The Complete Inventory Tyrant:
A complete inventory control system

By

Brian Seabolt

Submitted to
The Faculty of the Computer Science Technology Program
In Partial Fulfillment of the Requirements
For the Degree of Bachelor of Science
In Computer Science Technology

© Copyright 2002 Brian Seabolt

The information in this document is proprietary and may not be reproduced or distributed in whole or in part without the permission of the owner.

Brian Seabolt — June 2, 2005
Brian Seabolt — Date

Russell E McMahon — June 2, 2005
Russ McMahon, Faculty Advisor — Date

Patrick L. Kumpf — 6-2-05
Patrick C. Kumpf, Ed.D.
Interim Department Head — Date

The Complete Inventory Tyrant:

A Total Inventory Control System

By

Brian Seabolt

Submitted to

The Faculty of the Computer Science Technology Program

In Partial Fulfillment of the Requirements for

The Degree of Bachelor of Science

In Computer Science Technology

University of Cincinnati
College of Applied Science

May 2005

The Abstract

The Complete Inventory Tyrant is a complete inventory control system targeting the game and hobby industry. The primary goal of the Complete Inventory Tyrant is to assist in management of retail and sales. The main focus of the Inventory Tyrant is to replace the traditional cash register and to monitor the transactions taking place within the store. The Inventory Tyrant also contains a point of sale system that allows for record keeping and conducts the completed sale transaction. By connecting the Point of Sale aspect with the record tracking and managerial side of the retail business, the managers are better able to gain an understanding of how and why the business is working the way it is. The Complete Inventory Tyrant is broken down into three programs: a Administrative Console, the partition of Inventory Tyrant that controls the security of Inventory Tyrant; the Managerial Toolset, a group of tools that allow a manager to control company information, contact information, and other managerial tasks; and finally, the Transaction Controller, which replaces the standard cash register with a Point of Sale system. The Complete Inventory Tyrant was created using the Visual Studio Dot Net 2005 Beta, with the database being powered with Sql Server 2005 Beta.

Table of Contents

Section **Page**

Table of Figures

1. The Problem

The goal of any retail business is to increase its profit margin. However, there are many small businesses in this Nation that cannot afford the cost of something as key to success as an Inventory management system, or Point of Sale system. Without a Point of Sale system, managers are forced to spend a huge sum of time manually computing numbers and conducting inventory audits via pen and paper. Obviously, time is money, and while a complete Point of Sale system is extremely expensive in the eyes of a small business, the time wasted on manually completing tasks that are easily automated slowly piles up. (8, pp. 1-3.)

More and more retailers are moving from a standard cash register to a Point of Sale system. They are doing this because a Point of Sale system increases productivity for various different reasons. A Point of Sale system can give a retailer the ability to generate better revenues, increase productivity by automating tasks, and finally allows the managers more time to live their lives instead of having to manually file paperwork. (4.)

As stated above, the upfront cost of a Standard Point of Sale System is extremely costly, and many small businesses cannot afford such a system, including the small hobby and game store I currently work for. This company needs a Point of Sale system, but cannot afford such a system. Currently the methods used involve pen, paper, and an Excel spreadsheet. Obviously this is horribly inefficient. This method also has huge security risks, and almost no ability to have access to a real time in-stock inventory report. If this business were audited, it would suffer horribly, because currently inventory by hand takes at least four hours to complete. This project benefited both me and the growth of the company because I focused my senior design project on building them something they need with little or no cost to them. I planned on building a Complete Point of Sale system targeting the game and hobby industry, conveniently named "Inventory Tyrant."

By conducting multiple interviews with the various owners and the manager of this store, I was able to learn about everything they want from a Point of Sale system. Some of the reoccurring items that both the owners and the manger needed include:

- The ability to scan an item via a barcode system so the employee does not have to type everything in by hand.
- The system would need to print a receipt via thermal printer, and would have the store's logo on the receipt
- The User Interface must be easy to use and learn. The employees should not have to have years of computer experience to be able to handle this system.
- Have the ability to print real time inventory data and other reports for both sales analysis and backup documentation.
- Have the ability to monitor which Employee commits the transaction.
- Have the data secure so that only the people that are suppose to view the data can view it.

While there are other features not yet organized by the different owners and the manager, these points are the basic requirements of the project. These are the main issues I took upon my shoulders to solve with The Complete Inventory Tyrant.

2. Solution

The Solution to this problem was a simple one. I was going to build ACME Games a state of the art Software driven solution that not only offers a Point of Sale system, but control inventory, sales analysis, and employee control. The Complete Inventory Tyrant's main goal was to handle inventory control through the point of sale system, but also handle other various aspects of the retail business. Like the name implies, my software handles all aspects of Inventory, from adding a new item to inventory to crunching of numbers for sales analysis. My Software met all the requirements stated by the management of ACME Games, plus some, and will be rolled out for full use by the end of September. After the initial roll out of my product, I look to

sell my software on the market as a major competitor of Quicken and WASP Point of Sale Software.

3. User Profiles

There are going to be three types of users for this application. There will be the teller, the manager, and the administrator.

3.1 Teller

The teller is the employee sitting behind the cash register. They will only be allowed to complete a sale, and entering sales data into the database. The Teller will not be required to know much, if anything about computers. All they will be required to know is how to operate a simple front end and the barcode scanner.

3.2 Manager

The manager will be able to do everything the teller can do, as well as generate reports and view the sales data. Managers will be allowed to change non-critical data. Managers will be required to know a little more about computers and will need basic understanding of databases and will need to understand Microsoft word style report documents.

3. 3 Administrators

Administrators will have access to everything in the system. They will be in charge of data maintenance and will be allowed to add Tellers and Managers to the system. Administrators will need to have a strong computer skill base, as they will be in charge of adding and modifying Employee information and view the logs created by the program.

4. Design Protocol

The "Inventory Tyrant" covers three aspects of the Computer Science field. The two main areas are windows based user front end powered by the Dot Net language, and a Sql Server powered backend. The "Inventory Tyrant" will also require some hardware setup for the barcode scanner, receipt printer and cash drawer. My design will be broken down into the three layers of enterprise development, which means there will be a core layer, a business logic layer, and the presentation layer.

4.1 Dot Net Framework

The "Inventory Tyrant" will be primarily developed using Visual Basic .Net developed using Visual Studio Dot Net 2005. I will be using the beta 2 version of the Visual Studio 2005, which means I will also be running version 2 of the Dot Net Framework. For the presentation layer of the project, I will be using windows forms to create a easy to use interface that is both highly functional and easy to use. The Business logic portion of this code will be made up of windows services, web services, and class libraries. The web services will have the ability to post information to a web site's database, allowing for a real time inventory view of the stock of the store.

4.2 SQL Server

The Core layer will be a normalized Microsoft SQL Server Express database engine. SQL Server Express is the next generation of the Microsoft Data Engine (MSDE), and utilizes the Dot Net Framework to create a more efficient database structure. SQL Server Express is a free download that I can freely distribute without cost. It is also in Beta 2 development, and will require version 2 of the Dot Net Framework. Because of the fact that this is already installed on the test computer, no extra work is required to prepare the computer for SQL Server Express.

4.3 Hardware

While a minor step in the overall project, making sure the additional hardware runs properly is extremely important to the project. Because this program will be replacing the current out of date cash register, the bar code scanner, receipt printer, and cash drawer must be completely working and in good order before I can roll out the final product.

4.4 User Interface Design

Inventory Tyrant will be divided into three separate sub applications. Each sub application will have a specific predefined role to fulfill, and each of the sub applications will be able to be run independently of each other. The first of the sub applications will be the "Transaction Controller", which will handle the functionality of the Point of Sale. The second of the two applications will be the "Inventory Manager" which will handle the reporting, viewing, development of reports, and maintenance of the data. The third and final sub application will be the "Administration Console." This sub application will handle the creation of new users and other high level administration and security. While all three of these programs are independent of each other, they will share the same core layer and business layer. Currently the plan is to maintain the overall look and feel for the system. All fonts, sizes, and colors will remain the same throughout the system. The buttons will be the same size and the screens will all look the same. Below is some screen shots of what the system will look like.

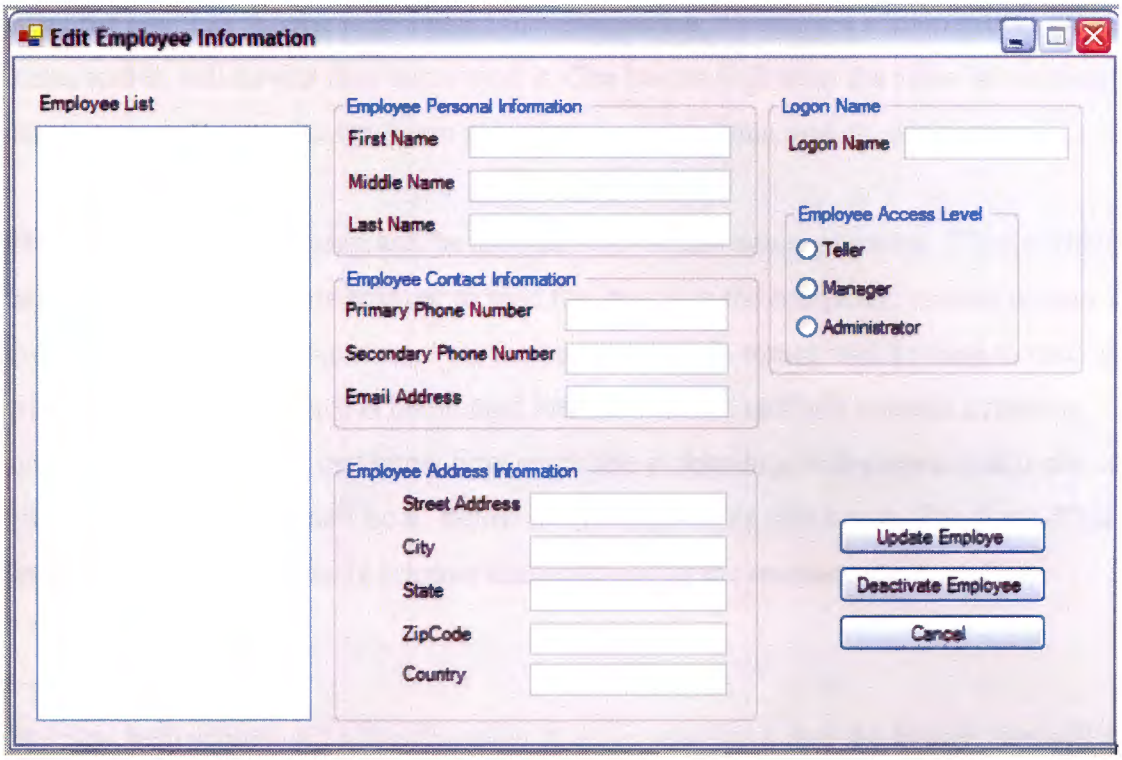


Figure 1: Screen Shot of User Interface.

4.4.1 Transaction Controller

The main function of the Transaction Controller is to ring up a sale and allow new items to be entered into inventory. This application will need the bar code scanner, cash drawer and receipt printer. Here the teller will log into the program, and will be able to add new stock and process sales. This application will contain four different screens: a Logon screen; a main menu screen; a sale transaction screen; and a screen that will allow the employee (if he or she has the proper access) to add new inventory items.

The Logon screen will be a generic screen that most people are use too in their everyday lives. There will be a user name field and a password field, and once the data is entered, will check with the database to verify the user's information and will then look up the rights of the user.

After the user has successfully logged into the system, he will see a main menu. This menu screen will have a four buttons on it. One button will open the sales transaction screen; one will open the new item screen; a log-out button; and an exit button.

