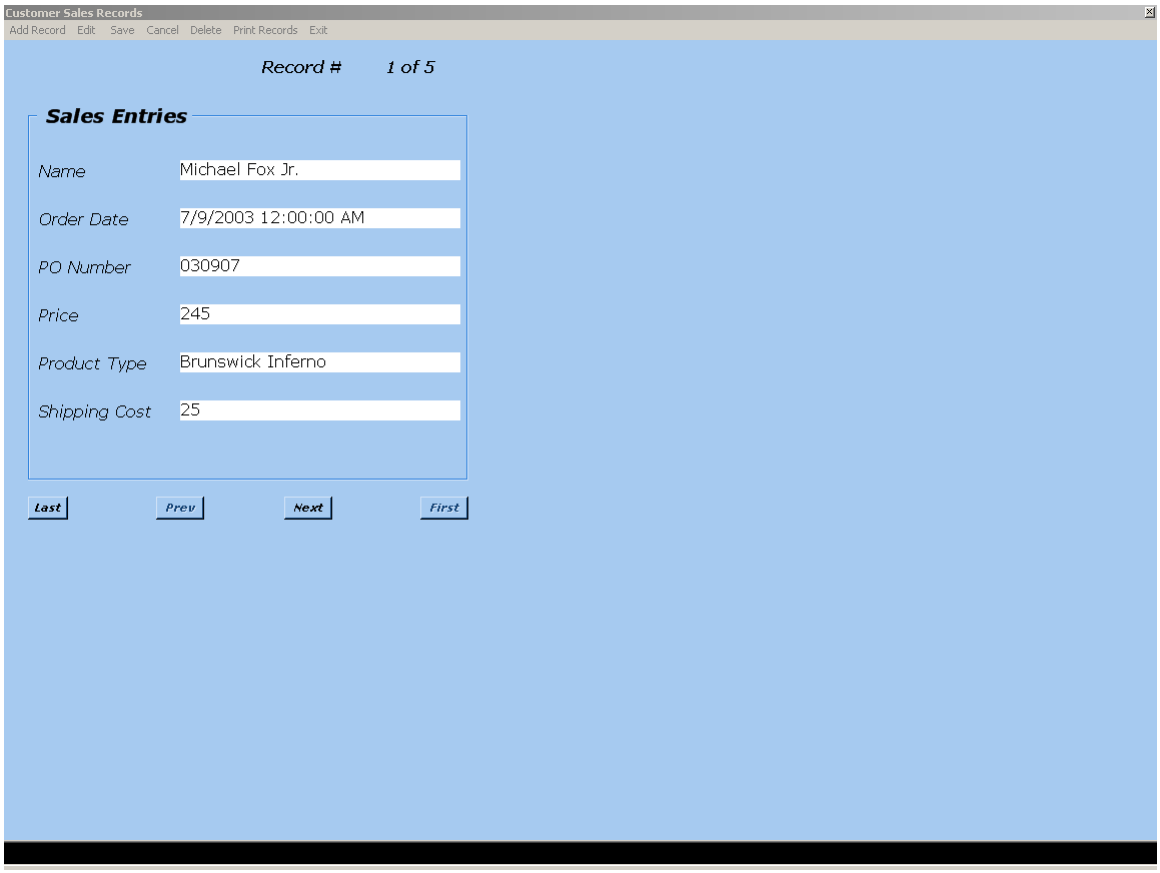

Figure 6. GUI Interface

5.2 Easy to Navigate Interface

Another key to creating a program that all pro-shop operators could use was to enable easy navigation throughout the Bowling Pro-Shop CIM. All, navigation is done through menus at the top of the page. There are a few buttons as well in the project which allow the user to print receipts and reports. Users with minimal knowledge of computers should be able to start using this program and understand it within minutes. Below, are examples of the menu driven system.

