Bookstore Management System

By

Jonathan Makcen

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

University of Cincinnati
College of Applied Science

May 2006

Bookstore Management System

by

Jonathan Makcen

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

© Copyright 2006 Jonathan Makcen

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

Jonathan Makcen	Date
Russ McMahon, Faculty Advisor	Date
Patrick C. Kumpf, Ed.D. Interim Department Head	Date

Acknowledgements

Special thanks to Bennie Durr for sponsoring this project and providing the hardware needed for the Point-of-Sale system.

Additionally, special thanks to Randy Burnett and Ideal Computer Solutions for donating the Point-of-Sale hardware that was used at the Tech Expo.

Table of Contents

List of Figures

Abstract

The Bookstore Management System is a software application that is designed to provide a complete management solution to a bookstore. *The Bookstore Management System* will help the owner make intelligent and informed decisions regarding his business by bringing all the data together into a meaningful set of tools. Provided in the application is a Point-of-Sale system, inventory management system, and reporting tools. By utilizing all aspects of the system, a business owner will be able to run a successful and efficient business. *The Bookstore Management System* is built on the latest .NET technologies: SQL Server 2005 and C#.NET. The system interacts with all common Point-of-Sale hardware, and is designed to also interact with credit card processors and warehouse distributors for efficient inventory management.

Description and Intended Use

Problem Statement

Bookstore owners are in need of systems to better manage and run their stores. These systems should not burden the owner and employees with confusing and hard-to-operate systems, but provide simple and easy-to-use automation where it makes sense.

The first need of a management system is a repository that contains all the needed data about the store. This repository will have to store information about the inventory of the store, the employees of the store, the inventory suppliers, and the customers of the store. This data will have to meet three requirements: it will need to be secure and at the same time it will need to be readily available and easily accessible. This repository will be the central item of the management system. All other requirements and features of the system will be driven by the data in the repository.

A second need of a management system is a Point-Of-Sale (POS) module. This POS module is needed for cashiers to quickly and efficiently assist customers in their purchase of items from the store. Without a POS module, the cashiers must manually add the price of each item, make quick decisions about the price of an item if it does not have a sticker, add receipts at the end of the day, and complete manual inventory counts. Therefore, the POS module must then make the job of a cashier easier and more efficient. The POS module must have a connection to the repository of inventory for price lookups; it must calculate the price of the overall sale; it must be able to calculate the tax of a sale; it must deduct the quantity of the item in the inventory list by the number of items ordered; and it must have the ability to calculate in special pricing like coupons or tax-exempt status; it must record all this information about a sale for the purpose of records and reports; it must be able to handle item returns by customers and generate the appropriate reports regarding such action; finally, the POS module must be able to make use of common POS hardware like a barcode scanner, cash drawer, and receipt printer.

A third need of a management system is reporting functionality. Management needs to be able to see the reports that detail the number of items sold, the amount of money being generated by sales, high or low selling items, high selling periods, or any other type of analytical or statistical reports. The reports must be able to pull data from the repository about sales as well as be able to relate data about the sales to inventory data.

A fourth need of a management system is inventory control functionality. This would include functionality to add to and to edit items in the repository, and create orders for either new or existing items. A subset of this functionality would also be several business components. This would include the ability to track accounts payable on orders placed for items, and also track accounts receivable on credit purchases.

In conclusion, there is a need for a system that makes the daily tasks and operations of bookstore owners simpler and more convenient. The system should be easy-to-use as well as clean and neat. Automation and efficiency is the ultimate goal, not confusion and more work.

Solution

My solution to the bookstore owners dilemma is a customized software and database application called the *Bookstore Management System*.

Some of the highlights of this product will be

- Two stand-alone consoles, a Point-of-Sale (POS) console and a Management console:
- POS integration with all major POS hardware, including a barcode scanner receipt printer, cash drawer, pole display, and credit card processor:

- Centralized database to which one or more consoles can connect:
- Complete control of inventory through Management console including the ability to add and edit items in the database, categorization of the items, management of distributors and their contacts, contact capabilities to the distributors for direct ordering, and employee management:
- Account tracking for accounts payable and receivable:
- Reports for sales analysis and accounts analysis:
- Separation of duties through user roles for management and cashier responsibilities

The POS console of the software application will be designed in a manner similar to current POS systems to allow for similarity in operation. It is important that the user interface is designed in such a manner that it is easy to operate and easy to teach others how to operate. The POS console will allow for the reading of item information from the database, and writing to the sales records. The POS console will not have access to write information to any of the item tables.

User Profiles

There are two user profiles that will be created for the system, the manager and the cashier.

Manager

The manager profile will be the administrator of the system. The manager will have complete control over the database and all the information in it. He/she will be able to add and edit items, add and edit employee information, view and print reports, make price adjustments, and create coupons. The manager will also be able to operate the POS console.

Cashier

The cashier will be able to only operate the POS console. This will restrict those that have been labeled as a cashier from changing item information. Through the POS console, the cashier will have the access to read only the item information and write to the sales records table only for the current sale or return. The cashier will not have access to go back and change sales information; a manager would be required to do that.

Design Protocols

Software Design

The Bookstore Management System will be a single software application with two modules. A user will be required to log into the system to begin use of the software. The functions that will be provided to the user after login will depend on their granted role and are shown in the use cases in Figure 1. This use case diagram shows that either a Cashier or a Manager can log into the system. The manager will have all the functionality available that the Cashier has, plus additional capabilities. The Cashier will only be able to operate the POS system.

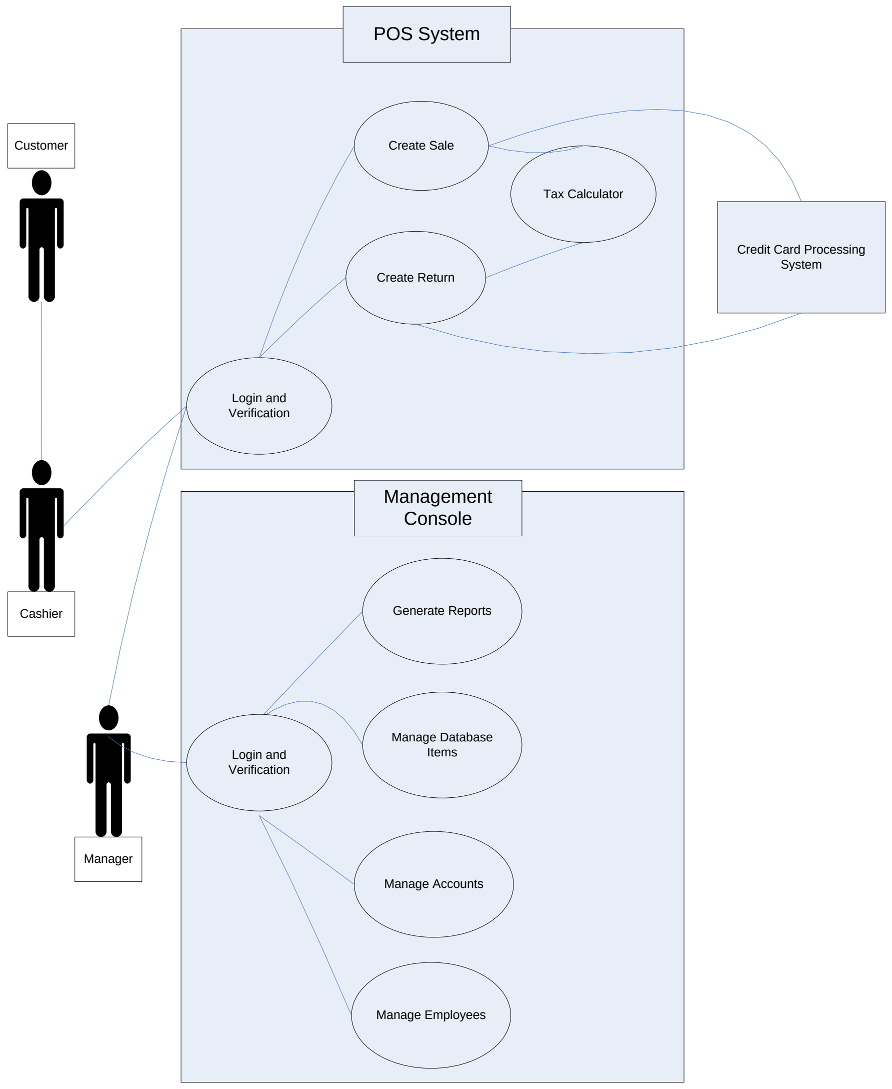


Figure 1 – Use Case Diagram

Class Design

The software component of the project has been developed using the n-tier application approach and as such has 3 layers: the user interface layer, the data access layer, and the business process layer. Figure 2 illustrates all the classes and their interaction with each other. The data access layer has an access class that corresponds to each object class in the business layer. For example, the ItemDA class is the data access class for the Item business layer class.