The sales transaction screen will be the mainstay of this sub application. This is where the teller will use the barcode scanner to read the item into the computer, receive money from the customer and print the receipt for the customer. This screen will be easy to read and will display each item as it is being read into the system, and will contain a running subtotal so that the user can know how much she is spending in the store, and if she can afford to add another small item. Below is an image of the sale transaction form. This screen will also have a daily printout that contains all the transactions the tellers completed.

The new item screen will allow the teller to enter new stock into the system. He will scan the item with the barcode scanner and will submit the information to the business layer, which will then process the information and push the data into the proper areas of the core layer.

4.4.2 Inventory Manager

The Inventory Manager is the portion of "Inventory Tyrant" where the managers can gain access to the data and view highly important information concerning data trends, marketing figures, and other retail information. This section will also house the reports that the manager can use to show all sorts of data. Here, the manager will also be able to edit data if something happened or the teller did not enter something correctly. This sub application will contain screens that will allow the Manager to edit data, print reports, export data to Excel, and compute marketing figures. This sub application will also reuse the Logon screen from the Transaction Controller.

The main part of this sub application will be the reports section. In the future I would like to base my reports using XML so that the managers can configure reports to their own liking, but currently the Reports run through Crystal Reports. These reports will hopefully be able to allow the managers to print any type of information necessary for their records. It will contain some default reports, like an inventory report, a monthly sales report, and vital tax information.

The next main form will be the Export to Excel form. This will allow the managers to export data so they can test out scenarios without harming the live data. This screen will allow the manager to export almost any information from the program into an excel spreadsheet.

While the goal of this project is to automate the transactions as much as possible, there will be times where the teller will have to manually enter data. That is why there is an edit screen for the managers. This way, if something happens and the manager needs to make changes to the data, he will have that ability. In order to keep a history of changes, when the manager commits a change to the data, that information will be logged. This way the company has a record of transferred data just in case something happens and a manager wants to fudge number.

The final option for the manager is the marketing data screen. This screen will allow the manager to view trends and information based on the transactions. The manager will be able to learn what products are selling the best, which items are not selling, and how much money the company is making verses spending.

Finally, the Inventory Manger will allow the mangers to create reports. If time allows, these reports will be completely created by the manager, and will be XML based so that any computer can view the reports. While this feature is not completely developed, I hope to add this as a major selling point of the project.

4.4.3 Administration Console

The Administration Console will allow the administrators of the system to not only maintain user information, but will also allow for the administrator to view the logs so that he can view information about data. This console will also contain various administration tools and will help the administrator to keep "Inventory Tyrant" running at full power. This application will also use the logon screen created in the "Transaction Controller" sub application.

The log viewer will look a lot like the event viewer provided by Microsoft. This viewer will contain all the information concerning data being edited and finding out anything odd with the system. This viewer will allow for the administrator to play the role of big brother and monitor the data.

The other section of the Administration Console will allow the Admin to maintain the high level parts of "Inventory Tyrant." Here is where the administrator will be able to create, edit, and delete a user, and will be able to change passwords and other various tasks as needed.

4.5 Program Diagrams

4.5.1 Class Diagrams

Below is my class diagram. Shown there is each class along with each of the attributes and methods of each of the classes.

Class Diagram

Brian Seabolt, Senior Design II

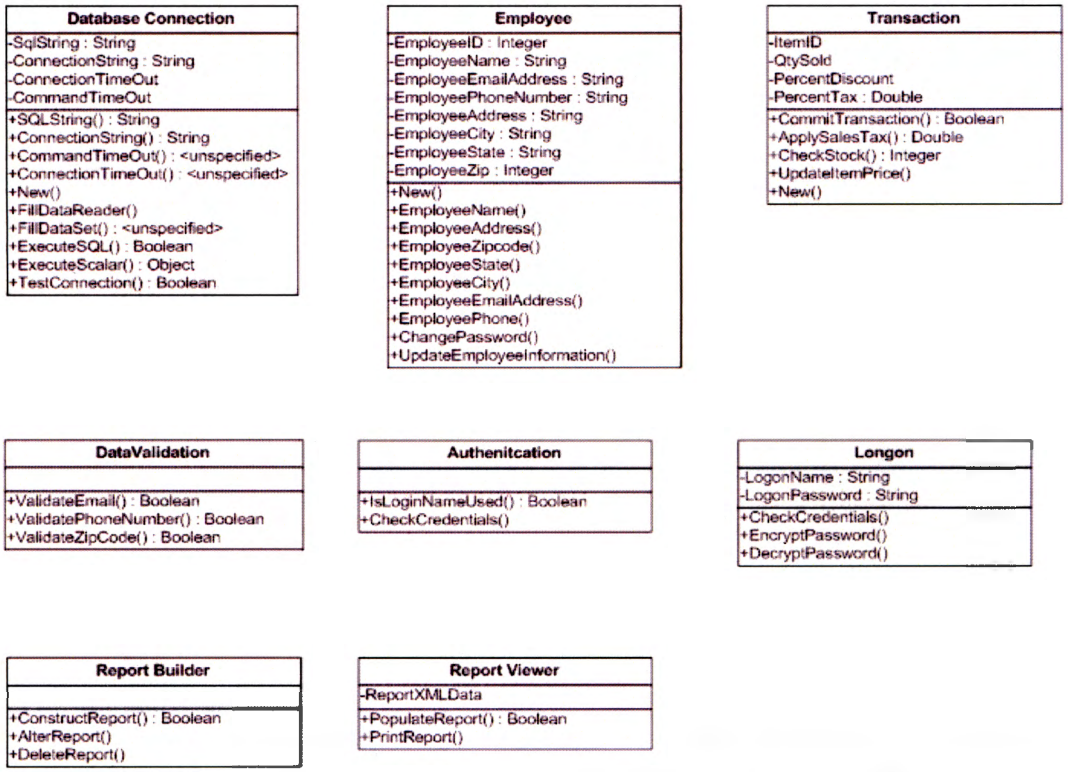


Figure 2: Class Diagram, showing the UML for the classes used in the Complete Inventory Tyrant

4.5.2 Database Diagrams

Below is the database diagram. Each table has a primary key, and the database is normalized to the third level. The bold columns in the tables are required columns. PK stands for Primary Key and FK stands for Foreign Key.

Database Diagram

Brian Seabolt, Senior Design II

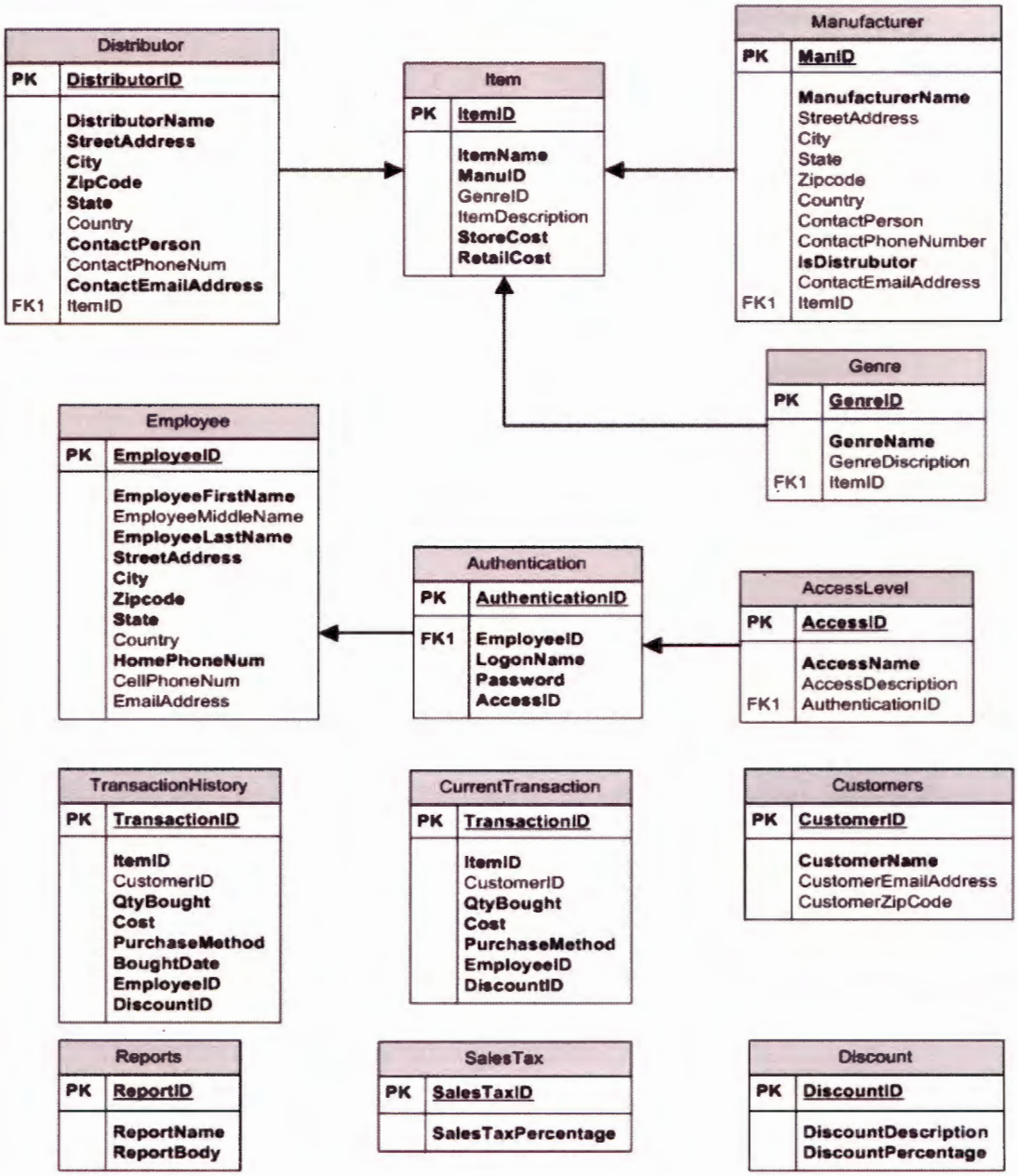
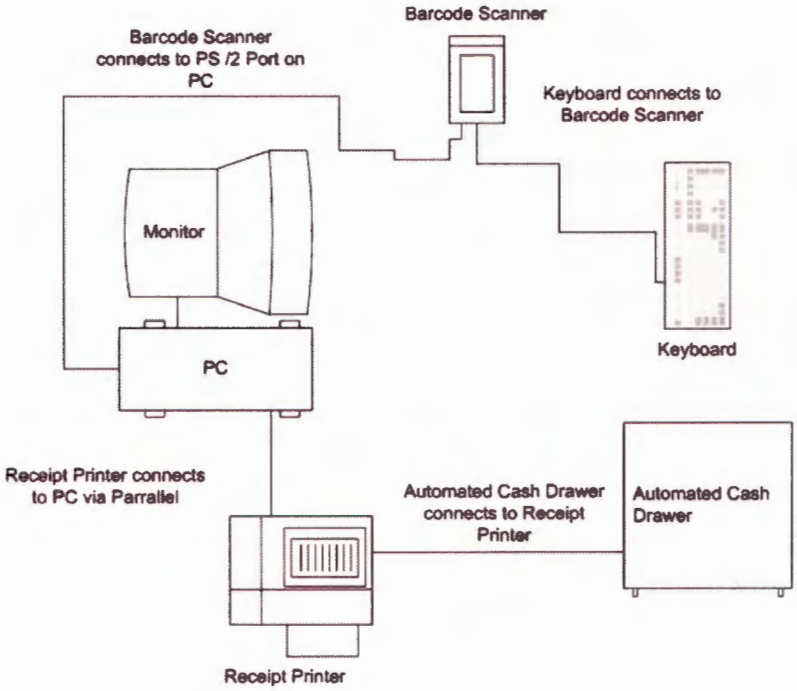


Figure 3: Table Structure of the Complete Inventory Tyrant

4.5.3 System Diagram

Below is a simple System Diagram. This diagram shows how actual hardware and how the hardware connects together.