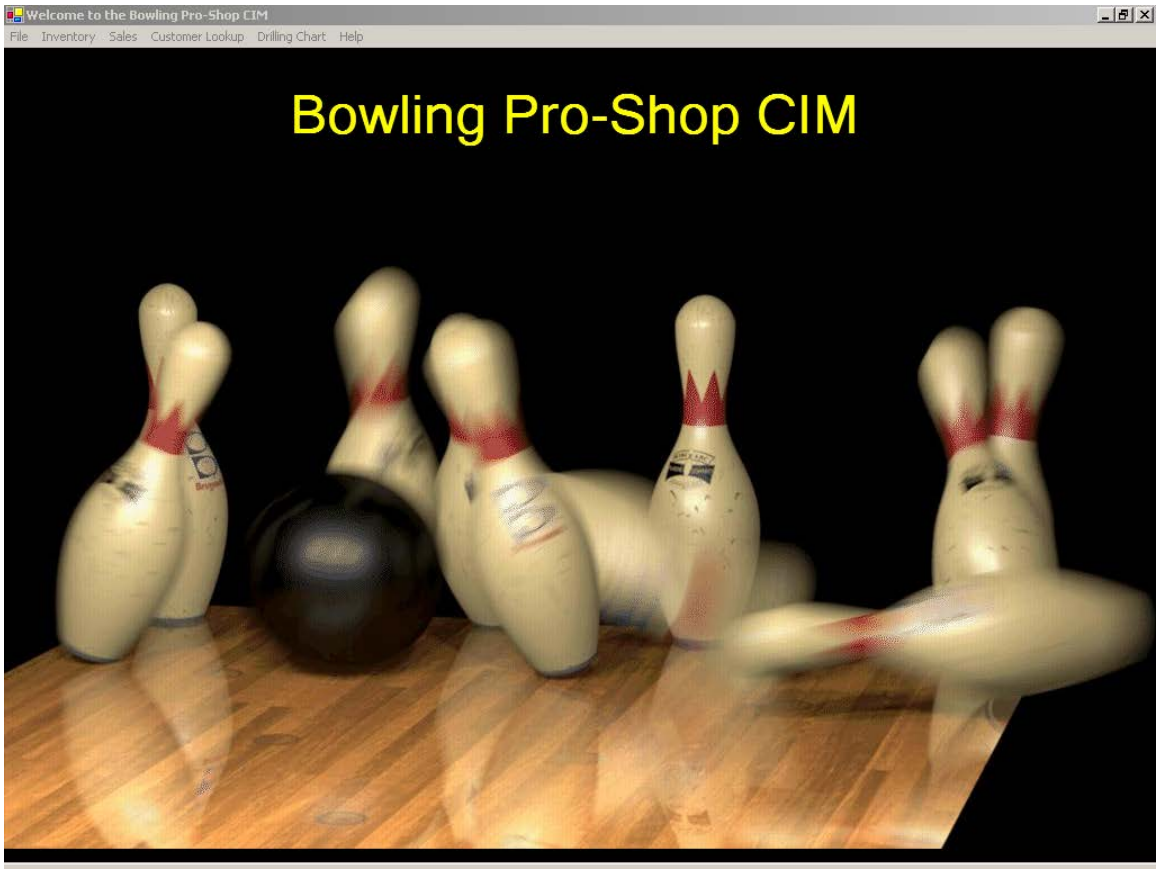

Figure 7. Navigation Example

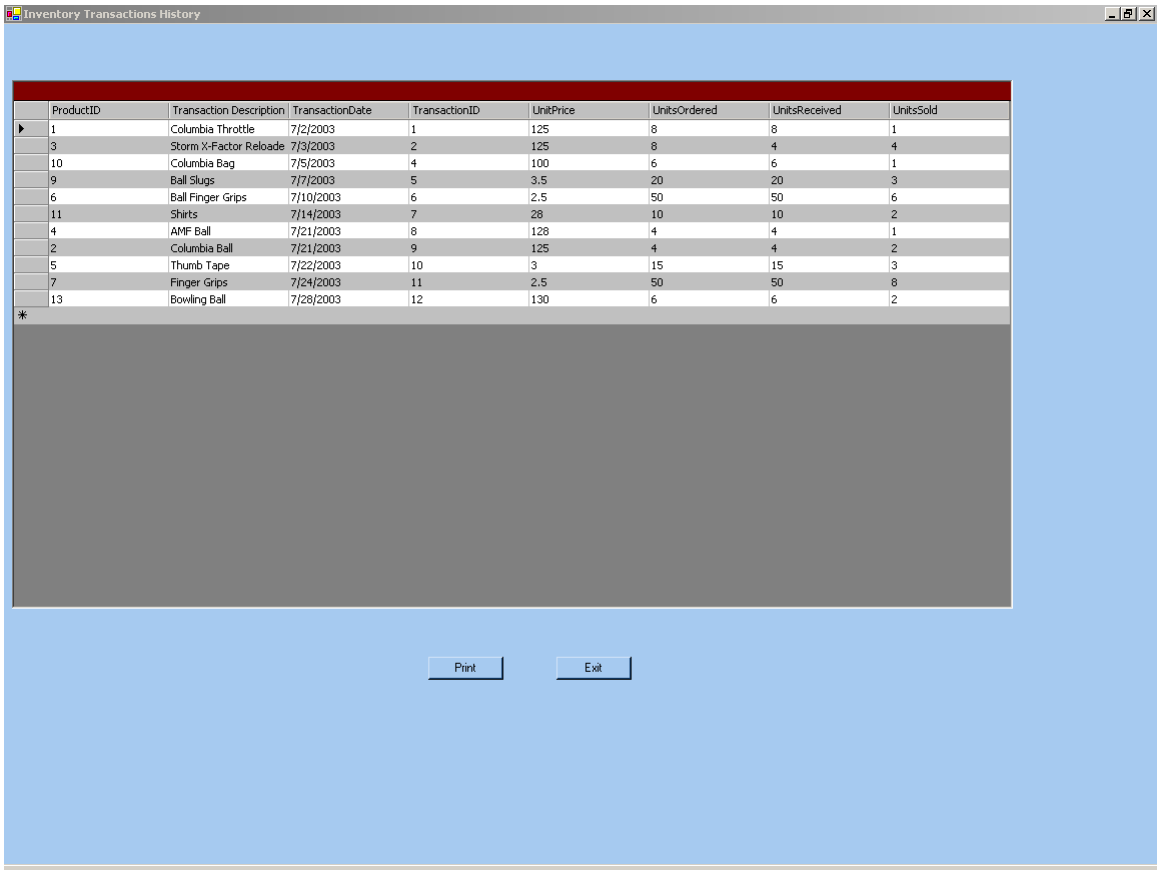

Figure 8. Navigation Example

5.3 Print Receipts and Reports

The ability for the user to be able to print receipts and reports for company and customer use is very important. Without this feature the Bowling Pro-Shop CIM would be a nice tool to keep track of inventory and sales. However, the program was designed to enable pro-shops more efficiently run their businesses. You can view screenshots on the next couple pages of reports and receipts that are available for printing.

