

# ***E-Commerce Application***

Software Requirements  
Raffi Khatchadourian  
CS598 Applications in .NET  
February 17, 2004  
Document Version 1.0

# Table of Contents

Table of Contents .....	2
1. Application Overview .....	3
1.1 Purpose.....	3
1.2 Value .....	3
1.3 Use .....	3
2. General Description .....	3
2.1 Product Functions .....	3
2.2 Similar Systems Information .....	3
3. Functional Abstract.....	4
3.1 General Functional Requirements.....	4
3.2 User Related Functional Requirements .....	4
3.3 Credit Card Authentication Functional Requirements.....	4
3.4 Warehouse Related Functional Requirements .....	5
3.5 Purchasing Related Functional Requirements .....	5
3.6 Shipping Related Functional Requirements.....	5
3.7 Administrative/Customization of the Application .....	6
3.8 Security Requirements .....	6
3.9 Data Reporting (Retrieval) Requirements .....	6
3.10 Data Storage (File Management) Requirements.....	6
4. Functional Overview.....	6
4.1 Static Description.....	6
4.2 Dynamic Description .....	7

# **1. Application Overview**

## **1.1 Purpose**

The purpose of this software requirements document is to plan, model, and state details of the functionality of E-Commerce Application. The document will state, in detail, what the software will be able to accomplish, the data it will store and retrieve, and how it will be organized. The intended audience is fellow software developers and the professors.

The purpose of the E-Commerce Application is solely educational. It will provide a stepping stone to familiarize the student with the .NET framework, C# syntax and mechanism concepts, development under the Visual Studio graphical IDE, and other object-oriented concepts. The application will feature object communication using object message passing and will provide a framework for these objects to reside on different machines in a distributed environment.

## **1.2 Value**

The application will contain no monetary value. However, the application will contain educational value as its development and maintenance will familiarize students with the Microsoft .NET framework. It will later be implemented in a distributed environment using various web services and, perhaps, providing some of its own.

## **1.3 Use**

The objective of the E-Commerce Application is to simulate a “Business to Consumer application.” It will provide a user, connected to a TTY, with a simulated e-commerce environment. The user, or customer, the main actor upon the system, will be able to browse and shop for five products. The user will be allowed to login, add items to their shipping cart, checkout, and provide personal information. The personal information pertaining to a customer’s order will include such information as shipping address, billing information, etc. The system will maintain a simulated warehouse of items, shipping department, purchasing, etc. The system will simulate the shipping of an order to a customer, the credit card authentication, and warehouse replenishment.

# **2. General Description**

## **2.1 Product Functions**

The software will be able to maintain a database of warehouse items and customers as well as initiate transactions among each entity. All transactions will take place at a TTY using a graphical or command line interface. It will provide the user, the main actor, to initiate transactions and have those transactions processed in a simulated E-Commerce environment. Functionality will be discussed in greater detail as the document continues.

## **2.2 Similar Systems Information**

The software will be a stand-alone system. The software will be monolithic in nature. The software will reside on a single machine and will interface with a local file system. There

will be no interaction between other modules as all work will be performed within the software's monolithic design.

### **3. Functional Abstract**

#### **3.1 General Functional Requirements**

1. The system must be able to be accessed via a TTY.
2. The system must provide a user interface (GUI or command line) for each entity interacting with the system.
3. The system must provide abstraction and maintain separate accounts for each user.
4. The system must be able to extract billing information from patients. System security is not within the scope of the system.
5. The system must be able to handle user input errors robustly.

#### **3.2 User Related Functional Requirements**

*The functionality discussed in this section specifies the actions the system will provide user.*

1. The system must provide for multiple users.
2. The system must provide the user with log in capabilities and retrieve specific user information, including attributes, billing information, and shipping information. This will be referred to as the user's "profile" thus forth.
3. The system must accept new user attribute information.
4. The system must maintain a username and password for each user.
5. The system must be able to accept new users.
6. The system must provide the user the ability to browse for items.
7. The system must provide the user with a temporary shopping cart that the user may add and subtract items from and to.
8. The user must be able to continue shopping after a shopping cart alteration.
9. The system must provide the user with means to checkout of the e-commerce application, thus converting the shopping cart to an order.
10. The system must be able to present the user with their choice of purchase at time of checkout.
11. The system must be able to calculate the total amount of a purchase and add New Jersey sales tax. There will be no charge for shipping.
12. The system must be able to authenticate credit card information either provided by the user at time of checkout or within the user's profile from past transactions.
13. The system may allow the user to view details of past orders/transactions.
14. The system may allow the user to alter their profile.
15. The system must allow the user to add additional billing information.

#### **3.3 Credit Card Authentication Functional Requirements**

*The functionality discussed in this section specifies the actions the system will provide for credit card authentication.*

1. The system must be able to access persistent data and attributes pertaining to credit cards.
2. The system must abide to the data representation provided by the assigned student's specifications.
3. The system must accept billing information from the user and return an approval or denial code based on that information.
4. The system must prompt the user for alternate billing information upon rejection.