Hardware Diagram **Brian Seabolt, Senior Design II**

Figure 4: Hardware Diagram for the Complete Inventory Tyrant

4.6 Deliverables

These deliverables are considered the core of the project. These are needed in order for the project to be considered complete.

- Normalized Sql Server 2005 DB Backend
- Windows interface powered by the Dot Net Framework version 2
- Program based security and authentication
- The ability to complete a sales transaction
- Print Receipt for Transaction and end of day report
- Allow users to add a new stock item
- Allow Managers to view sales data
- Support Addition of new companies, new contacts, and new item types
- Have the ability to track Transactions by Employees
- Have the ability to track customer purchases
- Have the ability to print real time up to the minute inventory information
- Allow Employees to be added and modified

5. Design and Development

This section includes the project time line, overall budget, and cost for materials such as books and learning.

5.1 Timeline

The Timeline section is divided into three subsections, one section for each of the three Senior Design sections.

5.1.1 Senior Design I

Senior Design I was primarily focused in research and the beginning stages of requirement gathering.

During the course of Senior Design I, these major milestones were met:

- Research Point of Sale system as well as Inventory monitoring systems
- Interviewed the managers and owners of ACME Games for requirements
- Monitored transactions and business processes of the retail industry
- Develop the basic models and diagrams for the project
- Acquire the needed software for development
- Created the required material for the Senior Design I course

The major portion of Senior Design I course was spent conducting research. I spent time searching the internet and testing demo Point of Sale systems. I also researched the hardware needed for completing the actual point of sale transaction. I looked into Barcode scanners, thermal receipt printers, and automated cash drawer. Some time was also spent researching Sql Server 2005 and Visual Studio 2005 Beta. I spent some time talking to the various managers and owners of ACME Games to find out what would be ideal for a Hobby industry Point of Sale system.

5.1.2 Senior Design II

The vast majority of Senior Design II was spent implementing the initial processes of the program.

During the course of Senior Design II, these major landmarks were met:

- Development and rollout of Sql Server 2005 based database
- Creation of the back end classes for the project
- Developed the Administration Console
- Began development of the Management Tools suite
- Creation of the necessary materials of the Senior Design II course

The vast majority of Senior Design II was spent focusing on the less glamorous backend development. The majority of the time was spent creating and testing the database portion and the interfaces between the database and the front end. I also began purchasing the hardware required for the completion of Transaction Controller.

5.1.3 Senior Design III

Senior Design III was spent completing the code and implementing the testing plan stated in the Design Freeze in Senior Design II.

During the course of Senior Design III, the major landmarks were met:

- Compete the coding to match the deliverables stated in the design freeze
- Complete the testing plan as stated in the Design Freeze
- Modified the code as required by the Managers of ACME Games
- Completed the needed documents for Senior Design III
- Presented my final product to the faculty.

Senior Design III was used to complete the remainder of the work required to graduate from the College of Applied Science.

Below is a basic timeline I followed for my Senior Design Sequence.

Task Name	Duration	Start Date	Finish Date
Senior Design I	**47 days**	**11/2/2004 8:00**	**1/5/2005 17:00**
Progress Report 1	13 days	11/2/2004 8:00	11/18/2004 17:00
Progress Report 2	14 days	11/22/2004 8:00	12/9/2004 17:00
Problem Statement	14 days	11/2/2004 8:00	11/19/2004 17:00
General Research	47 days	11/2/2004 8:00	1/5/2005 17:00
Proposal	18 days	12/10/2004 8:00	1/4/2005 17:00
Present Proposal	1 day	1/5/2005 8:00	1/5/2005 17:00
Senior Design II	**47 days**	**1/6/2005 8:00**	**3/11/2005 17:00**
Barcode Scanner Research	14 days	1/6/2005 8:00	1/25/2005 17:00
Receipt Printing Research	14 days	1/6/2005 8:00	1/25/2005 17:00
Sale Transaction Research	7 days	1/6/2005 8:00	1/14/2005 17:00
Database Constuction	14 days	1/17/2005 8:00	2/3/2005 17:00
VB .NET GUI Development	14 days	1/17/2005 8:00	2/3/2005 17:00
Complete Prototype	10 days	2/4/2005 8:00	2/17/2005 17:00
Construct Design Freeze Document	15 days	2/18/2005 8:00	3/10/2005 17:00
Present Design Freeze	1 day	3/11/2005 8:00	3/11/2005 17:00
Senior Design III	**63 days**	**3/14/2005 8:00**	**6/8/2005 17:00**
Add features to Phototype	21 days	3/14/2005 8:00	4/11/2005 17:00
Testing	20 days	4/12/2005 8:00	5/9/2005 17:00
Roll out Product to Company	7 days	5/10/2005 8:00	5/18/2005 17:00
Final Presentation	15 days	5/19/2005 8:00	6/8/2005 17:00

Figure 5: Simplified Timeline for development of Complete Inventory Tyrant

5.2 Budget

Below is a budget breakdown for the Complete Inventory Tyrant. Most of the items required for the project were free. Visual Studios Dot Net 2005 and Sql Server 2005 were free copies and most of the development was done on my personal laptop. The only major cost of this project was the hardware needed for the Point of Sale system.

Senior Design Project Budget		
Item	**Street Price**	**Actual Price**
Dell Dimension 3000 Desktop PC w/ Keyboard and Mouse	$600.00	$0.00
Windows XP Professional	300.00	0.00
Wasp Barcode Scanner	300.00	150.00
Star Thermal Receipt Printer	190.00	175.00
APG X100 Cash Drawer	250.00	0.00
SQL Server Express Beta 2	0.00	0.00
Visual Studio .Net 2005 Enterprise Beta 2	$2500.00	$0.00
Total	**$4140.00**	**$325.00**

6. Proof of Design

This section of the paper will prove that I completed each of my deliverables as required of my through the Senior Design course requirement.

6.1 Normalized SQL Server 2005 DB Backend

The database backend powering the Inventory Tyrant is Sql Server 2005 Beta. The database is normalized to the third degree, with the exception of the Transaction and Transaction history table. These tables were designed with repetition in mind. The Idea was that be having two tables to store transaction would allow for better flow of data, without the risk of data being lost.

Figure six is a screen shot of SQL Server Management Studio, the application that is included in the SQL Server 2005 package. This Software is supposed to replace Enterprise Manager found in the SQL Server 2000 package.

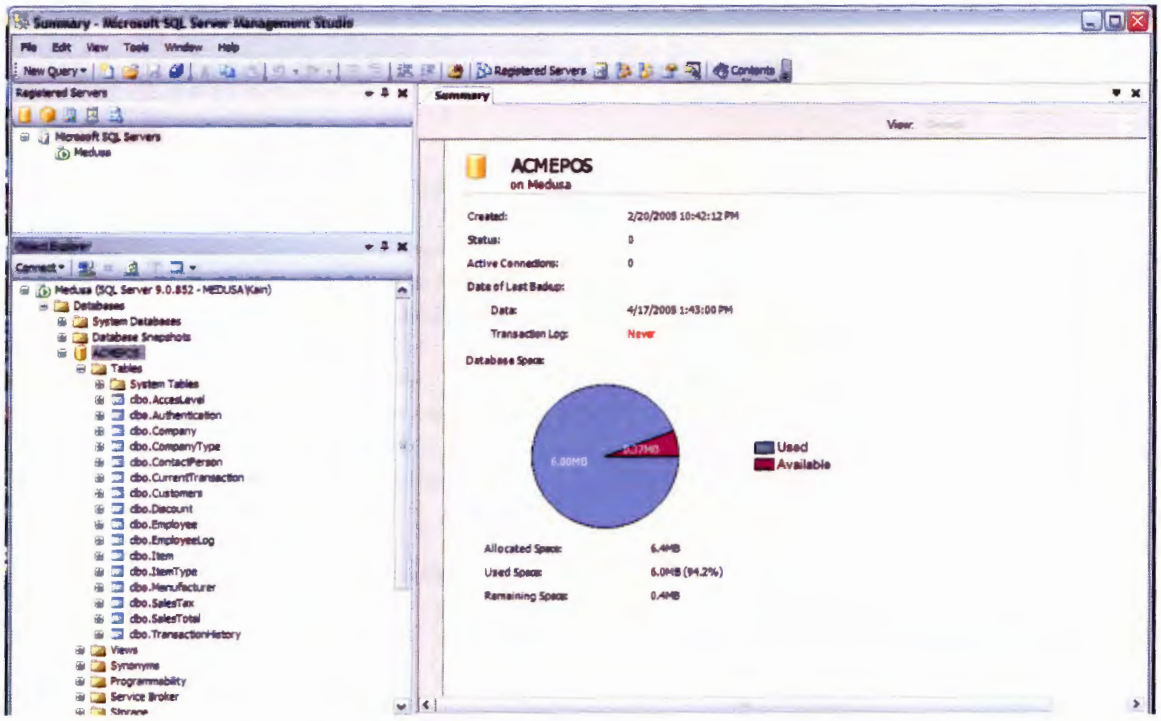


Figure 6: Screenshot of Server Management Studio and the tables of the Complete Inventory Tyrant

The tables used to store the information for the Complete Inventory Tyrant is seen on the left of the screen shot. As you can see, the new Server Management Studio has a different look and a different feel than the Enterprise Manager of old. If you look towards Appendix A, you will see the custom built Sql Connection class that connects the database to the user interface.

6.2 Windows interface powered by the Dot Net Framework version 2

The Complete Inventory Tyrant was built using the new Visual Studio Dot Net 2005 Beta. This Beta Version runs using the Dot Net Framework 2.0 Beta, and is required to be installed in order for software to be developed using Visual Studio 2005.