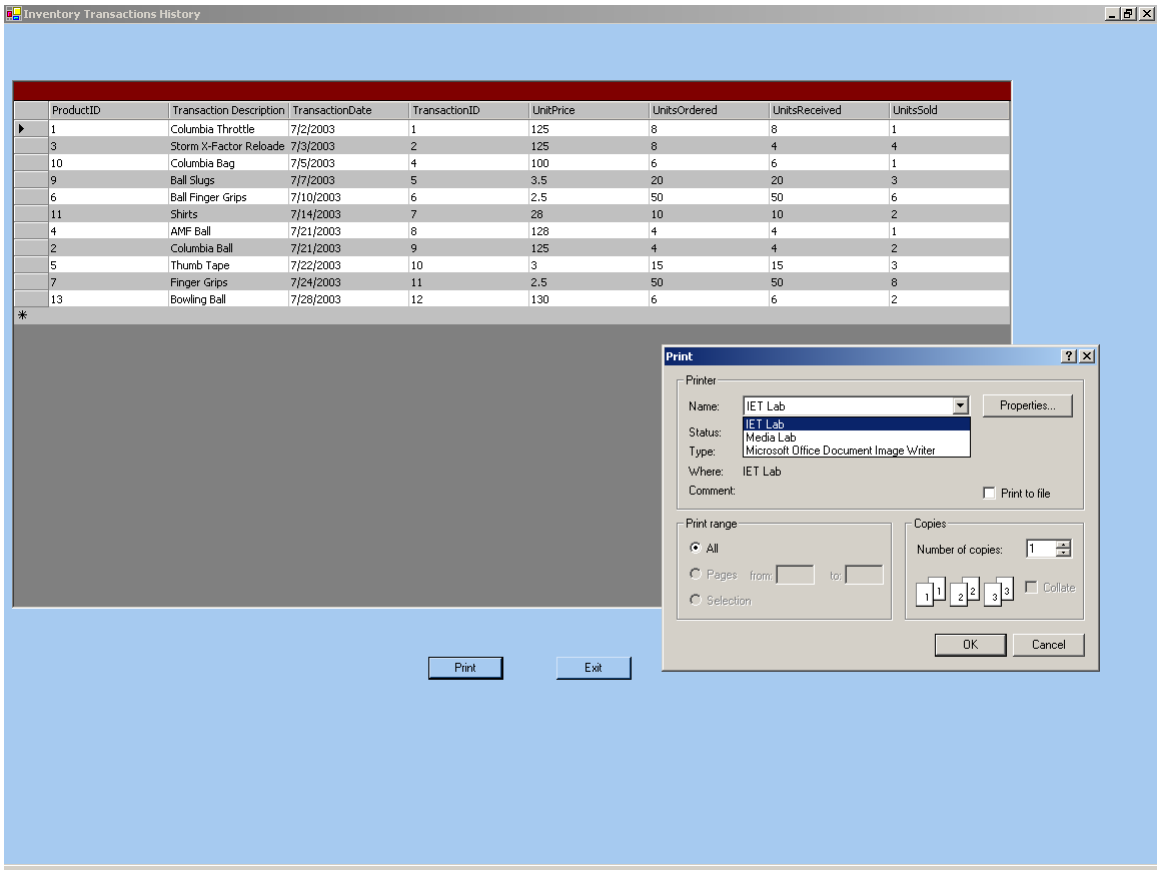

Figure 9. Printing Capability

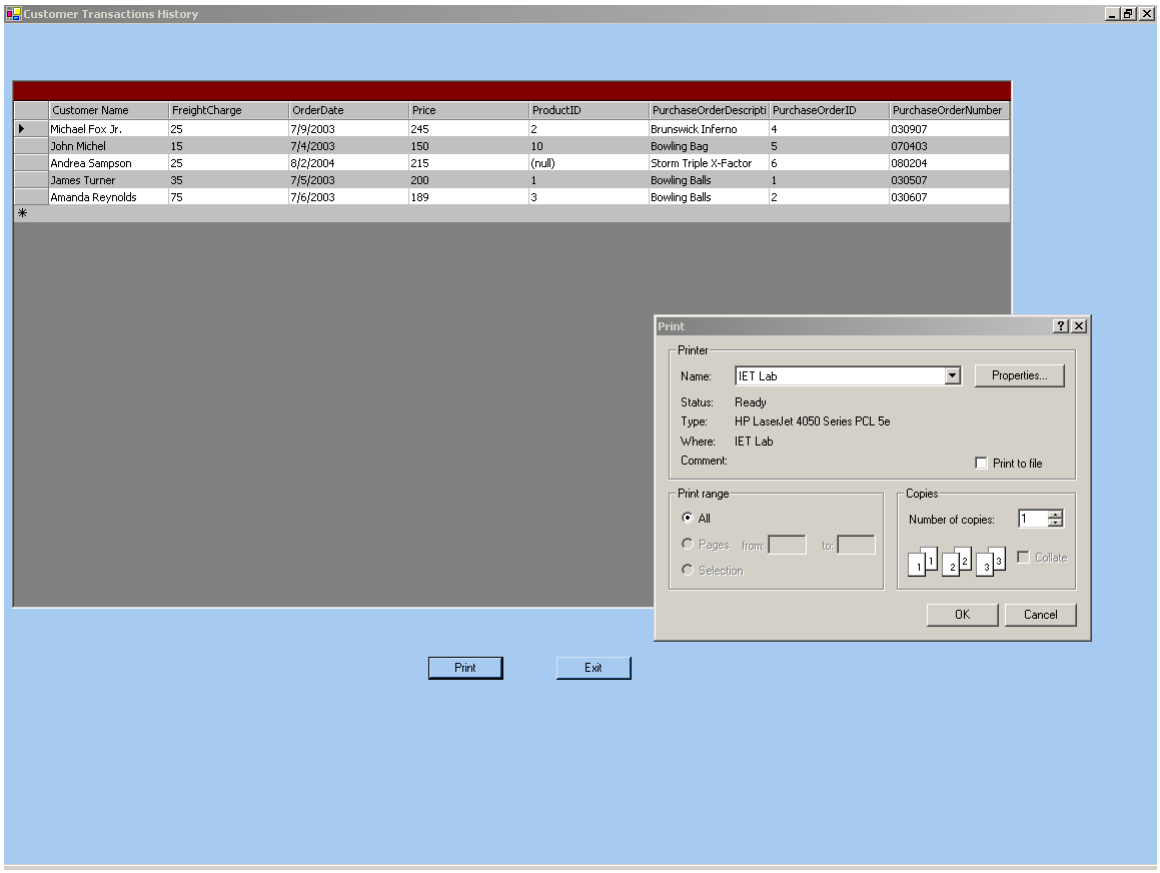

Figure 10. Printing Capability

5.4 Input Drilling Measurements

This is the most crucial element of the project as it separates other inventory database programs from a bowling specific program. The ability to input customer measurements through the use of a form and keep them on record is a very important feature. Refer to figure 5 on page 16 to see the form to input the customer drilling information can be seen on the next page.

5.5 Ability to View Customer Purchase History