### **3.4 Warehouse Related Functional Requirements**

*The functionality discussed in this section specifies the actions the system will provide to simulate a warehouse of items.*

1. The system must provide an abstraction for an item that will be purchased by the users.
2. The system must maintain and store attributes associated with different items.
3. The system must be able to retrieve attributes of items.
4. Of these attributes, the system must store the quantity of each item within the warehouse and contact the simulated "purchasing department" when that quantity is less than five.
5. The system will initially contain five different fixed items of which the only attribute that may be modified will be the quantity.
6. The system must be able to contact the simulated "shipping department" with information relating to a specific order.
7. The system must be able to "listen" for new items that are to be added to the warehouse from the purchasing department.
8. Upon an addition request from purchasing, the system must be able to update the warehouse with the addition quantities of items.

### **3.5 Purchasing Related Functional Requirements**

1. The system must be able to listen for requests from the warehouse to restock an item.
2. The system must be able to simulate item purchasing.
3. The system must be able to send information to the warehouse that the requested item has been restocked.

### **3.6 Shipping Related Functional Requirements**

1. The system must simulate a shipping department with the ability to "ship" orders consisting of specific items with their associated quantities to the associated user's selected shipping address.
2. The system must be able to store and retrieve information on the history of shipped orders.

### **3.7 Administrative/Customization of the Application**

1. System administrative tasks for the end-user will not be included in this version but may be included in future versions.

### **3.8 Security Requirements**

1. The system, as of this version, will not provide security features between user and system interaction. These features are considered outside the scope of the system.

### **3.9 Data Reporting (Retrieval) Requirements**

1. The system may provide minimal reporting of information of the user's profile.
2. Other types of reporting may be implemented in future systems.

### **3.10 Data Storage (File Management) Requirements**

1. The system must be able to store, maintain, and modify of orders.
2. The system must be able to store all attributes associated with each entity in a persistent and robust fashion.
3. The system may provide for updates of user information.
4. The system must be able to store entity changes.
5. The system must be able to retrieve entity information from the file system.

## **4. Functional Overview**

### **4.1 Static Description**

The following will provide a description of possible abstract data types (classes) in a static fashion. See the *Functional Specification* for a detailed version of this overview.

- Store
- Warehouse
- CreditCardProcessor
- Item
  - itemID : int
  - type : String
  - quantity : int
  - price : double
  - pictures : ArrayList
- Shipping
- Purchasing
- *StoreFront* (Abstract)
- GraphicalStoreFront
- CommandLineStoreFront
- User
  - Name : String
  - Phone : String

- email : String
- password : String
- userID : int
- CreditCard
  - number : int
  - expiration : DateTime
  - owner : String
- Address
- ShippingAddress
- BillingAddress
- Order
- ShoppingCart

## 4.2 Dynamic Description

The following will provide possible method declarations depicting system functionality in a dynamic fashion. See the *Functional Specification* for a detailed version of this overview.

*StoreFront*:: showLoginScreen() : void

Description:	Displays the login screen to the user.
Returns:	Void
Arguments:	Void

*Store*::loginUser(userName: String, password: String) : User

Description:	Attempts to login a user to the store. Returns a non-guest user from the database if successful, otherwise null.
Returns:	User: A non-guest user if exists, otherwise null.
Arguments:	userName: The user to attempt to log in. password: The user's password.

*Warehouse*:: getItem(itemNumber: int, itemQuantity: int) : Item

Description:	Returns a memory representation of a Item from the warehouse given a specific quantity. Throws an OutOfStock exception if the Item is out of stock.
Returns:	Item: A memory representation of the item.
Arguments:	itemNumber: The item's primary key. itemQuantity: The quantity of the item to retrieve.

*Warehouse:: addItem(anItem: Item) : void*

Description:	Adds an item to the warehouse.
Returns:	Void
Arguments:	anItem: The Item to add to the warehouse.

*Warehouse:: removeItem(itemNumber: Item) : void*

Description:	Removes an item from the warehouse given the item's primary key.
Returns:	Void
Arguments:	itemNumber: The primary key of the item to remove.

*Item:: addPicture(anIcon: ImageIcon) : void*

Description:	Add's a picture to the item's collection of pictures.
Returns:	Void
Arguments:	anIcon: The picture to add.

*Item:: removePicture(pos: int) : void*

Description:	Removes a picture from the item given it's position in the collection.
Returns:	Void
Arguments:	Pos: The pictures position in the collection.

*Shipping:: shipItem(anItem: Item, quantity: int, customer: User, address: int) : void*

Description:	Ships an item to a customer.
Returns:	Void
Arguments:	anItem: The item to ship. quantity: The quantity to ship. customer: The user to ship the item to. address: The primary key of the shipping address to send the item to.

*Purchasing:: purchaseItem(anItem: Item, quantity: int) : void*

Description:	Purchases a new item, given the quantity to reorder, for warehouse replenishment.
Returns:	void
Arguments:	anItem: The item to be purchased. quantity: The quantity of the item to be purchased.