Figures seven and eight shows that both the Framework 2 Beta and Visual Studios *2005 beta is installed on the machine used to develop The Complete Inventory Tyrant.*

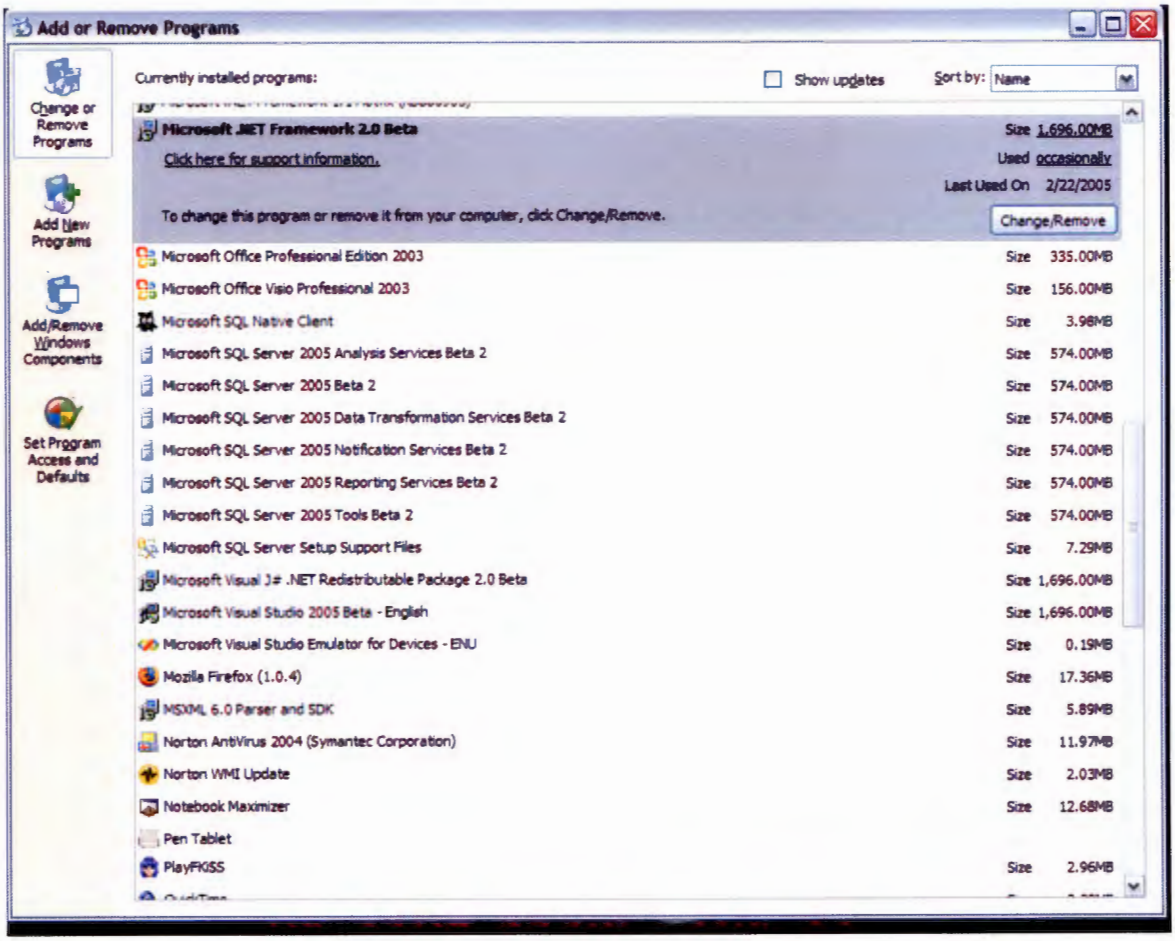


Figure 7: Showing the Framework 2 Beta is installed on the development machine

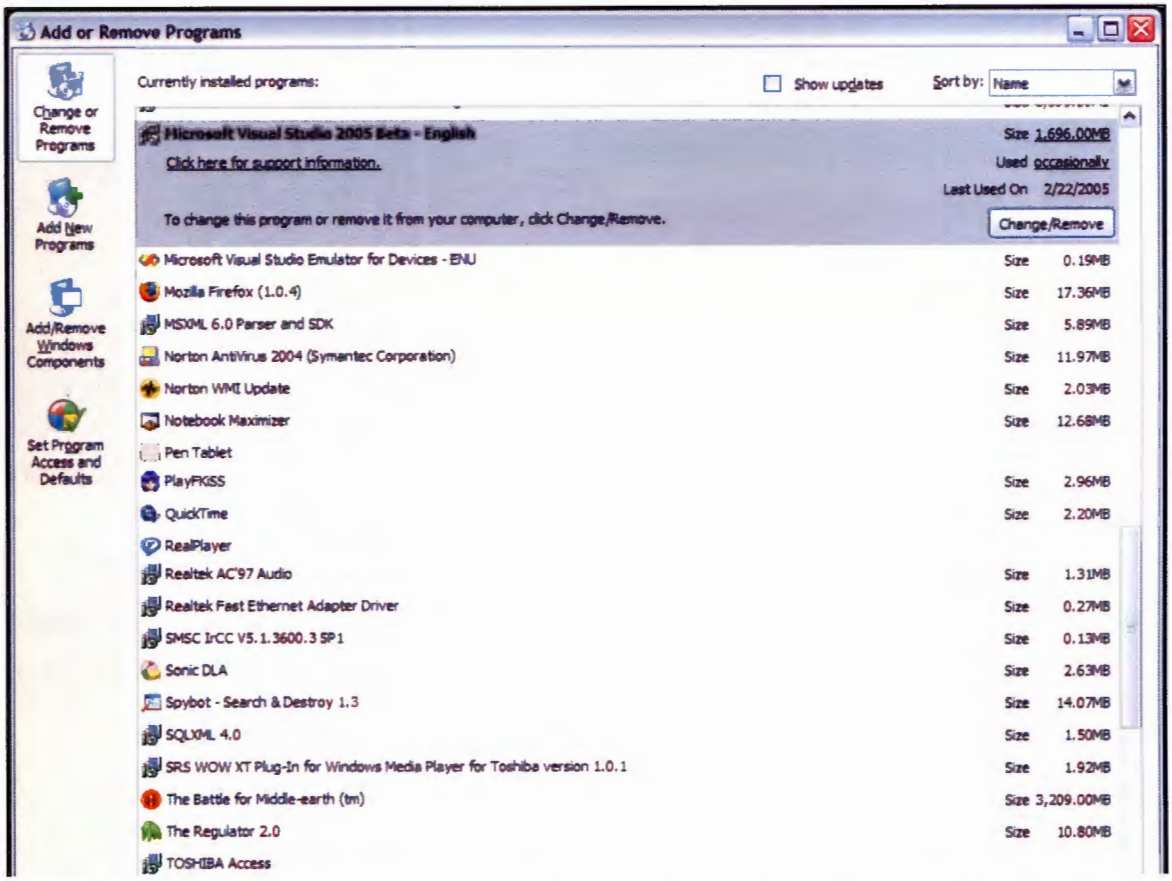


Figure 8: Showing that Visual Studio 2005 Beta is installed on the development machine

The Visual Studio package was the software development environment used to create the Complete Inventory Tyrant. The Framework is the middleware that translates the Dot Net code into what the computer can understand, binary.

6.3 Program based security and authentication

All three aspects of the Complete Inventory Tyrant uses Program based security. The logon information is stored in the database and is required to log into the system. The Security Class can be found in Appendix B.

Figure nine located on the following page shows the universal logon screen used throughout the three aspects of the Complete Inventory Tyrant. These logon screens have the same user-friendly feel to them as most logon screens used in today's popular applications.

Figure 9: The standard logon screen of the Complete Inventory Tyrant

The user types in his or her user name into the box under User name and then her password in the Password box. The user name is not case sensitive, but the password has to be to maximize the security of the program.

6.4 The ability to complete a sales transaction

The ability to complete a sales transaction is handled in the Transaction Controller portion of the Complete Inventory. The ability to complete a sales transaction section includes the ability to print a receipt and open the cash drawer.

The Transaction Recorder portion of the Inventory Tyrant handles the transaction and sale and pushes the data from the user interface portion to the Sql database backend. The item the customer wants to purchase is scanned in using a barcode scanner. The SKU number is read in by the barcode scanner and by using the Sql connection class shown in Appendix A, the item's information is populated back to the Transaction Recorder. Once all of the sale items have been recorded and populated back to the Transaction Recorder, the Teller has the option to complete the sale. The Teller enters the type of payment, the total amount tendered, any discount information, and the costumer information if the customer is willing. Figure 10 shows the Transaction Recorder Screen.

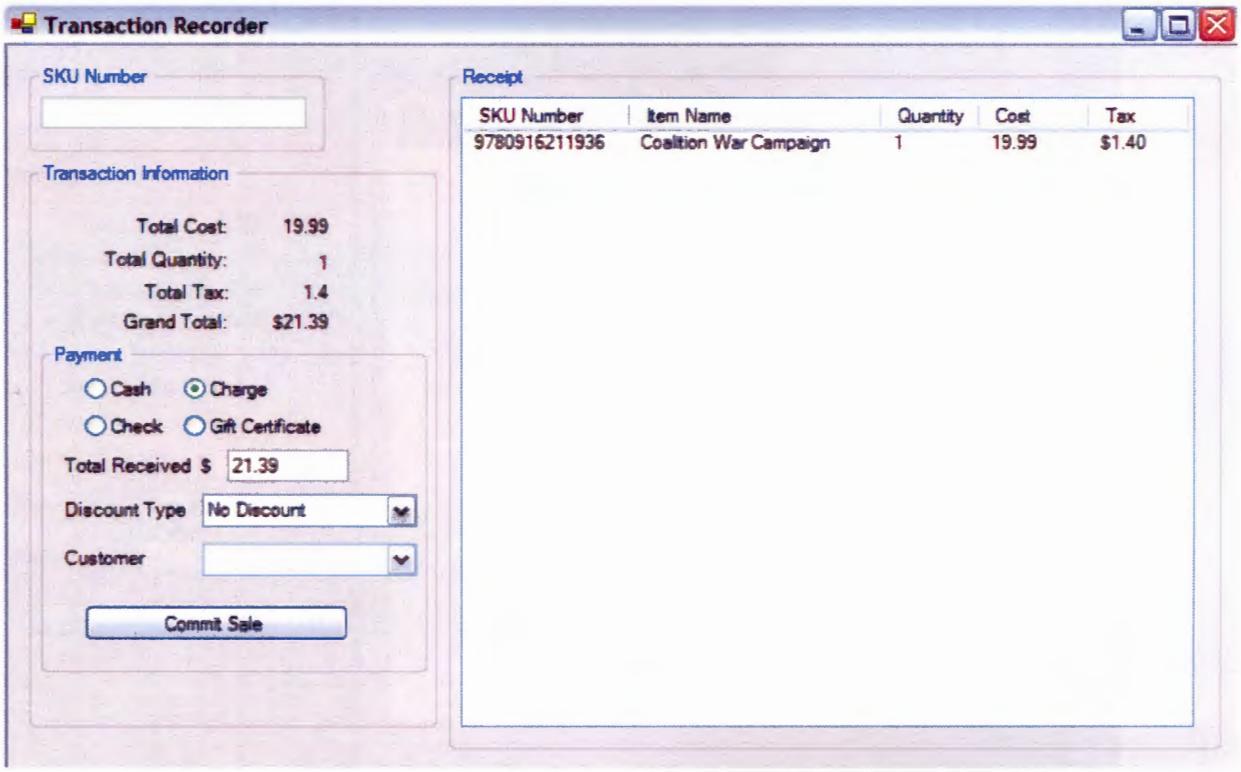


Figure 10: Screen shot of the Transaction Recorder

As you can see, the item is added to the Receipt section of the Recorder. The totals on the left section of the Recorder are updated after each item is added to the Receipt. This allows for a running sum, which helps the customer know how much he or she is spending before committing the sale.