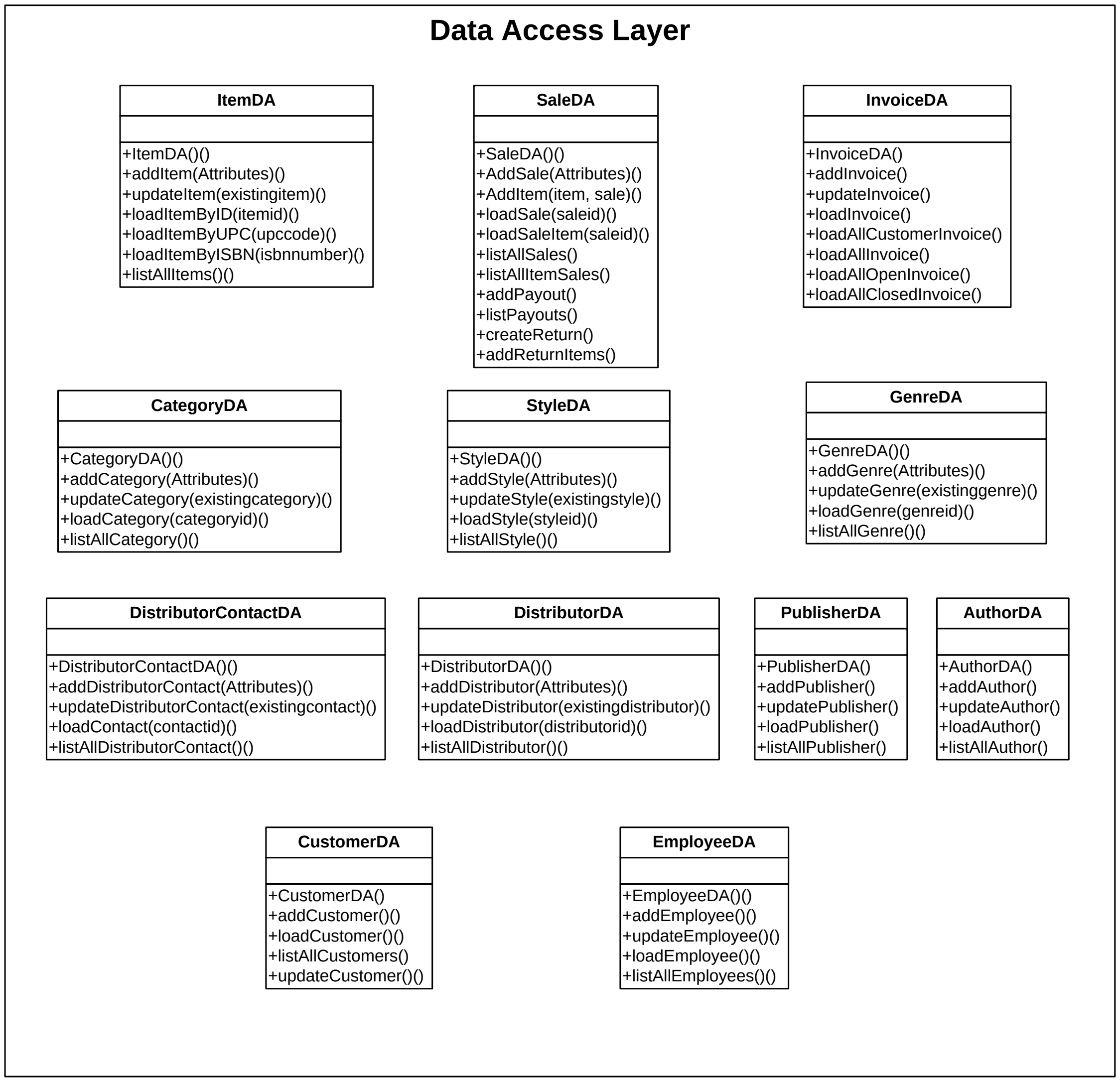


Figure 2 – Data Access Layer

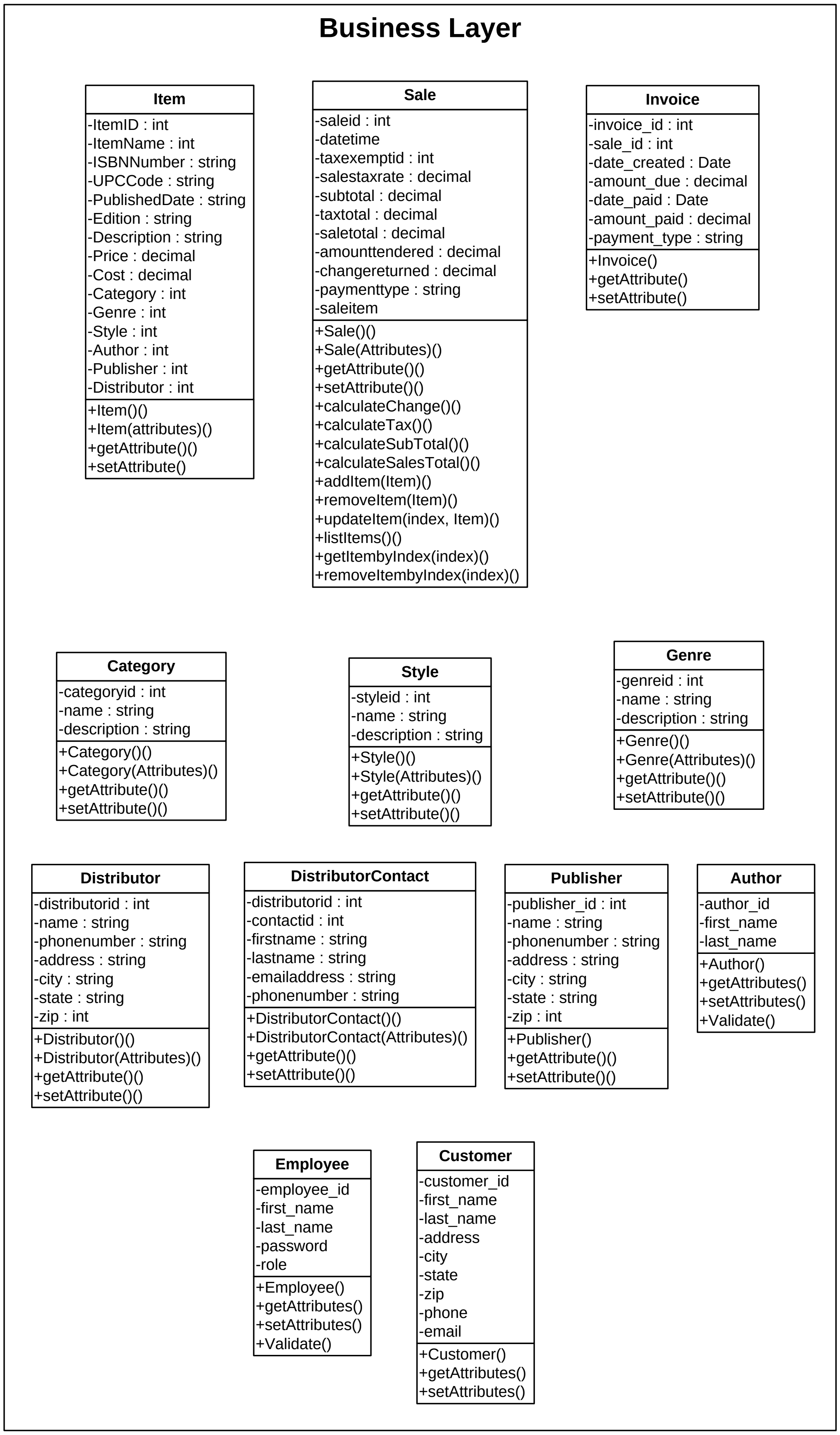


Figure 3 – Business Process Layer

Database Design

The database for this project is a collection of highly normalized tables. The central table is the Item table, and the majority of the other tables are related in some way to the items. The Publisher, Distributor, Category, Genre, and Style tables are all relational secondary tables that contain additional information about the items. The Sales and Return tables relate to the Item table in that a sale or return can contain multiple items.

