Empowered Storefront

By

Chris Jaehnen

Submitted to
the Faculty of the Information Engineering 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

June 2005

Empowered Storefront

By

Chris Jaehnen

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

Chris Jaehnen	Date
Russ McMahon, Faculty Advisor	Date
Patrick C. Kumpf, Department Head	Date

Acknowledgements

I would like to thank Steve Davis, former Senior Web Developer of WWW Internet Solutions, for his guidance and mentorship during my three years with him developing Web applications. Mr. Davis taught me the necessary skills for e-commerce application development that played an important role in the development of this project. I would also like to thank Information Engineering Technology faculty advisors Annu Prabahakar, Tom Wulf, and Russ McMahon, for their advice and support during the development of this project.

Table of Contents

List of Figures

Abstract

Empowered Storefront is a redistributable e-commerce application suite that Web hosting providers can incorporate into their Web site hosting packages and allows small businesses to establish an e-commerce presence. Small businesses commonly turn to Web hosting providers to establish an e-commerce presence since they are cost effective. Current e-commerce products include too many complex features, poorly designed user interfaces, insufficient application performance, and costly license fees. Empowered Storefront incorporates elements found in other products that small businesses need in an e-commerce application suite, and it improves on the functionality and usability that other products lack. This application suite provides a cost effective e-commerce solution for Web hosting providers and an easy-to-use and consistent solution for small businesses. Empowered Storefront is built with the latest technologies: Visual Basic .NET, ASP.NET, and SQL Server. The application suite provides small businesses with: an on-line store, a management application for customers, and an on-line store management application. The on-line store allows customers to browse for products, add products to a shopping cart, and place orders. The management application for customers allows them to manage their billing and shipping information, and it allows them to manage and track their orders. The on-line store management application allows small businesses to manage products, product categories, orders, customers, payment gateways, shipping providers, taxes, and users. Overall, Empowered Storefront promotes performance, usability, and reliability for Web hosting providers and small businesses.

Empowered Storefront

1. Statement of the Problem

1.1 Introduction to Commerce

In order for a business to succeed, it needs products and services to sell to its customers. To do this, a business needs a set of processes to develop and deliver these products and services to its customers. These processes make up the foundation of commerce, which, according to Gary Schneider, is, "a negotiated exchange of valuable objects or services between at least two parties and includes all activities that each of the parties undertakes to complete the transaction" (14, p. 5). These two parties often consist of a buyer and a seller. A seller develops products and services to meet the needs of buyers, while a buyer will purchase these products and services to fulfill the seller's needs. When a transaction or exchange takes place, the needs of both the buyer and seller are fulfilled, and thus commerce has occurred.

Whether one is a buyer or a seller, a sequential process of commerce between the two groups must take place. A buyer, according to Schneider's model, will follow this commerce sequence: identify a specific need, search for products or services that will satisfy the specific need, select a vendor in which to purchase the product or service, purchase the product or service from the vendor, and make warranty claims (14, p. 6). Conversely, a seller, according to Schneider's model, will follow this commerce sequence: conduct market research to identify customer needs, create products and services to meet customers' needs, advertise and promote products and services, negotiate a sale transaction with a customer, ship the goods and invoice the customer, receive and process customer payments, and provide customer support services after a

transaction is complete (14, p. 8). The two models show that each group performs different activities to fulfill the other's needs. Ultimately, the activities of both groups reach the end of the transaction by fulfilling the needs of one another.

To further extend one's understanding of these activities in commerce, Schneider states, "the activities in which businesses engage as they accomplish a specific element of commerce are often referred to as business processes" (14, p. 9). Business processes are steps in the commerce sequence that companies and organizations perform in order to meet the needs of customers. Gradually, as business processes are completed during a transaction, the needs of both groups get closer to being fulfilled, until the needs of both groups are fulfilled and the transaction is complete. From observation, businesses represent the "seller" group, so these processes follow Schneider's "seller" model and form the foundation of traditional commerce. Traditional commerce shows a general, systematic process businesses perform when engaging in commerce with customers, and provides one with a general view of what commerce is and why it is important to business. The success of a business' commerce will ultimately decide whether a business will succeed.

1.2 Electronic Commerce

The methods of traditional commerce usually involve physical interaction between businesses and its customers. Businesses will often use retail merchants to sell products and services. These retail merchants provide customer sales representatives to sell products and services directly to customers. In turn, customers have to come to these retail stores in order to purchase products and services. While some businesses may sell their products and services directly to their customers, many will use retailers to eliminate

another step in the commerce process on their part. This creates another party in the commerce process. The retailer is often considered the coordinator between the businesses and the customers. Retailers buy products and services from businesses, and they sell these products and services directly to customers. Effectively, this meets the needs of businesses and customers, thus completing the commerce process. With the additional retail group, the commerce process becomes more difficult to follow, and a direct link between the business and the customer no longer exists. Communication problems may arise where the goals of the business and the needs of the customer are not directly translated through the retailer to the opposite party. Depending on the needs of the business, the methods of traditional commerce may not be sufficient. Therefore, a more efficient system of commerce is needed to meet these needs.

To escape the limits of traditional commerce, a new method of commerce was needed. Technological innovations brought the creation of electronic commerce (e-commerce), which is a method of commerce that uses "electronic data transmission to implement or enhance business processes" (14, p. 10). For example, "banks have used EFTs (Electronic Funds Transfers) to move customers' money around the world, all kinds of businesses have used EDI (Electronic Data Interchange) to place orders and send invoices, and retailers have used television advertising to generate telephone orders from the general public for all kinds of merchandise" (14, p. 9). This example shows several ways businesses use e-commerce to perform business transactions. In this example, methods of e-commerce are used not only to reach the customers through different electronic media, but they are also used to enhance business processes; for example, bank transactions are handled electronically rather than by paper. In some cases, methods of e-

commerce can simplify methods of traditional commerce. For example, computer software is a good choice for e-commerce since it can be distributed through electronic media. The software can be downloaded directly to a customer's personal computer at anytime without the need of a compact disc (CD) and updates to the software are always available on a Web site (14, p. 13). Thus, with today's advances in technology, the need to conduct business processes electronically is evident, and e-commerce will only continue to become important as more businesses rely on technology to automate and enhance their business processes.

In the realm of e-commerce, many methods of electronic data transmission can be used to perform business transactions. Although e-commerce has existed for some time, it has only recently become well known to the business world. The advent of the Internet sparked a revolution in e-commerce by removing many of the physical barriers of traditional commerce. Customers no longer had to visit retail stores to purchase products and services. On-line, or virtual, ordering systems were created to allow customers to purchase products and services from home via a personal computer and an Internet connection. This new type of e-commerce has changed the way many companies do business and at times, even automated traditional commerce and already existing e-commerce business processes. This allows both customers and businesses to meet their goals more quickly and efficiently.

"Internet commerce," according to Schneider, is a type of e-commerce that "specifically uses the Internet or the Web as its data transmission medium" (14, p. 10). Although Internet commerce is a type of e-commerce, it is often mistaken solely to be e-commerce since many people correlate the birth of e-commerce with the large scale

arrival of the Internet and on-line shopping. From this point onward, any references to e-commerce will refer to Internet commerce only for simplicity. Not only did the arrival of e-commerce allow for easier methods to complete business transactions, it completely changed the way many do business and it changed the way many traditional commerce activities were performed.

1.3 The Need for E-Commerce in Small Businesses

Since the arrival of the e-commerce, companies, both large and small, can now reach customers on a global scale. Traditionally, small companies have limited competition to local or regional markets because small companies cannot compete with larger corporations in national and international markets (5). In order for a small company to develop a presence in a national or international market, the company would need to partner with a larger corporation already established in one of these markets (5). E-commerce removes this competition barrier since any company, regardless of its size, can sell directly to its customers. However, do to so, an e-commerce Web application, commonly known as an on-line store, is needed to allow customers to interface with a business. Many larger companies have developed custom e-commerce Web applications to suit their individual business needs. However, these custom built applications can cost hundreds of thousands of dollars to develop. Larger companies can afford this expense, but smaller companies cannot afford a custom solution to establish an e-commerce presence.

Typically, when a small business establishes a basic Web presence, it will host its Web site in a shared hosting environment where multiple Web sites are stored on the same Web server. This is cost effective since a monthly fee ranging from ten to fifty

dollars is all that is needed to host small business Web sites. In contrast, large companies will house their Web sites on their own Web servers. These companies often have an in-house or outsourced Information Technology (IT) staff to manage these servers. Small companies cannot afford the cost of purchasing and maintaining these servers, and if required, to pay the salaries of the IT staff. For this reason, small businesses turn to shared Web hosting providers to host their Web sites since Web hosting providers incur the cost of purchasing and maintaining Web servers, and they pay the salaries of their IT staff.

In order to meet the e-commerce needs of small businesses, many Web hosting providers purchase e-commerce applications rather then implement their own because of the large cost of such an endeavor. Typically, these e-commerce applications are offered as an add-on service to an existing Web hosting service. Again, small businesses simply pay a monthly fee for this additional service. The Web hosting provider handles the cost and maintenance of the e-commerce application. In this case, business processes are created to meet the low cost needs of establishing an e-commerce presence for small businesses. This allows small businesses to compete with large companies on a global scale. However, when small Web hosting companies are first established, they may not have the financial resources to purchase large e-commerce applications. Depending on the cost of the application, Web hosting providers may raise their e-commerce service costs to balance the cost of the application. In the next section, I outline several popular existing e-commerce applications that include their costs and associated problems if they were to be implemented.

1.4 Existing E-Commerce Applications

Although there are many e-commerce applications available, many of the costly e-commerce applications contain hundreds of features that are meant for large corporations. From my research, I found that demonstrations of several of these products offered many useful features, but lacked proper design, usability, and support for the price each vendor requested. Moreover, these products were not cost effective to Web hosting providers, and the layout and navigation of these applications did not follow the principles of good design. From this, small businesses would have a difficult time running an electronic business through one of these applications. From my analysis of three e-commerce applications, discussed below, I include reasons why these products do not suit the needs of Web hosting providers and small businesses (See Figures 1., 3., 4.)

Microsoft Commerce Server (See Figure 1.) provides a platform for e-commerce solutions.

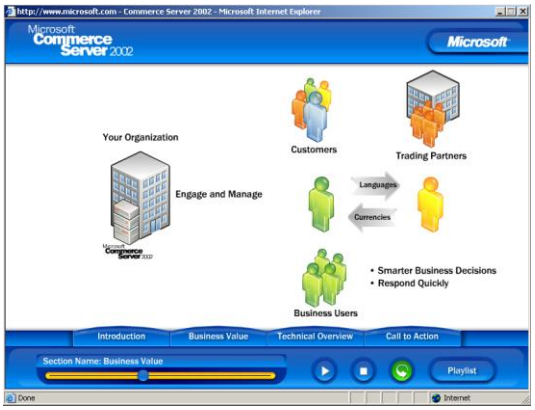

Figure 1. Microsoft Commerce Server

It includes common, built-in features for transaction-based applications, such as catalog management, order processing, and merchandising (8). Using Commerce Server, e-commerce applications would interface with these features to get information and perform operations. These features are built into Commerce Server to simplify the

implementation of e-commerce applications, and to speed up their deployment. This system provides the necessary features to implement an e-commerce application, but it lacks a pre-built on-line store interface. This interface would need to be developed to communicate with Commerce Server. As a result, the e-commerce application could not simply be setup and configured for a given Web site. A Web hosting provider would need to invest time and resources to develop for Commerce Server. This does not give Web hosting providers an adequate solution to meet the needs of small businesses. Above all, as outlined below (See Figure 2.), Commerce Server is too costly to even consider.

E-Commerce Application	Cost
Microsoft Commerce Server 2002 Standard Edition	$6,999 per processor
Comersus ASP Shopping Cart	Free under an open source license
Miva Merchant	$695.00 per on-line store

Figure 2. Analysis of Existing E-Commerce Applications (7, 1, 11)

Comersus (See Figure 3.) provides a complete e-commerce solution for any type of business.

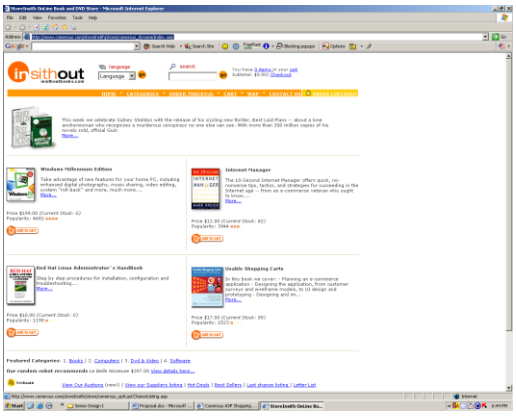

Figure 3. Comersus ASP Shopping Cart

Comersus is an open source product that is free for any type of use (1). Although this is a good incentive, Comersus has many problems associated with it. The project was developed by many programmers with diverse styles of programming. Different programmers developed add-ons and new features as Comersus evolved. As a result of these different programming styles, a lack of consistent code exists in the application. Maintenance of this application requires more time and resources since application logic and interface code is scattered throughout the application. Moreover, Comersus lacks an object-oriented design, which makes maintenance even more difficult.

Through e-commerce experience, Steve Davis, former senior Web developer of WWW Internet Solutions, found that certain functions of Comersus did not even work even though they are stated to do so (2). Davis stated, "Comersus provides the framework for a good on-line store, but it lacks consistency, which is important in any software project" (2). Since Comersus is an open source project, detailed documentation is scarce and technical support is limited. Most importantly, the user interface and navigation design is complex and confusing. This makes using Comersus and maintaining an operating business difficult. Above all, the product functions well for its cost, but its inherited problems make it not worth consideration.

Miva Merchant (See Figure 4.) provides another complete e-commerce solution for any type of business. This product was developed by a third-party company called Miva. Even though Miva Merchant is a fully-functioning product, the on-line store management application lacks consistent navigation, and instructions on how to perform different operations are vague.

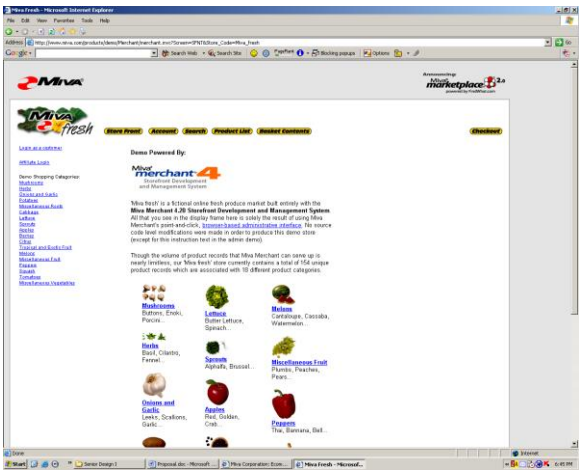

Figure 4. Miva Merchant

Customizing features of Miva Merchant is difficult since Miva developed its own scripting language, MivaScript, for the on-line store. One would have to learn the scripting language in order to customize its features.

Many of the features in Miva Merchant, such as advanced reporting, are designed for larger companies. This is evident from the cost of Miva Merchant. As outlined in the cost analysis of these e-commerce applications (See Figure 2.), Miva Merchant is licensed on a per Web site basis. This means that each time a company wants to use Miva Merchant, a license from Miva needs to be purchased. Miva does not offer an unlimited use license on a per Web server basis. This is not efficient for Web hosting providers since they will need to obtain a license each time from Miva before an on-line store can be setup. This effectively causes a delay in the setup process. Overall, Miva Merchant has the capabilities required to meet the needs of small businesses, but its high cost and associated problems is not sufficient for Web hosting providers and small businesses.

All of these products have both positive and negative elements. Even though all of them contain the necessary features required for small businesses, the products are

either too costly or lack good usability. An application that incorporates elements of each of these products, that provides a consistent design and navigation scheme, and that is cost effective is needed to meet the needs of Web hosting providers and small businesses.

2. Description of the Solution

In order to meet the needs of Web hosting providers and small businesses, I incorporated elements of other e-commerce products that small businesses need in an e-commerce solution, and I improved on the functionality and usability that other products lacked. This application provides a cost effective e-commerce solution for Web hosting providers and an easy-to-use and consistent solution for small businesses. Key elements of the application include:

- Common e-commerce features required for any type of business:
    - Product catalog with stocking capabilities
    - Multiple levels of product categorization
    - Shopping cart and checkout functionality
    - Order processing and tracking
    - Pre-built payment gateways
    - Real-time shipping rates
    - Customizable static shipping rates
    - Customizable tax rates
    - Customer account registration and management capabilities
    - Stored customer information for reoccurring customers
    - Customer only management center
    - On-line store management center

- Object-oriented design:
    - Application logic and user interface code is separated to improve performance and maintainability.
    - The application's design and code structure follows good programming practices and techniques to ensure consistency and reliability in the application.
- Simple and consistent user interface:
    - The look of the user interface is consistent on every page in the application.
    - The navigation scheme groups similar elements together and they are located in logical areas of the user interface that are common among e-commerce applications.
    - Hyperlinked navigation elements are iconic and text-based to support future globalization of the application.
    - The checkout process includes a graphical representation showing users where one is in an order from start to finish.
    - Friendly data input hints and reminders are provided to ensure users can complete forms in the proper format.
- Quick installation and configuration
    - Application files are packaged into an executable installation file that installs application files into a given Web site.
    - The database connection information is handled in one configuration file.
    - The look of the user interface can be changed globally through the application' pre-built template files.
    - On-line store settings are configured through the application's on-line store management application.
- Cost effective product licensing for Web hosting providers:
    - A single server license allows one to purchase this application and use it for an unlimited amount of Web sites for that server.
    - The license fees are as follows:

        - New single server license: $249.99
        - Upgraded single server license: $124.99
    - These fees are set at a rate similar to a competing product's rate for a single Web site license. The purpose of this is to give Web hosting providers an incentive to purchase a single server license that can be used on as many sites as a given server can hold for the same price as a single Web site license from a competitor's product. These fees are much less than competitor fees for single server licenses.
    - Version upgrades licensing fees are discounted.
    - No setup fees are required.
    - Web hosting providers are free to incorporate this product into their Web hosting packages at their own discretion.

The application is broken into three sub subsystems: an on-line store, a customer management system, and an on-line store management system. The development of Empowered Storefront was not financed by a particular company or organization, and I hold sole proprietary rights to Empowered Storefront. I priced Empowered Storefront below standard market value for comparable systems to meet the low cost needs of Web hosting providers and small businesses.

2.1 User Profile

There are three types of users in Empowered Storefront. The three types of users are customers, operators, and administrators. The characteristics of each user and the user's expected level of IT literacy are described as follows:

2.1.1 Customers

Customers are the primary users of the application. They use the on-line store and customer management applications. Customers use the application to purchase products and services from the on-line store's owner and track the status of orders. Since

the on-line store subsystem is a public application, customers can be anonymous. Depending on the business's marketing plan, the customer base can include: existing customers, reoccurring customers, new customers, or specific business vendors. Customers could also be international customers since this application will be distributed globally. The nature of a given company's business will decide what customers will use the application.

Since customers can be anonymous, their level of IT literacy will vary from business to business. Customers are basic computer users with an understanding of the Internet and how e-commerce works. Customers are expected to know how to log onto the Internet, use a Web browser, complete on-line forms, and log into a secure section of a Web site. The user interface is intuitive so customers can easily navigate through the application. Customers should have a basic understanding of the on-line store's privacy policy, which will be provided by the store's owner, since sensitive information will be collected from a given customer. The data will be collected and stored securely by the application. Overall, a basic understanding of how to use Internet will be sufficient for this type of user.

2.1.2 Operators

Operators are the secondary users of the application. They use the application's on-line store management subsystem to manage customer orders and store products. Logically, this type of user can include several physical job positions, such as sales clerks, shipping clerks, and customer service representatives. Sales clerks are responsible for preparing the details of an order and the managing its payment. They will also be responsible for maintaining prices and other information for each product. From there,

the order would be passed along to shipping clerks who are responsible for packaging the order and shipping it out to the customer. A shipping clerk would then assign a tracking code for the shipment and mark the order's status as complete. Customer service representatives are be responsible for handling any questions or problems brought forth by a given customer. Operators only have access to customer, product, and order specific information in the on-line store management subsystem.

Operators are intermediate computer users with an understanding of how the Internet and e-commerce works. Operators will need to be trained on how to use the on-line store management subsystem for managing customer, product, and order specific data. They should know how to complete on-line forms and how log to into a secure section of a Web site. Depending on a given company's policies, different operators may perform different tasks, such as the tasks described about sales representatives and shipping clerks. The operator user group will be defined to handle many physical job positions that a given company may define.

2.1.3 Administrators

Administrators are the third and final level of users in the application. They use the application's on-line store management subsystem to manage all aspects of the application, such as application settings. Administrators have the same access capabilities that operators do. They are responsible for maintaining the application's configuration, payment types, shipping types, tax rates, and user security. Operators will be denied access to administrator only sections of the on-line store management subsystem. Overall, administrators will have complete control of what appears in the on-line store.

Administrators are intermediate computer users with an understanding of how the Internet and e-commerce works. Administrators will need to be trained on how to use the on-line store management subsystem. They should know how to complete on-line forms and how log into a secure section of a Web site. A basic understanding of group-based security is required when assigning permissions to users. Logically, a given company's policy will determine what physical users will be administrators and operators.

2.2 Design Protocols

2.2.1 Project Design

Empowered Storefront is a redistributable e-commerce application suite that uses Microsoft Visual Basic .NET as the programming platform for application logic and ASP.NET as the programming framework for the application's Web-based user interface. The application's data storage facilitates the use of a Microsoft SQL Server, relational database. The application executes in a transaction-based environment where data is dynamically accessed and managed through the application's Web interface. This project involves two major areas of Information Technology (IT): Web development and database.

2.2.1.1 Web Development

Empowered Storefront runs as a Web application in a client/server environment, specifically in a Web site hosting environment. It incorporates client-side Extensible Hypertext Markup Language (XHTML) syntax, cascading style sheets (CSS), and JavaScript, and it incorporates the use of ASP.NET using the Visual Basic .NET programming language. The application uses object-oriented programming standards that are widely accepted throughout the industry. Database access and utilities are

encapsulated inside classes to promote maintainability and reusability. Sixty percent of application's code consists of user-defined classes, so classes can be reused throughout the application and in future development projects. The user interface design and navigation scheme are template-based so they can be changed globally throughout the application from just a few files. The XHTML and CSS markup follow standards as outlined by the World Wide Web Consortium (W3C).

The application's interface is broken into three sub subsystems: an on-line store, a customer management system, and an on-line store management system. The on-line store subsystem serves as the public Web site where customers interact with the application to purchase orders on-line. The customer management subsystem allows a given customer to manage information specific to that customer. The on-line store management subsystem allows operators and administrators of the on-line store to control different elements of the store.

2.2.1.2 Database

The backend for the application consists of a relational database design using Microsoft SQL Server 2000. This application runs in a transaction-based environment where data is constantly being inserted, updated, and deleted. Depending on the company, the database might experience a high volume of traffic at a given time. An enterprise-level database management system was needed to meet this demand. For this reason, SQL Server was my choice for the database management system. Relational tables were created to eliminate the possibility of having redundant data and to enforce the rules of normalization. The database works with the application logic to perform data access operations. The user interface interacts with the application logic by creating

references to it, when needed, and any data access requests are passed to the database through the application logic.

2.2.2 Organizational Schema

The organization of Empowered Storefront is divided into four components: the physical database layout, object-based application logic, the application's domain, and the user interface. The physical database layout includes entity relationship diagrams for the database's tables and specific details about each table. The application logic section includes an object model of Empowered Storefront's objects that are organized into logical namespaces and the specific details about each object. The application's domain section includes a domain model of how the application would function in a real world scenario. The user interface section includes a use case diagram of how users will use the application.

2.2.3 Physical Database Layout

The physical database layout includes several entity relationship diagrams and descriptions for each diagram. The database diagrams are divided into several sub components: product relationships, customer relationships, order relationships, user relationships, and unrelated tables. Each diagram contains a figure for each table and its column name, data type, and the length of the data type. The physical database was designed following the rules of normalization. All variable length columns were designed to use Unicode for future globalization of the application.

2.2.4 Product Entity Relationships

The product entity relationships (See Figure 5.) are defined for tables that relate to the products' table.

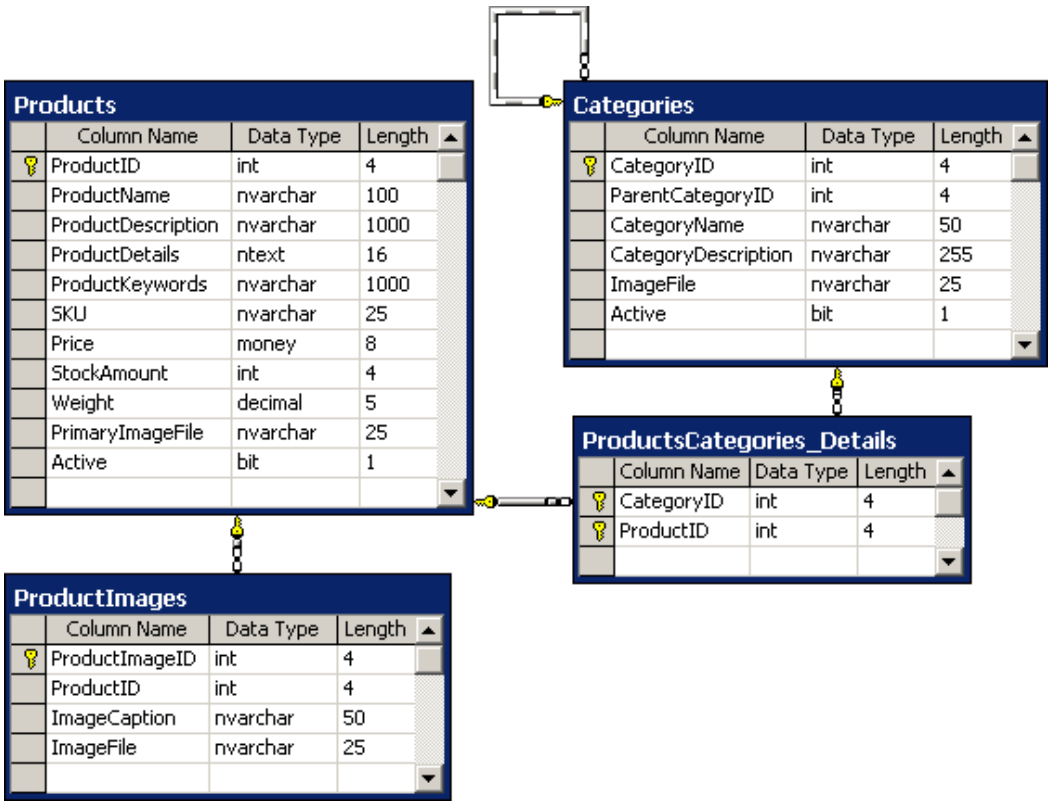

Figure 5. Product Entity Relationships Diagram

The products' table contains columns that relate specifically to product only data in order to satisfy the third normal form. Products are organized into categories and sub categories. In the categories' table, a primary key, CategoryID, is defined to identify a specific category, and a foreign key, ParentCategoryID, is defined to identify a parent category for a group of sub categories. Root-level categories are identified by a ParentCategoryID of zero. A single table relationship was defined on the CategoryID and ParentCategoryID columns. This keeps all categories contained in a single table and provides the capability of creating a category tree structure based on the ParentCategoryID column.