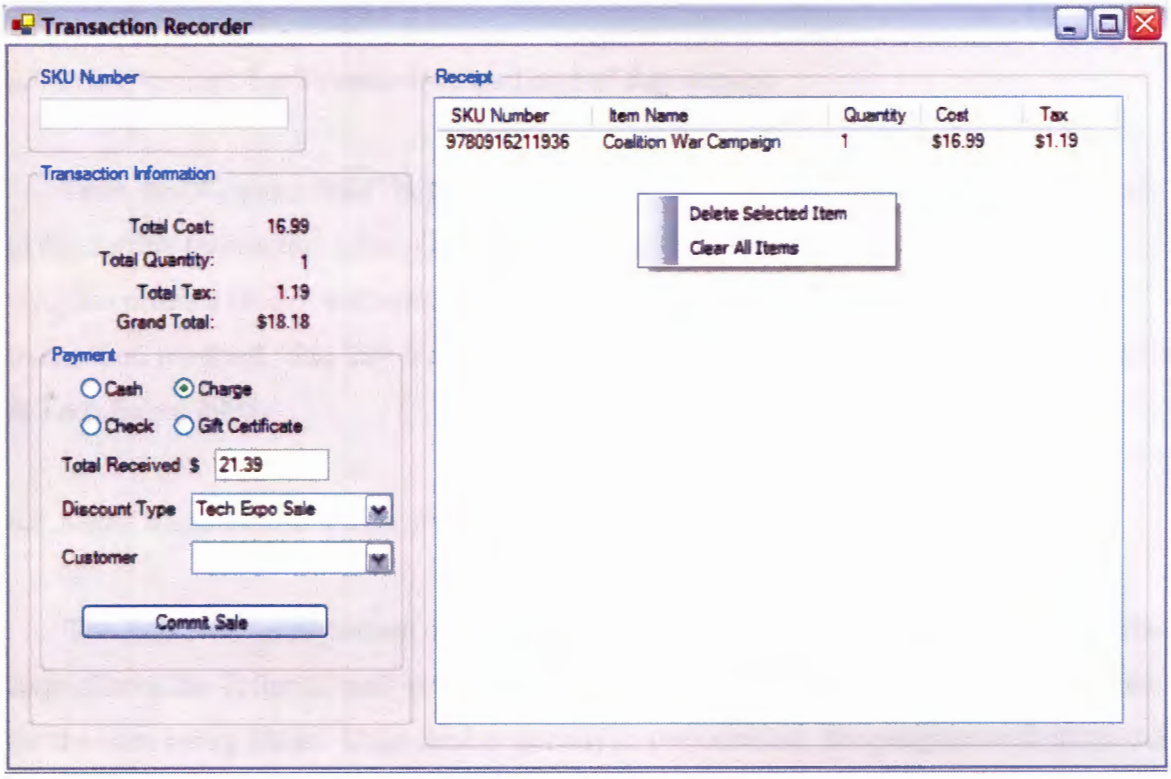


Figure 11: More Functionality of the Transaction Recorder

Figure eleven shows some of the other features of the Transaction Recorder. Figure ten showed the normal processes and standard costing. Figure eleven shows the discount applied to the item (The Tech Expo discount is fifteen percent.) Figure eleven also shows the context menu that appears when the Teller right clicks on an item in the Receipt section of the Recorder. The context menu allows for items to be removed either singly or completely. After which the total information is updated to handle items being removed.

The customer information section just above the "Commit Sale" button is a list of all the customers' email addresses. This allows for sales to be tracked to the customers for better market research. If the "New Customer" selection is chosen, then a small form opens to allow a new customer to be entered into the system. The Customer information is not required, so the customer does not feel like he or she has to give the company his personal information.

6.5 Print Receipt for Transaction and end of day report

Once the "Commit Sale" button is clicked by the Teller, the information is recorded to the current transaction table. Once the information is inserted into the table, the program prints a receipt and opens the cash drawer to allow the final portion of the transaction reported. This key feature was demonstrated to the faculty and fellow students at Tech Expo, 2005.

6.6 Allow users to add a new stock item

The Add New Item portion of the Inventory Tyrant can be seen in figure twelve. This form allows the Teller to read in the item's SKU number as well as the other information for the item being added. If the item is already in the database, the program will fill in the item's information from the database.

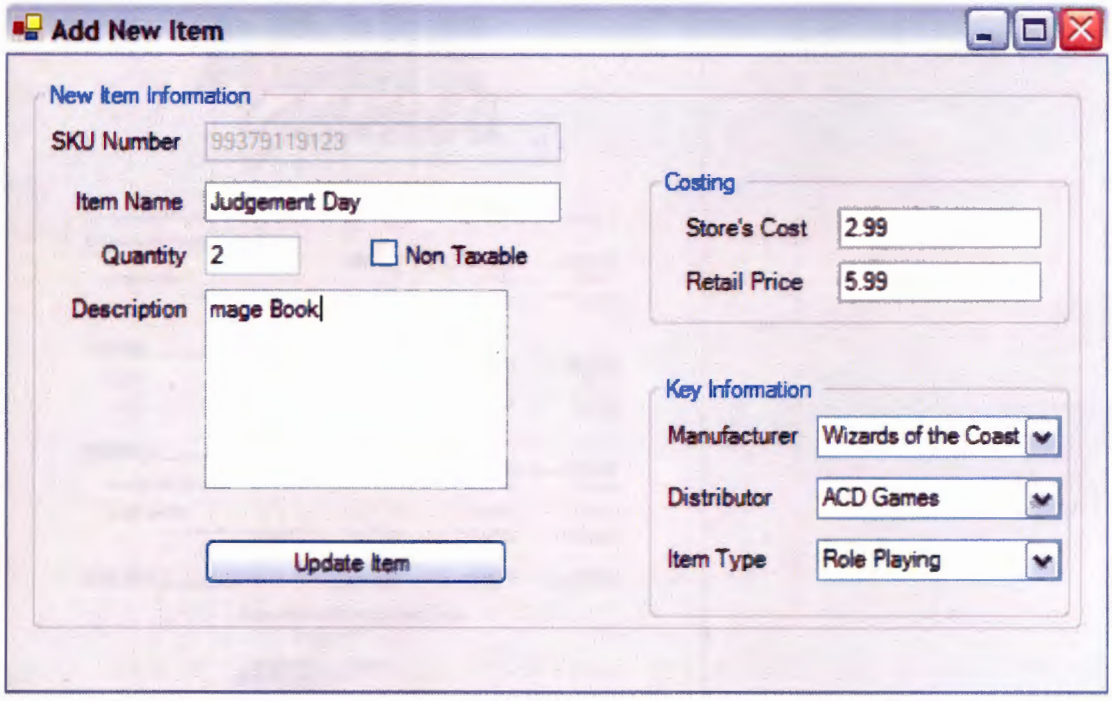


Figure 12: The Add New Item portion of the Inventory Tyrant

The Key information portion of the Add New Item form allows items to be linked to certain companies and item types. This allows for the managers to sort items apart, allowing for more accurate sales analysis.

6.7 Allow Managers to view sales data

The Management Console allows for the Managers to print out sales reports so they can better understand the market and the items they have sold. The Sales Data Report shows the items sold in the past thirty days. This report breaks down the items sold by item type, which is also controlled in the Management section of the Complete Inventory Tyrant. Figure thirteen shows a screenshot of the report that the manger would print out using this functionality. To better help the managers understand how the money is being gained; a dynamic pie graph was added to the bottom of the report. The ACME Games logo can be easily replaced with any company's logo.

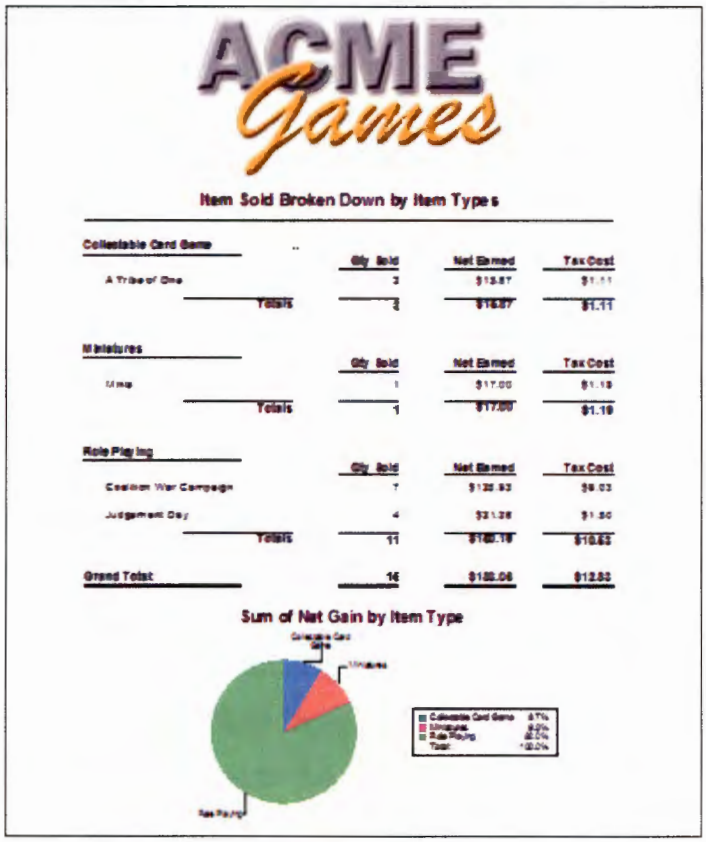


Figure 13: Sales Data Report

6.8 Support Addition of new companies, new contacts, and new item types; which are added by the Manager

Figure fourteen shows the Add New Company form found in the Management Console. All of the Company's information is required, and the data validation code used to make sure all of the information is properly entered into the database is correct can be found in Appendix C.

When adding a new company, the program requires you to add a Primary Contact Person. When I interviewed the managers at ACME Games, they all said that each company has a primary contact person, thus requiring a contact person when a new company is added to the program.

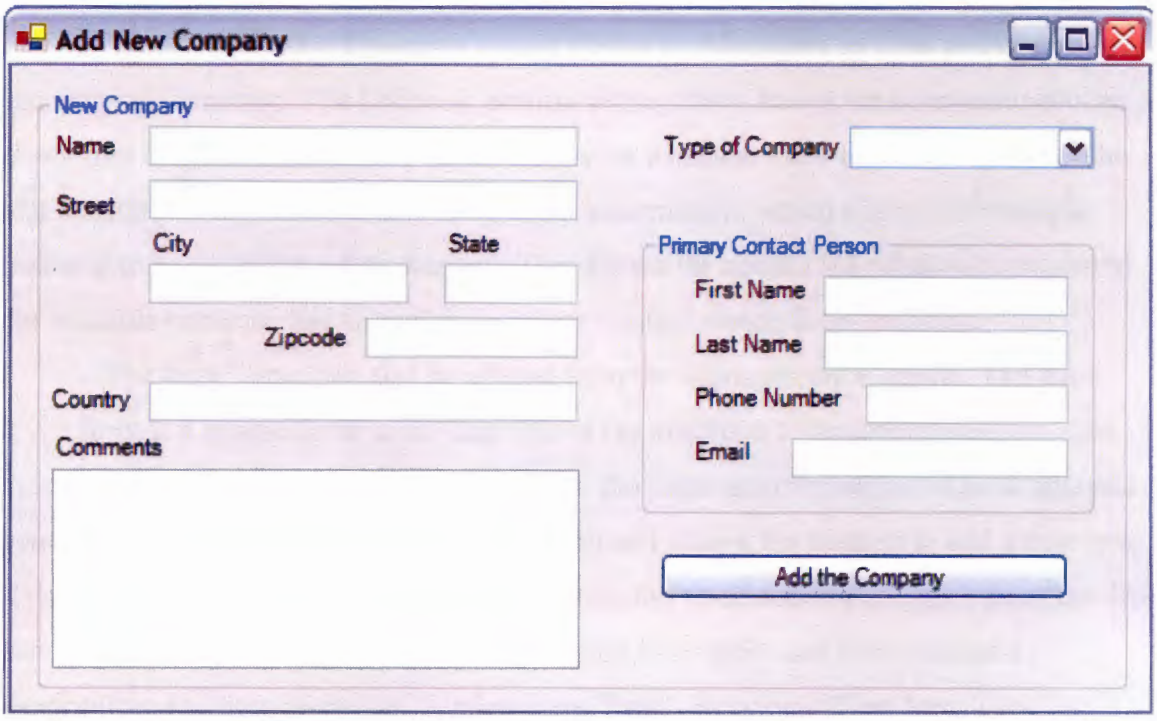


Figure 14: The Add New Company Section of Complete Inventory Tyrant

Once the company has been added, it can be modified in the Update Company screen found in the Management Console. This section also allows for additional contact people to be added to the company's list of contacts. Figure fifteen shows the Update Company Screen.