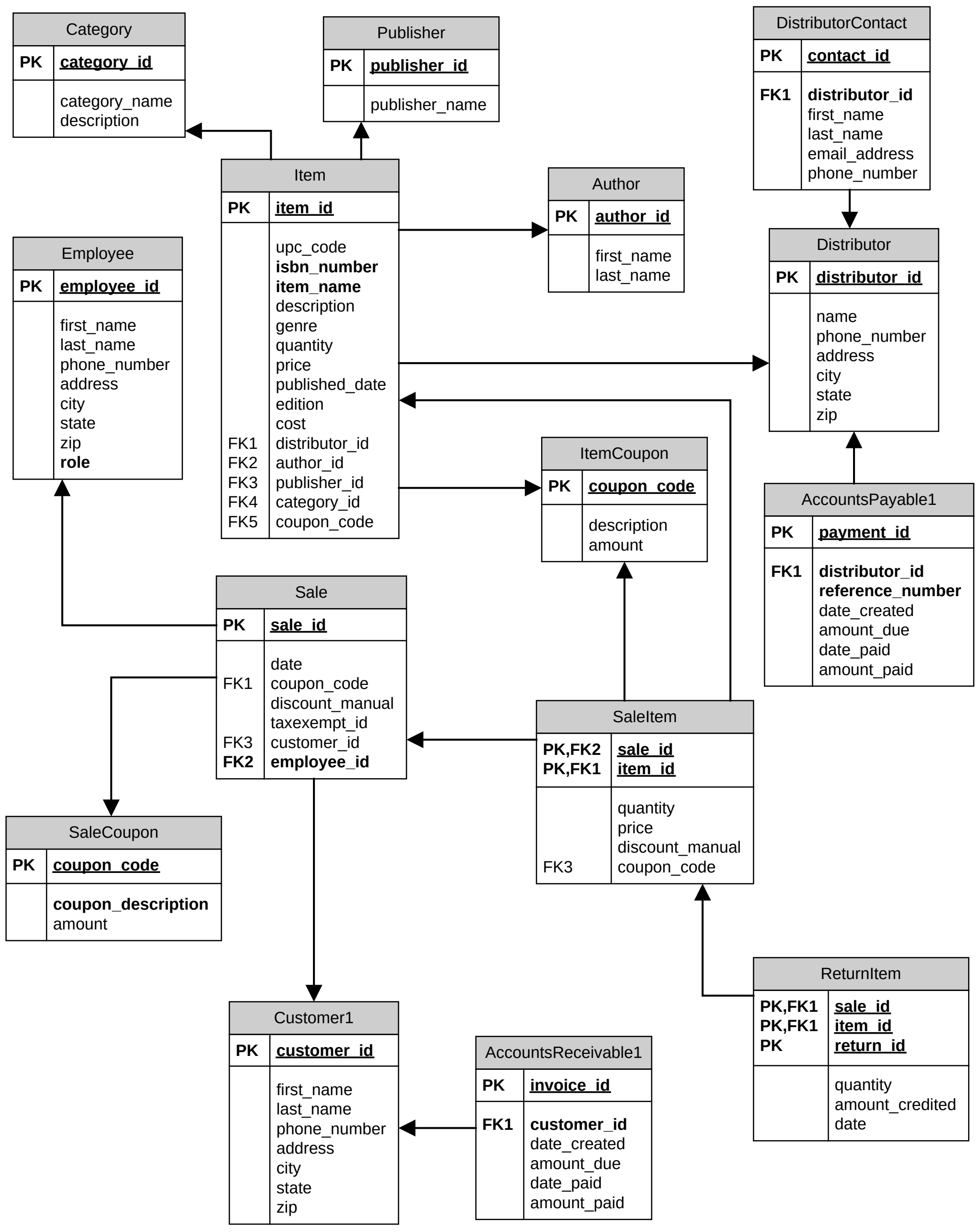


Figure 4 – Database Diagram

User Interface

The user interface is an important part of this project. The user must be able to understand the functionality available to him through the screens they are looking at and know how to apply it. The overall design will be a toolbar at the top of the screen with buttons that take the user to the main subsections of the program. On each section, the functionality available there will be specific to that given area.

POS Console

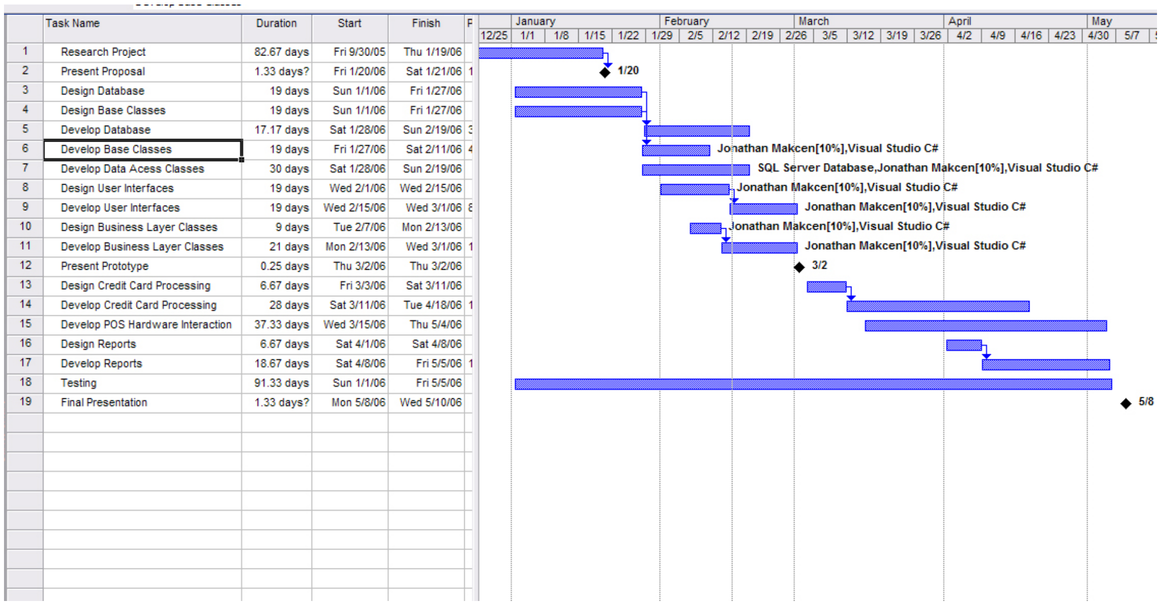
It is important that the user interface design of the POS Console be designed with a similar look and feel compared to current POS Consoles. This similarity would allow for familiarity in usage and ease of transitioning from one system to another. The design components that are outputs from this requirement include: large, easy-to-read buttons; a list of the items in the current sale or return; on-screen hints for keyboard shortcuts; clear flow of process from start of new sale or return to completion of checkout; and prompts at appropriate milestones in the sale or return process flow.

Management Console

The user interface design of the Management Console will contain a large amount of functionality, but should be presented in such a way that it is not confusing or hard to use. This simplicity will be achieved by breaking down the functionality into similar categories, and presenting each category on a separate screen.

Project Plan

Timeline



	Task Name	Duration	Start	Finish	P
1	Research Project	82.67 days	Fri 9/30/05	Thu 1/19/06	
2	Present Proposal	1.33 days?	Fri 1/20/06	Sat 1/21/06	1
3	Design Database	19 days	Sun 1/1/06	Fri 1/27/06	
4	Design Base Classes	19 days	Sun 1/1/06	Fri 1/27/06	
5	Develop Database	17.17 days	Sat 1/28/06	Sun 2/19/06	3
6	Develop Base Classes	19 days	Fri 1/27/06	Sat 2/11/06	4
7	Develop Data Acess Classes	30 days	Sat 1/28/06	Sun 2/19/06	
8	Design User Interfaces	19 days	Wed 2/1/06	Wed 2/15/06	
9	Develop User Interfaces	19 days	Wed 2/15/06	Wed 3/1/06	8
10	Design Business Layer Classes	9 days	Tue 2/7/06	Mon 2/13/06	
11	Develop Business Layer Classes	21 days	Mon 2/13/06	Wed 3/1/06	1
12	Present Prototype	0.25 days	Thu 3/2/06	Thu 3/2/06	
13	Design Credit Card Processing	6.67 days	Fri 3/3/06	Sat 3/11/06	
14	Develop Credit Card Processing	28 days	Sat 3/11/06	Tue 4/18/06	1
15	Develop POS Hardware Interaction	37.33 days	Wed 3/15/06	Thu 5/4/06	
16	Design Reports	6.67 days	Sat 4/1/06	Sat 4/8/06	
17	Develop Reports	18.67 days	Sat 4/8/06	Fri 5/5/06	1
18	Testing	91.33 days	Sun 1/1/06	Fri 5/5/06	
19	Final Presentation	1.33 days?	Mon 5/8/06	Wed 5/10/06	

January
February
March
April
May
12/25
1/1
1/8
1/15
1/22
1/29
2/5
2/12
2/19
2/26
3/5
3/12
3/19
3/26
4/2
4/9
4/16
4/23
4/30
5/7
1/20
Jonathan Makcen[10%],Visual Studio C#
SQL Server Database,Jonathan Makcen[10%],Visual Studio C#
Jonathan Makcen[10%],Visual Studio C#
Jonathan Makcen[10%],Visual Studio C#
Jonathan Makcen[10%],Visual Studio C#
Jonathan Makcen[10%],Visual Studio C#
3/2
5/8

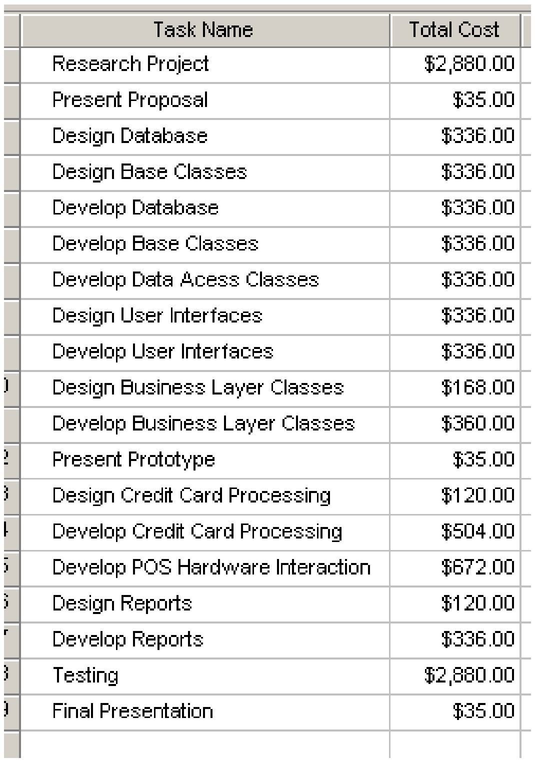
Figure 5 – Project Timeline

Resources

- Developer (myself)
- SQL Server 2005 Express
- Visual C# 2005 Express
- Visio 2005
- Project 2005
- Advisor (Russ McMahon)

Budget

The chart in Figure 6 is an estimate of the cost of labor for this project. Since I am using SQL Server 2005 Express and Visual C# 2005 Express, there are no software costs for this project. These are free downloads from Microsoft.