Since a situation may arise where one product could exist in multiple categories, an intermediate table, ProductsCategories_Details, was created to allow many products to

exist in many categories. This table contains CategoryID and ProductID columns that serve as foreign keys to the categories and products' tables respectively.

The products' table allows one to store an image for a given product. The physical image file is stored on a given Web server's hard drive for performance and efficiency. Only the physical file name is stored in the products table. This image file serves as the main image for a given product. One might want to store multiple images for a given product and the ProductImages table allows one store multiple images for a given product. Again, the physical file name is stored in this table and a foreign key, ProductID, is used to relate many product images to one product. This satisfies the first normal form by not including repeating groups of columns for product images.

2.2.5 Customer Entity Relationships

The customer entity relationships (See Figure 6.) are defined for tables that relate to the customers' table. The customers' table contains columns that relate specifically to customer only data in order to satisfy the third normal form. The customers' table contains specific contact information, such as an address, for a given customer. In e-commerce applications, there are billing and shipping addresses. This is setup so a given customer can have an order billed to one address and the products shipped to another address. The customers' table contains columns for the billing address while the shipping addresses' table contains columns for shipping addresses. One might want to use multiple shipping addresses when placing orders in the application. The shipping addresses' table contains a foreign key, CustomerID, which relates back to the customers' table. This allows a given customer to store many shipping addresses and this satisfies

the first normal form. For security reasons, the password column stores encrypted passwords that utilize a hashing algorithm.

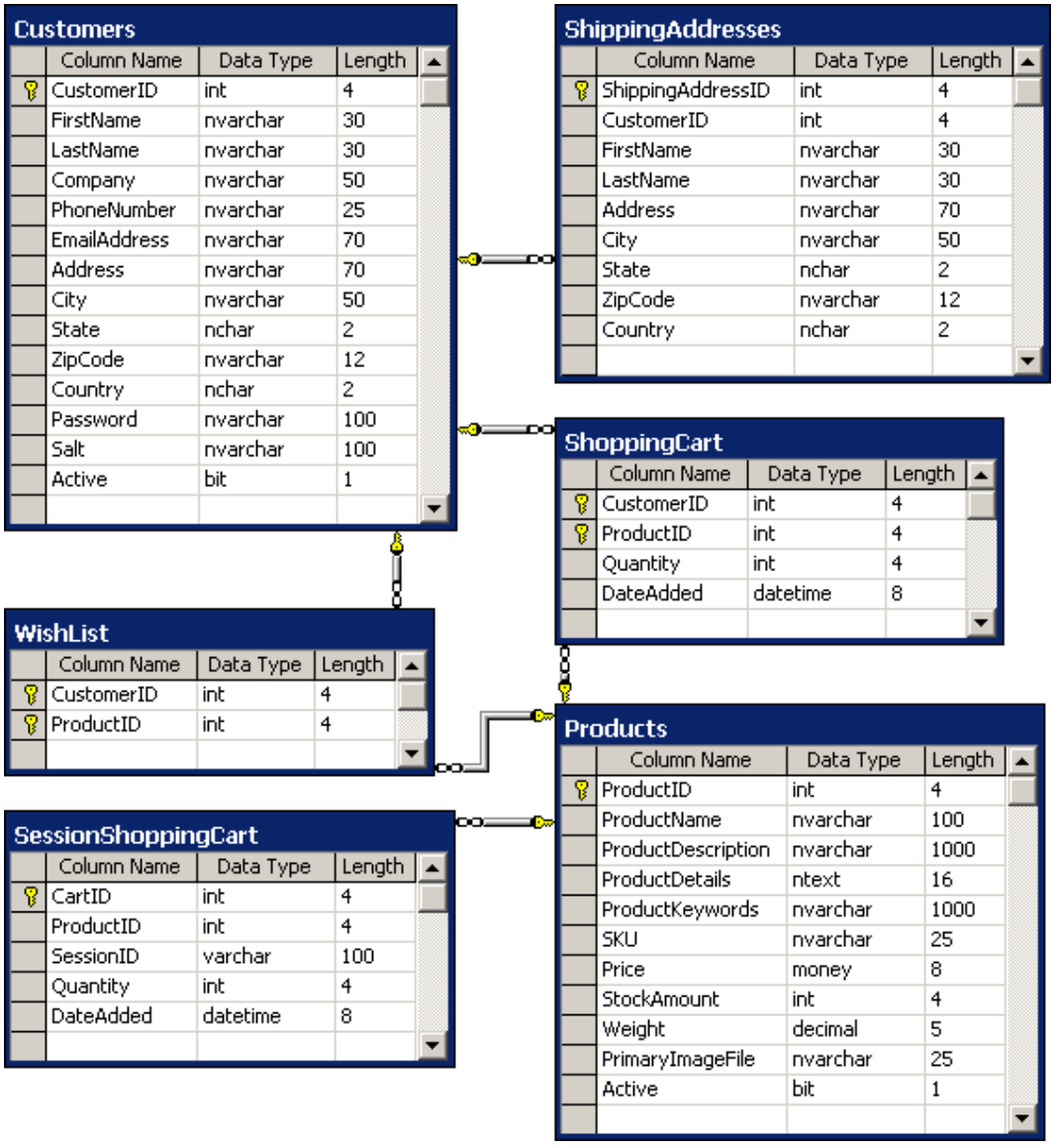

Figure 6. Customer Entity Relationships Diagram

The wish list table stores a list of products a given customer might want to purchase. This table contains CustomerID and ProductID columns that serve as foreign keys to the customers and products' tables respectively. This allows many customers to create a wish list of many products. The shopping cart table stores a list of products that

a given customer wishes to purchase. It follows the same format as the wish list table since it contains CustomerID and ProductID columns that serve as foreign keys to the customers and products' tables respectively. This intermediate table can then relate any customer or product specific information to the application. Rows are added to this table when a customer adds a product to the shopping cart. When an order is completed, those rows will be removed from the shopping cart table. A second shopping cart table was implemented to allow anonymous customers to add items to the shopping cart. Anonymous users are identified by a SessionID that is used to related items in the shopping cart back to them. Once an anonymous customer creates an account, the items in the session shopping cart will be copied to the main shopping cart. This design provides a solution for both anonymous and account-based customers.

2.2.6 Orders Entity Relationships

The order entity relationships (See Figure 7.) are defined for tables that relate to the orders' table. The orders' table contains columns that relate specifically to order only data in order to satisfy the third normal form. Once a given customer completes an order, the products in the shopping cart will be inserted into the Order_Details table. Since multiple customers can place multiple orders, the Order_Details' intermediate table is needed. The ProductID and OrderID serve as foreign keys to the products and orders' tables respectively. The Order_Details' table stores a product's name, its price, and its weight for a given order. This table is de-normalized so a history of orders will remain present if a customer or a product is removed.

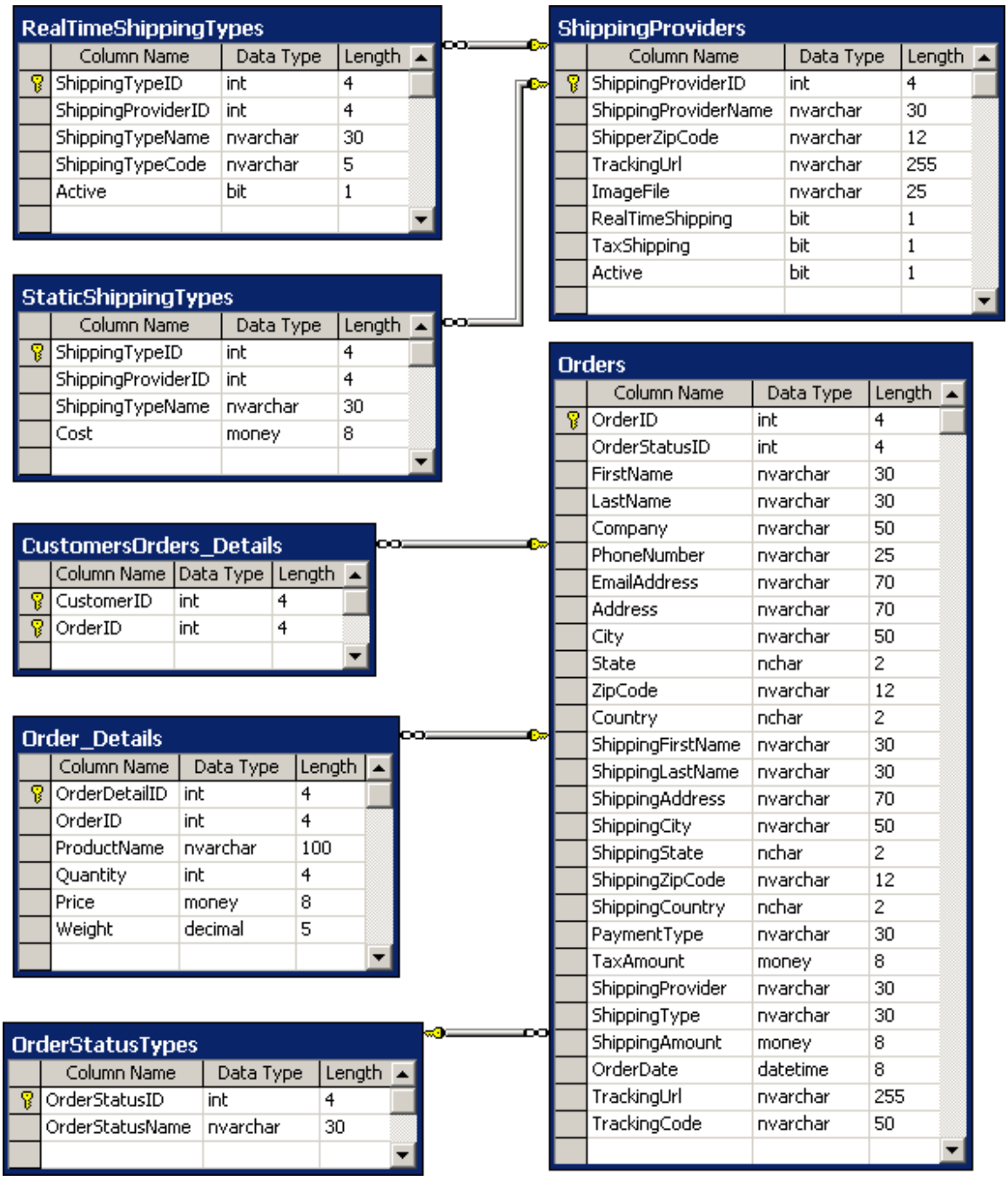

Figure 7. Order Entity Relationships Diagram

The shipping providers' table stores a list of shipping providers that a given company can use for product shipping. A shipping provider can be enabled or disabled as needed. The real time shipping types' table stores a list of real time shipping types provided by a given shipping provider. These are pre-defined shipping types and they

can be enabled or disabled for a given shipping provider. In this case, a shipping type and its cost will not need to be entered since this data will come directly from the shipping provider. Shipping information is passed to a given shipping provider's Web site and the shipping cost is returned to the application. The static shipping types' table allows one to create custom shipping methods, such as next day, overnight, or ground delivery. These are considered static shipping types since they are user-defined and can be changed by administrators as needed. The cost for that type of shipping can be defined in this table. A foreign key, ShippingProviderID, is used relate a given real time or static shipping type back to a shipping provider. For both real time and static shipping types, customers will be able to select the type of shipping they want during the checkout process. This design provides a scalable solution for both scenarios.

The order status types' table stores a list of status types for an order. For example, an order's status could be processing or completed. The orders' table uses an OrderStatusID to relate an order status type to an order. This table stores billing, shipping, and payment information for a given order. The order's table is de-normalized so a history of orders will remain present if a customer, shipping address, or payment type is removed. The orders' table also includes tracking URL and code columns where one can assign a tracking URL and code to an order so customers can track their orders from a given shipping provider. The orders' table also includes several columns for shipping information. Since there are multiple methods of shipping, static and real time shipping, the shipping specific data is stored in the orders' table for a given order to accommodate both scenarios. The shipping specific columns include: the shipping provider, the shipping type, and the shipping amount. A column for the tax amount is

included in the order's table to store the amount that was taxed for the order. Finally, the orders' table stores an order date that specifies when the order was placed.

The shipping providers and order status types' tables will include pre-defined data generated during the installation of the application. This will provide one with built-in data during the setup of the application.

2.2.7 User Entity Relationships

The user entity relationships (See Figure 8.) are defined for tables that relate to the users' table.

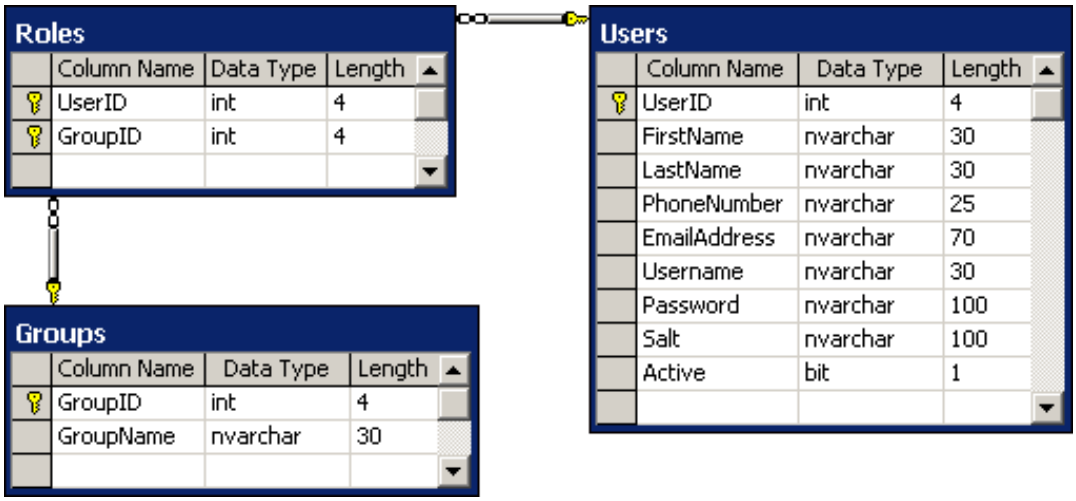

Figure 8. User Entity Relationships Diagram

The users' table contains columns that relate specifically to user only data in order to satisfy the third normal form. The users' table stores information about users and who can administer the application. For security reasons, the password column stores encrypted passwords that utilize a hashing algorithm.

The groups' table defines the user groups for the application. The default groups that will be added during the installation of the application are operators and administrators. User-based security is enforced at the code-level in the application based

on groups defined in this table. The application defines the permissions for a given user group.

The roles' table defines the user group for a given user. The UserID and GroupID serve as foreign keys to the users and groups' tables respectively. Users do not have the ability to define their own groups since permissions are defined at the code-level. By using these relationships, future expansion of the application can easily allow for more types of the user groups.

2.2.8 Unrelated Tables

The remaining database tables (See Figure 9.) do not have any relationships defined. The settings' table is a one row table that contains settings that are used throughout the application. In each column, one can change configuration settings for the application.

The tax table allows one to define how to tax customers during the checkout process. In essence, one will create "tax rules" depending on a customer's location. The location includes one's country, state, or zip code. A description for the tax rule and the tax rate, in the form of a percentage, is available in the tax rule. When a customer checks out, their location information will be queried against any tax rules. If their location information matches a tax rule, the tax will be applied to the order. Otherwise, no tax will be charged. This provides administrators with control over how customer orders get taxed.

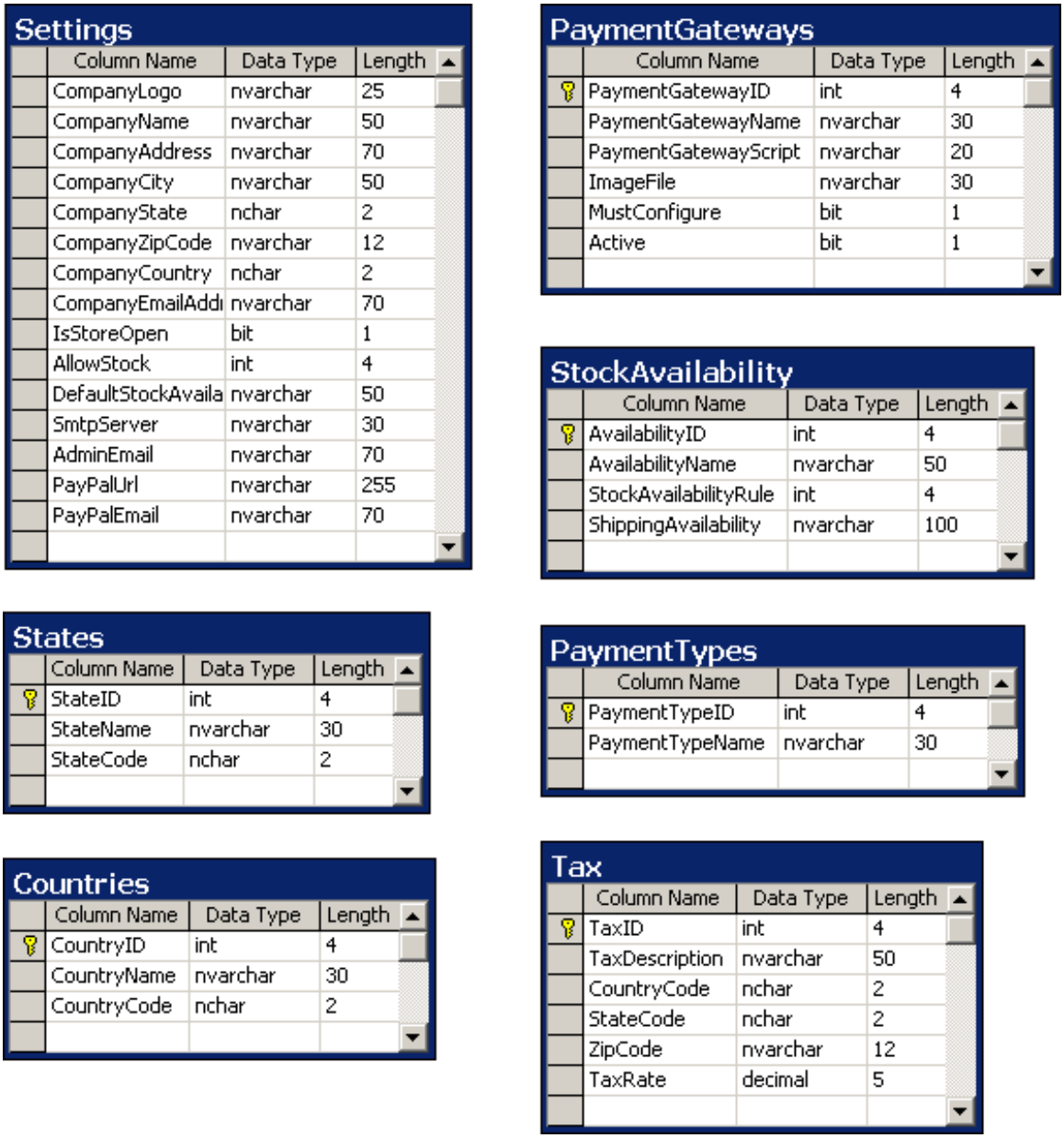

Figure 9. Unrelated Tables

The payment gateways' table defines all the payment gateways available in the application. During the installation of the application, this table will contain pre-defined payment gateway data that one can use to configure the application. The payment gateway script column points to a script in the on-line store that will execute when a customer purchases products. The active column can enable or disable the payment

gateway. The payment types' table allows one to define types of payment that can be accepted during the checkout process.

The stock availability table allows one to create an "availability rule" based on how many products are in stock for a given product. For example, one might create a rule where a given product is on back order when the stock amount drops below thirty. The rule column specifies the stock amount in which the rule should take affect. The name column specifies the stock availability when the rule goes into affect. The shipping availability column is used to let customers know when a product will ship from a given company's shipping facility.

The states and countries' tables are lookup tables that are used in various sections of the application. Since these values will not change, their data is inserted into a state or country column as is. One has the ability to specify which states and countries can be used in the application. By default, all states and countries are added to their respective tables during the installation of the application.

2.2.9 Object-based Application Logic

This section includes an object model diagram (See Figure 10.) that is organized into logical namespaces in the application. The descriptions of the objects are divided into each namespace. The row with a gray background signifies a namespace with the objects in each namespace listed beneath it.

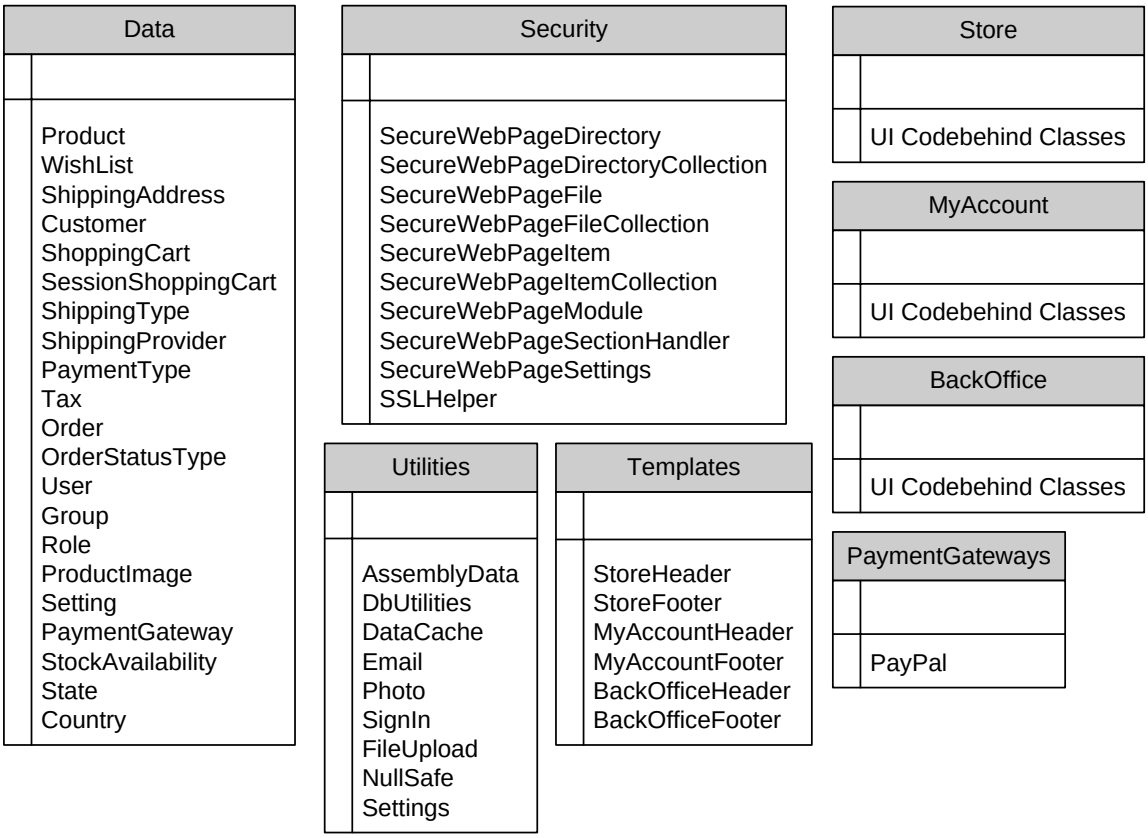

Figure 10. Object Model Diagram

2.2.10 Data Namespace

The data namespace objects (See Figure 11.) are derived from tables found in the database. Each object provides data access methods and properties for reading, inserting, updating, and deleting data in a given table. The data connection and access methods are encapsulated inside each class and are not exposed to the user interface. The object functions return pre-filled datasets when performing read operations. For a unique row of data for a given object, the data is populated into the object's properties so it is easily accessible to the user interface. Each method executes a stored procedure in the database when performing data access operations. Stored procedures are used to prevent SQL injection attacks that commonly plaque applications that embed SQL code into the application.

Object	Description
Product	This object provides methods for storing and retrieving product information.
WishList	This object provides methods to allow customers to store and retrieve their favorite products before purchasing them.
ShippingAddress	This object provides methods to allow customers to store and retrieve multiple shipping addresses.
Customer	This object provides methods for storing and retrieving customer information.
ShoppingCart	This object provides methods to allow customers to store and retrieve products in a virtual shopping cart.
SessionShoppingCart	This object provides methods to allow anonymous customers to store and retrieve products in a virtual shopping cart.
ShippingType	This object provides methods to allow users to define shipment types that are offered to customers during the checkout process.
ShippingProvider	This object provides methods to allow users to select the shipping providers they will use when they ship products to customers.
PaymentType	This object provides methods to allow users to define what types of payment are offered to customers during the checkout process.
Tax	This object provides methods for storing and retrieving tax information.
Order	This object provides methods for storing and retrieving order information.
OrderStatusType	This object provides methods to allow users to define a status for an order. For example, an order's status could be processing.
User	This object provides methods for storing and retrieving user information.
Group	This object provides methods for storing and retrieving group information.
Role	This object provides methods for storing and retrieving roles. Roles relate a user to a group.
ProductImage	This object provides methods for storing and retrieving multiple product images for a given product.
Setting	This object provides methods for storing and retrieving application configuration settings.
PaymentGateway	This object provides methods to allow users to decide what payment gateway should be used for the payment process.
StockAvailability	This object provides methods to allow users to define stock availability rules when customers view product information.
State	This object provides methods to allow users to define what states are allowed in the application.
Country	This object provides methods to allow users to define what countries are allowed in the application.

Figure 11. Data Namespace Objects

2.2.11 Utilities Namespace

The utilities namespace includes objects (See Figure 12.) that perform common operations in conjunction with the user interface. These utilities are encapsulated into classes for reusability throughout the application.

Object	Description
AssemblyData	This object provides methods and properties from reading information out of an assembly.
DbUtilities	This object provides shortcut methods for performing common pre-data access operations, such as string encoding and decoding.
DataCache	This object provides methods for updating XML files that are used for data caching.
Email	This object provides the capability of sending e-mails from the application.
Photo	This object provides methods to manipulate the sizing of photographs.
SignIn	This object provides methods for performing customer and user authentication and authorization.
FileUpload	This object provides methods for uploading files.
NullSafe	This object provides methods for converting data types and checking for bad data input that is a common problem when performing basic data type conversions.
Settings	This class provides methods for retrieving store settings.

Figure 12. Utilities Namespace Objects

2.2.12 Templates Namespace

The templates namespace (See Figure 13.) includes user control objects that are global templates for each section of the application. The three different sections of the application include: the on-line store subsystem, customer management subsystem, and on-line store management subsystem.

Object	Description
StoreHeader	The top portion of the on-line store template.
StoreFooter	The bottom portion of the on-line store template.
MyAccountHeader	The top portion of the customer management template.
MyAccountFooter	The bottom portion of the customer management template.
BackOfficeHeader	The top portion of the on-line store management template.
BackOfficeFooter	The bottom portion of the on-line store management template.