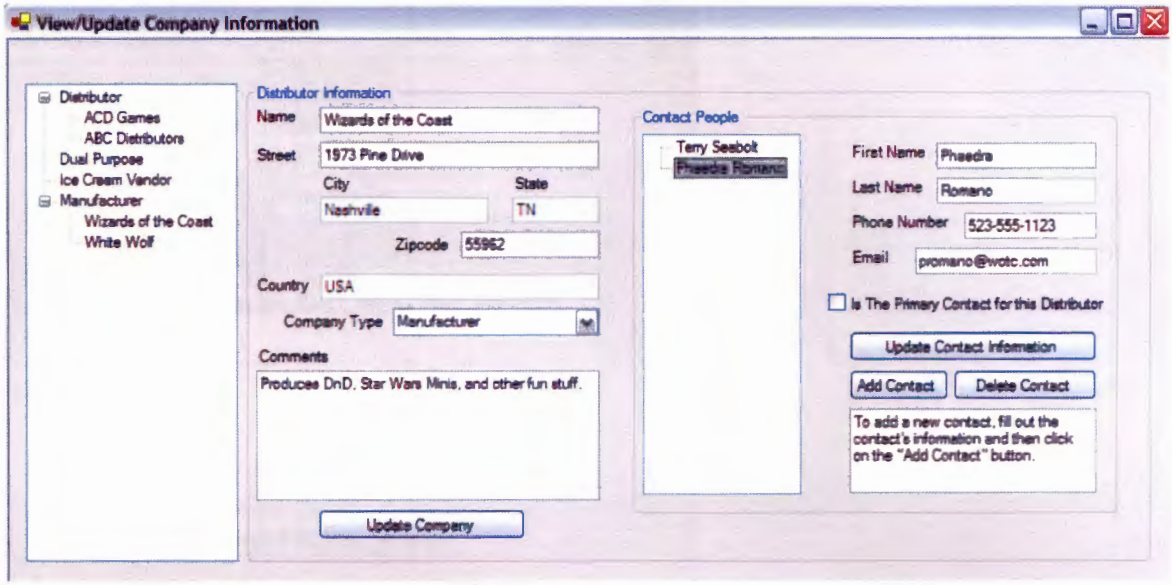


Figure 15: The Update Company information screen

The View/Update Company screen allows for Managers to view and update company information. The Leftmost portion of the screen shows the companies broken down into their company types, which allows for a cleaner view of the companies. The right portion of the screen shows the contact information, which allows for multiple contacts to be listed for each company. This allows for contact information to be stored for multiple contacts, just in case the primary contact person is on vacation.

The Item Types can also be altered from the Management Console. The Item Type form is a simple form to use that allows the managers to update or add new item types. The item types are used to control how the items are divided in the sales analysis report. The Item Type allows for a description and allows the Manger to add a new Item Types to the database, They do this by selecting the "Add New Item Type" from the drop down selection. At that point, the manager types in a name, and if they chose a description, and then clicks the "Update Item Type", but when "New Item Type" is selected, the button says "Add New Item." Figure sixteen shows the Item Type form.

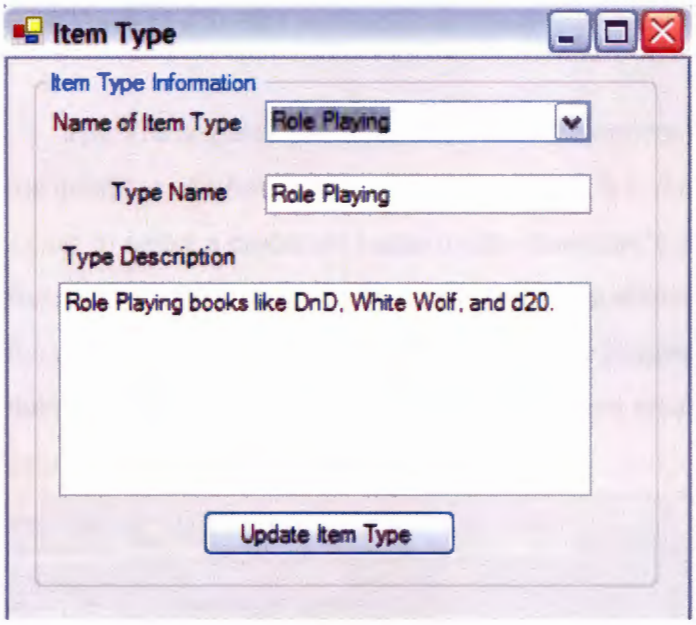


Figure 16: The Item Type Form

6.9 Have the ability to track Transactions by Employees

Each Employee is assigned an employee number when he or she is added to the program. This number, or Employee ID, is stored by the program when a user logs into the system. Once the sale is final and the employee clicks the "Commit Sale" button (shown in figure ten and eleven) her employee ID is recorded along with the sale. As seen in Figure seventeen, the Employee's ID (in this case, 1 is my ID) is recorded for each of the transaction.

	TransactionID	SKU	QtyBought	Cost	Tax	TotalCost	TransactionDate	CustomerID	EmployeeID	PurchaseMethod
1	1	46363004958	1	5.29	0.37	5.66	2005-05-18 01:1...	0	1	Check
2	2	9780916211936	1	19.99	1.40	21.39	2005-05-18 01:1...	0	1	Check
3	3	46363004958	1	5.29	0.37	5.66	2005-05-18 01:1...	0	1	Charge
4	4	46363004958	1	5.29	0.37	5.66	2005-05-18 01:1...	0	1	Charge
5	5	9780916211936	1	19.99	1.40	21.39	2005-05-18 01:1...	0	1	Charge
6	6	9780916211936	1	19.99	1.40	21.39	2005-05-18 21:0...	0	1	Charge
7	7	9780916211936	1	17.99	1.26	19.25	2005-05-20 13:1...	0	1	Charge
8	8	99379119123	1	5.99	0.42	6.41	2005-05-20 13:1...	0	1	Cash
9	9	9780916211936	1	16.99	1.19	18.18	2005-05-20 14:0...	0	1	Charge
10	10	99379119123	1	5.09	0.36	5.45	2005-05-20 14:0...	0	1	Charge
11	11	5011921942930	1	17.00	1.19	18.19	2005-05-20 14:4...	0	1	Charge
12	12	99379119123	1	5.09	0.36	5.45	2005-05-20 14:4...	0	1	Charge
13	13	9780916211936	1	16.99	1.19	18.18	2005-05-20 14:5...	0	1	Charge
14	14	9780916211936	1	16.99	1.19	18.18	2005-05-20 15:2...	0	1	Charge
15	15	99379119123	1	5.09	0.36	5.45	2005-05-20 15:2...	0	1	Charge

Figure 17: Shapshot of the Item Table

6.10 Have the ability to track customer purchases

The Transaction Recorder allows for customers to be linked with sales committed to the database. As briefly mentioned in section 6.4, the customer selection box allows the Teller to select a customer based on the customer's email address. This information is not required to complete the sale. Figure eighteen shows the Add Customer screen. It allows the customer's information to be added to the program. It also has the choice to not have the customer's email be shown. Figure eighteen also shows what happens when the information entered is incorrect.

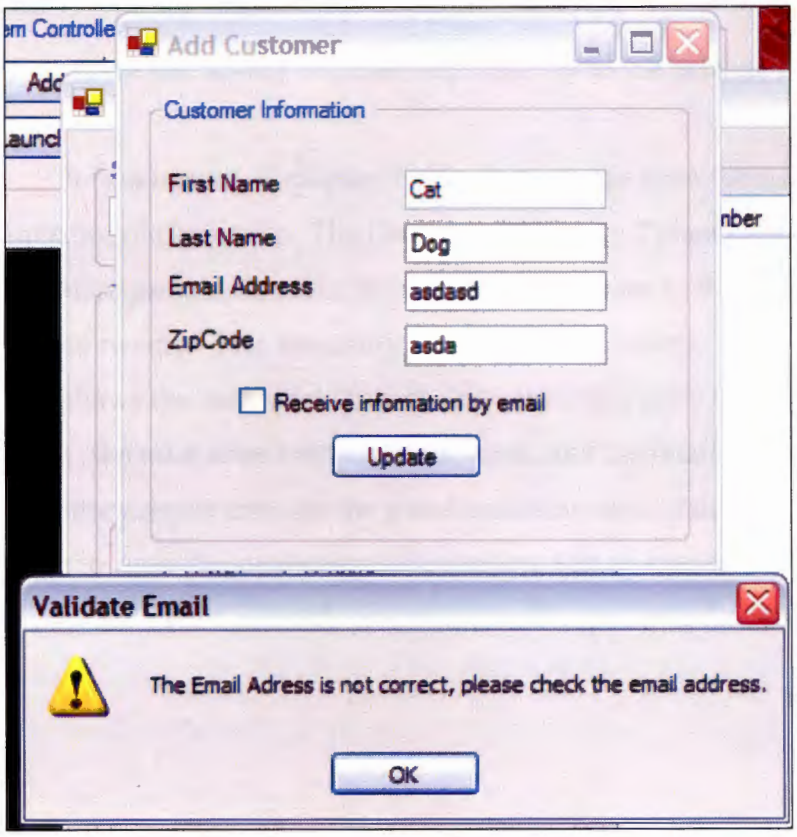


Figure 18: The Add Customer information screen, as well as proof that the Validation works.

Figure nineteen shows the customer information being saved into the database. The last record in the image shows the customer information being saved into the database. Also notice that a different employee id is being stored in the database as a different log in name was used to commit this transaction.

	TransactionID	SKU	QtyBought	Cost	Tax	TotalCost	TransactionDate	CustomerID	EmployeeID
1	1	46363004958	1	5.29	0.37	5.66	2005-05-18 01:1...	0	1
2	2	9780916211936	1	19.99	1.40	21.39	2005-05-18 01:1...	0	1
3	3	46363004958	1	5.29	0.37	5.66	2005-05-18 01:1...	0	1
4	4	46363004958	1	5.29	0.37	5.66	2005-05-18 01:1...	0	1
5	5	9780916211936	1	19.99	1.40	21.39	2005-05-18 01:1...	0	1
6	6	9780916211936	1	19.99	1.40	21.39	2005-05-18 21:0...	0	1
7	7	9780916211936	1	17.99	1.26	19.25	2005-05-20 13:1...	0	1
8	8	99379119123	1	5.99	0.42	6.41	2005-05-20 13:1...	0	1
9	9	9780916211936	1	16.99	1.19	18.18	2005-05-20 14:0...	0	1
10	10	99379119123	1	5.09	0.36	5.45	2005-05-20 14:0...	0	1
11	11	5011921942930	1	17.00	1.19	18.19	2005-05-20 14:4...	0	1
12	12	99379119123	1	5.09	0.36	5.45	2005-05-20 14:4...	0	1
13	13	9780916211936	1	16.99	1.19	18.18	2005-05-20 14:5...	0	1
14	14	9780916211936	1	16.99	1.19	18.18	2005-05-20 15:2...	0	1
15	15	99379119123	1	5.09	0.36	5.45	2005-05-20 15:2...	0	1
16	16	9780916211936	1	19.99	1.40	21.39	2005-06-01 23:5...	0	4
17	17	9780916211936	1	19.99	1.40	21.39	2005-06-02 00:1...	4	4

Figure 19: Snapshot of CustomerID being stored in the Transaction Table

6.11 Have the ability to print real time up to the minute inventory information

It is extremely important for the manager to have access to up to the minute inventory information. The Complete Inventory Tyrant has this functionality built in. In the Management Console, the manager has access to the Inventory report shown in Figure twenty. This inventory report breaks down the stock stored into it's item types, and shows the store cost for each item, the retail price for each item, the total amount in stock, the total store cost for entire stock, and the retail price for the entire stock. The inventory report also has the grand totals for each of the categories. This report has a pie chart to show the percentage of inventory tied up into each of the item types. Only the items that have a quantity greater than zero appear in the Inventory Summary.