Task Name	Total Cost
Research Project	$2,880.00
Present Proposal	$35.00
Design Database	$336.00
Design Base Classes	$336.00
Develop Database	$336.00
Develop Base Classes	$336.00
Develop Data Acess Classes	$336.00
Design User Interfaces	$336.00
Develop User Interfaces	$336.00
Design Business Layer Classes	$168.00
Develop Business Layer Classes	$360.00
Present Prototype	$35.00
Design Credit Card Processing	$120.00
Develop Credit Card Processing	$504.00
Develop POS Hardware Interaction	$672.00
Design Reports	$120.00
Develop Reports	$336.00
Testing	$2,880.00
Final Presentation	$35.00

Figure 6 – Budget Estimation for Labor

Proof of Concept

Login

When a user starts the program, they are presented with the Welcome Screen (Figure 7). The welcome screen provides two options to the user, launching either the POS Console or the Management Console.

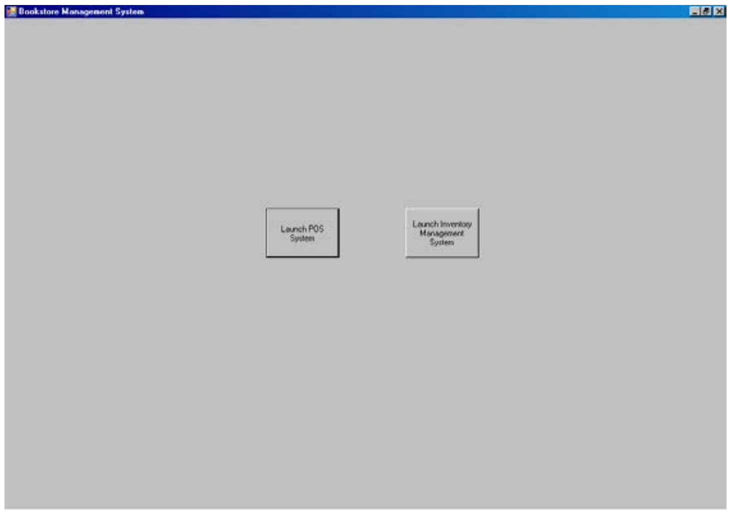

Bookstore Management System
Launch POS System
Launch Inventory Management System

Figure 7 – Welcome Splash Screen

When the user attempts to launch one of the consoles, they will be prompted for their username and password combination. The username is their employee id number, and their password is a phrase that the user sets.

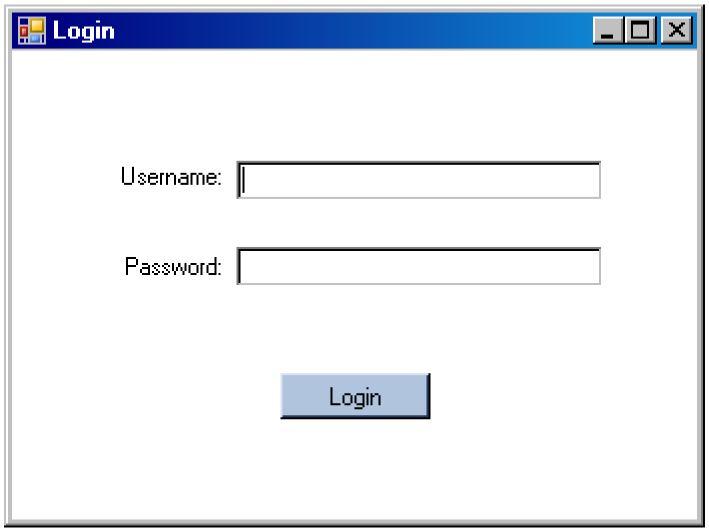


Figure 8 – Login dialog box

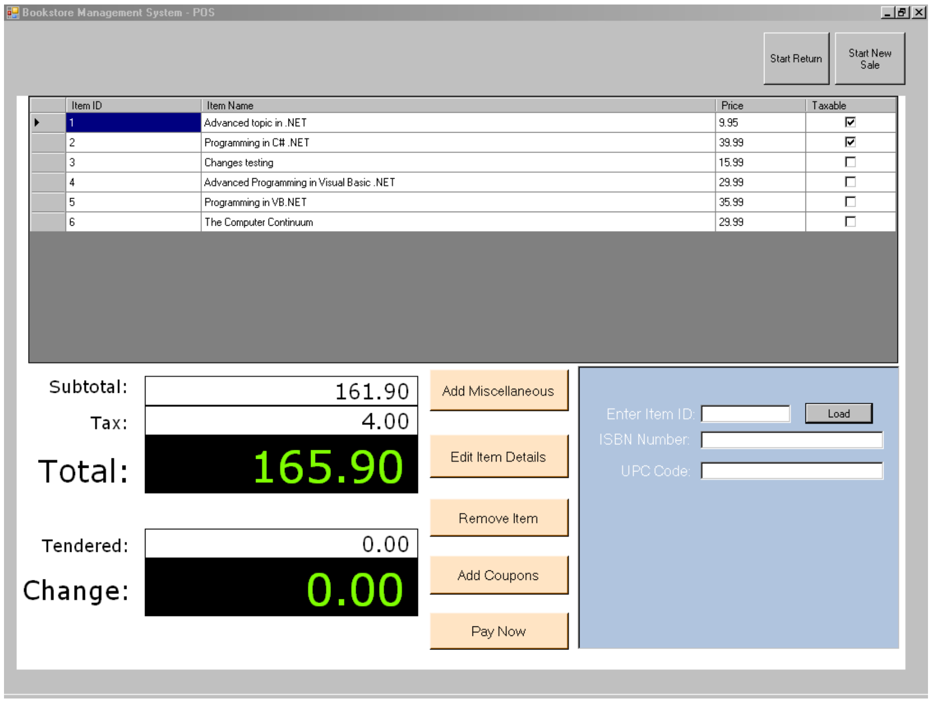
Two checks are performed. The first is to validate that the user attempting access to the console is a registered user in the system. The second check happens when the management console is launched. The user must be a Manager. A cashier cannot log into the management console.

Point-of-Sale Console

There are three pieces of functionality in the POS console. A cashier has the option to start a new sale, start a new return, or perform a payout.

Sale

The sale functionality provides the tools for the cashier to create a sale for a customer. This includes adding items to the sales list, editing the price of those items on the sales list, adding a miscellaneous item, changing the taxable status of items on the sales list, adding a coupon to the sale, and processing the payment for the sale either by cash, check, credit card, or charge account.   Figure 8 depicts the system at the point at which items can be added to the sale and figure 9 depicts the system at the point of payment with the Cash options shown.

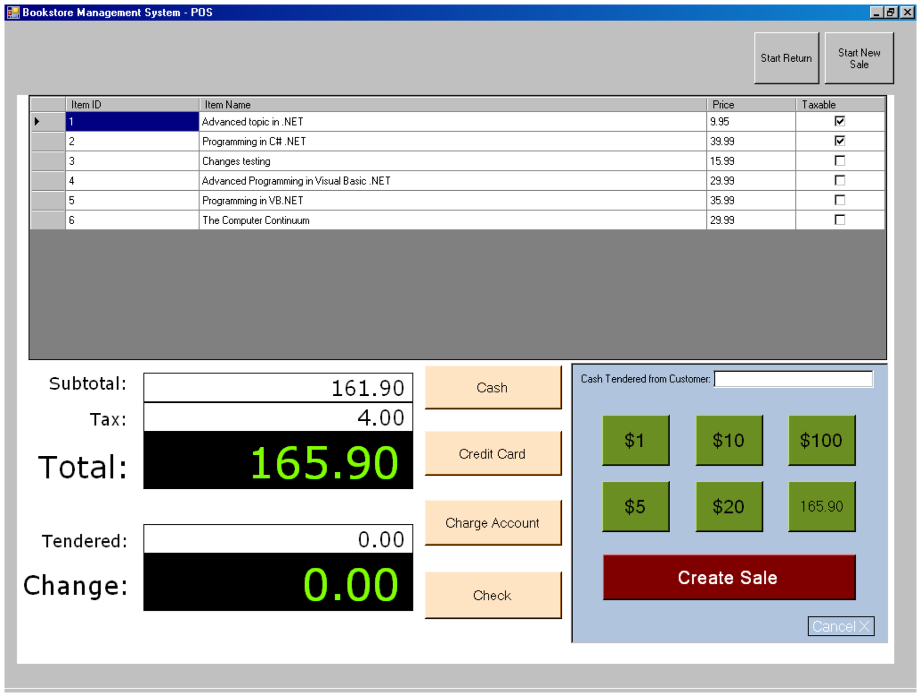


Item ID	Item Name	Price	Taxable
1	Advanced topic in .NET	9.95	✓
2	Programming in C# .NET	39.99	✓
3	Changes testing	15.99	☐
4	Advanced Programming in Visual Basic .NET	29.99	☐
5	Programming in VB.NET	35.99	☐
6	The Computer Continuum	29.99	☐