Figure 13. Templates Namespace Objects

2.2.13 Application Domain

This section includes a domain model (See Figure 14.) for how the application would function in a real world scenario. The domain model begins with the product specifications. These specifications contain information about the product, such as its description and price. The product specifications describe each item in the on-line store. The product catalog is the central repository where all products can be found. The product catalog is then used by the on-line store to display the products to customers. Customers will use the on-line store as an interface to view items in the product catalog. In reality, a warehouse will stock the items found in the on-line store so they can be shipped to customers. A sales line item records the sale of each item that a customer wishes to purchase. Then, each sales line item is entered into a shopping cart. During the checkout process, each sales line item's details are transferred to a sale that contains information about a given customer and the sale, such as the customer's payment information and the sale's date and time. The sale is initiated by the customer and completed when the customer pays for the sale with a given type of payment, such as a credit card. The sale is recorded in the warehouse and a shipping clerk completes the sale by shipping the items to the customer. A sales clerk can generate reports and analyze sales data from the warehouse. Finally, a system administrator monitors the activity of

the on-line store for security purposes and manages the users of the on-line store. This describes a possible scenario of how the application would work in the real world.

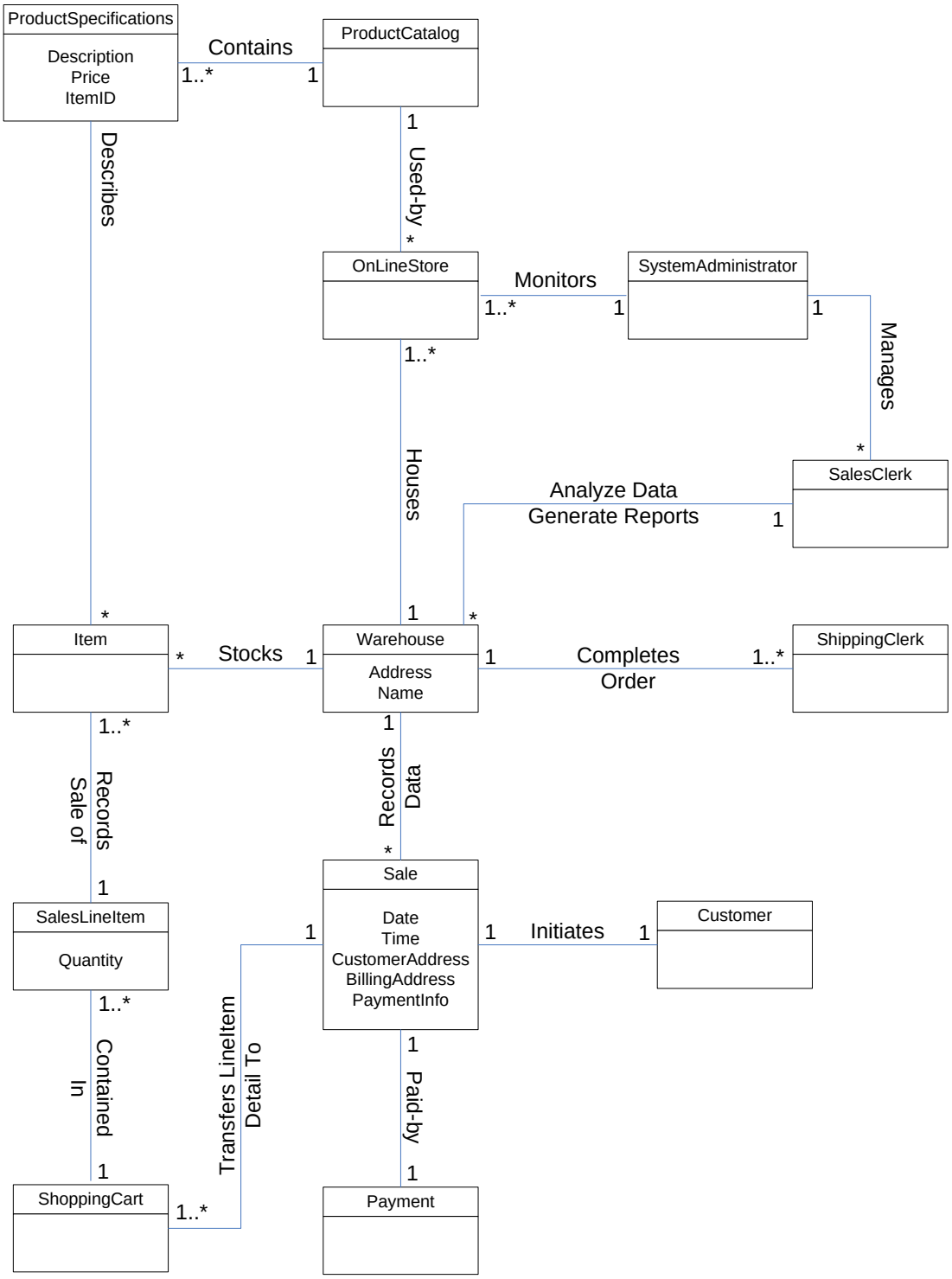

Figure 14. Application Domain Model

2.2.14 User Interface

This section includes a use case diagram (See Figure 15.) that shows how users interact with the application.

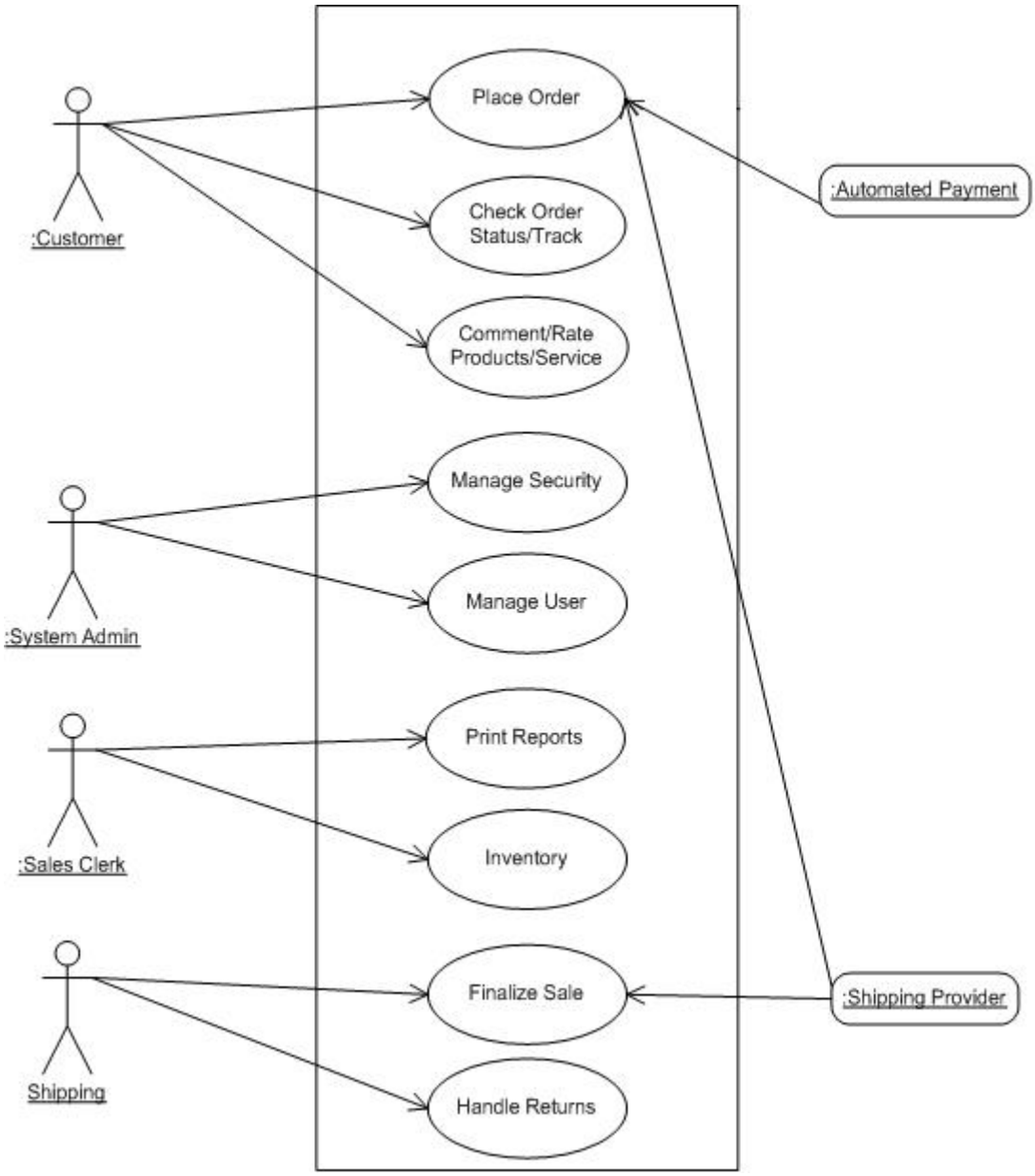

Figure 15. Use Case Diagram

Figure 15. describes the interaction between users and the application. The circled items represent elements of the application and the figures to the left represent users. The

figures to the right represent external applications that the application will use to perform various tasks. External applications are connected to the application by the use of Web services. Rather than implement these services, the application uses the services since they already exist and have been thoroughly tested. This promotes good reusability and efficiency in the application. The operator user can encompass many physical positions, such as sales clerks, shipping clerks, and customer service representatives. The descriptions of the sales and shipping clerks below encompass the role of the operator user group.

2.2.15 Customers

Customers use the application's on-line store interface as a means to browse items found in the product catalog. Details of each product will help them decide what products to buy. Once a customer decides to buy a product, the product is added to a virtual shopping cart. This process continues until the customer is ready to checkout. Once the customer is ready checkout, the customer will enter their contact, billing, and shipping address information. After this is complete, the customer will select how they want the products to be shipped. Then, the customer will select a method of payment and enter the payment information. After a customer reviews the details of the order, the customer will then place the order. While the order is being processed, the customer can use the application's customer management subsystem to view the order's status and track the shipment once the order has been shipped.

2.2.16 Sales Clerk

Sales clerks use the application's administrative subsystem to manage orders and the product catalog. Once an order is placed, a sales clerk prepares the details of an

order, and sends the information to the shipping department so an order can be shipped. A sales clerk can use the administrative subsystem to periodically manage products in the application. The sales clerk can use the subsystem's functions to keep a company's product catalog updated.

2.2.17 Shipping Clerk

The shipping clerk receives notifications through the application when an order is ready for shipping. The shipping clerk will physically package the order and then ship it using one of the shipping providers defined in the application. Once the order is shipped, the shipping clerk can use the application's administrative subsystem to assign a tracking code to the order and update the order's status to complete.

2.2.18 System Administrator

The system administrator will use the application's administrative subsystem to manage the application's configuration, payment types, shipping types, tax rates, and users. The administrator will have access to all sections of administrative subsystem where as operators will only have access to order, product, and customer sections of the administrative subsystem. The administrator will handle application security by assigning authentication credentials to users. Authorization permission for different groups will be determined at the code-level for security reasons.

2.2.19 External Applications

The automated payment Web service automatically charges credit cards for a given customer. This serves as the payment gateway for the application. The amount will then be deposited to an account created by the company that owns the on-line store. The shipping provider Web service allows the application to calculate the shipping cost

for a customer during the checkout process. The products' weight and the customer's zip code are passed to the shipping provider and the service returns a shipping price that will be used by the application to calculate the total for the order. The shipping clerk can also be used to assign shipping tracking codes to an order. Once a shipping clerk ships an order, the shipping provider will generate a tracking code. The shipping clerk can then enter the code for the order so customers can track the order.

2.2.20 User Interface Design

The design of the user interface meets the goals of two types of users: the customers and users of the on-line store management subsystem. Customers use the on-line store and customer management interfaces to perform order operations in the application. Operators and administrators use the application's on-line store management subsystem to perform operations within the application.

2.2.21 Customer Interfaces

The design of the on-line store interface (See Figure 16.) follows a look commonly found in all e-commerce applications. The page layout fills the entire browser screen to utilize an efficient use of space. Links to the main sections of the on-line store interface appear as tabs across the top of the page. Likewise, the same links appear at the bottom of page to promote good usability for navigating through the interface. A quick search option appears on all pages in the application which allows customers to easily search for a product. A categories menu appears to the left that displays the application's root categories. When a user clicks on a category, iconic sub categories are displayed. The images allow customers to find what they are looking for quickly. Products are displayed in tables with all the details for a customer to view.

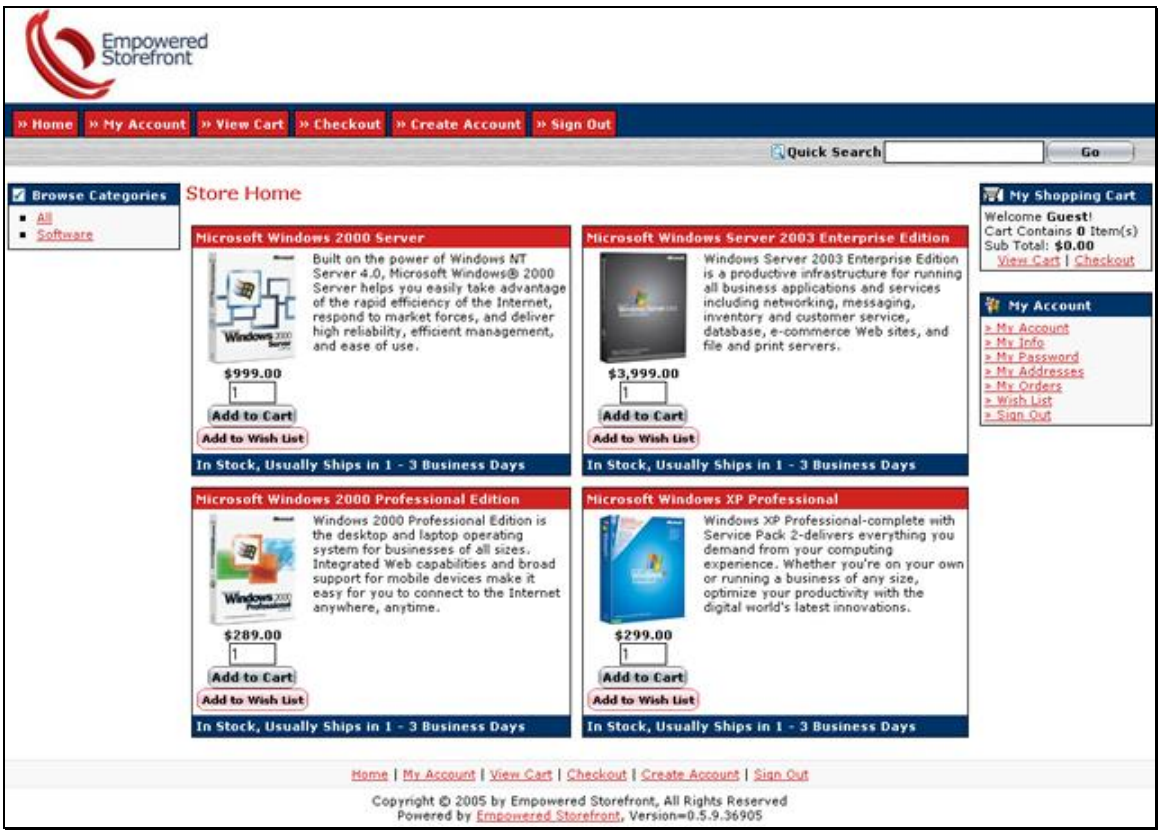

Figure 16. On-line Store Interface

On each product page, customers can easily specify the quantity for a given product to add to the shopping cart. Page headers and icons are used to inform customers where they are in the application.

Page specific content appears in the center portion of the interface. The page layout is consistent throughout the application to promote good usability. A section to the right provides quick details about the shopping cart, such as the number of the items in the cart and its sub total. This allows customers to get a quick view of the shopping cart's details. The checkout process has icons that show the progress of the checkout from start to finish. The start stage is when a customer first decides to checkout and finish stage is when the customer places an order. The customer management interface (See Figure 17.) follows the same page layout as the on-line store.

Figure 17. Customer Management Interface

Here customers are able to track orders, update their account information, and manage their shipping addresses. When a customer enters data into an on-line form, hints about what type of data and its format are provided to the customer when an input error occurs.

2.2.22 Operator and Administrator Interface

The design of the operator and administrator interface (See Figure 18.) follows the same design and navigation scheme of the customer interface. These users use the application's on-line store management subsystem to manage the on-line store. Tabs will appear across the top of the page to allow these users to easily navigate to the main sections. Page headers and icons are used to inform these users where they are in the application. When these users enter data into an on-line form, hints about what type of data and its format will be provided to these users when an input error occurs.

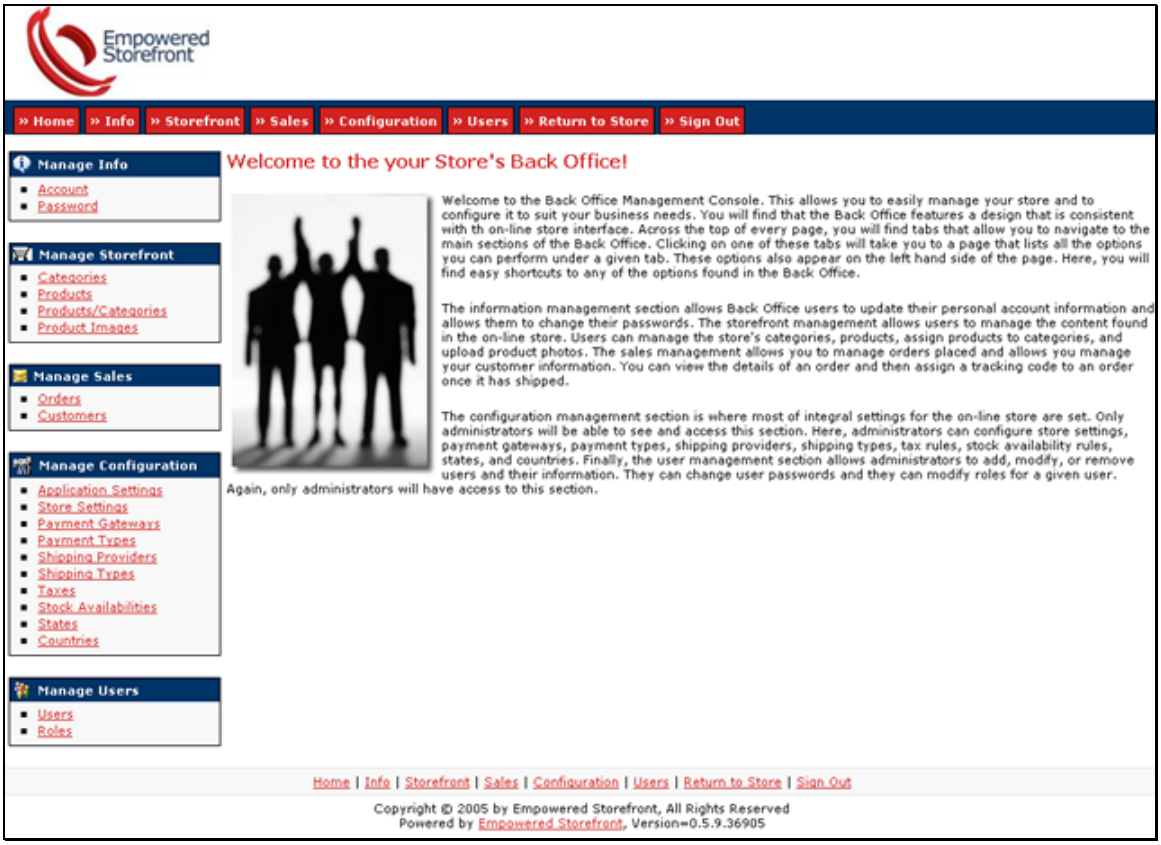

Figure 18. On-line Store Management Interface

2.2.23 Color Scheme

The application uses a default color scheme that is configured through an external CSS file. When the application is distributed to companies, it will use this default color scheme. This color scheme was chosen because it is appealing to the eye and it allows users to easily read and comprehend elements found in the interface. Companies can easily change the color scheme by modifying the global CSS file. The colors can be changed for all three interfaces found in the application.

3. Deliverables

Due to the complexity of this project, realistic deliverables were defined during the design phase. The deliverables are as follows:

1. A redistributable e-commerce application suite that can be incorporated in a Web hosting provider's hosting packages for small businesses.

2. An application that will allow small businesses to publish their product catalogs globally across the Internet, a centralized application for performing business transactions, an application that is easy for small businesses and customers to use, and an application that is easy to install and maintain for Web hosting providers.

3. A Web-based user interface that uses Microsoft Visual Basic .NET as the programming platform for application logic and ASP.NET as the programming framework.

4. An enterprise-level database that can handle the traffic of an on-line transaction application.

5. Secure logon for customers, operators, and administrators authenticated against credentials stored in the database.

6. Code-level security for the operator and administrator user groups.

7. Customers will be able to do the following:

    a. Browse for products to buy
    b. Search for products
    c. Add products to the shopping cart
    d. Select a type of shipping
    e. Select a type of payment
    f. Place orders
    g. Track an order's status
    h. Manage orders
    i. Manage account information
    j. Manage wish lists

8. Operators will be able to do the following:

    a. Manage products
    b. Manage categories
    c. Manage orders
    d. Manage shipping tracking codes
    e. Manage customer account information

9. Administrators will be able to do the following:

    a. Performs tasks that operators can perform
    b. Manage user accounts and roles
    c. Manage allowed states and countries
    d. Manage types of payment
    e. Manage types of shipping and shipping providers
    f. Manage stock availability rules
    g. Manage tax availability rules
    h. Manage payment gateways
    i. Manage configuration settings

4. Design and Development

The next sections of this report describe the project's budget including the hardware and software cost and its timeline.

4.1 Budget

The budget for the entire project is shown in the table below (See Figure 19). Since the software that I used was purchased or obtained through academic licenses, the budget below lists real world figures if one were to purchase regular licenses for each piece of software. The hardware costs are based current values for the hardware contained in my development environment.

Hardware	Cost
Workstation Computer for the Development Environment	$ 800.00
DiscountASP.NET Web Hosting for One Year	90.00
Microsoft SQL 2000 Database for One Year	120.00
Software	**Cost**
Microsoft Visual Studio .NET 2003 Professional Edition	799.00
Microsoft SQL Server 2000 Developer Edition	49.95
Macromedia Fireworks MX 2004	254.00
GlobalSCAPE CuteFTP Professional 6.0	59.99
Total Cost:	**2,162.94**
Actual Cost for Purpose of this Project:	**$ 210.00**

Figure 19. Project Budget (13, 10, 9, 6, 4, 3)

4.2 Timeline

Listed below (See Figure 20.) is a timeline of tasks and milestones that occurred during the development of this project. Milestones represent the black diamond and the blue bar represents tasks completed in the Gant chart below. A detailed discussion of the tasks and milestones for each Senior Design sequence follows the timeline listed below.

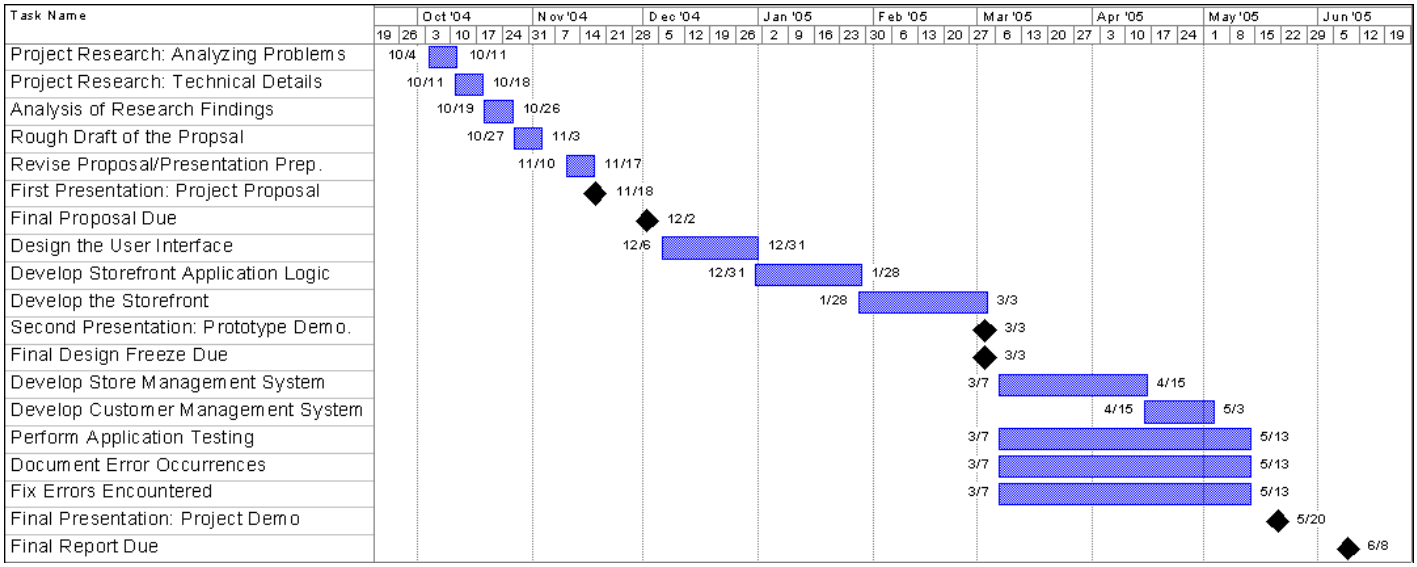

Figure 20. Project Timeline

4.2.1 Senior Design I Tasks and Milestones

The following tasks and milestones were accomplished in Senior Design I:

- Began and completed the requirements gathering phase of the project
- Analyzed common features of existing e-commerce applications
- Performed research on problems with existing e-commerce applications
- Performed research on the e-commerce the needs of small businesses and Web hosting providers
- Performed research on the technical details of e-commerce applications
- Performed an analysis of my research findings

- Developed a draft the project's proposal detailing its requirements and its initial design
- Revised the initial proposal and prepared the final proposal
- Prepared the presentation of the proposal
- Milestone: Gave the first presentation of the project's proposal
- Milestone: Completed the final proposal

4.2.2 Senior Design II Tasks and Milestones

The following tasks and milestones were accomplished in Senior Design II:

- Began and completed the design phase of the project
- Designed the database model and the physical database
- Designed the user interface through templates and a global style sheet
- Designed the application's classes and organized them into namespaces
- Developed all the classes and stored procedures to be used by the application
- Developed the on-line store application
- Developed a draft of the project's design freeze detailing its design and structure
- Developed a working prototype by the end of Senior Design II
- Revised the initial design freeze and prepared the final design freeze
- Prepared the presentation of the design freeze
- Milestone: Gave the second presentation of the project's design freeze
- Milestone: Completed the final design freeze

4.2.3 Senior Design III Tasks and Milestones

The following tasks and milestones were accomplished in Senior Design III:

- Began and completed the implementation, testing, and deployment phases of the project
- Developed the on-line store management application
- Developed the customer management application
- Performed iterative testing of small subsets of the completed project
- Documented errors that occurred during the testing phase
- Fixed errors that occurred during the testing phase
- Developed a draft of the project's final report
- Revised the initial document and prepared the final report
- Prepared the presentation of the completed project
- Milestone: Gave the final presentation of the completed project
- Milestone: Completed the final report

5. Proof of Design

This section discusses how the deliverables, the objectives of the project, were met and what problems were encountered during the project's development.