The ability to track and view customer purchase history is vital to any business. This program allows the pro-shop operator to do this through point and click functionality. Users are able to edit previous purchases, print all purchases, and view

individual purchases. Below is an example of printing all customer purchases in the shop's history.

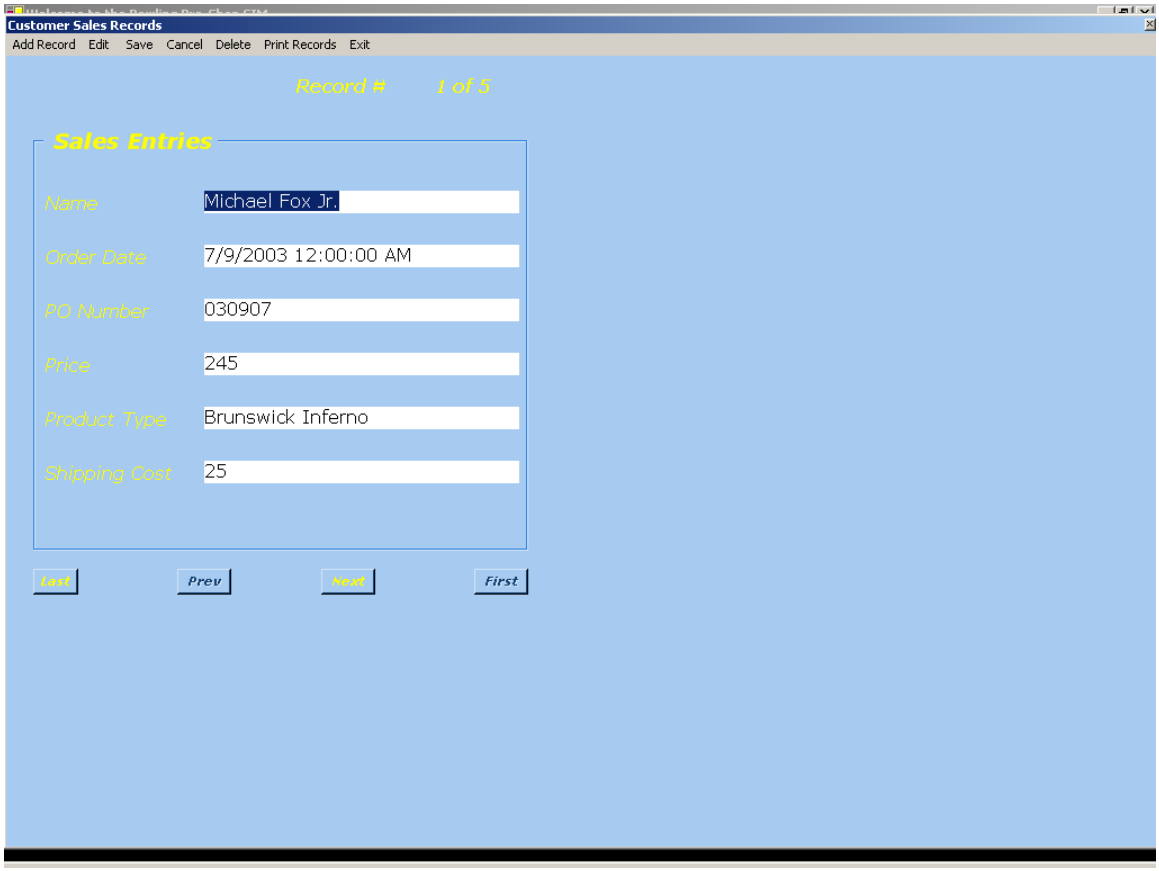

Figure 11. Customer Purchases History

5.6 Backend Relational Database

The backend relational database is a Microsoft Access 2000 database. This database contains all information pertaining to the customer and store. Any information that is input or edited in the project is stored in this database. Refer to figure 1 on page 10 to see the relationships that exist amongst the tables in the database.