Subtotal: 161.90
Tax: 4.00
Total: 165.90
Tendered: 0.00
Change: 0.00

Enter Item ID:
ISBN Number:
UPC Code:

Figure 9 – Sale: Adding Items to the sale



Item ID	Item Name	Price	Taxable
1	Advanced topic in .NET	9.95	✓
2	Programming in C# .NET	39.99	✓
3	Changes testing	15.99	☐
4	Advanced Programming in Visual Basic .NET	29.99	☐
5	Programming in VB.NET	35.99	☐
6	The Computer Continuum	29.99	☐

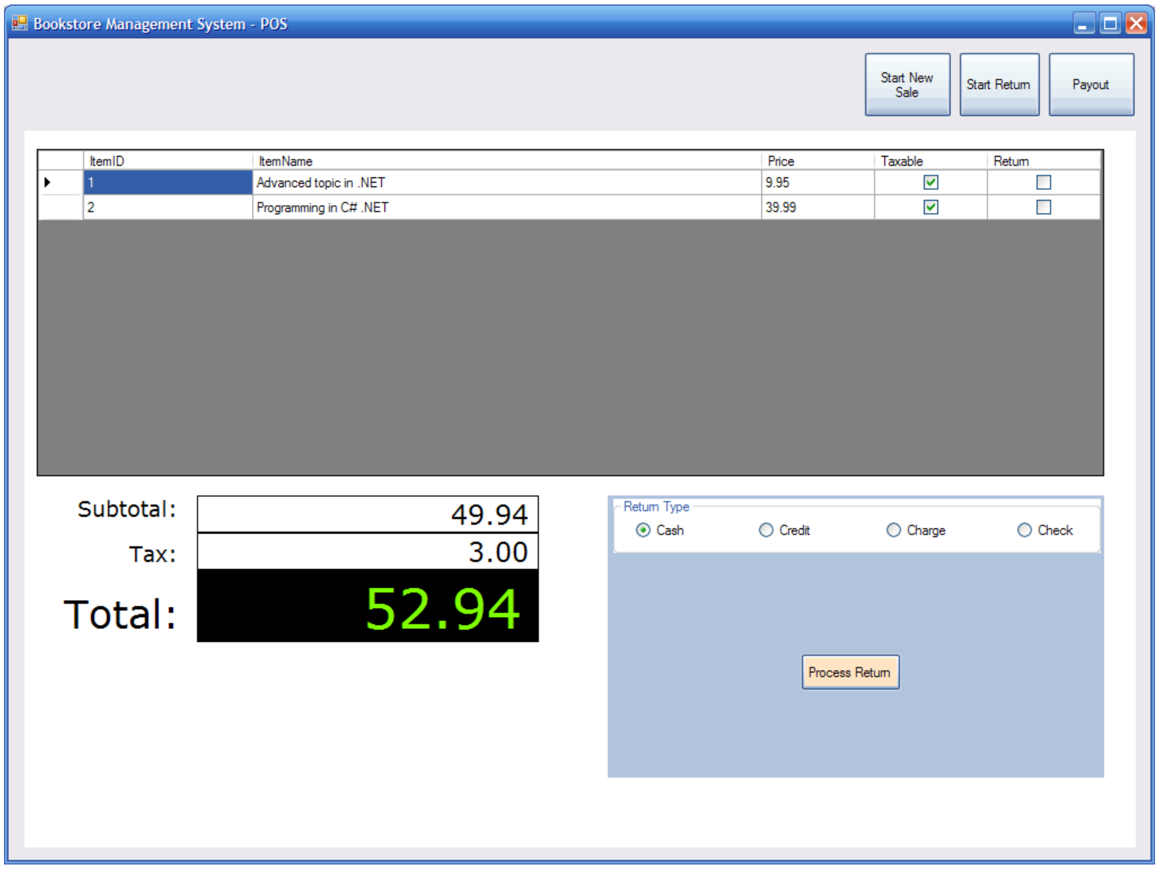
Subtotal: 161.90
Tax: 4.00
Total: 165.90
Tendered: 0.00
Change: 0.00

Cash Tendered from Customer:

Figure 10 – Sale: Cash Payment

<u>Return</u>

The return functionality provides the tools for the cashier to create a return on particular items that had previously been sold. This includes loading a previously created sale, selecting the items to return, and processing the return of payment for the items selected.



ItemID	ItemName	Price	Taxable	Return
1	Advanced topic in .NET	9.95	✓	☐
2	Programming in C# .NET	39.99	✓	☐

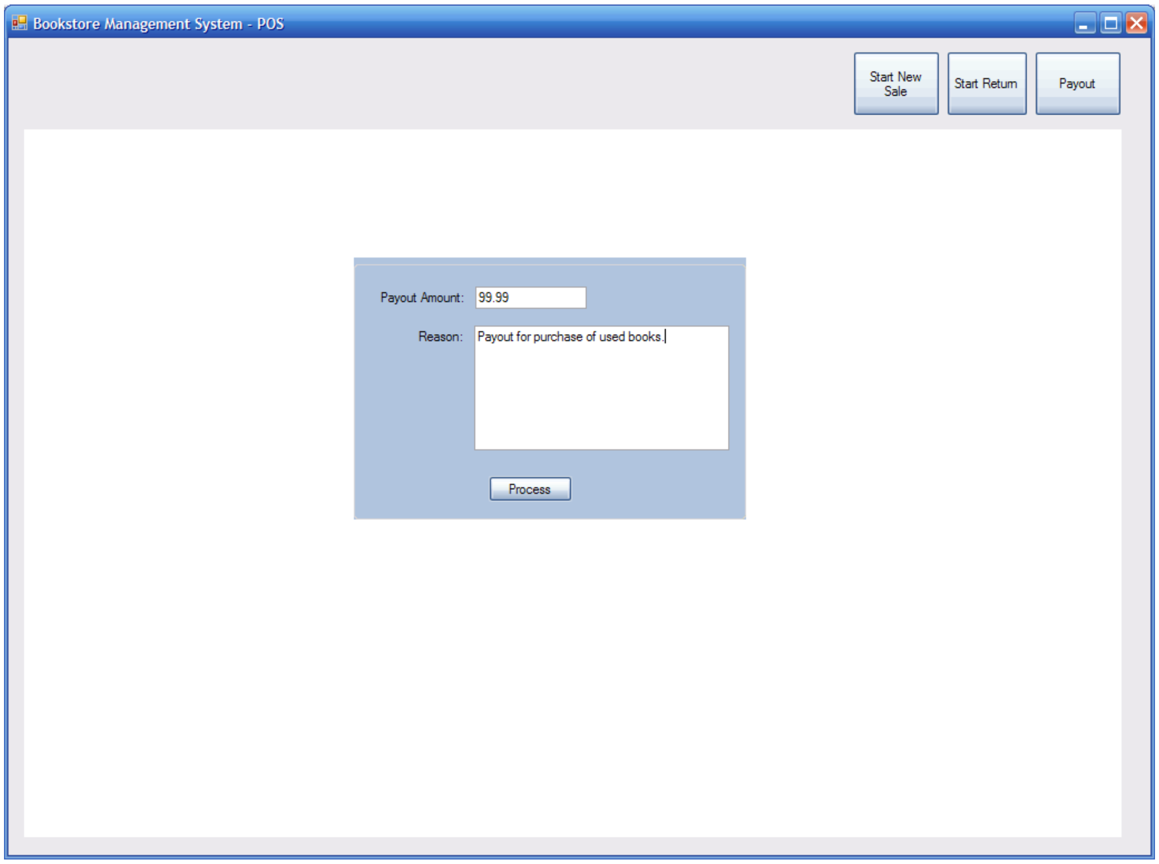
Subtotal: 49.94
Tax: 3.00
Total: 52.94

Return Type: [✓ Cash] Credit Charge Check

Figure 11 – Return on a sale

<u>Payout</u>

At times, it is necessary for the person managing the cash register to perform a payout. The reasons for this could be: Manual return of an item where there is no receipt, purchase of an item, or a need for cash for some other reason. The system requires entering a reason for the payout for tracking and history purposes.



Payout Amount: 99.99

Reason: Payout for purchase of used books.