5.1 Redistributable E-Commerce Application Suite

Empowered Storefront contains three applications that form an application suite. The design of the application is generic and therefore, it is not designed for a particular company or industry. The completed project features an installation application that allows Web hosting providers to copy the application files to an existing Web site by the means of a Wizard. The Wizard creates the necessary directory and file structure for the application and it sets up the root directory of the application as an application directory. Once the application files are created, an install directory off the application's root directory contains two SQL script files that are used to setup the database. Web site hosting providers can simply open the Database.sql file in SQL Server's Query Analyzer and execute the contents of the file. This file creates the database's tables, store procedures, relationships, and constraints. The DefaultData.sql file is used to insert default application data into the database's tables. Web site hosting providers can add the database connection string in the Web.config file off the application's root. From here, they can log into the on-line store management application to complete the setup process by assigning configuration settings and application users. This allows Web hosting providers to easily install and maintain these applications in a distributed environment.

Small businesses will then be able to access the on-line store management application to setup and configure their on-line store. At this point, small businesses will have complete control of their on-line stores. The on-line store management application

allows small businesses to publish their product catalogs to the on-line store. It also provides them with the ability to save all their product information in a centralized location. Customers can access the on-line store globally and small businesses have a centralized application for performing business transactions. The consistent "look and feel" incorporated into the application's templates and its simple user interface makes it easy for customers and small business to use the application.

5.2 Application Design

Each user interface in the three applications is Web-based, which can be accessed from any in the world. The completed application uses Microsoft Visual Basic .NET as the programming platform for application logic and ASP.NET as the programming framework. The completed application uses the enterprise edition of SQL Server as the database management system. This edition can handle the expected high volume of traffic of this on-line transaction application in a distributed Web hosting environment. The application can be configured to use Secure Socket Layer (SSL) certificates that provide secure connections on certain pages in the application. Those secure pages are pre-configured in the Web.config during the installation of the application. Once a SSL certificate has been installed by the Web hosting provider, the secure pages can be enabled in the Web.config file. A centralized logon page is available that handles the authentication and authorization process for all three sub applications (See Figure 21.).

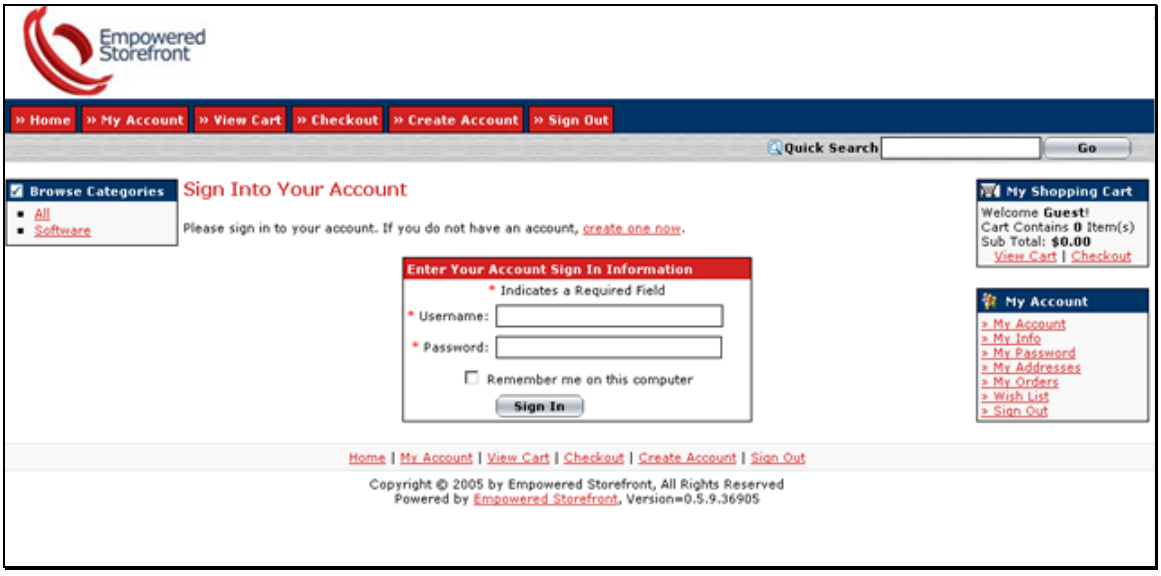

Figure 21. Global Logon Page

Passwords use a one-way encryption algorithm, SSH1, and the concept of "salting" a password when they are stored in the database. Code-level security is built into all secure pages in the application and authorization to resources is handled through user roles (See Figure 22.). Roles are assigned to users when they attempt to authenticate.

```
' This event fires when a given page loads.
Private Sub Page_Load(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles MyBase.Load

      ' Ensure the user is authenticated
      If Not User.Identity.IsAuthenticated Then
            Response.Redirect("../Store/SignIn.aspx")
      End If

      ' Ensure the user is authorized
      If Not User.IsInRole("Administrators") And _
            Not User.IsInRole("Operators") Then
            Response.Redirect("../Errors/403.aspx")
      End If
End Sub
```

Figure 22. Code-level Security Snippet

5.3 Customer Actions

This section shows how the application allows customers to perform their actions as defined in the deliverables.

5.3.1 Browsing for Products

The first page customers see is the home page for the on-line store (See Figure 23.). It shows four randomly selected products in a table. Customers can view detailed information about a product, or a add product to the shopping cart or wish list.

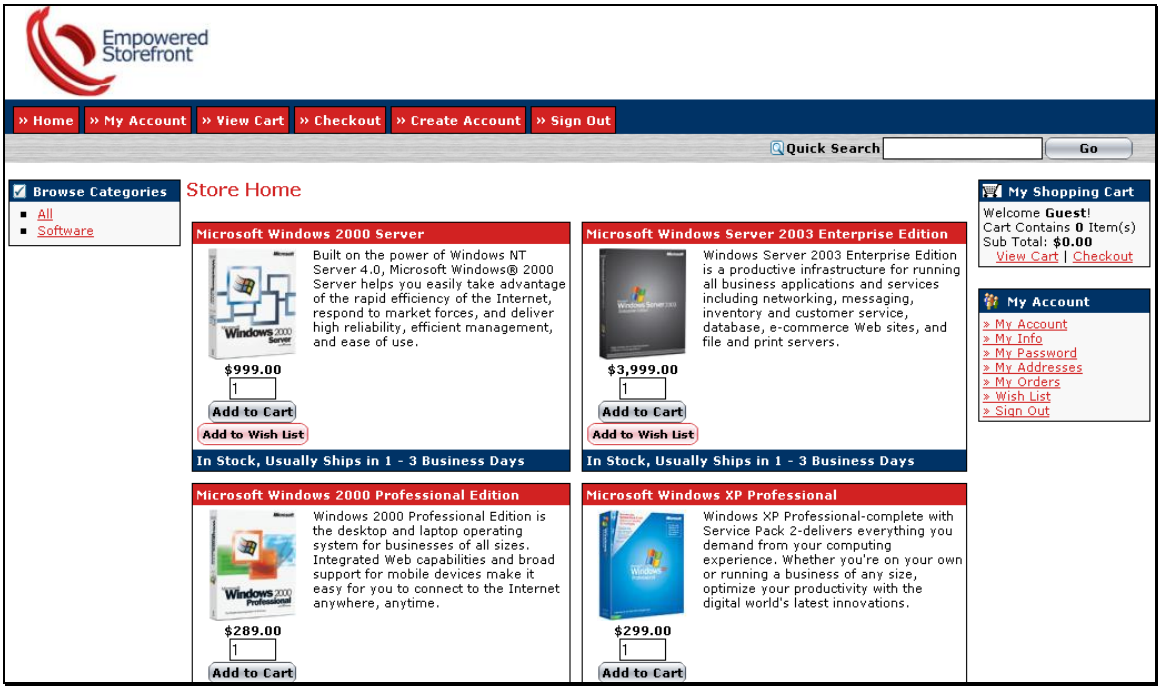

Figure 23. On-line Store Home Page

At this point, customers can browse for products using the "Browse Categories" sub menu to the left. All products are organized into categories so customers can easily navigate to products they wish to purchase. This menu lists root-level categories that customers can easily identify. Empowered Storefront allows users to create a category tree structure in the form of parent-child category relationships. When a customer clicks

on a root-level category link to the left, it shows a page that lists all the sub categories in a given parent category (See Figure 24.)

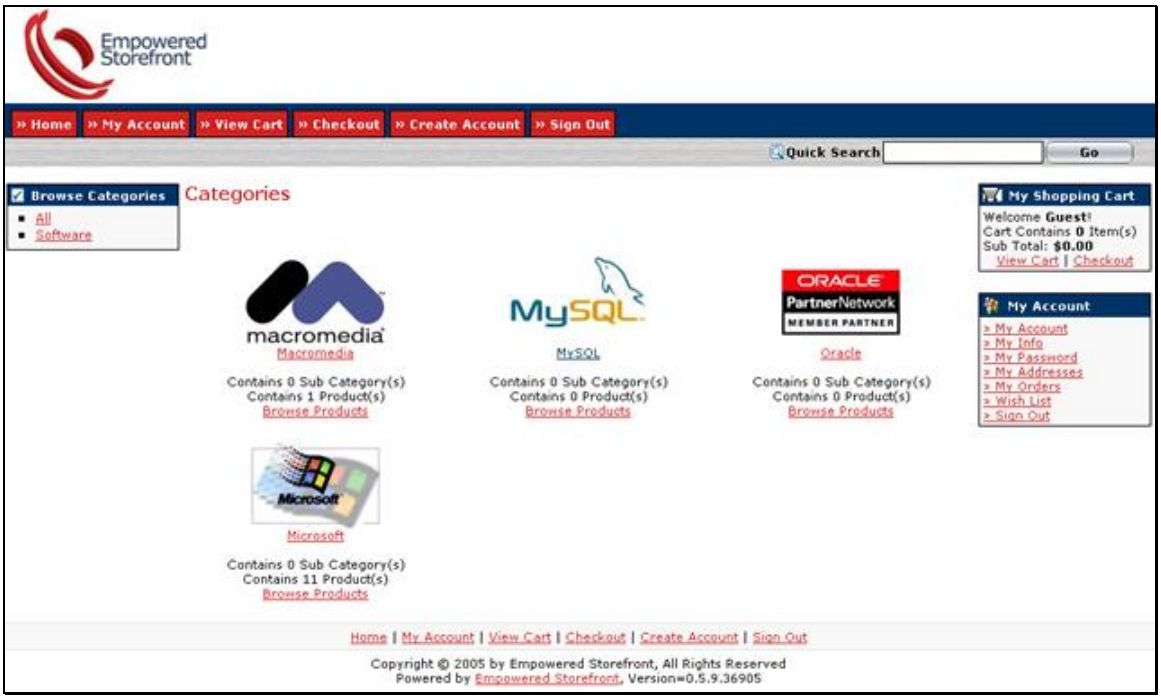

Figure 24. Sub Category Listing

Each sub category contains an optional image that allows customers to easily relate to a sub category. Under the name of the sub category, a display tells customers how many sub categories and products exist in the shown sub category. Customers can click on the image or sub category name to view the next set of sub categories or they can click on "Browse Products" link to see a listing of products in a given sub category. After clicking on the sub category's image or name, if no sub categories exist in that category, the customer is automatically redirected to a page that lists all the products found in that category (See Figure 25.). This design allows customers to easily browse for products.

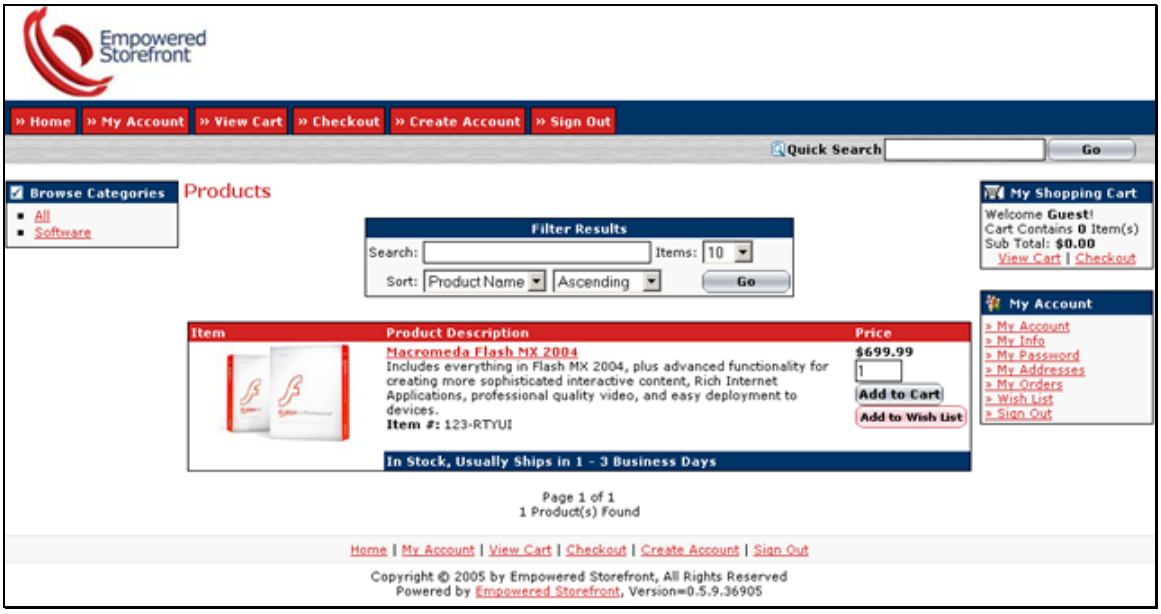

Figure 25. Product Listing Page

5.3.2 Searching for Products

From any page in the on-line store, customers can type search terms into the "Quick Search" form found in the upper-right portion of the page (See Figure 26).

Figure 26. Quick Search Form

After a customer types in a search term, the customer must click the “Go” button. The search results are retrieved from the database and the products matching the search terms are shown as follows (See Figure 27):

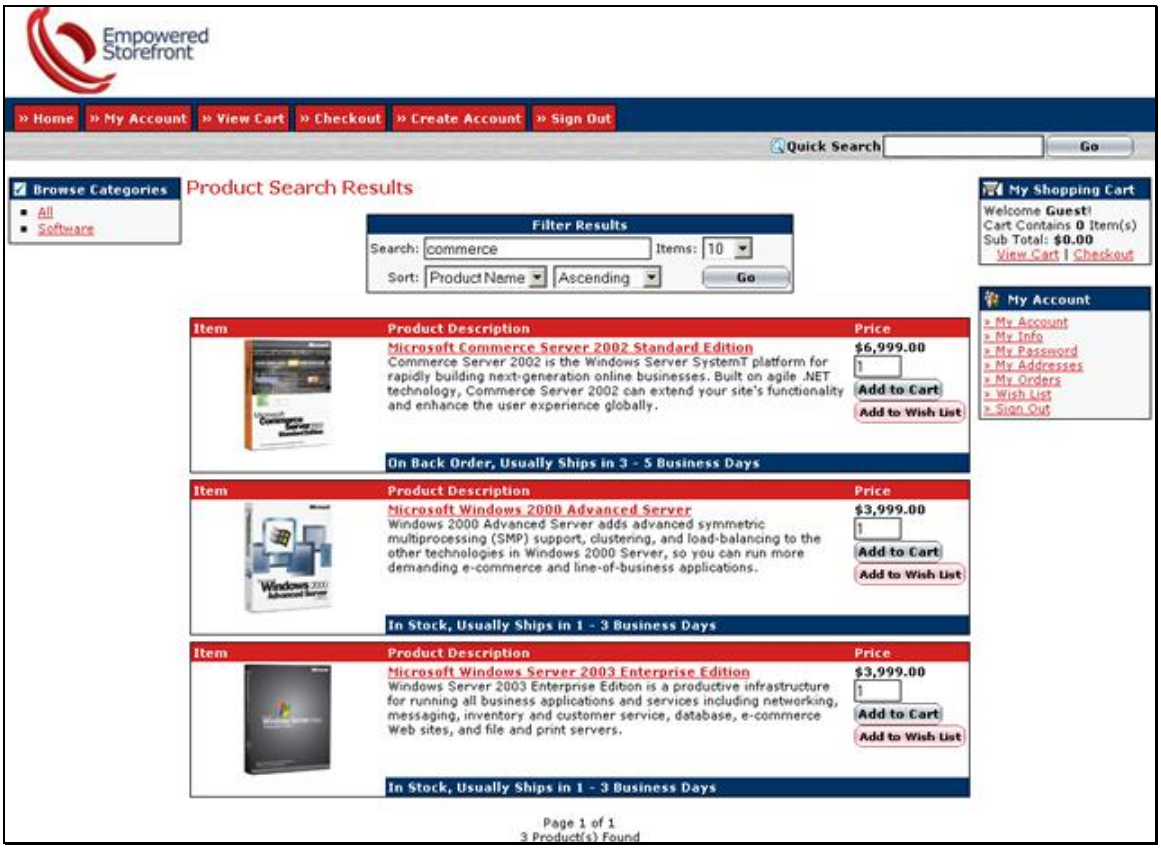

Figure 27. Search Results

The filter results table allows customers to sort products by type and in a certain order. Customers can also set how many records per page they want. By default, only ten records per page are shown. The display at the bottom of the page show how many pages there are, how many products were found in the search, and has navigation links that allow customers to easily navigate through all the pages in the search results. The product listing page, when browsing by category, also contains this functionality. This feature is consistently used throughout the application suite.

5.3.3 Adding products to the Shopping Cart

As shown from several product pages in the on-line store, customers can specify a quantity for a given product they want to add to their shopping carts (See Figure 28).

Figure 28. Adding a product to the Shopping Cart

After the quantity is specified, the customer must click the "Add to Cart" button. This action adds the items to the shopping cart in the database and redirects the customer to the "View Cart" page (See Figure 29.).

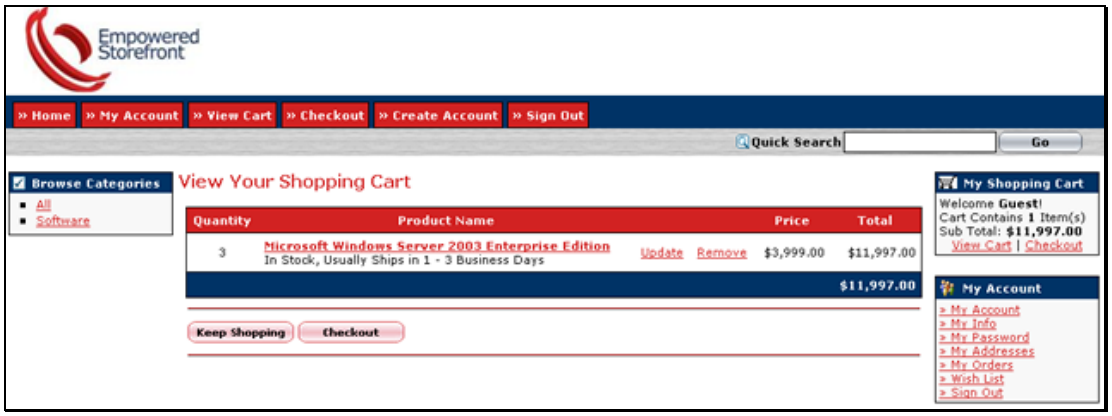

Figure 29. Viewing the Shopping Cart

5.3.4 Checking Out

When a customer is ready to check out, the customer can click on the "Checkout" button from the "View Cart" page or the "Checkout" tab at the top of the page as shown in previous screens. The customer will be redirected to a page where the customer can select the check out method (See Figure 30.).

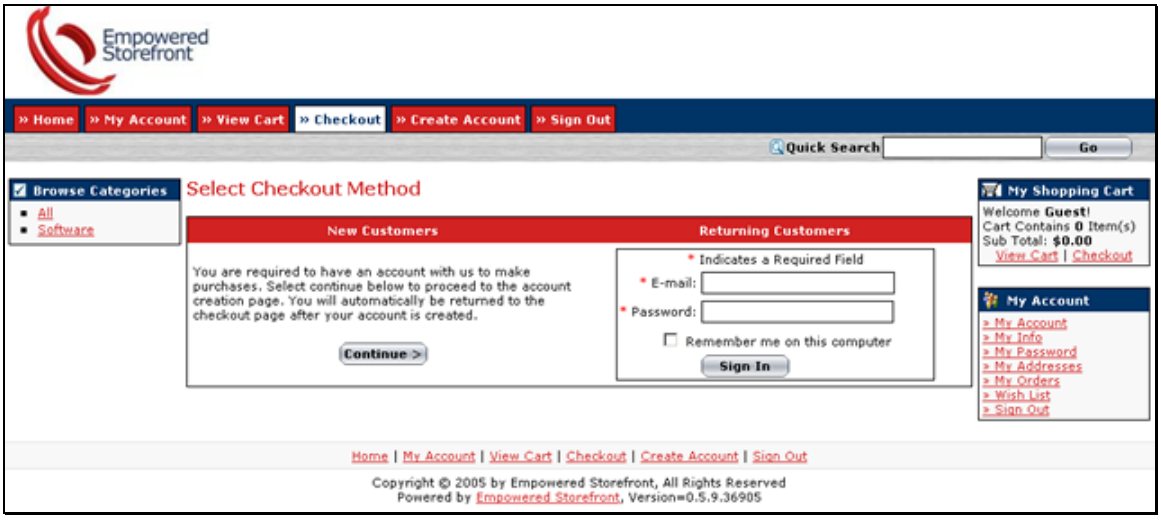

Figure 30. Select Checkout Method

New customers can simply click on "Continue" or existing customers can sign into the application. Upon completing either check out method, the customer will be redirected to the checkout page. If no items exist in the shopping cart, the customer will be returned to the shopping cart page where it will state there are no items in the shopping cart. The first page in the check out process will ask the customer to complete billing information and enter a type of payment (See Figure 31.). A progress indicator is shown at the top of the page that shows customers where they are in the check out process. The shopping cart icon moves forward as the customer moves through the check out process.

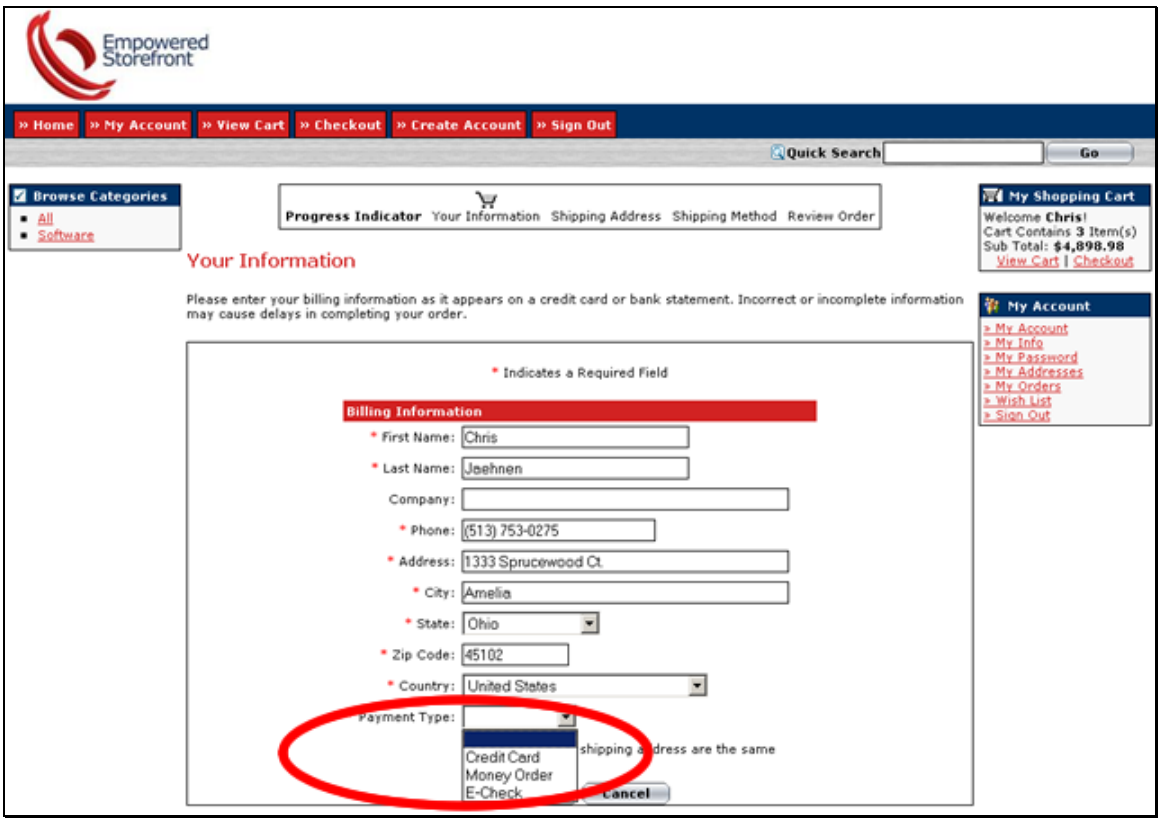

Figure 31. Billing Information and Payment Type

After clicking on "Continue," the customer is taken to a page where the customer can enter a shipping address (See Figure 32.). The customer will have the option to enter a new address or use a stored shipping address for the current order. A table of shipping addresses will be shown if the customer already has shipping addresses stored in the database. The customer can click on the "New Address" button to add a new address. If no shipping addresses are found, a blank shipping address form is shown.

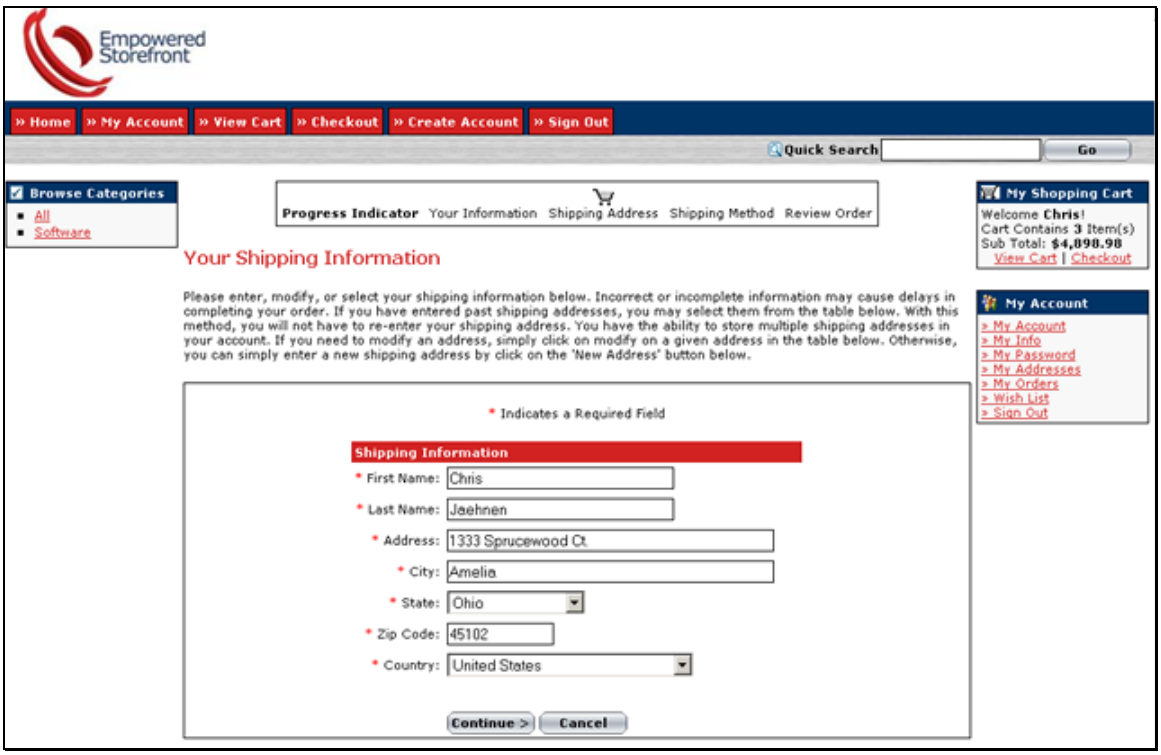

Figure 32. Shipping Address Form

After clicking again on the “Continue” button, the customer is taken to a page that allows the customer to select a shipping type and a delivery type, residential or commercial, (See Figure 33.).