5.7 Proof of Testing

To test the Bowling Pro-Shop CIM I went to Strikeline Pro-Shop and had Bob Katzler test the product. You can refer to our initial conversation about the creation of the product on page 2. I installed Visual Studio .Net and Microsoft Office as he had neither. He went through the various forms and found the product easy to navigate. He felt the overall layout was good and that he would be able to implement the product into his business if he chose to use the program.

Other than Mr. Katzler I also had a few peers test the program and one told me that they felt I needed a help section in the program. Thus, I added that section. Another told me that color scheme was too dark. Thus, I lightened the background to blue and made the words yellow.

Through testing I also found a major flaw in the database connection with the GUI interface. I had all of the database connections set to a path of a zip disk location of e:\. This caused errors on systems that did not have this path or that were pointed to a CD-Rom location. Thus, I changed the path to c:\ and have provided a text file instructing the user where to place the database file on their system before using the Bowling Pro-Shop CIM. To conclude, testing of the product was one of the major keys in creating a successful product.

6. Conclusions and Recommendations

6.1 Conclusions

This project was created due to the lack of bowling specific customer inventory software available. I created a Windows GUI interface that uses a database backend to

enable virtually any pro-shop operator to use the program with very little learning time required. The project was designed using the Visual Basic .Net programming language and Microsoft Access 2000 for the backend relational database. The project wa s completed over the three quarter Senior Design sequence which took place from 1/03-6/04. The budget of $2600 is an estimate for the completion of this project if it were to be completed outside of an educational institution. The project fulfilled all Design Freeze deliverables. Testing was performed both at OCAS and at a few bowling pro-shops around Cincinnati to ensure the product's ease of use and marketability.

6.2 Recommendations

Creating this project has been both a challenging and rewarding experience. When I started this project in January of 2003 I chose to use Visual Basic .Net because I wanted to learn the programming language and I was unable to learn it through the IET program. Looking back I would choose no differently. I could have decided to use a programming language that I had learned such as C# or Visual Basic 6.0. But, I felt that learning something that was not taught at the time in IET would only enhance my marketability once I graduated. I recommend to anyone just starting the Senior Design sequence to try a project that is both challenging and something that you will truly learn from.

As far as learning Visual Basic .Net and connecting the backend relational database designed in Access goes, this was very challenging for me. I had to use many sources throughout the 15 month period of designing the project. But, my best source for discovering how to program VB .Net was the Internet. I was able to find many examples of programs through tutorial websites that enabled me to move quickly from phase to

phase in the project. The movement through the project phases was quite a bit faster than if I would have just used books to learn the coding. So, this is another recommendation to future Senior Design students, use the Internet often.

I also strongly suggest that all Senior Design participants test their projects at least a couple weeks prior to presenting their projects at the end of Senior Design III. This can prevent errors in code or design that you might not have been able to find if you had not tested the project.

Finally, I believe the best way to make it through the Senior Design sequence is to be able to trust your advisor. Don't be afraid to ask questions, have them read your papers, look at your code, or have them watch a practice presentation. The advisors are there for your own good and are often a great source for help.

Appendix A.