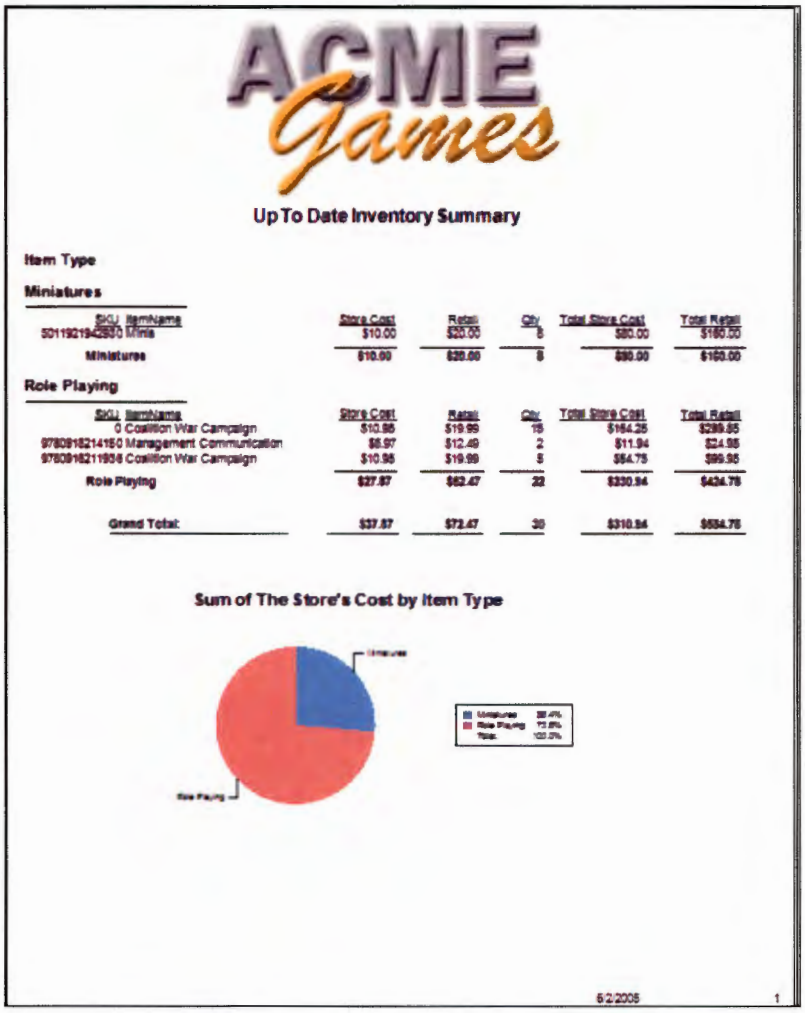


Figure 20: Up to date Inventory Report

6.12 Allow Employees to be added and modified

Employees are added and modified in the Administration section. There are two sections for employees. One section is to add the employees and the other section to update or modify employees. Figure twenty one shows the form to add a new employee. After the Administrator fills out the information in the required fields the Administrator clicks on the Submit button, which submits the information to the database. Once the information is entered into the database the new employee can access the information. The Clear button is used to remove all the information from the fields. As with the other forms, the data validation class found in appendix C is used to make sure the data is correct. The authentication class in appendix B is used to verify that the information is

unique. The Employee access level is used to make sure the employee has access to the proper information and functionality.

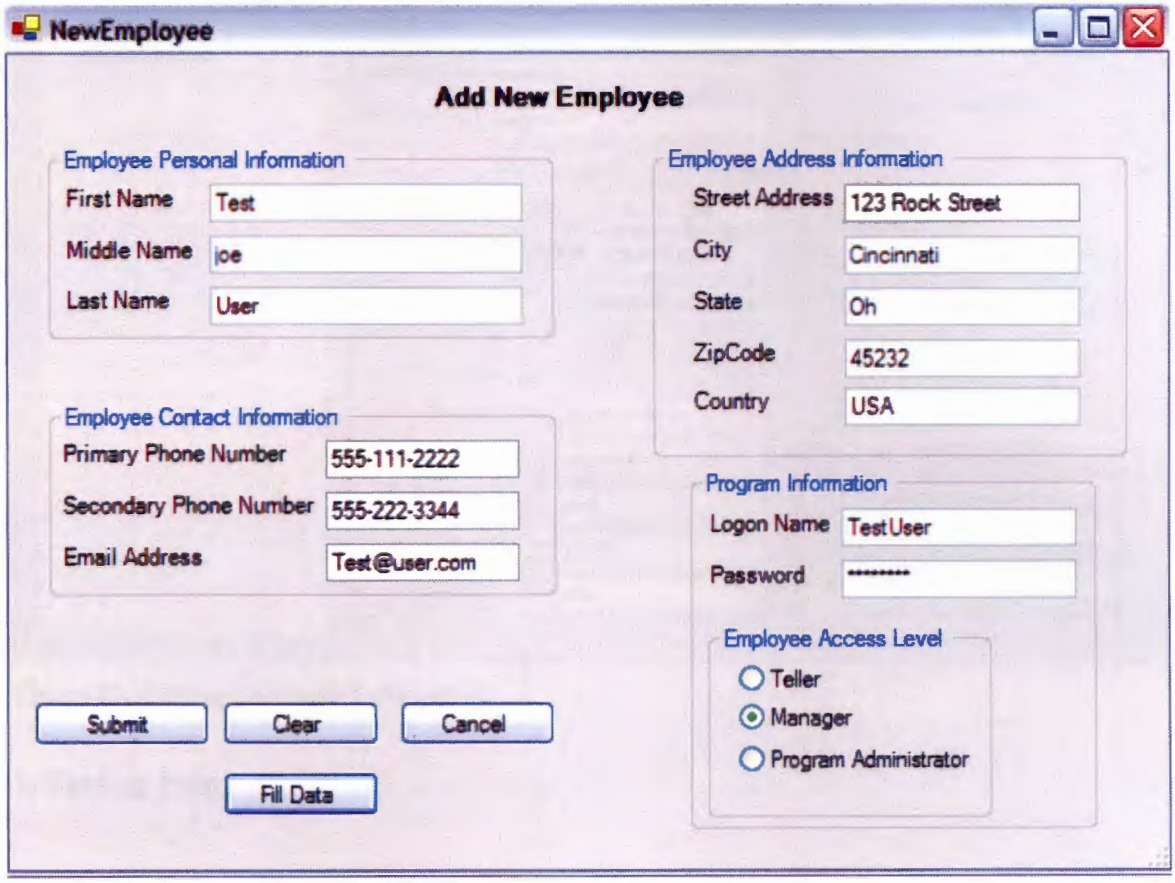


Figure 21: Adding a new employee

The second portion of the employee section is for updating and viewing employee information. Figure twenty two shows this form. The left section of this form shows the list of employees. By default only the active employees are shown. The Complete Inventory Tyrant does not allow employees to be removed from the database. Instead, the employees can be deactivated, which tells the program who can and cannot log into the system. This allows for a history of all the employees that worked for the company to be kept. This form makes use of both data validation found in Appendix C and the authentication found in Appendix B.

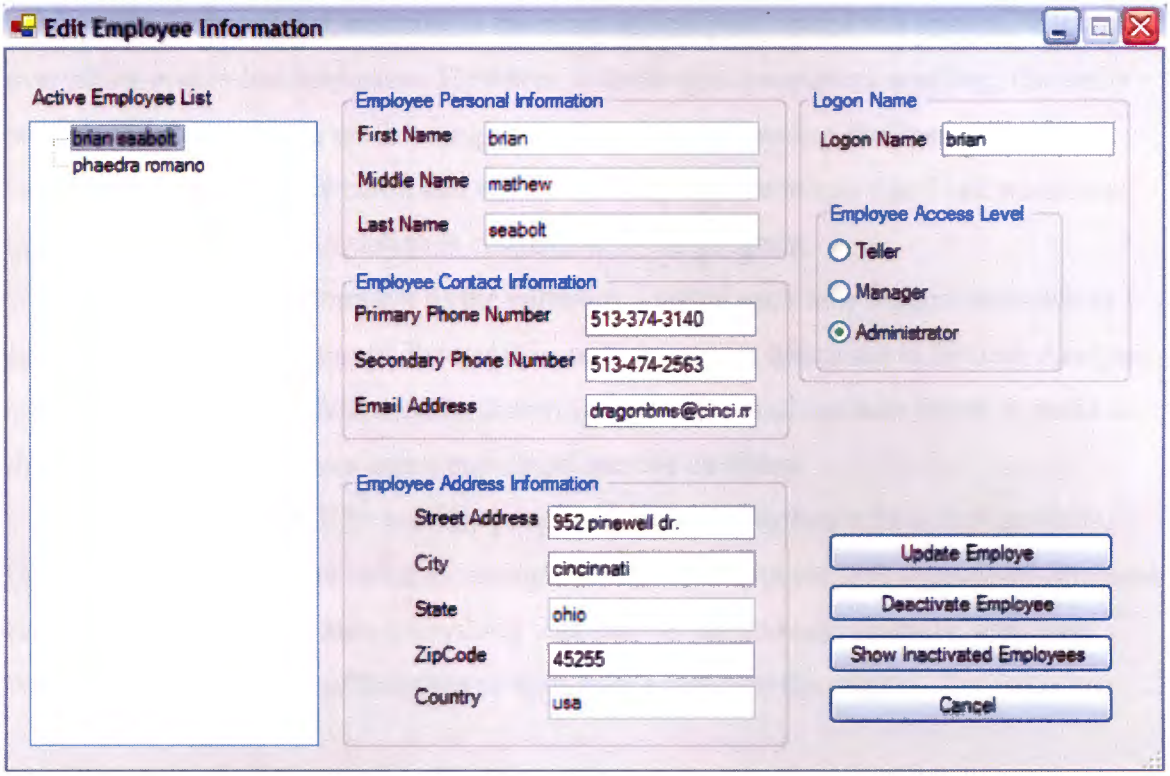


Figure 22: Editing Employee Information

7. Testing Plan

In order to test "Inventory Tyrant" I have offered to donate a copy of this program to a local game store, ACME Games. This roll out will occur in September, 2005. They will be my test bed for this project, so I will be able to test each aspect of the program in iterative steps. First and foremost I plan on rolling out the completed version of the Transaction Controller. That way I will be able to test out the core aspect of the program without having to roll out the other two aspects. After the rollout of the Transaction Controller, the Administration Console will follow, and then finally I will roll out the Inventory Manager.

The Transaction Controller is the most critical partition of this system. If it works, everything else is less important. However, without this component working, the entire product is useless. I plan on focusing a good deal of time testing the Transaction Controller. I will actually be on site when I test this application so that I can watch and take notes when each transaction is committed to the program.

During the development of the software, I tested each new feature and code as I completed them. I ran through the iterative testing process discussed in System Analysis and Design I as well as Software Engineering. I also checked the data tables to make sure the proper information was being populated into the database.

This program will be used by people with various technological backgrounds, so I received input on the user interface design from people with varying degrees of computer skills. I wanted to make sure everything was easy to use and user friendly, and what better way to make sure of that than to have user's input on the project.