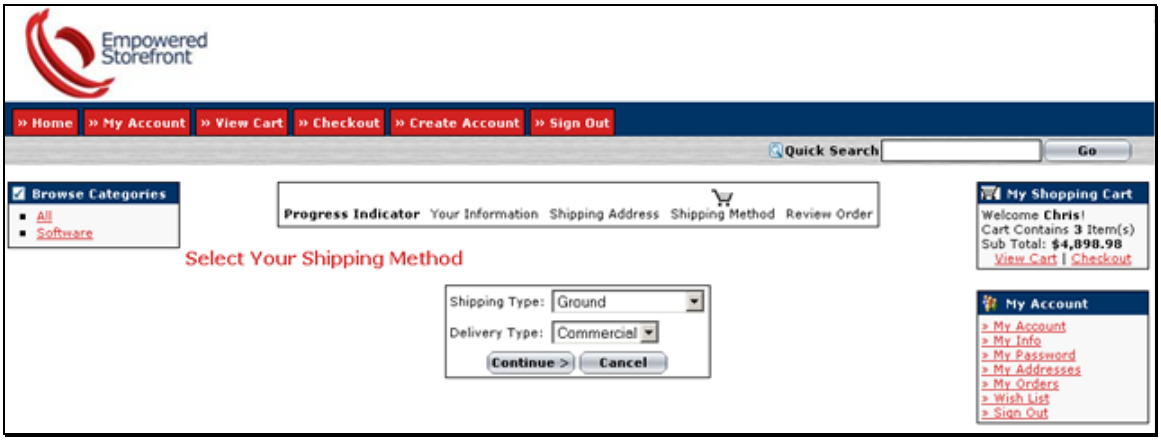

Figure 33. Shipping Method Form

After clicking on the “Continue” button a third time, the customer is taken to a page where the customer can review the details of the order. The order review page contains the following: order company information, billing information, shipping information, products in the order, and the order details that outlines the charges for the order (See Figure 34.).

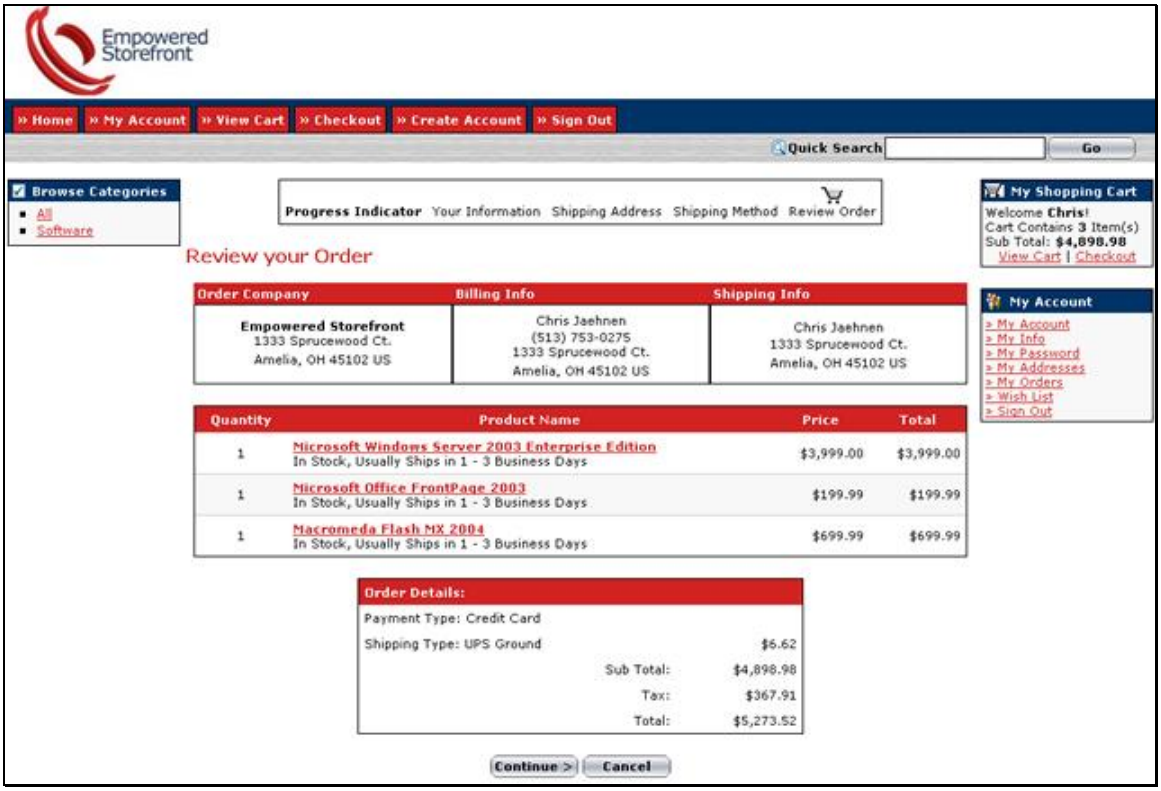

Figure 34. Review Order Page

After clicking on the “Continue” button a fourth time, the order is placed and the customer is redirected to page where the customer can pay for the order. The customer will be taken to a page that handles the enabled payment gateway. The operations on a given payment gateway page will vary on the requirements of the payment gateway. Once the payment is received, the customer’s order will get shipped and the order process is complete.

5.3.5 Manage Orders

After an order has been placed, a customer can sign into the customer management application, as shown below (See Figure 35.), and click on "My Orders."

Figure 35. Customer Management Application Home Page

The customer will be taken to a page where the customer can select the type of order: open, completed, or canceled (See Figure 36).

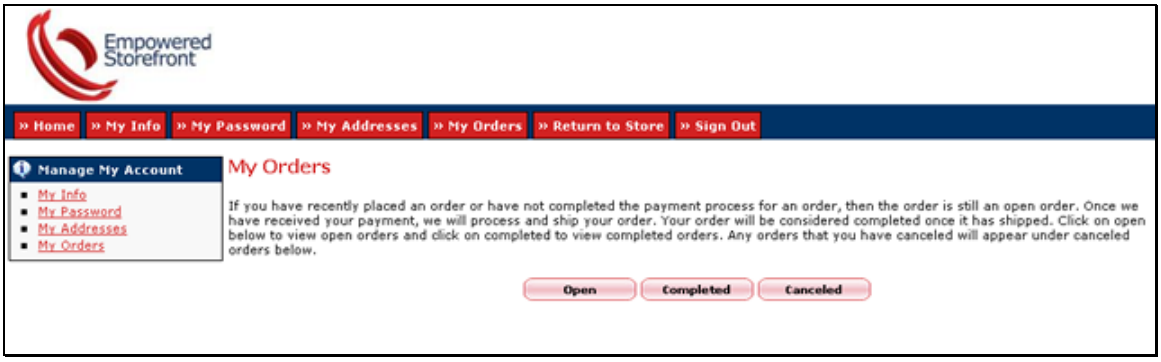

Figure 36. My Orders Page

The open orders page shows orders that have been placed but still need payment (See Figure 37.).

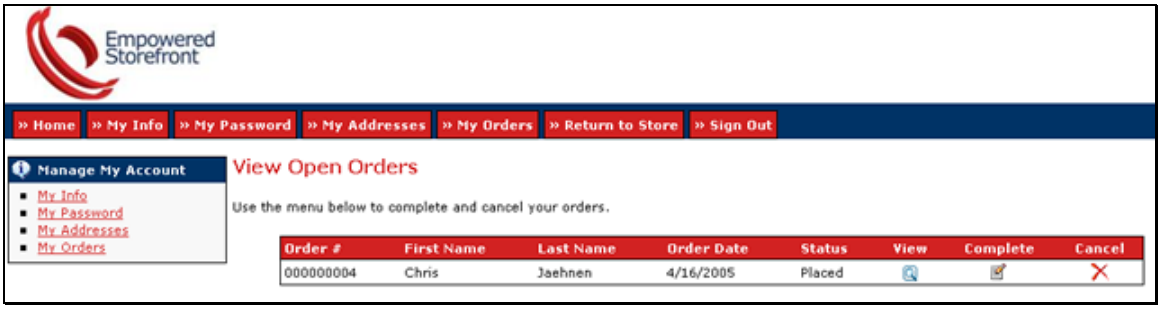

Figure 37. Open Orders Page

The customer can click on "Complete" to pay for the order or "Cancel" to cancel the order. Only open orders can be canceled. The completed orders page shows orders that have received a payment and that have been shipped by a given company (See Figure 38.).

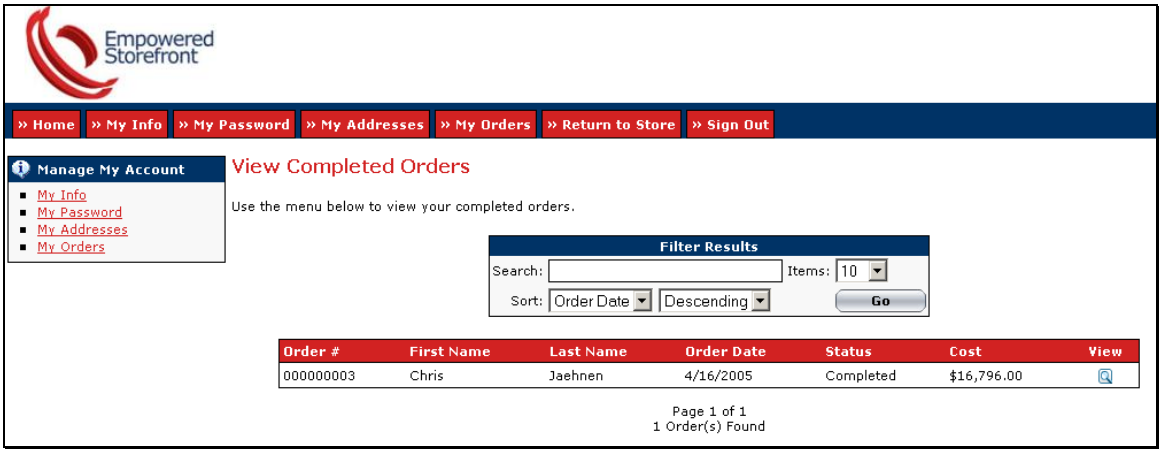

Figure 38. Completed Orders Page

Finally, the canceled orders page shows orders that have been canceled by the customer. Orders can only be canceled when the order status is "placed" or "in processing." Once an order is canceled, it cannot be re-opened by the customer. Application operators will have to re-open the order at the customer's request (See Figure 39.).

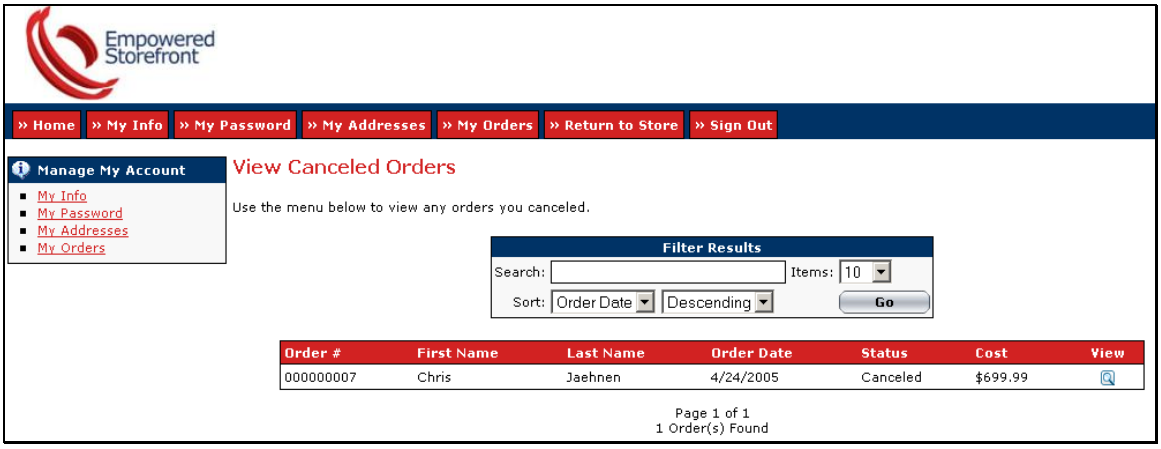

Figure 39. Canceled Orders Page

5.3.6 Tracking an Order's Status

From any of the order management pages, a customer can click on the "View" button to view the details of an order. This page mirrors the order review page shown in the check out process (See Figure 40.).

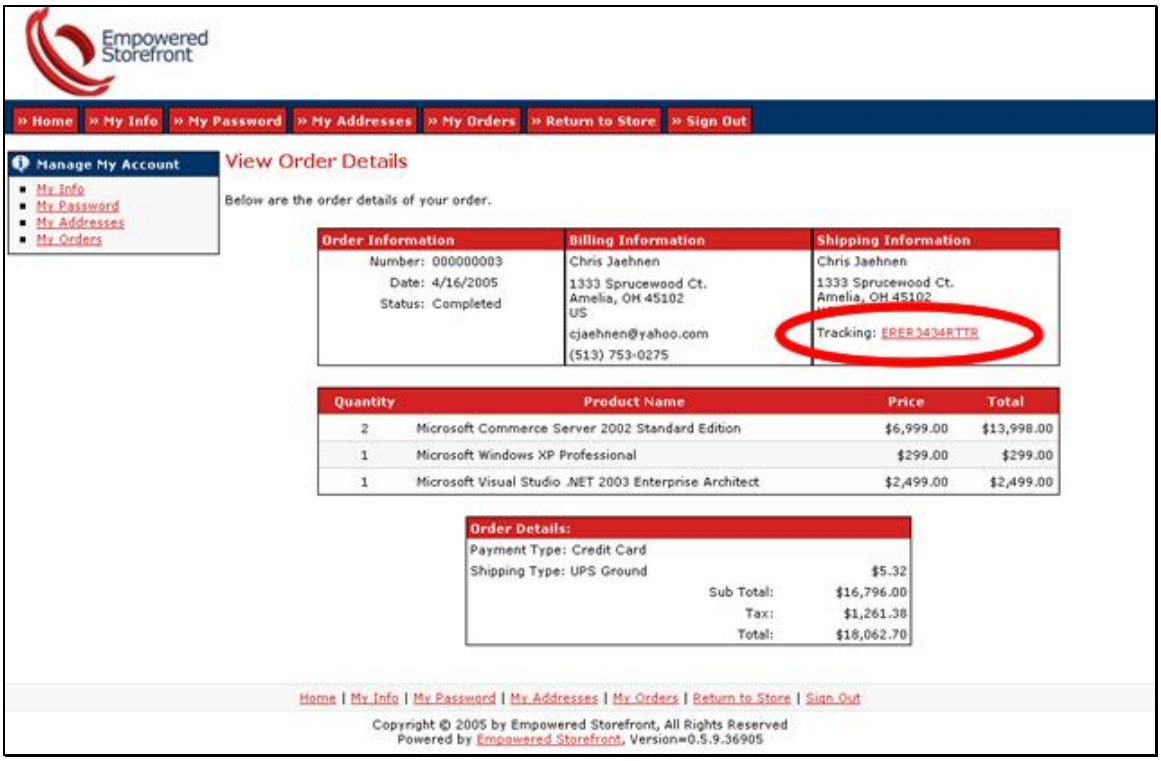

Figure 40. Order Details Page

If a tracking code has been assigned to an order, a customer can click on the tracking code, as shown in Figure 40., and a new window will open linking to the selected shipping provider's order tracking Web site. If a tracking code has not been assigned, the order details page will show "Not Assigned" for the tracking code.

5.3.7 Manage Account Information

From the customer management application, as shown earlier, customers can manage their billing account information. They simply need to click on "My Info" in the customer management application (See Figure 41.)

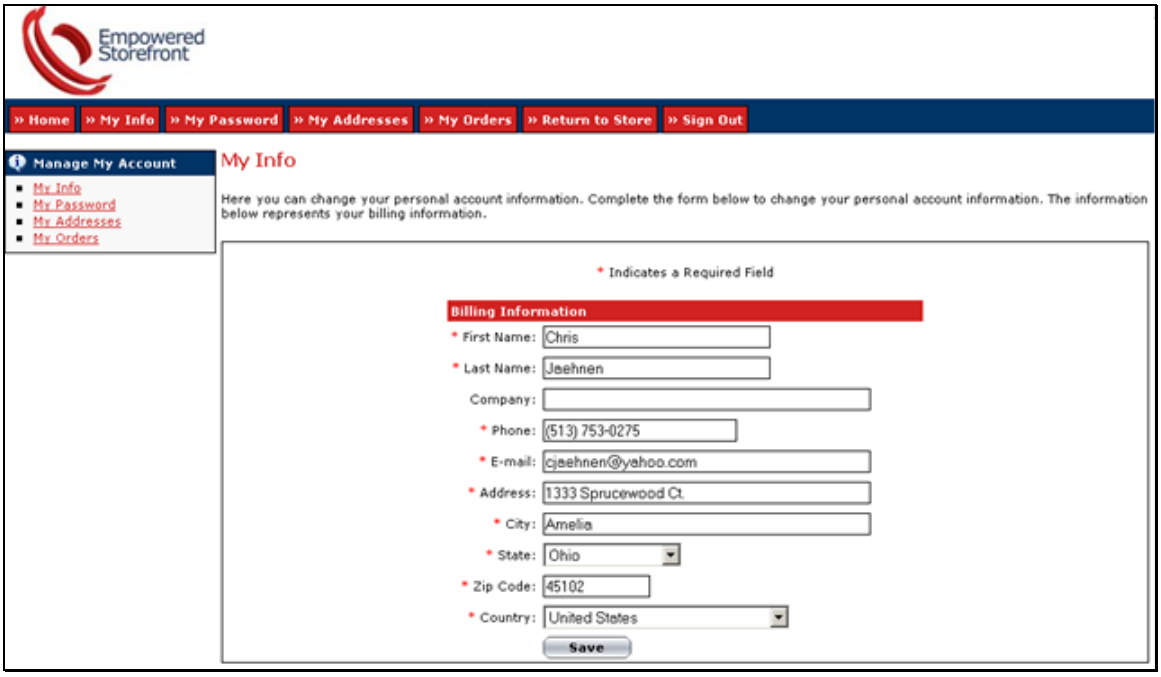

Figure 41. My Info Page

Here, the customer can change any information and click the "Save" button to save the updated billing account information.

5.3.8 Manage Wish Lists

From the on-line store, customers can click on the “Wish List” link found in the lower-right corner of page under the “My Account” sub menu. This will take customers to page that lists all the items in their wish lists (See Figure 42.).

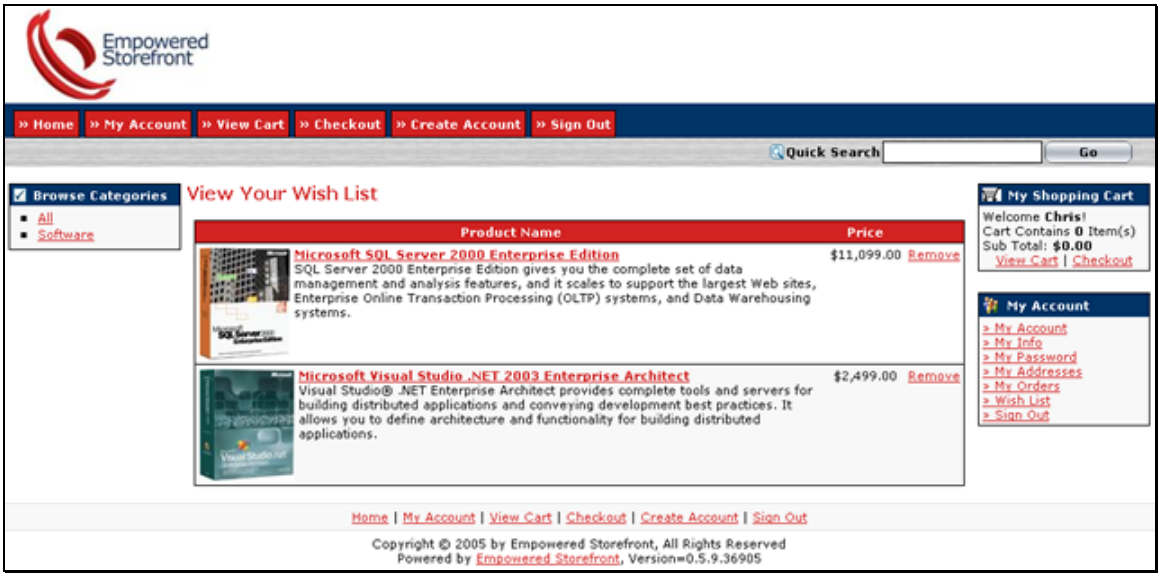

Figure 42. Wish List Page

This page provides a link to each individual product details page and a “Remove” link that allows customers to remove items from their wish lists.

5.4 Operator Actions

This section shows how the application allows operators to perform their actions as defined in the deliverables. Administrators are able to perform the same actions that operators can perform.

5.4.1 Managing Products

After an operator signs into the on-line store management application, the operator must click on “Products” link under the storefront management menu (See Figure 43.).

Figure 43. Operator On-line Store Management Home Page

Depending on the type of user, a different set of menus will appear for an operator or an administrator. The menus shown in Figure 43. represent the operator menus. The operator will be taken to a page that lists the categories defined in the on-line store. The operator will navigate to the category that contains the product that the operator wishes to add, modify, or remove. This category navigation scheme works just like the browsing for products by category in the on-line store. Once an operator has selected a category, a list of products will be shown where the operator can add, modify, or remove a product (See Figure 44.)

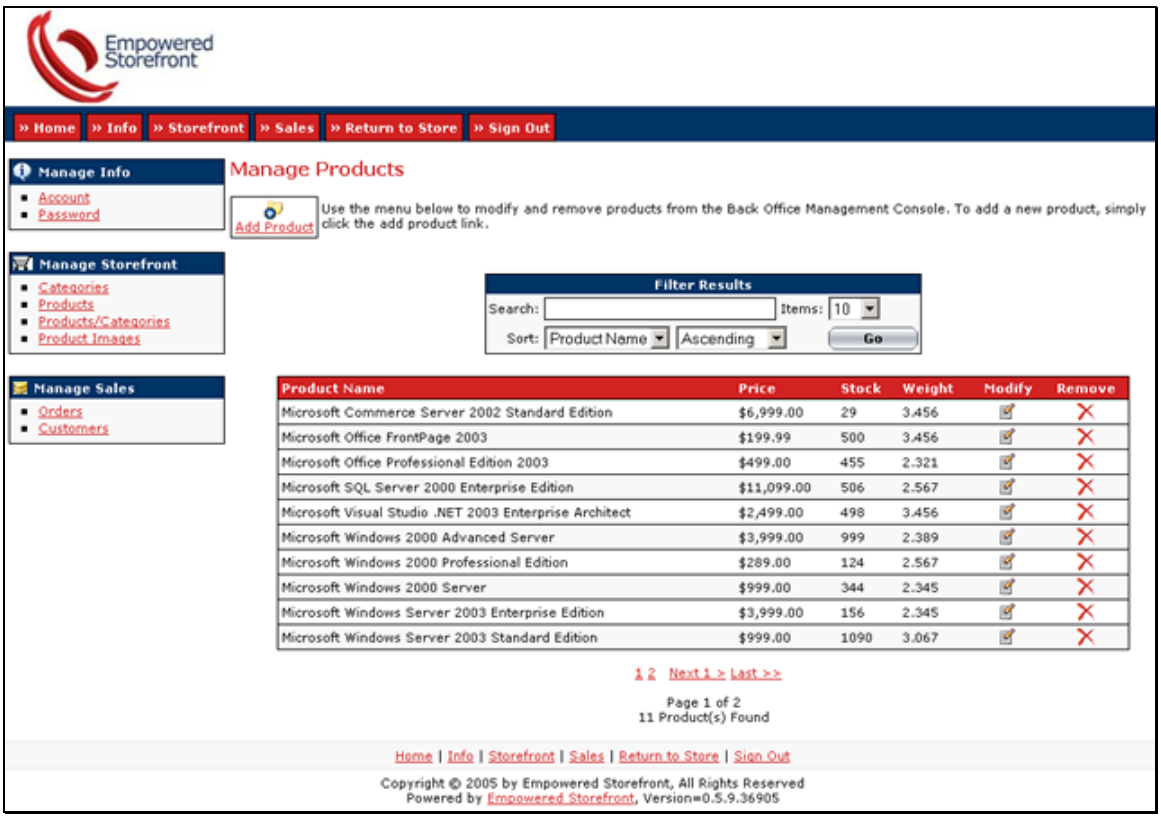

Figure 44. Manage Products Page

Clicking on the “Remove” button will show a pop up message asking the operator to confirm the removal of the product. After selecting “OK,” the product will be removed. Clicking on either the “Modify” or “Add” buttons will take the operator to a page where the operator can add or modify a selected product (See Figure 45.).