Code Snippets

C 1. Menu Navigation Sample Code

The code seen on the following pages is basically the subs that are called when the user clicks on the menus in the Bowling Pro-Shop CIM. For instance, when the user wants to edit an entry, they press the button edit on the menu. The Edit sub is then called and the message appears to enter the changes and press save.

```
Private Sub Add()
    '[Add a new entry]
    Try
        Me.BindingContext(DsSales1, _
          "CustomerInfo").EndCurrentEdit()
        Me.BindingContext(DsSales1, _
          "CustomerInfo").AddNew()

    Catch eAdd As System.Exception
        MessageBox.Show(eAdd.Message)
    End Try
    '[Add a new entry/]

    '[See custom procedures]
    Me.dssales_PositionChanged()
    '[See custom procedures/]

    '[Focus the first textbox]
    txtName.Focus()
    '[Focus the first textbox/]

End Sub

Private Sub Edit()
    MessageBox.Show("Please enter the changes and press save")
End Sub

Private Sub Save()

    '[Update]
    Try
        Me.BindingContext(DsSales1, _
          "CustomerInfo").EndCurrentEdit()
        daSales.Update(DsSales1, _
          "CustomerInfo")

    Catch eSave As System.Exception
        MessageBox.Show(eSave.Message)
    End Try
    '[Update/]
```

```
    '[See custom procedures]
    Me.dssales_PositionChanged()
    '[See custom procedures/]


End Sub

Private Sub Cancel()

    '[Ask for confirmation]
    If MessageBox.Show("This will cancel the current entry." _
      & ControlChars.NewLine & "Do you want to continue?", _
      "Sample", MessageBoxButtons.YesNo, _
      MessageBoxIcon.Question) = DialogResult.Yes Then
        '[Ask for confirmation/]
        '[If the user confirms, cancel the entry]
        Me.BindingContext(DsSales1, _
        "CustomerInfo").CancelCurrentEdit()
        '[If the user confirms, cancel the entry/]
        '[See custom procedures]
        Me.dssales_PositionChanged()
        '[See custom procedures/]

    End If

End Sub

Private Sub Delete()

    '[Ask for confirmation]
    If MessageBox.Show("This will delete the current entry." _
      & ControlChars.NewLine & "Do you want to continue?", _
      "Sample", MessageBoxButtons.YesNo, _
      MessageBoxIcon.Question) = DialogResult.Yes Then
        '[Ask for confirmation/]

        '[Delete the entry]
        If (Me.BindingContext(DsSales1, _
          "CustomerInfo").Count > 0) Then
            Me.BindingContext(DsSales1, _
            "CustomerInfo").RemoveAt(Me.BindingContext(DsSales1, _
            "CustomerInfo").Position)
            '[Delete the entry/]

            '[See custom procedures]
            Me.dssales_PositionChanged()
            '[See custom procedures/]

        End If

    End If

End Sub
```

C 2. Database Connection

This code changes the database connection to the application startup path. Therefore, the database will connect when placed in the application's path.

```
Try
    Me.odcsales.ConnectionString = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source= bowlingproshopsd.mdb"
Catch eConnection As System.Exception
    MessageBox.Show(eConnection.Message)
End Try
```

C 3. Printing Capability

This code enables the user to print receipts and reports throughout the program. This sets the program apart from others and allows pro-shop operators to run their businesses more efficiently.

```
Private Sub frmSalesPrint_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Try
            Me.dcPur.ConnectionString = _
            "Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source= bowlingproshopsd.mdb"
          Catch eConnection As System.Exception
            MessageBox.Show(eConnection.Message)
      End Try
      dapur.Fill(Dspur)
 End Sub
```

C 4. Database Connection

This code loads the correct data from the database into the Windows GUI interface. The example below shows the Sales form loading data from the database. The odasales represents the data adapter and the dssales represents the dataset. The Purchase_Orders represents the table in the database that the data is being pulled from.

```
Try
    odasales.Fill(Dssales)
    lblNavNum.Text = _
    Me.BindingContext(Dssales.Purchase_Orders).Position
Catch eLoad As System.Exception
    MessageBox.Show(eLoad.Message)
End Try
```

References

1. Katzler, Bob. Bowling Pro-Shop Owner. Email communication. January 20, 2003.

2. "Product Overview for Visual Basic .Net". http://msdn.microsoft.com/vbasic/productinfo/overview/default.asp. February 15, 2003.

3. "Product Overview". http://www.microsoft.com/sql/evaluation/overview/default.asp. February 15, 2003.

4. "System Requirements for Visual Basic .NET". http://msdn.microsoft.com/vbasic/productinfo/sysreqs/default.asp. March 5, 2003.

5. "Dell- Dell Home Systems Dimension 2350 Desktop". http://www.dell.com/us/en/dhs/products/model_dimen_1_dimen_2350.htm. March 5, 2003.

6. Riordan, Rebecca. *Microsoft ADO .NET Step by Step*. Redmond: Microsoft Press, 2002.

7. Waymire, Richard. *Sams Teach Yourself SQL Server 2000 in 21 Days.* Indianapolis: Sams Publishing, 2003.