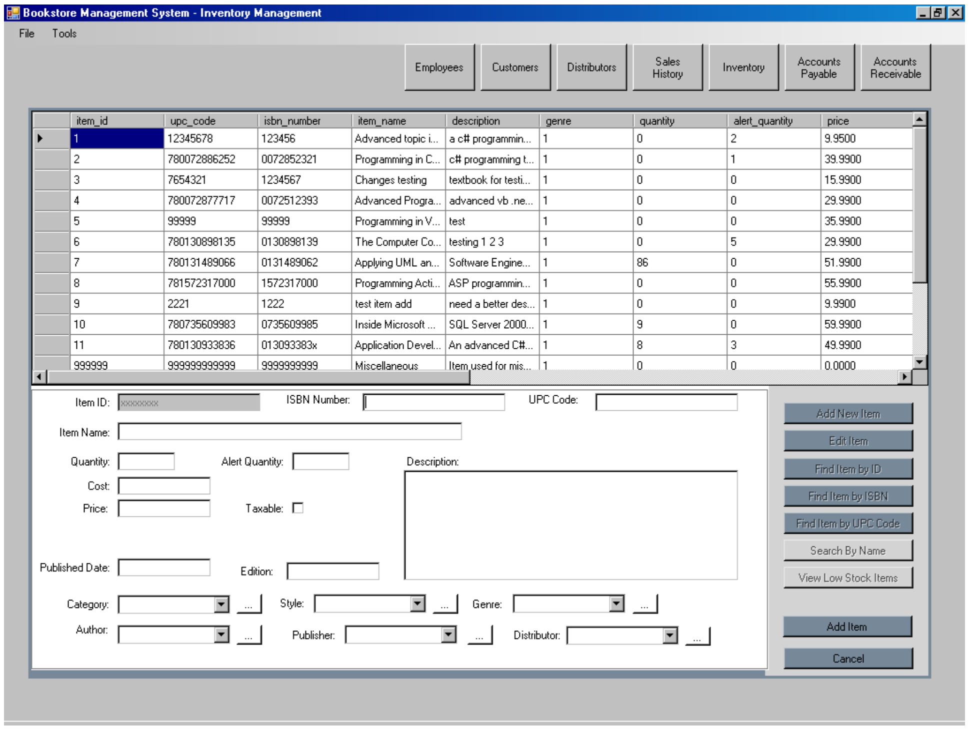
Figure 12 – Payout functionality

Management Console

The management console provides all the functionality needed to maintain the store inventory, sales data, customer data including outstanding invoices, and employee data. In addition, it also provides the necessary reports to enable proper management of the store.

Inventory

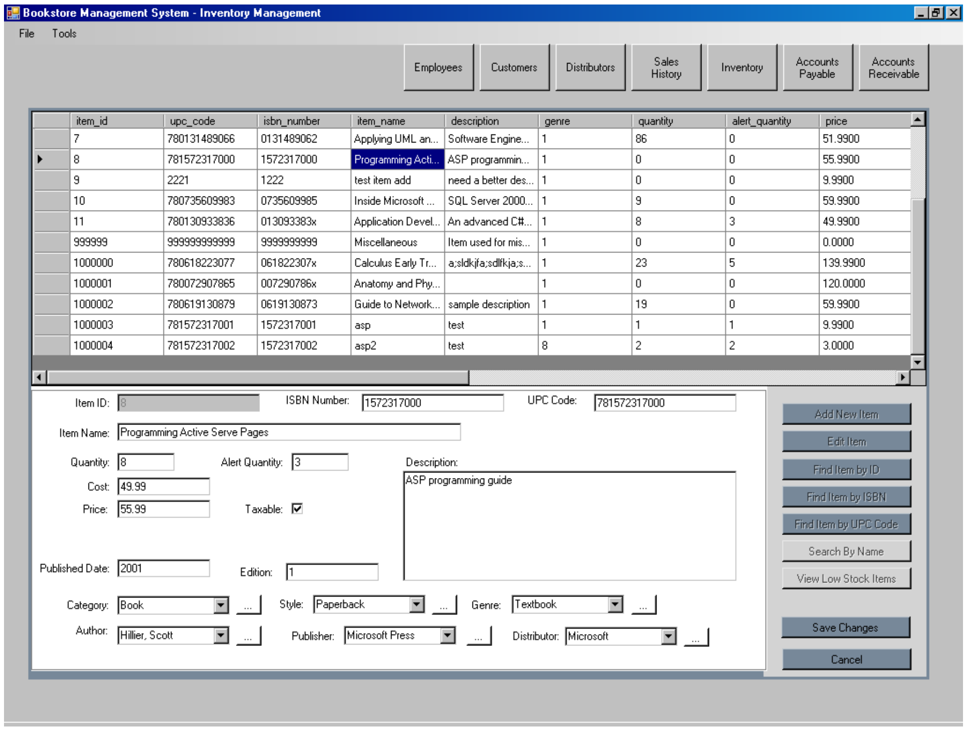
The management console provides management of the inventory by providing the necessary processes to add new items and update item information. The inventory can also be searched by item id, isbn number, upc code. Additionally, searches can be run on the item name field, where the system will return all items where the string the user enters is in the item name. Finally, reports can be run to show the user low stock items and selling history on items.



item_id	upc_code	isbn_number	item_name	description	genre	quantity	alert_quantity	price
1	12345678	123456	Advanced topic i...	a c# programmin...	1	0	2	9.9500
2	780072886252	0072852321	Programming in C...	c# programming t...	1	0	1	39.9900
3	7654321	1234567	Changes testing	textbook for testi...	1	0	0	15.9900
4	780072877717	0072512393	Advanced Progra...	advanced vb .ne...	1	0	0	29.9900
5	99999	99999	Programming in V...	test	1	0	0	35.9900
6	780130898135	0130898139	The Computer Co...	testing 1 2 3	1	0	5	29.9900
7	780131489066	0131489062	Applying UML an...	Software Engine...	1	86	0	51.9900
8	781572317000	1572317000	Programming Acti...	ASP programmin...	1	0	0	55.9900
9	2221	1222	test item add	need a better des...	1	0	0	9.9900
10	780735609983	0735609985	Inside Microsoft ...	SQL Server 2000...	1	9	0	59.9900
11	780130933836	013093383x	Application Devel...	An advanced C#...	1	8	3	49.9900
999999	999999999999	9999999999	Miscellaneous	Item used for mis...	1	0	0	0.0000

Item ID: xxxxxxxx
ISBN Number:
UPC Code:
Item Name:
Quantity:
Alert Quantity:
Description:
Cost:
Price:
Taxable: ☐
Published Date:
Edition:
Category:
Style:
Genre:
Author:
Publisher:
Distributor:

Figure 13 – Inventory: Adding an item



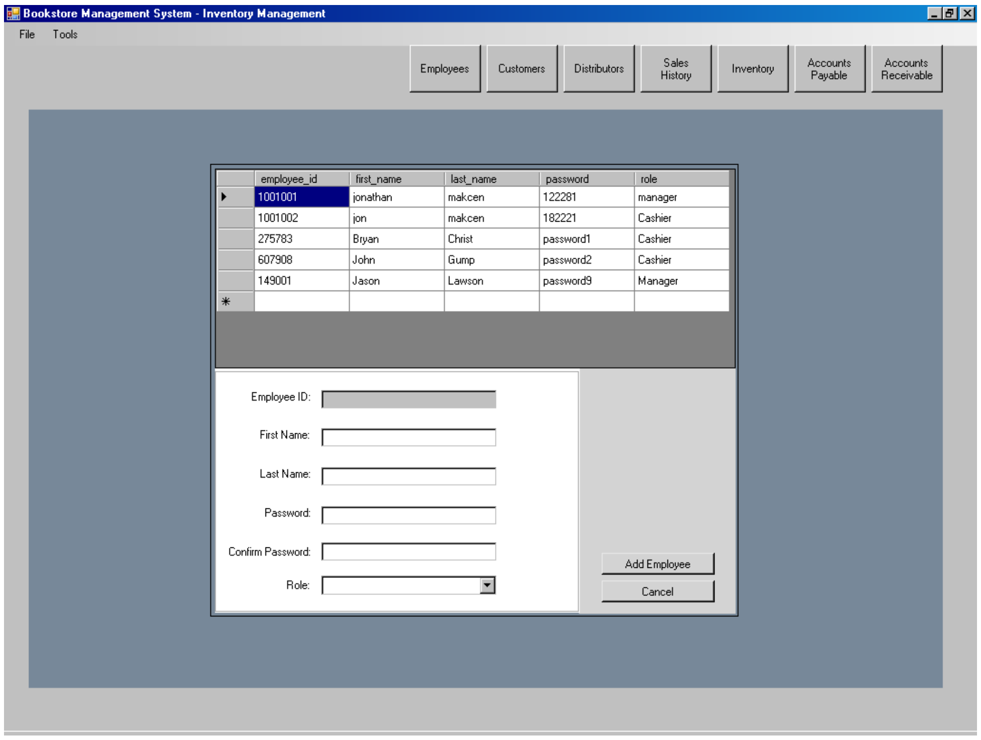
item_id	upc_code	isbn_number	item_name	description	genre	quantity	alert_quantity	price
7	780131489066	0131489062	Applying UML an...	Software Engine...	1	86	0	51.9900
8	781572317000	1572317000	Programming Acti...	ASP programmin...	1	0	0	55.9900
9	2221	1222	test item add	need a better des...	1	0	0	9.9900
10	780735609983	0735609985	Inside Microsoft ...	SQL Server 2000...	1	9	0	59.9900
11	780130933836	013093383x	Application Devel...	An advanced C#...	1	8	3	49.9900
999999	999999999999	9999999999	Miscellaneous	Item used for mis...	1	0	0	0.0000
1000000	780618223077	061822307x	Calculus Early Tr...	a;sldkjfa;sdlfkja;s...	1	23	5	139.9900
1000001	780072907865	007290786x	Anatomy and Phy...		1	0	0	120.0000
1000002	780619130879	0619130873	Guide to Network...	sample description	1	19	0	59.9900
1000003	781572317001	1572317001	asp	test	1	1	1	9.9900
1000004	781572317002	1572317002	asp2	test	8	2	2	3.0000