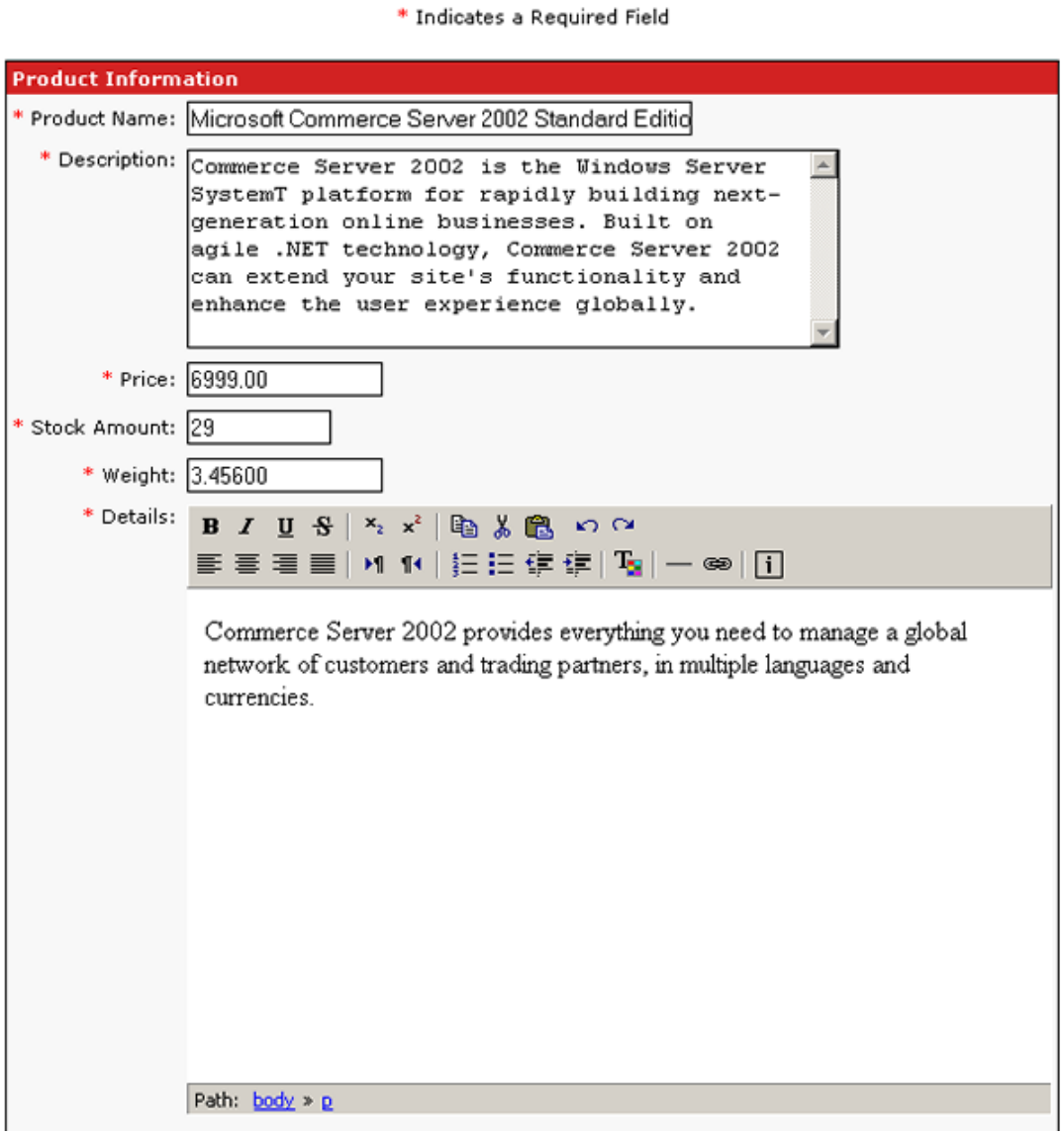

Figure 45. Adding or Modifying a Product

After the completing the product information form, an operator can click the "Save" button to save the product information. Products are assigned to categories using the "Products/Categories" link under the storefront management menu.

5.4.2 Managing Categories

An operator must click on the "Categories" link under the storefront management menu to manage categories. The operator will be taken to a page that lists the categories

defined in the on-line store. The operator will navigate to the category that the operator wishes to modify or remove. This category navigation scheme works just like the browsing for categories under product management. Once an operator has selected a category, the operator can modify or remove the category. When removing a category, any sub categories or products that exist within that category will need to be assigned to a different category. First, the removal process will check to see if any products exist within the category. If products do exist in that category, a page will allow an operator to assign these products to a different category (See Figure 46).

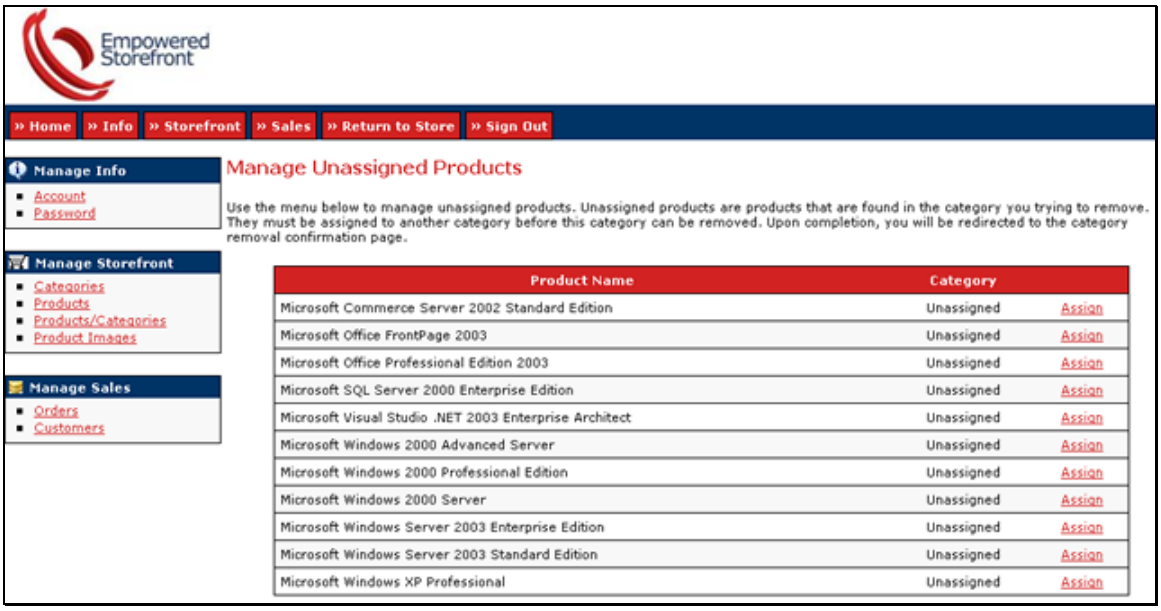

Figure 46. Manage Unassigned Products Page

The operator clicks on the "Assign" link and a drop down list filled with categories will allow an operator to move these products to a selected category. Once all the products have been assigned to another category, the removal process will check if any sub categories exist within this category. If sub categories do exist in that category, a page will allow an operator to assign these sub categories to a different category (See Figure 47.).

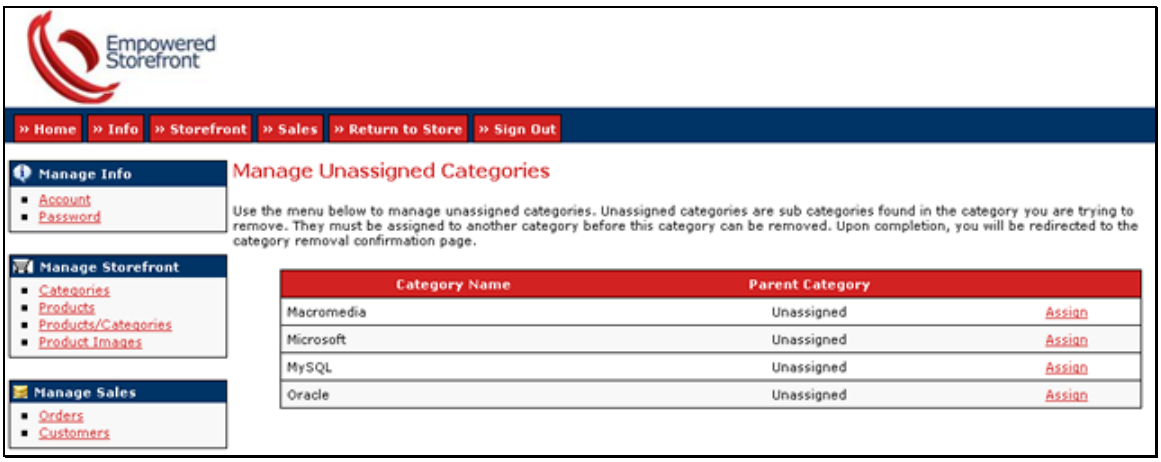

Figure 47. Manage Unassigned Categories Page

The operator clicks on the "Assign" link and a drop down list filled with categories will allow an operator to move these sub categories to a selected category. Once all the sub categories have been assigned to other categories, the removal process will confirm the removal of the category (See Figure 48).

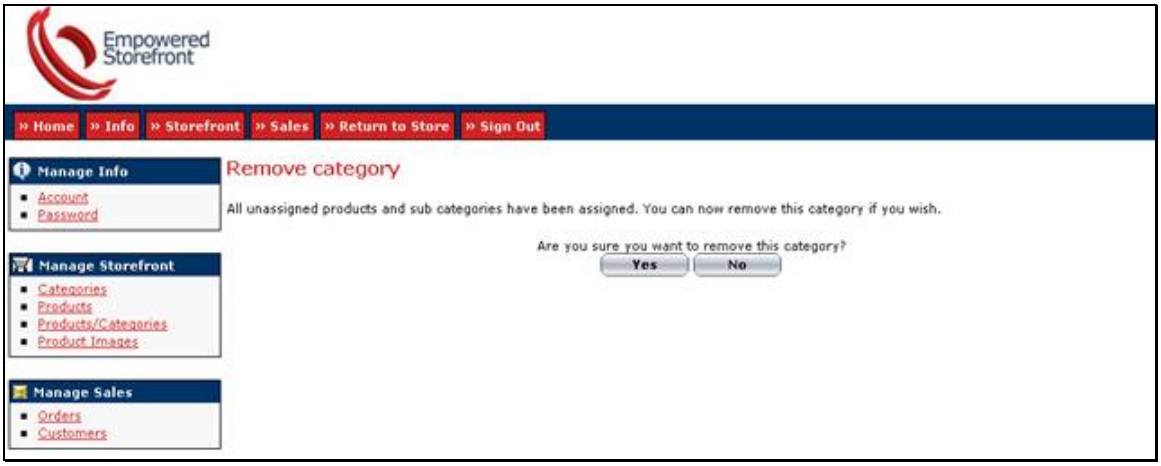

Figure 48. Remove Category Confirmation Page

After an operator clicks the "Yes" button, the category will be removed. When an operator adds or modifies a category, the following page is shown (See Figure 49.).

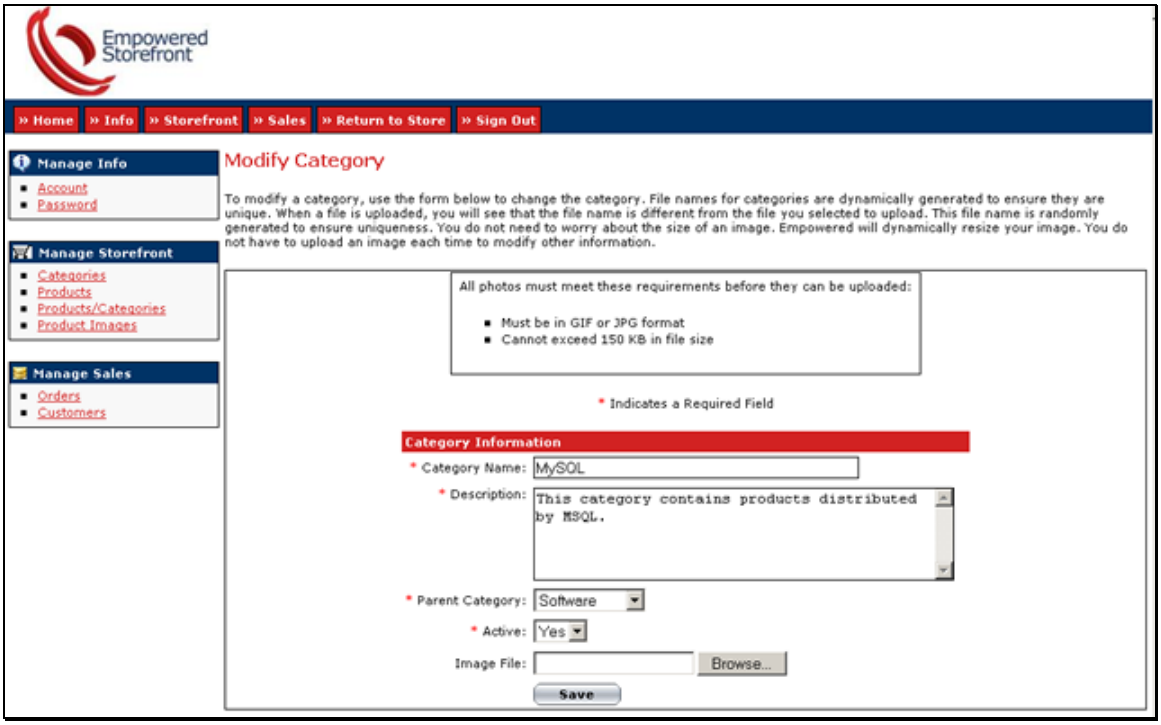

Figure 49. Adding or Modifying a Category

After completing the category information form, an operator clicks on the "Save" button and the category information is saved.

5.4.3 Managing Orders

An operator must click on the "Orders" link under the sales management menu to manage orders. The operator will be taken to a page that lists orders placed in the on-line store (See Figure 50.). Operators can view, modify, or remove orders. Clicking on the "Remove" button will show a pop up window asking the operator to confirm the removal of an order. After the operator clicks "OK," the order will be removed. When an operator views an order, the operator will be taken to a page that shows details of an order. This page mirrors the order details page found in the customer management application.

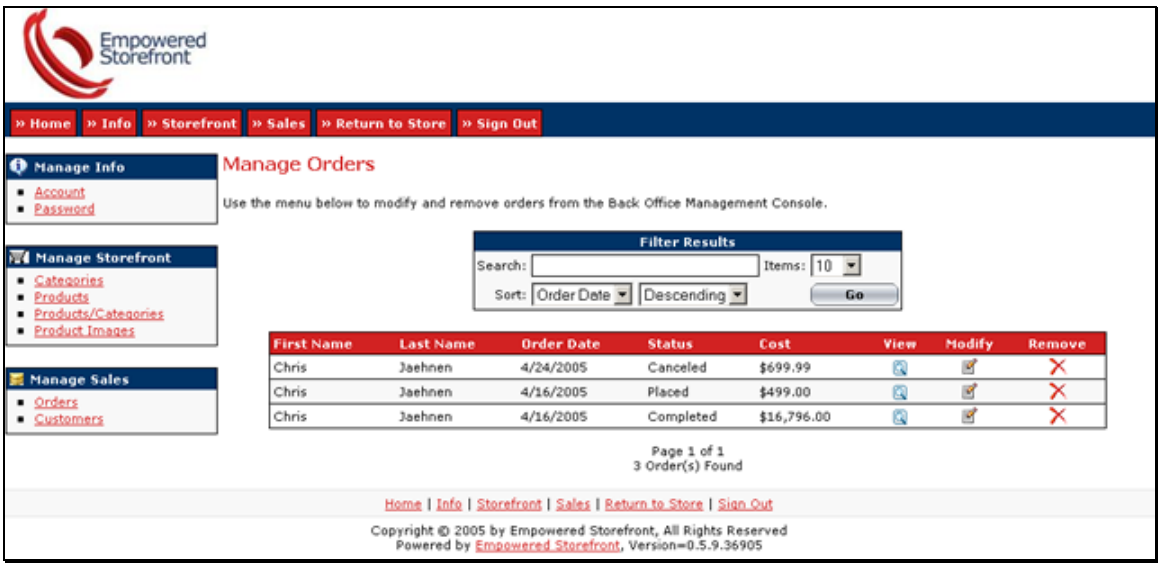

Figure 50. Manage Orders Page

When an operator clicks on the "Modify" button for an order, the operator will be taken to page where the operator can change details of the order, such as the shipping tracking code (See Figure 51.).

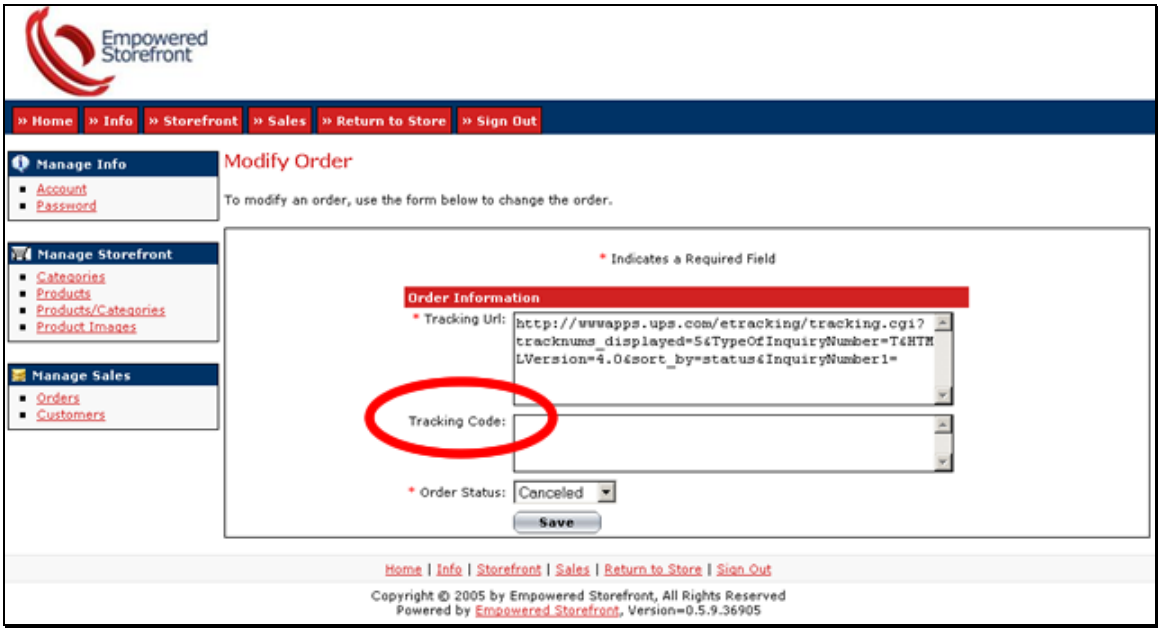

Figure 51. Modify Order Page

5.4.4 Managing Customer Account Information

An operator must click on the "Customers" link under the sales management menu to manage customers. The operator will be taken to a page that lists customers that have created accounts in the on-line store (See Figure 52).

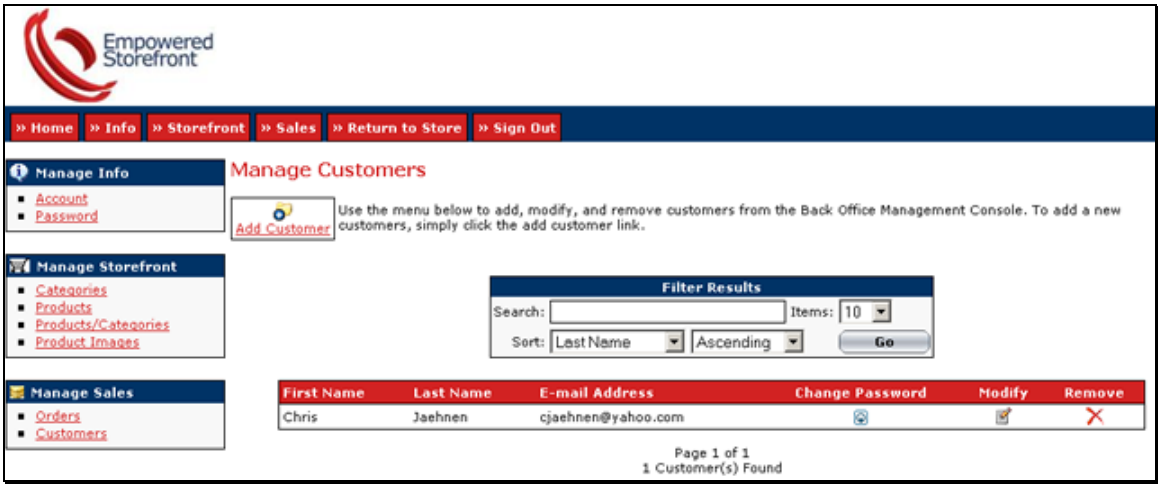

Figure 52. Manage Customers Page

Operators can add, modify, or remove customers. Clicking on the "Remove" button will show a pop up window asking the operator to confirm the removal of a customer. After the operator clicks "OK," the customer will be removed. When an operator adds or modifies a customer, the following page will be shown (See Figure 53.). Once an operator has completed the customer information form, the operator clicks the "Save" button and the customer information is saved.

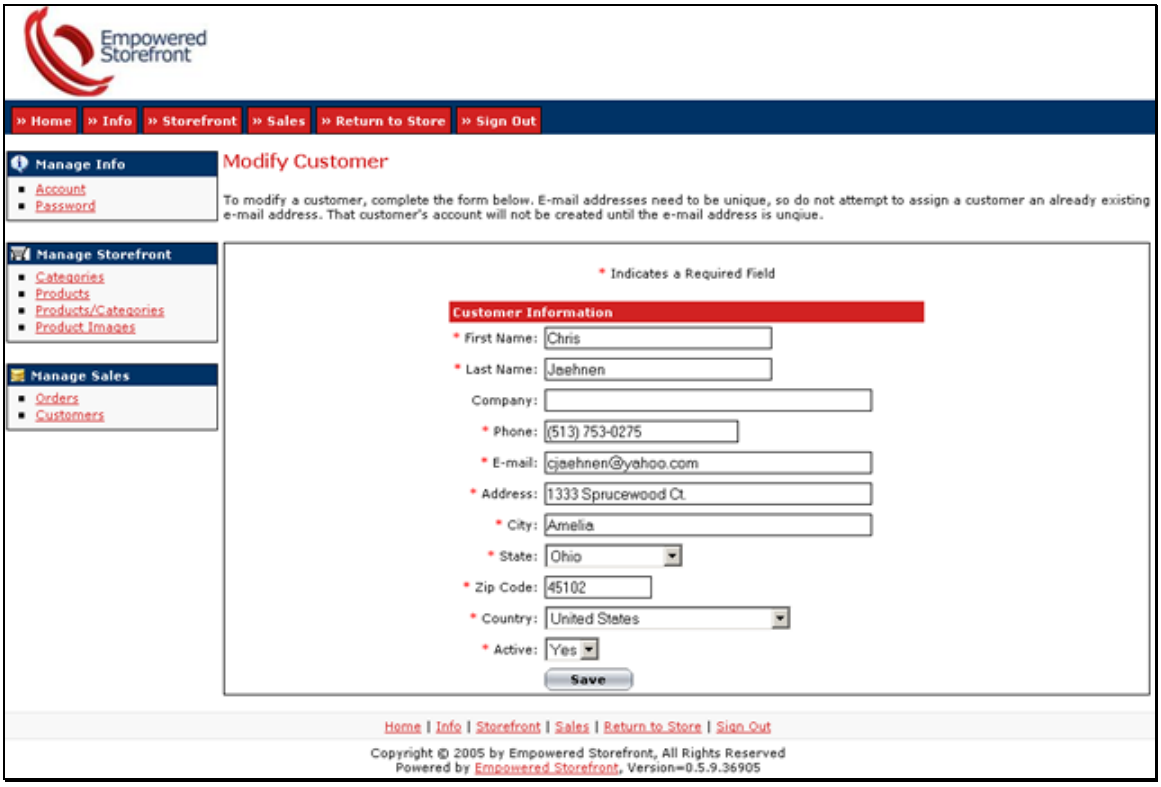

Figure 53. Adding or Modifying Customers

5.5 Administrator Actions

This section shows how the application allows administrators to perform their actions as defined in the deliverables. Administrators are able to perform the same actions that operators can perform.

5.5.1 Managing User Accounts

After an administrator signs into the on-line store management application, the administrator must click on the "Users" link under the users' management menu (See Figure 54.).

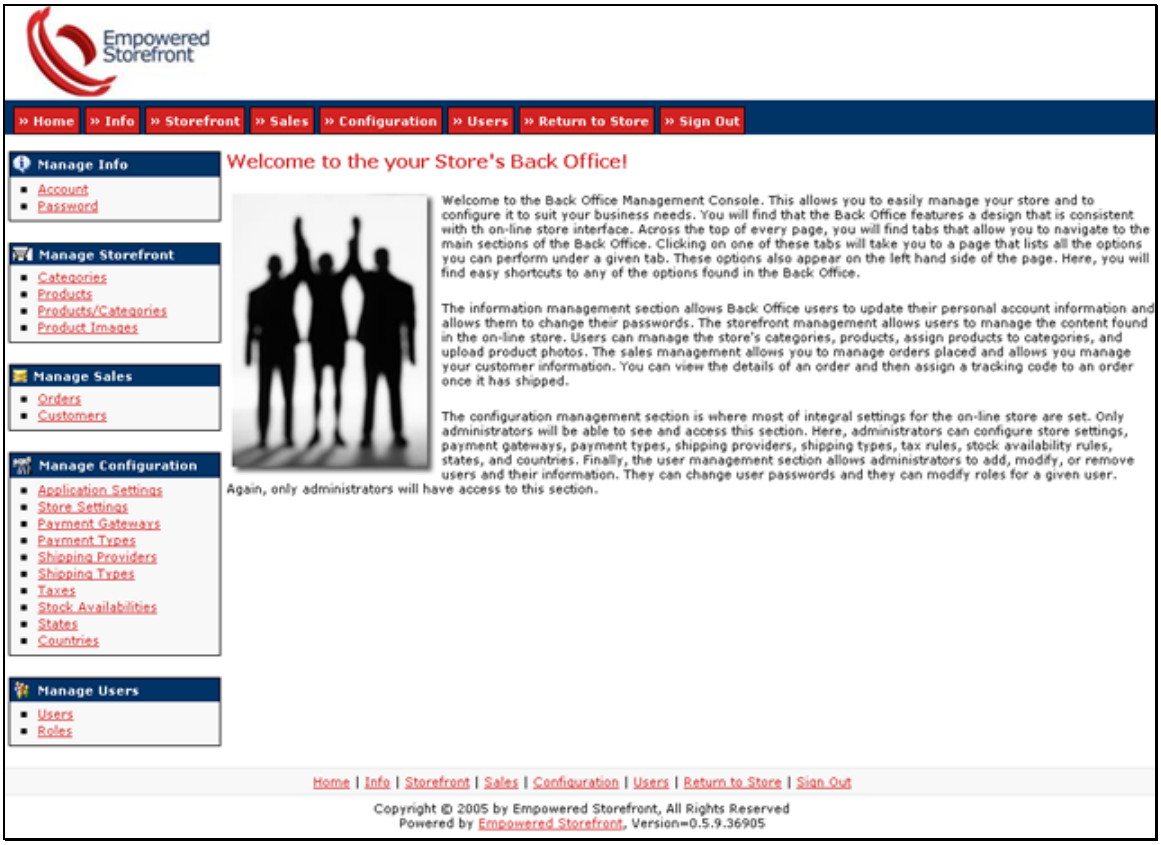

Figure 54. Administrator On-line Store Management Home Page

Depending on the type of user, a different set of menus will appear for an operator or administrator. The menus shown in Figure 54. represent the administrator menus. The administrator will be taken to a page that lists the users found in the on-line store management application (See Figure 55.). This page allows administrators to add, modify, or remove users, and it allows them to change a given user's password. Clicking on the "Remove" button will show a pop up window asking the administrator to confirm the removal of a user. After the administrator clicks "OK," the user will be removed.