8. Conclusions and Recommendations

8.1 Conclusion

This project was originally created to help ACME Games carry its business into the twenty first century. Not only did I provide ACME Games a complete Point of Sale system, but I built a system in which they can monitor their inventory, cash flow which will allow ACME Games, and any Game and Hobby retail store to grow their business and better attack the Market in Cincinnati Ohio. This software was developed using Microsoft Visual Studios 2005 Beta and Sql Server 2005 beta. The deliverables were met and the testing plans went through without issue. Full roll out of the Complete Inventory Tyrant will be deployed in September at ACME Games, and after that, I plan on marketing the software to the general public.

8.2 Recommendations

The biggest thing I can recommend to someone is to make sure they are into the project they are working on. The biggest problem I had with the project is I lost interest with the project for a while. If I was more into the project, I think I could have done something bigger and better with the project. I spent about two hundred hours on the project, and most of that was during the last section of Senior Design II and throughout Senior Design III.

The next thing I can recommend is that you should work on a project that is challenging and fun. I wanted to provide a service to ACME Games, but in the end the project was a little boring for me, and was not all that challenging. The hardest part of the build was getting the transaction information to properly be stored in the database. That and learning the crystal report functionality.

It was exciting to me to get to use the new Dot Net beta as well as the SQL Server 2005 beta. I wish my project was a little more complicated so I could have gotten into the more advanced features of the project, but I at least attempted to do something different, something other than an ASP Dot Net Web Page that connects to an MS-Access database.

Appendix A: Sql Connection Class

Below is the code used by the Inventory Tyrant to connect to the Sql Server 2005 database. Notice how the functions that handle sql interaction are overloaded to handle different methods. Also note that some of the functions contain "Shared" which allows the function to be called with creating the SqlConnection class.

```
Imports System.Data
Imports System.Data.SqlClient

Public Class SQLConnection

#Region "Declarations"
    Private _SqlString As String
    Private _ConnectionString As String
    Private _CommandTimeOut As Data.CommandBehavior
    Private _ConnectionTimeOut As Data.CommandBehavior
#End Region

#Region "Properties"
    Public Property SqlString() As String
        Get
            Return _SqlString
        End Get
        Set(ByVal value As String)
            _SqlString = value
        End Set
    End Property

    Public Property ConnectionString() As String
        Get
            Return _ConnectionString
        End Get
        Set(ByVal value As String)
            _ConnectionString = value
        End Set
    End Property

    Public Property CommandTimeOut() As Data.CommandBehavior
        Get
            Return _CommandTimeOut
        End Get
        Set(ByVal value As Data.CommandBehavior)
            _CommandTimeOut = value
        End Set
    End Property

    Public Property ConnectionTimeout() As Data.CommandBehavior
        Get
            Return _ConnectionTimeOut
        End Get
        Set(ByVal value As Data.CommandBehavior)
            _ConnectionTimeOut = value
        End Set
    End Property
```

```
#End Region

#Region "Methods"

#Region "Constructors"


    Public Sub New()
        MyBase.New()
    End Sub

    Public Sub New(ByVal sqlcommand As String)
        MyBase.New()
        _SqlString = sqlcommand
    End Sub

    Public Sub New(ByVal SqlCommand As String, ByVal ConnectionInfo As
String)
        MyBase.New()
        _SqlString = SqlCommand
        _ConnectionString = ConnectionInfo
    End Sub
#End Region

#Region "SqlDataReader methods"
    Public Function ExecuteReader() As SqlDataReader
        Return ExecuteReader(_SqlString, _ConnectionString,
CommandBehavior.Default)
    End Function

    Public Function ExecuteReader(ByVal sqlcommand As String) As
SqlDataReader
        Return ExecuteReader(sqlcommand, _ConnectionString,
CommandBehavior.Default)
    End Function

    Public Function ExecuteReader(ByVal sqlcommand As String, ByVal
connectioninfo As String) As SqlDataReader
        Return ExecuteReader(sqlcommand, connectioninfo,
CommandBehavior.Default)
    End Function

    Public Function executeReader(ByVal SqlCommand As String, ByVal
Connection As String, ByVal Behavior As CommandBehavior) As
SqlDataReader
        Dim sqlcntn As New Data.SqlClient.SqlConnection(Connection)
        Dim sqlcmd As New Data.SqlClient.SqlCommand(SqlCommand,
sqlcntn)
        Dim sqlreader As Data.SqlClient.SqlDataReader

        sqlcntn.Open()
        _ConnectionString = Connection
        sqlreader = sqlcmd.ExecuteReader(Behavior)

        Return sqlreader
    End Function
```

```
#End Region

#Region "DataSet methods"

    Public Function ExecuteDataSet() As Data.DataSet
        Return ExecuteDataSet(_SqlString, _ConnectionString)
    End Function

    Public Function ExecuteDataSet(ByVal sqlcommand As String) As
Data.DataSet
        Return ExecuteDataSet(sqlcommand, _ConnectionString)
    End Function

    Public Function executeDataSet(ByVal SqlCommand As String, ByVal
Connectioninfo As String) As Data.DataSet
        Dim sqlcntn As New Data.SqlClient.SqlConnection(Connectioninfo)
        Dim sqlcmd As New Data.SqlClient.SqlCommand(SqlCommand,
sqlcntn)
        Dim sqlDA As New SqlDataAdapter(sqlcmd)
        Dim dsResults As New Data.DataSet

        _ConnectionString = Connectioninfo
        sqlcntn.Open()
        'dsResults =
sqlcmd.ExecuteResultSet(ResultSetOptions.Updatable)

        sqlDA.Fill(dsResults)

        Return dsResults
    End Function
#End Region

#Region "Execute SQL Methods"

    Public Function executeSqlCommand() As Integer
        Return executeSqlCommand(_SqlString, _ConnectionString)
    End Function

    Public Function executeSqlCommand(ByVal sqlcommand As String) As
Integer
        Return executeSqlCommand(sqlcommand, _ConnectionString)
    End Function

    Public Function executeSqlCommand(ByVal sqlcommand As String, ByVal
connectioninfo As String) As Integer
        Dim sqlcntn As New Data.SqlClient.SqlConnection(connectioninfo)
        Dim sqlcmd As New Data.SqlClient.SqlCommand(sqlcommand,
sqlcntn)
        Dim numrowsaffected As Integer

        _ConnectionString = connectioninfo

        sqlcntn.Open()
        numrowsaffected = sqlcmd.ExecuteNonQuery()
        sqlcntn.Close()
        sqlcmd.Dispose()
        sqlcntn.Dispose()
```

```
        Return numrowsaffected
    End Function
#End Region

#Region "Scalar Execution Methods"

    Public Function Scalar() As Object
        Return scalar(_SqlString, _ConnectionString)
    End Function

    Public Function scalar(ByVal sqlcommand As String) As Object
        Return Scalar(sqlcommand, _ConnectionString)
    End Function

    Public Function Scalar(ByVal sqlcommand As String, ByVal
ConnectionInfo As String) As Object
        Dim sqlcntn As New Data.SqlClient.SqlConnection(ConnectionInfo)
        Dim sqlcmd As New Data.SqlClient.SqlCommand(sqlcommand,
sqlcntn)
        Dim returnedItem As Object

        _ConnectionString = ConnectionInfo


        sqlcntn.Open()
        returnedItem = sqlcmd.ExecuteScalar

        sqlcntn.Close()
        sqlcmd.Dispose()
        sqlcntn.Dispose()

        Return returnedItem

    End Function
#End Region

#End Region

End Class
```

Appendix B: Security Class for Complete Inventory Tyrant

This class is used to connect to the database to make sure the logon information is correct. It also contains a function that checks to make sure the logon name is currently no being used by another employee.

```
Imports System.Security
Public Class Authentication

    Public Function CheckCredentials(ByVal LogonName As String, ByVal
Password As String, ByVal AccessLevel As String) As Boolean
        Dim sqldb As New SQLConnection
        Dim strSQL As String
        Dim intAuthLevel As Integer
        Dim dbpassword As String
        Dim authenticated As Boolean

        strSQL = "select Password from Authentication where
LogonName='" & LogonName & "'"
        dbpassword = sqldb.Scalar(strSQL, DFT_CONNECTION_STRING)

        strSQL = "select AccessLevel from Authentication where
LogonName='" & LogonName & "'"
        intAuthLevel = sqldb.Scalar(strSQL, DFT_CONNECTION_STRING)


        If Password = dbpassword And intAuthLevel >= AccessLevel Then
            authenticated = True
        Else
            authenticated = False
        End If

        Return authenticated
    End Function

    Public Shared Function IsLogonNameUsed(ByVal LongonName As String)
As Boolean
        Dim sqldbCnt As New SQLConnection
        Dim sqldrLogonName As Data.SqlClient.SqlDataReader

        sqldrLogonName = sqldbCnt.ExecuteReader("SELECT LogonName FROM
Authentication WHERE LogonName ='" & LongonName & "'",
DFT_CONNECTION_STRING)

        If sqldrLogonName.HasRows Then
            Return True
        Else
            Return False
        End If

    End Function
End Class
```

Appendix C: Data Validation Class

The data validation class is used to validate the information being inserted into the database. The data validation class uses Regular expressions to make sure the data fits the proper standards used for the data.

```
Imports System.Text.RegularExpressions

Public Class DataValidation

    '^(([A-Za-z0-9]+_+)|([A-Za-z0-9]+\-+)|([A-Za-z0-9]+\.+)|([A-Za-z0-
9]+\++))*[A-Za-z0-9]+@((\w+\-+)|(\w+\.))*\w{1,63}\.[a-zA-Z]{2,6}$

    Public Shared Function ValidatePhoneNumber(ByVal PhoneNumber As
String) As Boolean
        Dim rgexValidatePhone As New Regex("^([\(]{1}[0-9]{3}[\)]{1}[
|\-]{0,1}|^[0-9]{3}[\-| ])?[0-9]{3}(\-| ){1}[0-9]{4}$")

        Return rgexValidatePhone.IsMatch(PhoneNumber)
    End Function

    Public Shared Function ValidateEmail(ByVal Email As String) As
Boolean
        Dim regexvalidateEmail As New Regex("^(([A-Za-z0-9]+_+)|([A-Za-
z0-9]+\-+)|([A-Za-z0-9]+\.+)|([A-Za-z0-9]+\++))*[A-Za-z0-9]+@((\w+\-
+)|(\w+\.))*\w{1,63}\.[a-zA-Z]{2,6}$")
        Return regexvalidateEmail.IsMatch(Email)
    End Function

    Public Shared Function ValidateZipcode(ByVal Zipcode As String) As
Boolean
        Dim regexvalidateZipcode As New Regex("^\d{5}(-\d{4})?$")
        Return regexvalidateZipcode.IsMatch(Zipcode)
    End Function

End Class
```

Appendix D: Work Cited

1. "APG Cash Drawer". www.apgcd.com/apghome/index.php. 20 October 2004.

2. "Bar Codes Info Center". www.skandata.com/infocenter.html. 20 October 2004.

3. Daugs, Bob. Salesmen, APG Cash Drawer. Email Interview. October 19, 2004.

4. Enbysk, Monte. "10 Reasons to Automate Your Retail Store." http://www.posdirect.com/articletenreasons.php . 20 October 2004.

5. "Epson: POS". http://pos.epson.com/posindex.htm. 20 October 2004.

6. Gillming, Joe. Store Manager, ACME Games. Personal Interview. October 10, 2004.

7. Mallory, Dave. Store Owner, ACME Games. Personal Interview. October 8, 2004.

8. Rosenblum, Paula. *Achieving Business Benefits from Point-of-Sale Systems.* Boston: Aberdeen Group, 2004.