Item ID: 8
ISBN Number: 1572317000
UPC Code: 781572317000
Item Name: Programming Active Serve Pages
Quantity: 8
Alert Quantity: 3
Description: ASP programming guide
Cost: 49.99
Price: 55.99
Taxable: ☑
Published Date: 2001
Edition: 1
Category: Book
Style: Paperback
Genre: Textbook
Author: Hillier, Scott
Publisher: Microsoft Press
Distributor: Microsoft

Figure 14 – Inventory: Editing item details

Employees

The management console provides the management of employees through providing the ability to add and update employee information.

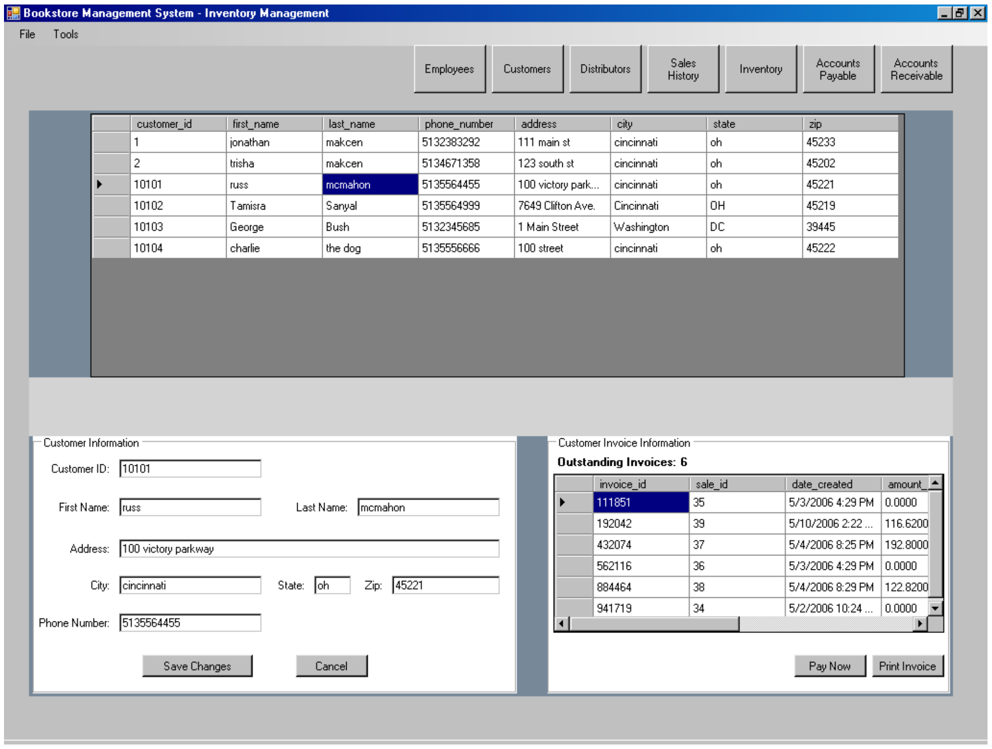


employee_id	first_name	last_name	password	role
1001001	jonathan	makcen	122281	manager
1001002	jon	makcen	182221	Cashier
275783	Bryan	Christ	password1	Cashier
607908	John	Gump	password2	Cashier
149001	Jason	Lawson	password9	Manager

Figure 15 – Employee Management

Customers

Customer management is provided as well by the ability to add and edit customer information. Additionally, customers are able to charge purchases to their account. Management of this is provided by the ability to view outstanding invoices and print individual invoices for billing purposes as well as complete outstanding invoices.



customer_id	first_name	last_name	phone_number	address	city	state	zip
1	jonathan	makcen	5132383292	111 main st	cincinnati	oh	45233
2	trisha	makcen	5134671358	123 south st	cincinnati	oh	45202
10101	russ	mcmahon	5135564455	100 victory park...	cincinnati	oh	45221
10102	Tamisra	Sanyal	5135564999	7649 Clifton Ave.	Cincinnati	OH	45219
10103	George	Bush	5132345685	1 Main Street	Washington	DC	39445
10104	charlie	the dog	5135556666	100 street	cincinnati	oh	45222

Customer Information

Customer ID: 10101

First Name: russ Last Name: mcmahon

Address: 100 victory parkway

City: cincinnati State: oh Zip: 45221

Phone Number: 5135564455

Customer Invoice Information

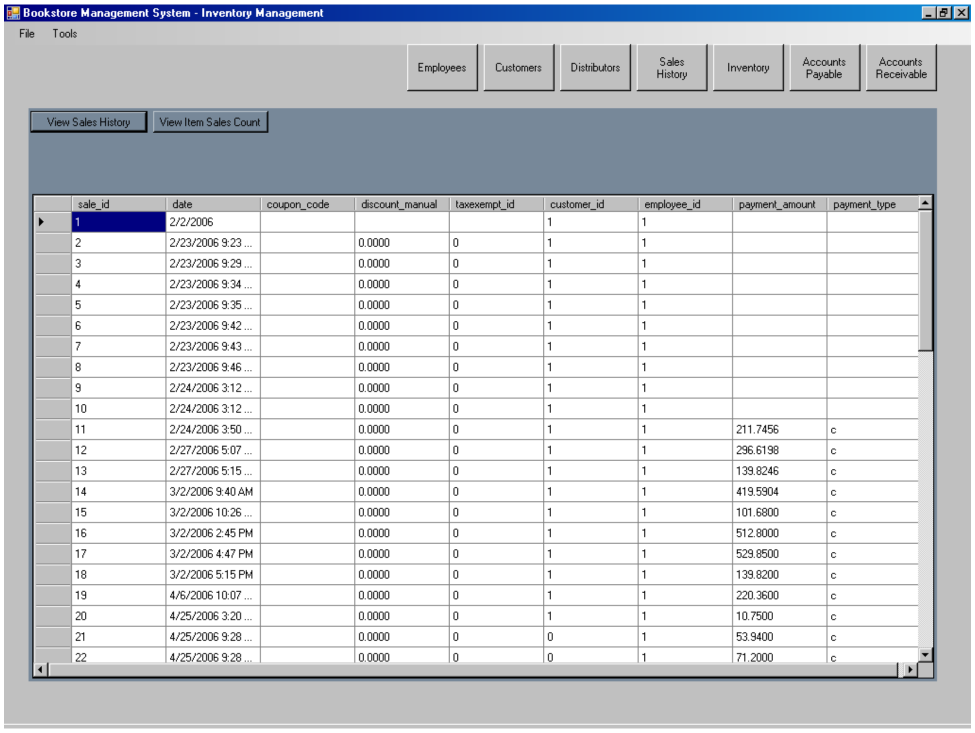
Outstanding Invoices: 6

invoice_id	sale_id	date_created	amount_
111851	35	5/3/2006 4:29 PM	0.0000
192042	39	5/10/2006 2:22 ...	116.6200
432074	37	5/4/2006 8:25 PM	192.8000
562116	36	5/3/2006 4:29 PM	0.0000
884464	38	5/4/2006 8:29 PM	122.8200
941719	34	5/2/2006 10:24 ...	0.0000