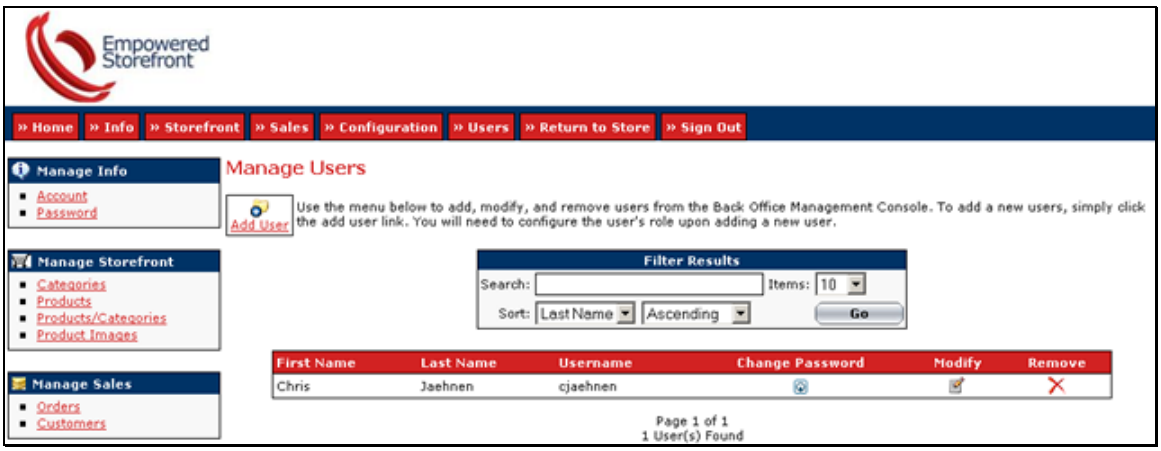

Figure 55. Manage Users Page

When an administrator adds or modifies a user, the following page will be shown (See Figure 56).

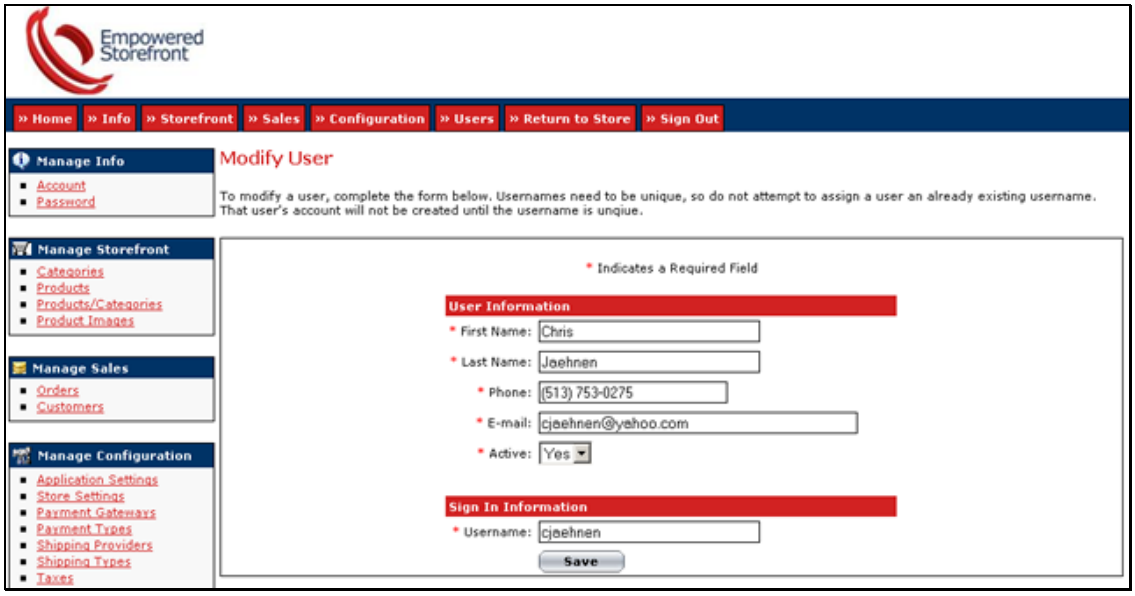

Figure 56. Adding or Modifying Users

Once an administrator has completed the user information form, the administrator clicks the “Save” button and the user information is saved. When an administrator needs to change a given user’s password, an administrator can click the “Change Password” button from the manage users page. The following page will be shown (See Figure 57.).

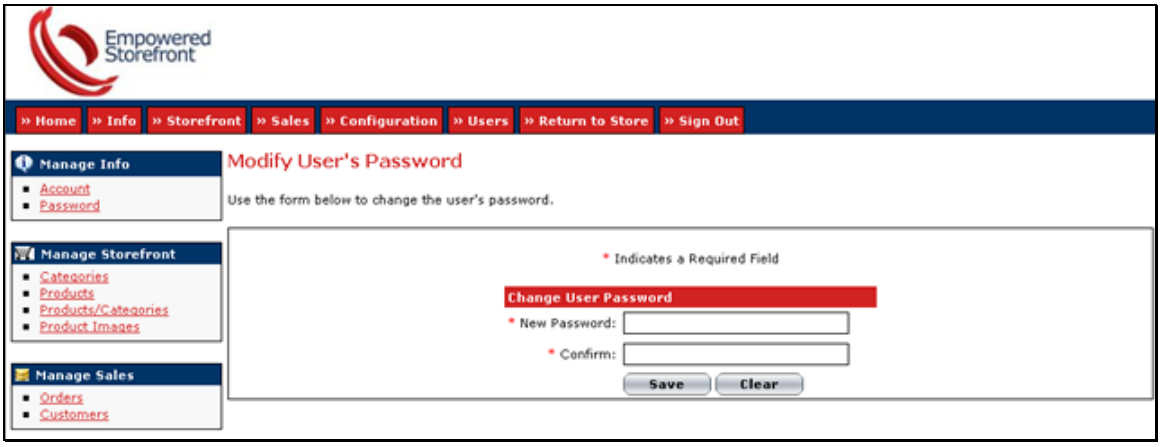

Figure 57. Modify User's Password Page

After an administrator enters the new password for a given user and confirms it is correct, an administrator clicks the "Save" button and the user's password is updated.

5.5.2 Managing User Roles

After an administrator clicks on the "Roles" link under the users' management menu, an administrator will be taken to a page that lists the users found in the on-line store management application. This page mirrors the user management page that was shown earlier. When an administrator clicks the "Modify" button for a given user, the administrator is taken to a page where the administrator can change a given user's access role, such as an operator or an administrator (See Figure 58.).

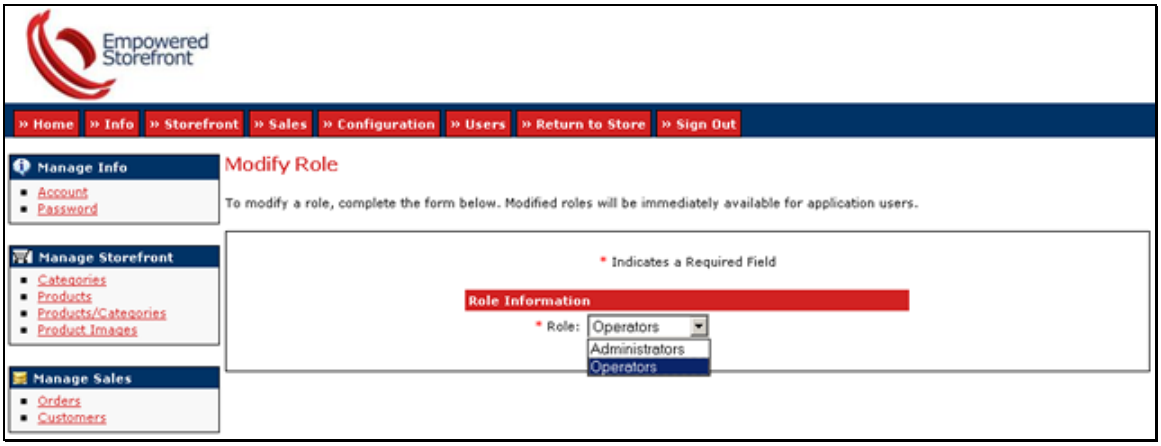

Figure 58. Modify Role Page

After an administrator selects a role, the administrator can click on the “Save” button to save the role for a given user.

5.5.3 Managing States

After an administrator clicks on the “States” link under the configuration management menu, an administrator will be taken to a page that lists the states found in the on-line store management application. This page mirrors the user management page that was shown earlier. Clicking on the “Remove” button will show a pop up window asking the administrator to confirm the removal of a state. After the administrator clicks “OK,” the state will be removed. When an administrator adds or modifies a state, the administrator is taken to a page where the administrator can enter information about a state (See Figure 59.).

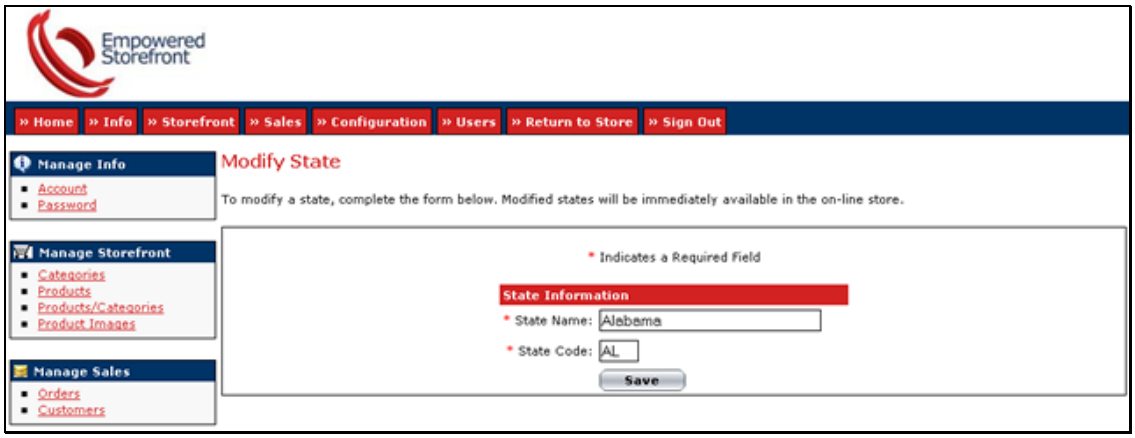

Figure 59. Adding or Modifying a State

Once an administrator has completed the state information form, the administrator clicks the “Save” button and the state information is saved.

5.5.4 Managing Countries

After an administrator clicks on the “Countries” link under the configuration management menu, an administrator will be taken to a page that lists the countries found

in the on-line store management application. This page mirrors the user management page that was shown earlier. Clicking on the "Remove" button will show a pop up window asking the administrator to confirm the removal of a country. After the administrator clicks "OK," the country will be removed. When an administrator adds or modifies a country, the administrator is taken to a page where the administrator can enter information about a country (See Figure 60.).

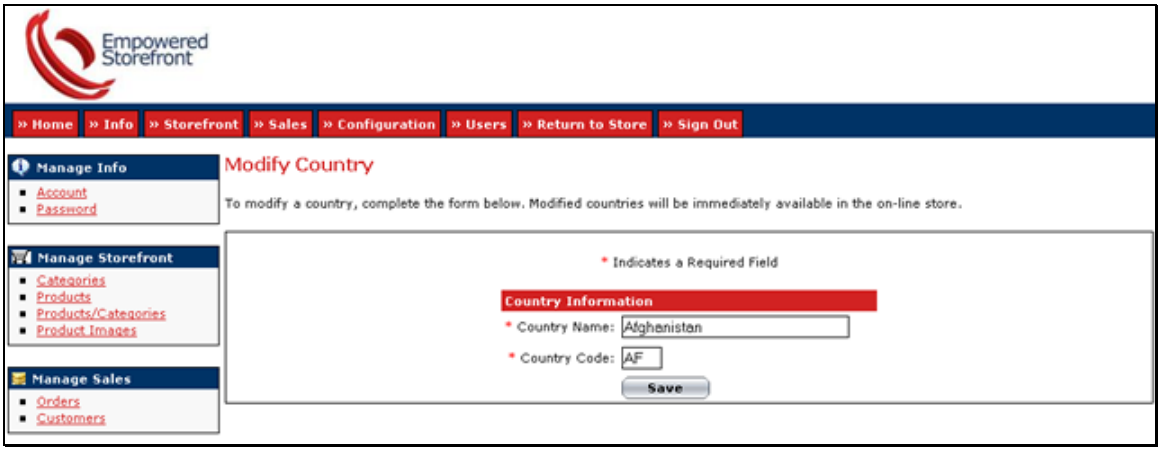

Figure 60. Adding or Modifying a Country

Once an administrator has completed the country information form, the administrator clicks the "Save" button and the country information is saved.

5.5.5 Managing Payment Types

After an administrator clicks on the "Payment Types" link under the configuration management menu, an administrator will be taken to a page that lists the payment types found in the on-line store management application. This page mirrors the user management page that was shown earlier. Clicking on the "Remove" button will show a pop up window asking the administrator to confirm the removal of a payment type. After the administrator clicks "OK," the payment type will be removed. When an

administrator adds or modifies a payment type, the administrator is taken to a page where the administrator can enter information about a payment type (See Figure 61.).

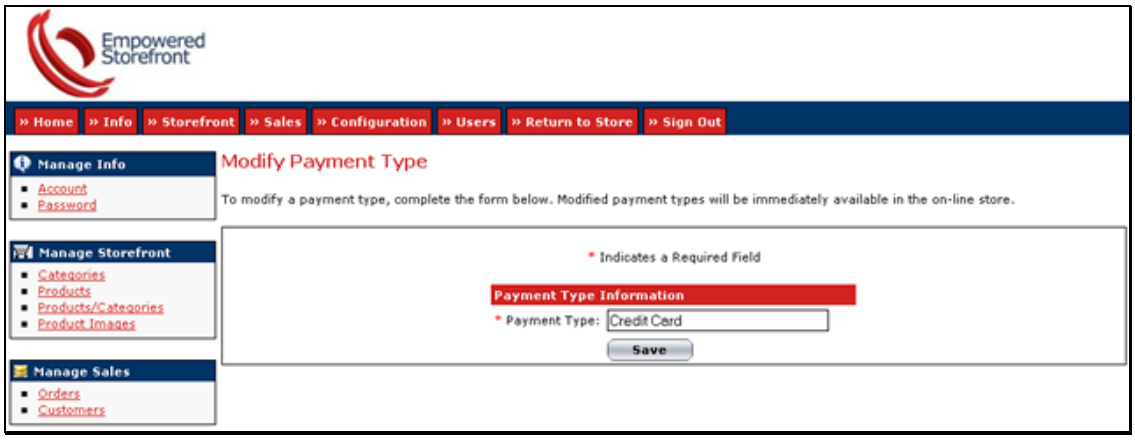

Figure 61. Adding or Modifying a Payment Type

Once an administrator has completed the payment type information form, the administrator clicks the "Save" button and the payment type information is saved.

5.5.6 Managing Shipping Types

After an administrator clicks on the "Shipping Types" link under the configuration management menu, an administrator will be taken to a page that lists the shipping providers available in the application. The administrator will have the option to manage real time or static shipping types (See Figure 62.).

Figure 62. Manage Shipping Types Page

If an administrator clicks on the “Real Time” shipping types’ button, the administrator will be taken to page where the administrator can enable or disable the real time shipping types provided by the selected shipping provider (See Figure 63.)

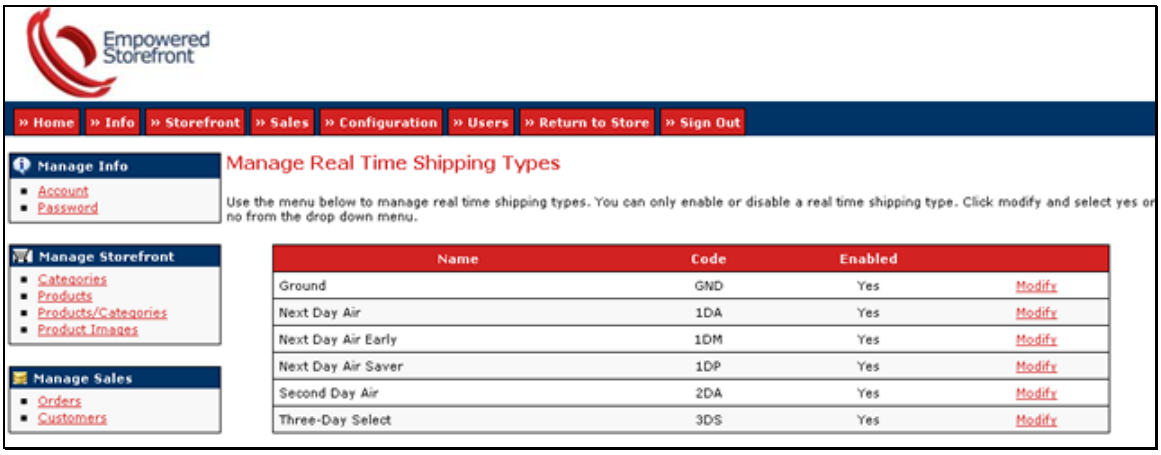

Figure 63. Manage Real Time Shipping Types Page

An administrator can click on the “Modify” link and a drop down list will show allowing an administrator to enable or disable a real time shipping type. If an administrator clicks on the “Static” shipping types’ button, the administrator will be taken to a page that lists the static shipping types found in the on-line store management application. This page mirrors the user management page that was shown earlier. Clicking on the “Remove” button will show a pop up window asking the administrator to confirm the removal of a static shipping type. After the administrator clicks “OK,” the static shipping type will be removed. When an administrator adds or modifies a static shipping type, the administrator is taken to page where the administrator can enter information about a static shipping type (See Figure 64.).

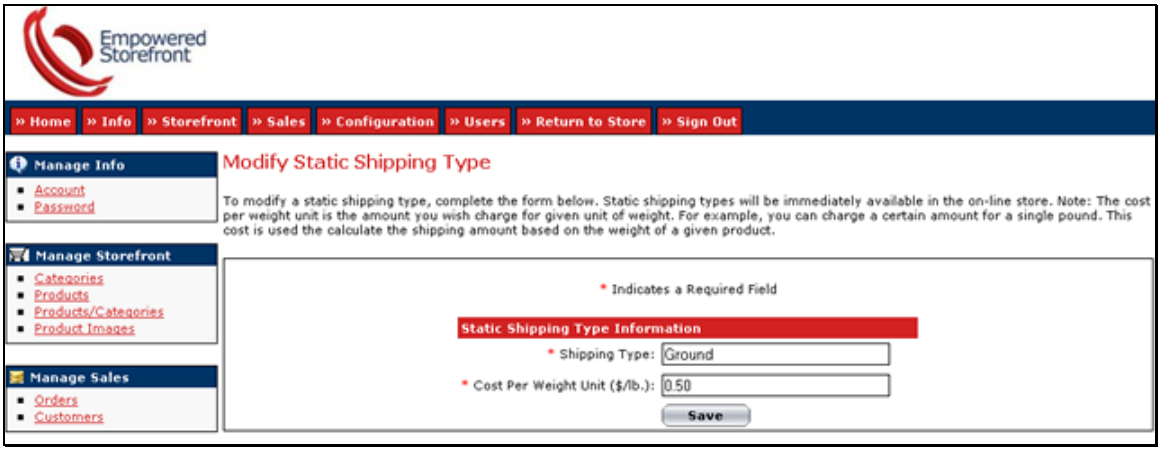

Figure 64. Adding or Modifying a Static Shipping Type

Once an administrator has completed the static shipping type information form, the administrator clicks the "Save" button and the static shipping type information is saved.

5.5.7 Managing Shipping Providers

After an administrator clicks on the "Shipping Providers" link under the configuration management menu, an administrator will be taken to a page that lists the shipping providers available in the application. An administrator will have the option to enable or configure a given shipping provider (See Figure 65.).

Figure 65. Manage Shipping Providers Page

If an administrator clicks on the "Enable" shipping types' button, that shipping provider will be enabled for the application. All other shipping providers will be disabled. If the administrator clicks on the "Configure" button, the administrator will be taken to a page where the administrator can configure settings for a given shipping provider (See Figure 66.)

Figure 66. Configure Shipping Provider Page

Once an administrator has completed the shipping provider information form, the administrator clicks the "Save" button and the shipping provider information is saved.

5.5.8 Managing Stock Availability

After an administrator clicks on the "Stock Availabilities" link under the configuration management menu, an administrator will be taken to a page that lists the stock availability rules found in the on-line store management application. This page mirrors the user management page that was shown earlier. Clicking on the "Remove" button will show a pop up window asking the administrator to confirm the removal of a stock availability rule. After the administrator clicks "OK," the stock availability rule will be removed. When an administrator adds or modifies a stock availability rule, the

administrator is taken to a page where the administrator can enter information about a stock availability rule (See Figure 67).

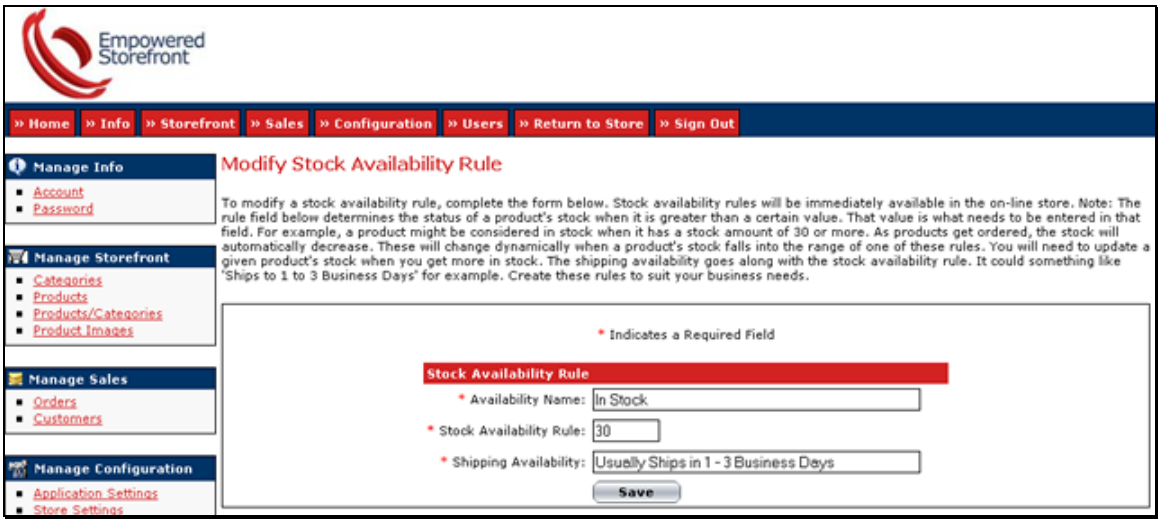

Figure 67. Adding or Modifying a Stock Availability Rule

Once an administrator has completed the stock availability rule information form, the administrator clicks the "Save" button and the stock availability rule information is saved.

5.5.9 Managing Taxes

After an administrator clicks on the "Taxes" link under the configuration management menu, an administrator will be taken to a page that lists the tax rules found in the on-line store management application. This page mirrors the user management page that was shown earlier. Clicking on the "Remove" button will show a pop up window asking the administrator to confirm the removal of a tax rule. After the administrator clicks "OK," the tax rule will be removed. When an administrator adds or modifies a tax rule, the administrator is taken to a page where the administrator can enter information about a tax rule (See Figure 68).

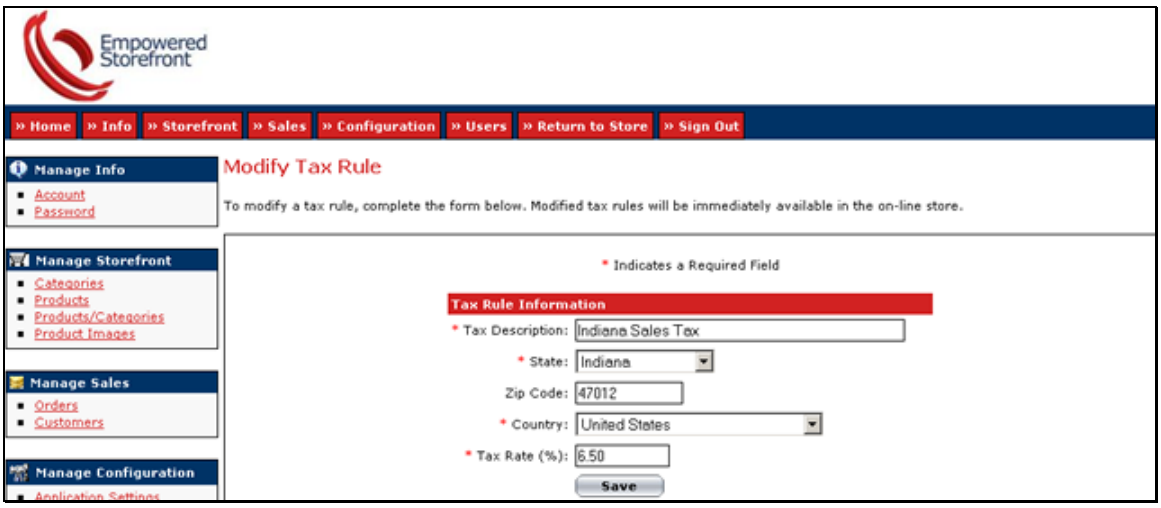

Figure 68. Adding or Modifying a Tax Rule

Once an administrator has completed the tax rule information form, the administrator clicks the "Save" button and the tax rule information is saved.

5.5.10 Managing Payment Gateways

After an administrator clicks on the "Payment Gateways" link under the configuration management menu, an administrator will be taken to a page that lists the payment gateways available in the application. An administrator will have the option to enable or configure a given payment gateway (See Figure 69.).

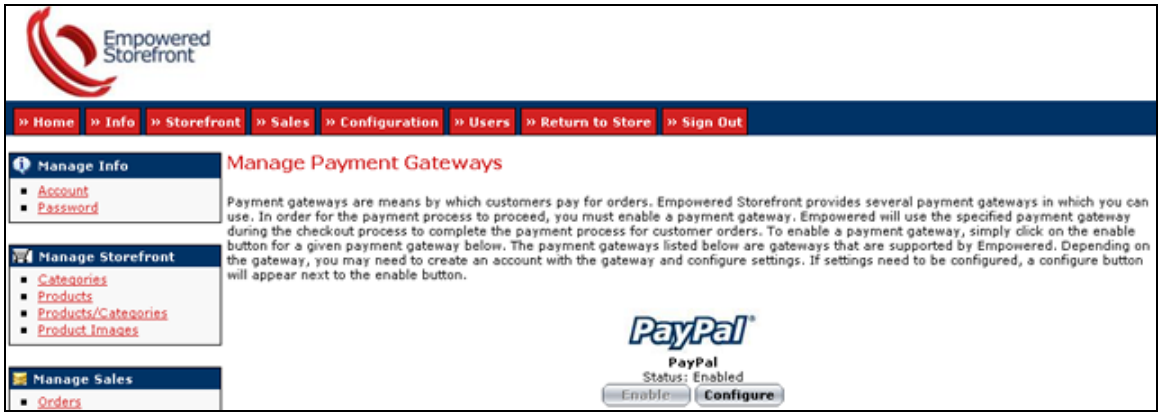

Figure 69. Manage Gateways Page

If an administrator clicks on the “Enable” payment gateway button, that payment gateway will be enabled for the application. All other payment gateways will be disabled. If the administrator clicks on the “Configure” button, the administrator will be taken to a page where the administrator can configure settings for a given payment gateway.

5.5.11 Managing Application Settings

After an administrator clicks on the “Application Settings” link under the configuration management menu, an administrator will be taken to a page where an administrator change application related settings (See Figure 70.).