Figure 16 – Customer Management

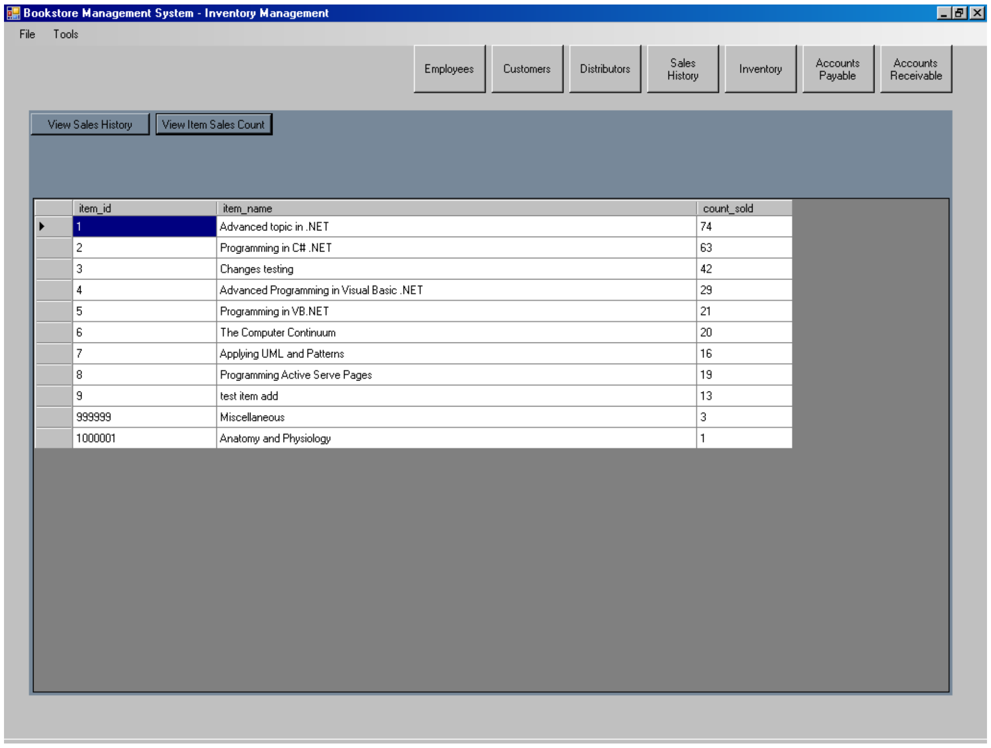
Sales History

Sales history is tracked by both item sales history and total sales history. These reports are able to be filtered several different ways, including date ranges, sale value and item counts.



sale_id	date	coupon_code	discount_manual	taxexempt_id	customer_id	employee_id	payment_amount	payment_type
1	2/2/2006				1	1		
2	2/23/2006 9:23 ...		0.0000	0	1	1		
3	2/23/2006 9:29 ...		0.0000	0	1	1		
4	2/23/2006 9:34 ...		0.0000	0	1	1		
5	2/23/2006 9:35 ...		0.0000	0	1	1		
6	2/23/2006 9:42 ...		0.0000	0	1	1		
7	2/23/2006 9:43 ...		0.0000	0	1	1		
8	2/23/2006 9:46 ...		0.0000	0	1	1		
9	2/24/2006 3:12 ...		0.0000	0	1	1		
10	2/24/2006 3:12 ...		0.0000	0	1	1		
11	2/24/2006 3:50 ...		0.0000	0	1	1	211.7456	c
12	2/27/2006 5:07 ...		0.0000	0	1	1	296.6198	c
13	2/27/2006 5:15 ...		0.0000	0	1	1	139.8246	c
14	3/2/2006 9:40 AM		0.0000	0	1	1	419.5904	c
15	3/2/2006 10:26 ...		0.0000	0	1	1	101.6800	c
16	3/2/2006 2:45 PM		0.0000	0	1	1	512.8000	c
17	3/2/2006 4:47 PM		0.0000	0	1	1	529.8500	c
18	3/2/2006 5:15 PM		0.0000	0	1	1	139.8200	c
19	4/6/2006 10:07 ...		0.0000	0	1	1	220.3600	c
20	4/25/2006 3:20 ...		0.0000	0	1	1	10.7500	c
21	4/25/2006 9:28 ...		0.0000	0	0	1	53.9400	c
22	4/25/2006 9:28 ...		0.0000	0	0	1	71.2000	c

Figure 17 – Sales history



item_id	item_name	count_sold
1	Advanced topic in .NET	74
2	Programming in C# .NET	63
3	Changes testing	42
4	Advanced Programming in Visual Basic .NET	29
5	Programming in VB.NET	21
6	The Computer Continuum	20
7	Applying UML and Patterns	16
8	Programming Active Serve Pages	19
9	test item add	13
999999	Miscellaneous	3
1000001	Anatomy and Physiology	1

Figure 18 – Sales history: Item count

Test Plan

The Test Plan for this project consists of two phases. The first phase is unit testing. As each piece of the software is developed, it will be tested both independently and in conjunction with the rest of the project. Additionally, all input forms will be tested for data quality and validity. All data access objects will be tested for correct functionality and for proper error capturing in the event of incorrect usage. All forms will be tested to ensure that they do not allow bad data to be entered. All data displayed by the system will be verified by comparing what the system displays to what is actually in the database. The POS system will be tested to ensure that all calculations are performed correctly and all receipts match the original sale request. When returns are created, original sales information must be verified to be correct to ensure correct calculation of the return amount.

The second phase of the testing will be the system testing and user acceptance testing. The system will be set up for the users to use for a period of time in a testing environment. This will ensure that the entire system is working together as expected, that the system meets the users needs, and that the solutions are capable of providing quality data.

Deliverables

- A database created in SQL Server 2005
- A POS front-end console designed in C# that has the capability to
    - Process the sales of items:
    - Process the returns of items:
    - Interface with common POS hardware:
    - Interface with a credit card processor:
- A Management system designed in C# that provides data management, account management, and employee management which includes
    - Add and edit items in the database:
    - Connect to distributors system to order/reorder items:
    - Manage employee data:
    - Create and view reports for all relevant data:
    - Manage outstanding debts and credits:
    - Create coupons for in-store use:
- User roles to provide security to the system
    - Cashier role – ability to run POS console only:
    - Manager role – ability to run either the POS console or the Management console:

Conclusions and Recommendations

Conclusion

In conclusion, I have developed a complete application that will assist a store owner in running a small commerce business. The application was built on the .NET 2.0 Framework using C# as the coding language. The database and all database procedures were developed using SQL Server 2005.

I failed to complete one deliverable, and that was providing the ability to create a connection to a distributor for the purpose of automating orders. This is on my list of things-to-do to get this project production ready. All other deliverables were successfully met.

Recommendations

My recommendation to future Seniors is to find a project that pushes the edge of whatever specialty they are in. I feel that my project was a little bland in that it was a desktop application that connects to a database. I was able to successfully make the POS hardware work, which was a great challenge; otherwise, it was a pretty standard project. I would recommend finding something that is cutting edge and a little more unique.

I did enjoy working with the POS hardware. In appendix A you will find code samples of what it took to get them working. This was an interesting endeavor because I had never programmatically worked with external hardware on the computer.

References

1. Frederick, David. Business Analyst. Personal Interview. October 2005.
2. Hoffman, Kevin and Lonny Kruger. *Microsoft Visual C# .NET 2003 Unleashed*. Sams: 2004.
3. Microsoft Windows Embedded Point of Sale Operating System website. http://msdn.microsoft.com/embedded/getstart/devplat/pos/default.aspx. October 2005.
4. Microsoft SQL Server 2005 Express website. http://lab.msdn.microsoft.com/express/sql/. October 2005.
5. Monroe Consulting Services - OLE for Point-of-Sale Systems website. http://monroecs.com/opos.htm. October 2005.
6. Montgomery, Benjamin T. *"Supplemental Sun Salon Management System"*. OCAS Timothy C. Day Technical Library. 2003
7. Point-of-Sale Superstore website. http://www.posguys.com. October 2005.
8. POSWorld – Point of Sale and Barcode Superstore website. http://www.posworld.com. October 2005.
9. Sales Consultant, PCAMERICA. http://pcamerica.com. Email interview. October 2005.
10. Seabolt, Brian. *"The Complete Inventory Tyrant: A Total Inventory Control System"*. OCAS Timothy C. Day Technical Library. 2002.

Appendix A – OPOS Implementation

The implementation of the POS hardware was completed by using the OLE for Point of Sale (OPOS) standards. Here are several code samples of those implementations.

Barcode Scanner

Code added to the designer.cs page to create a data event handler for the scanner.

```
//
//scanner
//
this.scanner.DataEvent += new
SCANNERLib._DScannerEvents_DataEventEventHandler(scanner_DataEven
t);
```

Object declaration

```
private SCANNERLib.ScannerClass scanner = new
SCANNERLib.ScannerClass();
```

Scanner initialization

```
scanner.Open("Scanner1");
if (scanner.Open("Scanner1") != 0)
{
    MessageBox.Show("Failed to connect scanner");
}
scanner.DecodeData = true;
scanner.DataEventEnabled = true;
if (scanner.ClaimDevice(1000) != 0)
{
    MessageBox.Show("Failed to claim scanner");
}
scanner.DeviceEnabled = true;
if (scanner.ResultCode != 0)
{
    MessageBox.Show("Failed to enable scanner");
}
```

```
Event Handler Procedure
```

```
private void scanner_DataEvent(int i)
{
    txtUPCCode.Text = scanner.ScanData;

    scanner.DataEventEnabled = true;
}
```

Pole Display

Object Declaration

```
private POS.Devices.OPOSLineDisplayClass ddd = new
POS.Devices.OPOSLineDisplayClass();
```

Object Initialization

```
try
{
    ddd.Open("LC-PD");
    ddd.ClaimDevice(1000);
    ddd.DeviceEnabled = true;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

Sending data to the display

```
ddd.ClearText();
ddd.DisplayTextAt(0, 0, i.getStrItemName(), 0);
ddd.DisplayTextAt(1, 0, i.getNumPrice().ToString(), 0);
```