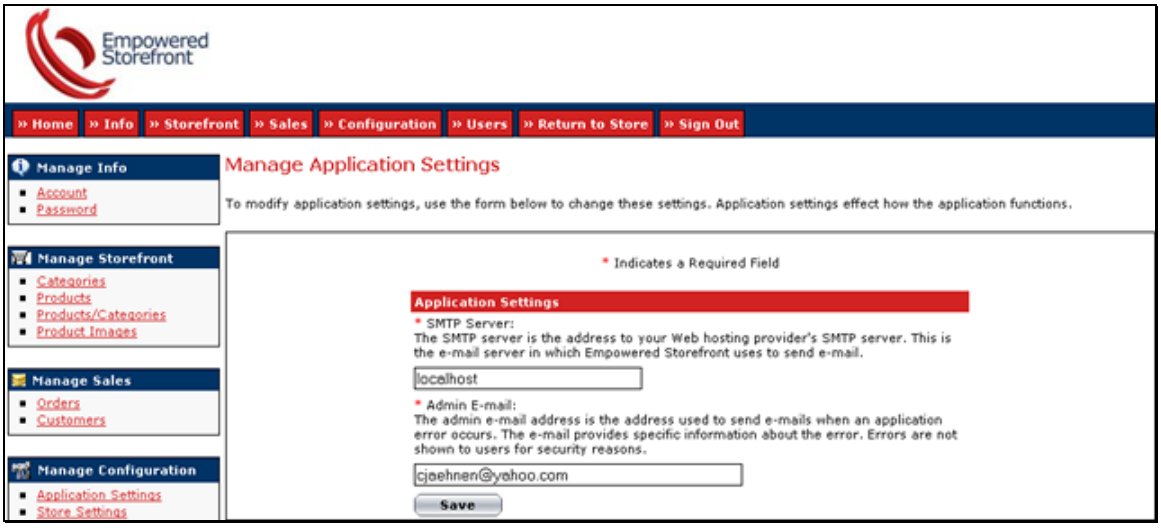

Figure 70. Manage Application Settings Page

Once an administrator has completed the application settings information form, the administrator clicks the “Save” button and the application settings are saved.

5.5.12 Managing Store Settings

After an administrator clicks on the “Store Settings” link under the configuration management menu, an administrator will be taken to a page where an administrator change store related settings (See Figure 71.).

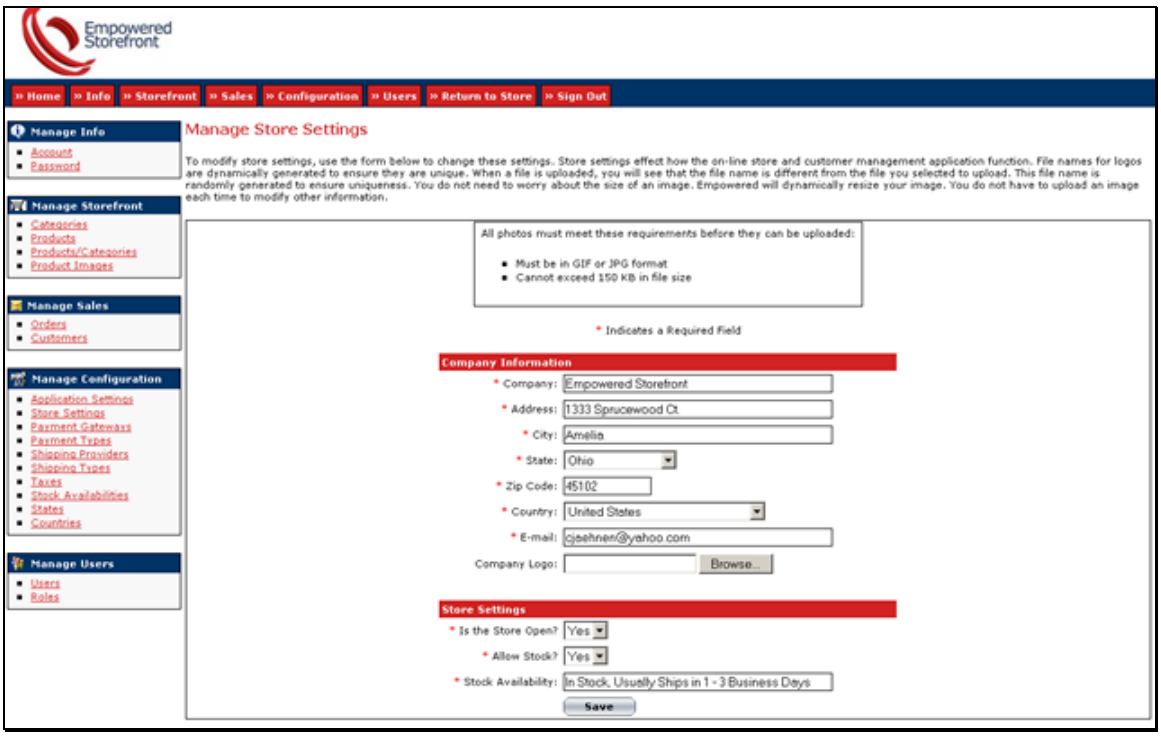

Figure 71. Manage Store Settings Page

Once an administrator has completed the store settings information form, the administrator clicks the “Save” button and the store settings are saved.

6. Design and Testing Procedures

The design and testing procedures for Empowered Storefront were based on processes used in the Unified Process for software development. When the design phase began, the project development was divided into short, one week iterations where I designed, implemented, and tested small, executable subsets of the project. First, I designed the application logic classes and implemented them through code. Next, I wrote the stored procedures that some of the application logic classes needed when performing data access operations. Then, I designed and implemented segments of the user interface and the application code required to interact with the user interface. I built several segments of the project on a daily basis and performed a project-wide build each day. Upon completion of the build, I thoroughly tested each segment of the project each day. My test plan included the following:

- Ensuring user input was in the proper format when entered into form fields using regular expressions
- Proactively handling potential errors and logging them through the application
- Catching unexpected errors and logging them through the application
- Handling improper values passed through the query string in the browser's URL
- Trapping attempts to perform dictionary attacks on the sign in page
- Handling attempts to impersonate as another user to gain access to another user's information
- Working with multiple sets of data to ensure segments of the application functioned properly

I rigorously tested each segment of the application during a given iteration. At the end of an iteration, or week, I performed a complete test on all the segments of the application built during that iteration. All application logic classes and user interface

code were carefully planned out before implementation. After this careful planning and rigorous testing, I asked several people to beta test completed segments of the application on a weekly basis. After reviewing known bugs and comments, I incorporated changes to those segments of the application at the beginning of the next iteration. Common changes included:

- User interface usability changes
- Performance changes that required data access optimization and data caching
- Application logic changes to improve reliability and performance
- Initial syntax conversion from Hungarian code style to .NET programming standards

I designed a system that would catch and log any runtime errors to an Excel spreadsheet using ASP.NET's Global.asax file. I have an included several segments of this document in this report with causes of the problem and its resolution (See Appendix A.) After the project was completed, a final, application-wide acceptance test was performed by beta testers. Their findings were thoroughly discussed and changes were incorporated into the application. This completed the design, implementation, and testing phases of the project.

7. Conclusions and Recommendations

7.1 Conclusions

This project was created to provide Web hosting providers with a cost effective solution for their customers and to provide small businesses with an easy-to-use and reliable e-commerce solution. This project was designed to improve on the usability and performance that existing e-commerce applications lack. The project provides visually appealing templates that can easily be changed and uses a global style sheet to control the

text styles and colors. Most of the application's user interface is text-based, which improves performance since image-based interfaces take longer to load. Icons are used to support future globalization of the application. Graphics for the project were prepared using Macromedia Fireworks MX 2004. The implementation of the user interface and application logic was built using Visual Basic .NET as the programming platform and ASP.NET as the programming framework. The database was designed and built using SQL Server 2000 Enterprise Manager. The project was completed over the three quarter Senior Design sequence. The real world budget of the project was estimated to be $2,162.94 while the actual budget of the project was $210.00. Testing was performed to ensure the application provided reliability and performance to Web hosting providers and small businesses. All the deliverables were fulfilled upon the project's completion.

7.2 Recommendations

During the development of Empowered Storefront, I encountered several problems and challenges. The main challenge was designing and implementing a redistributable e-commerce application. Designing a product that should work for any business was difficult. It required detailed requirements gathering and making observations of existing e-commerce applications. Many of the applications I observed were complex and required much time to make the necessary observations. Making observations on the faults with existing e-commerce applications added to the complexity. Another challenge was researching what small businesses and Web hosting providers need in an e-commerce application. Once I had the requirements in place, I was able to create a design for the application.

The design of the database and application architecture was challenging. The definition of detailed project requirements helped in the design process. Using my knowledge from previous courses and professional work with e-commerce, I was able to design a unique system that improved on the usability and performance of existing e-commerce applications. Suggestions from Steve Davis, former Senior Web Developer of WWW Internet Solutions, also helped in the design process. The final design of the architecture provided an efficient application that works in a Web hosting environment.

Another major challenge was optimizing the performance for a Web hosting environment. As one might see, e-commerce applications are resource intensive, and optimizing those resources in Web hosting environment was difficult. I made use of the .NET framework's application variables and data caching. Global data is stored in application variables when the application first starts and rarely changing datasets are cached in the Web server's memory indefinitely until they are changed. Global data is populated into templates on each page load from the application variables rather than making a round trip to the database each time. Datasets are retrieved from memory and bound to controls rather than from the database.

One major issue I encountered in the implementation of the project was during the development of the check out process. All the check out logic is handled from a single page. Initially, I decided to break the four steps of the check out process into separate user controls. ASP.NET has a feature called the ViewState. Essentially, ASP.NET stores all information about a Web form in an encrypted, hidden field called the ViewState. I use the ViewState to maintain state in the check out process. However, when using user controls, the ViewState is not maintained between the parent page and the user control.

State values are cleared as a user progresses through the check out process. I could not find a viable solution to this problem, so removed the user controls and encapsulated all the logic into the parent page. User controls simplified the amount of maintenance on the page since each step is handled inside a separate user control, but the current system works fine. I had never encountered this problem, so it was an interesting learning experience.

After reviewing recommendations from users on the completed application, I plan to add more features in the coming months to the application. While the application is fully functional, I would like to add more customer modeling features, such as product and service ratings. This will improve usability for customers and should improve the image of a small business in the process. I would also like to add reporting and database export features. I also want to do some performance tuning with the application as well. Before marketing the product, I need to provide support for more payment gateways and shipping providers. After I add support for several payment gateways and shipping providers, this product should compete well in the e-commerce application market.

Appendix A.

Error Resolution Report

The following report describes runtime and logic errors I encountered during the implementation of Empowered Storefront. A statement of the problem and its resolution is included in this report. The following table outlines logic errors that occurred during the implementation of the application (See Figure 72).

Problem	Resolution
The initial shopping cart would not allow customers to add items if they were not registered.	I added a session-based shopping cart table that tracks cart items based on a SessionID. This allows anonymous users to add items to the shopping cart
When adding items to the shopping cart, the quantity was showing as zero. Checks were in place to prevent this.	The values that were being passed to the script were null and was not checking for null values. Adding a null check solved the problem.
Initially, I planned to have two separate sign in pages, one for customers and one for users. ASP.NET only allows one sign in page per application.	I could have created two applications, but to converse memory, I wrote logic to dynamically translate the two sign in scenarios based on the URL that users were trying to access.
All global information is retrieved from the database and stored in global application variables when the application first starts. At times, the variables would clear and all the global data was not showing in the application's template files.	Apparently, IIS 5.0 recycles memory when it is not being used and clears global variables. IIS 6.0 clears the variables but an application start fires when a user hits the application. I put checks into place to retrieve the application data if does not exist.
When the application was calculating sub totals for the review order page, it was returning an inconsistent value.	I discovered the sub query that performed the calculation was calculating sub totals for all orders. I adjusted the SQL code so it only provided a total for a specific order.
Authorization is handled through roles in the application. At times, the application was getting confused between operators and administrators versus customers. Customer roles were not initially defined, so if a user signed in as an operator and tried to access the on-line store, runtime errors occurred because the authenticated	I added a default customer role to a user if the authenticating user did not have any other roles. Then, I added additional checks to ensure on-line store users were in the customer role before attempting to get information from the database.

user did not exist since the application was looking for a customer.	
I implemented custom data grid paging where only the required records are retrieved instead of all records that is handled by default. The data grid paging was encapsulated in a user control for reusability. The page index was not setting properly when a user clicked on the paging navigation links.	I had to raise an event in the user control and build an event handler inside the parent page that executes when the event is raised in the user control. This process is known as "event bubbling." In the event handler, I set the page index before the data is retrieved. This solved the page index problem.
When I was implementing the real time shipping rates with UPS, it returns a delimited string of information in which I can retrieve the shipping amount. It was initially returning a null value.	I found out that I was not parsing the string properly. After I modified the parsing code, I was able to get the shipping value.
Initially, I could not get the Web request for the UPS real time shipping rate to fire. The code compiled fine but it would not send the request to UPS.	I found out in Service Pack 1 of the .NET Framework that the Web request creation method was obsolete. After modifying the code, the Web request fired.
When searching products by category, I saw at times products from different categories were being returned. This should not happen.	I modified the SQL code for the search by putting that segment into parenthesis. This way it looked for products in a given category and products based on the search terms. Those products from different categories no longer showed.
I had developed an algorithm for determining stock availability based on the stock amount for a given product and the stock availability rules defined. Everything was working fine until I noticed that after a stock value dropped below a certain value, the rule was not working properly.	I discovered that the rule comparison should have been checking for stock amounts greater than the rule instead of less than the rule. Rigorous testing helped me find the logic error and I fixed it.
When a customer was adding a shipping address during the check out process, duplicate shipping addresses where being added. This should not have happened.	I added some additional SQL code that checks if the shipping address exists. If it does, the address does not get added, and if it does not, the address gets added.

Figure 72. Logic Error Resolution Report

The following table outlines common runtime errors that occurred during the implementation of the application (See Figure 73).

Problem	Resolution
The specified cast is not valid.	This occurred when I attempted to convert an object to another data type. The object was either null or not in the proper format. I designed a NullSafe object that performs safe conversions.
Could not find row at position 0.	This occurred when I attempted to reference a data table row in a dataset that was empty. I returned a record count data table alongside the dataset to ensure the data table was not empty.
System.SqlException	This occurred when a database error was thrown when performing data access operations. This was always a problem with SQL code inside one of the stored procedures.
Index out of range.	This occurred when I would perform looping operations based on a count from a collection object. Adding count minus one resolved the problem.

Figure 73. Runtime Error Resolution Report

Other than the runtime errors mentioned above, I did not encounter anymore common runtime errors. There seemed to be more logic errors than runtime errors. Most of the time, builds would succeed without incident.

Appendix B.

Code Snippets

B.1 File Upload Algorithm

I designed and implemented a Web-based file upload algorithm that allows multiple files to be uploaded at once. It provides properties for setting a maximum file size and an accepted list of file extensions. A save method allows one to save a file to the Web server's hard drive. The upload method reads the file's binary data and stores it into a property, so the binary data can be inserted into a database. If the file is an image, the image's width and height are set as properties. The width and height can then be accessed through these properties.

```
Public Class FileUpload

        ' Data members
        Private _FileSize As Integer
        Private _ContentType As String
        Private _FileName As String
        Private _FileExtension As String
        Private _Binary As Byte()
        Private _PostedFile As HttpPostedFile
        Private _Files As HttpFileCollection
        Private _AcceptedFileTypes As String
        Private _MaxFileSize As Integer
        Private _ImageWidth As Integer
        Private _ImageHeight As Integer
        Private _Message As String

        ' This property gets the file size.
        Public ReadOnly Property FileSize() As Integer
            Get
                Return _FileSize
            End Get
        End Property

        ' This property gets the content type.
        Public ReadOnly Property ContentType() As String
            Get
                Return _ContentType
            End Get
        End Property
```

```
        ' This property gets the file name.
        Public ReadOnly Property FileName() As String
            Get
                Return _FileName
            End Get
        End Property

        ' This property gets the file extension.
        Public ReadOnly Property FileExtension() As String
            Get
                Return _FileExtension
            End Get
        End Property

        ' This property gets the file binary data.
        Public ReadOnly Property Binary() As Byte()
            Get
                Return _Binary
            End Get
        End Property

        ' This property gets the http posted file.
        Public ReadOnly Property PostedFile() As HttpPostedFile
            Get
                Return _PostedFile
            End Get
        End Property

        ' This property gets the file collection.
        Public ReadOnly Property Files() As HttpFileCollection
            Get
                Return _Files
            End Get
        End Property

        ' This property gets/sets the accepted file types.
        Public Property AcceptedFileTypes() As String
            Get
                Return _AcceptedFileTypes
            End Get
            Set(ByVal Value As String)
                _AcceptedFileTypes = Value
            End Set
        End Property

        ' This property gets/sets the max file size.
        Public Property MaxFileSize() As Integer
            Get
                Return _MaxFileSize
            End Get
            Set(ByVal Value As Integer)
                _MaxFileSize = Value
            End Set
        End Property

        ' This property gets the image width.
```

```
    Public ReadOnly Property ImageWidth() As Integer
        Get
            Return _ImageWidth
        End Get
    End Property

    ' This property gets the image height.
    Public ReadOnly Property ImageHeight() As Integer
        Get
            Return _ImageHeight
        End Get
    End Property

    ' This property gets the error message.
    Public ReadOnly Property Message() As String
        Get
            Return _Message
        End Get
    End Property

    ' This default constructor stores the files into memory.
    Public Sub New()
        ' Declarations
        Dim Request As HttpRequest = HttpContext.Current.Request

        ' Get the files collection
        _Files = Request.Files

        ' Dispose of objects
        Request = Nothing
    End Sub

    Public Function Upload(Optional ByVal Index As Integer = 0) _
        As Boolean

        ' Instantiate objects
        _PostedFile = _Files.Get(Index)

        ' Get the file properties
        _FileSize = _PostedFile.ContentLength
        _ContentType = _PostedFile.ContentType
        _FileName = Path.GetFileName(_PostedFile.FileName).ToLower
        _FileExtension = _
              Path.GetExtension(_PostedFile.FileName).Replace( _
              ".", _"").ToLower
        _FileName = GenerateUniqueFilename() & "." & _FileExtension

        ' Ensure the file type matches the accepted file type list
        If InStr(_AcceptedFileTypes, _FileExtension) = 0 Then
            _Message = "The uploaded file type cannot be " & _
            accepted. The accepted file types are: " & _
            _AcceptedFileTypes
            Return False
        End If
```

```
        ' Ensure the file size does not exceed the max file size
        If _FileSize > _MaxFileSize Then
            _Message = "The uploaded file, " & _FileName & _
                ", exceeded the max file size of " & _
                _MaxFileSize & " bytes."
            Return False
        End If

        ' Allocate buffer for reading the file
        Dim BufferedData(_FileSize) As Byte

        ' Read upload file from stream
        _PostedFile.InputStream.Read(BufferedData, 0, _FileSize)

        ' Set the binary data
        _Binary = BufferedData

        ' Set the image properties if the file is an image
        If IsImage() Then
            _ImageWidth = GetImageWidth(_PostedFile)
            _ImageHeight = GetImageHeight(_PostedFile)
        End If

        Return True
    End Function

    ' This procedure saves the current file in the file collection
    ' to a specified path.
    Public Sub Save(ByVal FilePath As String)
        ' Declarations
        Dim FileStream As FileStream
        Dim FileToSave As String = FilePath & _FileName

        ' Try to save the file
        Try
            ' Create the file
            FileStream = New FileStream(FileToSave, _
                 FileMode.Create)

            ' Write data to the file
            FileStream.Write(_Binary, 0, _Binary.Length)

            ' Close file
            FileStream.Close()

            _Message = "The file, " & _FileName & ", " & _
                 "saved successfully!"
        Catch ex As Exception
            Throw New ApplicationException("The file, " & _
                 _FileName & _
                 ", could not be saved: " & ex.Message)
        Finally
            ' Dispose of objects
            FileStream = Nothing
        End Try
    End Sub
```

```
' This function generates a dynamic file name.
Private Function GenerateUniqueFilename() As String
    ' Randomize the seed generator
    Randomize()

    ' Declarations
    Dim Random As New Random

    Return Random.Next.ToString()
End Function

' This function determines if specified file is an image.
Private Function IsImage() As Boolean
    Select Case _FileExtension.ToLower
        Case "jpg"
            Return True

        Case "gif"
            Return True

        Case Else
            Return False
    End Select
End Function

' This function returns the image's width.
Private Function GetImageWidth(ByVal PostedFile As _
    HttpPostedFile) As Integer

    Dim Image As Image

    ' Ensure file is an image
    If IsImage() Then
        Image = Image.FromStream(PostedFile.InputStream())
        Return Image.Width
    Else
        Return 0
    End If
End Function

' This function returns the image's height.
Private Function GetImageHeight(ByVal PostedFile As _
    HttpPostedFile) As Integer

    Dim Image As Image

    ' Ensure file is an image
    If IsImage() Then
        Image = Image.FromStream(PostedFile.InputStream())
        Return Image.Height
    Else
        Return 0
    End If
End Function
```

```
        ' This destructor removes the posted file and files objects.
        Protected Overrides Sub Finalize()
            ' Dispose of objects
            _PostedFile = Nothing
            _Files = Nothing
            MyBase.Finalize()
        End Sub

    End Class
```

B.2 UPS Real Time Shipping Web Service

I created a Web service class that exposes a get price method that creates a Web request based on passed parameters, builds a payload string, send the request to UPS, and returns a shipping price. I decided to create this class as a Web service so it can be reused in future projects.

```
    Public Class UPS
        Inherits WebService

        ' UPS request URL
        Private Const _UPSRequestUrl As String = _
            "http://www.ups.com/using/services/rave/qcost_dss.cgi?"

        <WebMethod()> _
        Public Function GetPrice(ByVal ServiceLevelCode As String, _
            ByVal RateChart As String, _
            ByVal ShipperZipCode As String, _
            ByVal ReceiverZipCode As String, _
            ByVal ReceiverCountry As String, _
            ByVal PackageWeight As String, _
            ByVal IsResidential As String, _
            ByVal IsCOD As String, _
            ByVal IsSaturdayPickup As String, _
            ByVal IsSaturdayDelivery As String, _
            ByVal PackageType As String) As String

            ' Declarations
            Dim WebResponse As WebResponse
            Dim Stream As StreamReader
            Dim ReturnValue As String
            Dim URLRequest As String
            Dim Line As String

            ' Build the UPS request
            URLRequest = BuildUPSRequest(ServiceLevelCode, RateChart, _
                ShipperZipCode, ReceiverZipCode, ReceiverCountry, _
                PackageWeight, IsResidential, IsCOD, _
                IsSaturdayPickup, IsSaturdayDelivery, PackageType)
```

```
        ' Create the Web request for the UPS data
        Dim WebRequest As WebRequest = _
             WebRequest.Create(URLRequest)

        ' Get the Web response
        WebResponse = WebRequest.GetResponse()

        ' Create a stream for reading
        Stream = New StreamReader(WebResponse. _
             GetResponseStream(),  Encoding.ASCII)

        ' Try the read data from the stream
        Try
            Line = Stream.ReadLine()

            Do While Not Line Is Nothing
                Line = Stream.ReadLine()

                If InStr(1, LCase(Line), "upsonline") <> 0 Then
                    ' Split the line's contents at the percent
                    Dim ArrayData() As String = Split(Line, "%")

                    If Left(ArrayData(3), 4) = "0000" Then
                        ReturnValue = ArrayData(12)
                    Else
                        ReturnValue = "Error: " & _
                            Right(ArrayData(3), ArrayData(3). _
                            Length - 4)
                    End If

                    Exit Do
                End If
            Loop
        Catch ex As Exception
            ReturnValue = ex.Message
        End Try

        Return ReturnValue
    End Function

    Private Function BuildUPSRequest( _
        ByVal ServiceLevelCode As String, _
        ByVal RateChart As String, _
        ByVal ShipperZipCode As String, _
        ByVal ReceiverZipCode As String, _
        ByVal ReceiverCountry As String, _
        ByVal PackageWeight As String, _
        ByVal IsResidential As String, _
        ByVal IsCOD As String, _
        ByVal IsSaturdayPickup As String, _
        ByVal IsSaturdayDelivery As String, _
        ByVal PackageType As String) As String

        ' Declarations
        Dim UPSPayload As String
```

```
        ' Build the UPS request string
        UPSPayload = _UPSRequestUrl
        UPSPayload &= _
           "AppVersion=1.2&AcceptUPSLicenseAgreement=YES&"
        UPSPayload &= "ResponseType=application/x-ups-" & _
            "_rss&ActionCode=3&"
        UPSPayload &= "ServiceLevelCode=" & ServiceLevelCode & _
           "&RateChart=" & RateChart & "&"
        UPSPayload &= "ShipperPostalCode=" & ShipperZipCode & _
            "&ConsigneePostalCode=" & ReceiverZipCode & "&"
        UPSPayload &= "ConsigneeCountry=" & ReceiverCountry & _
            "&PackageActualWeight=" & PackageWeight & "&"
        UPSPayload &= "ResidentialInd=" & IsResidential & _
            "&CODInd=" & IsCOD & "&SatDelivInd=" & _
            IsSaturdayDelivery
        UPSPayload &= "&SatPickupInd=" & IsSaturdayPickup & _
            "&PackagingType=" & PackageType

        Return UPSPayload
    End Function
```

References

1. Comersus. "Comersus ASP Shopping Cart: Purchase." http://www.comersus.com/purchase.html. October 30, 2004.

2. Davis, Steve. Senior Web Developer, Internet Solutions of Cincinnati (ISOC) Personal Interview. September 30, 2004.

3. DiscountASP.NET. "DiscountASP.NET Web Hosting Package Features." http://www.discountasp.net/features.aspx. October 31, 2004.

4. GlobalSCAPE. "Purchase CuteFTP Professional." http://www.cuteftp.com/store/purchase.asp?product=cuteftppro. October 26, 2004.

5. Irish Higher Education Consultancy. "Why is e-Commerce Important." http://www.ihec.net/ebusiness/ecommerce3.htm. October 2, 2004.

6. Macromedia. "Macromedia Worldwide Store." http://www.macromedia.com/cfusion/store/. October 15, 2004.

7. Microsoft. "Microsoft Commerce Server: How to Buy." http://www.microsoft.com/commerceserver/howtobuy/production.asp. May 17, 2004.

8. Microsoft. "Microsoft Commerce Server: Product Overview." http://www.microsoft.com/commerceserver/evaluation/overview/. May 17, 2004.

9. Microsoft. "Microsoft SQL Server: How to Buy." http://www.microsoft.com/sql/howtobuy/development.asp. October 5, 2004.

10. Microsoft. "Microsoft Visual Studio .NET 2003: How to Buy." http://msdn.microsoft.com/howtobuy/vstudio/. October 5, 2004.

11. Miva Merchant. "Miva General Store: Miva Merchant Domain." http://www.miva.com/purchase/Merchant2/merchant.mvc?Screen=PROD&Store_Code=General_Store_3.01&Product_Code=mnderc_1&Category_Code=merc_prods. October 13, 2004.

12. PayPal. "PayPal - Fees." http://www.paypal.com/cgi-bin/webscr?cmd=_display-fees-outside. October 17, 2004.

13. Price Watch. "Price Watch Computer Inventory." http://www.pricewatch.com.

October 31, 2004.

14. Schneider, Gary P. and James T. Perry. *Electronic Commerce*. Boston: Course Technology, 2001.

15. StormPay. "StormPay - Fees." http://www.stormpay.com/stormpay/user/about_us.php?v=f. October 25, 2004.