Software Requirements Specification (SRS)

Book _Maza (Online Book Purchasing)

Authors: **Utkarsh Naidu (1541036)**

Vijay Palwade (1541038)

Ankush Patil (1541039)

Manasi Patil (1541040)

1 Introduction

The Software Requirements Specification is designed to document and describe the agreement between the customer and the developer regarding the specification of the software product requested [5]. Its primary purpose is to provide a clear and descriptive "statement of user requirements" [5] that can be used as a reference in further development of the software system. This document is broken into a number of sections used to logically separate the software requirements into easily referenced parts.

This Software Requirements Specification aims to describe the Functionality, External Interfaces, Attributes and Design Constraints [4] imposed on Implementation of the software system described throughout the rest of the document. Throughout the description of the software system, the language and terminology used should unambiguous and consistent throughout the document.

1.1 Purpose

Defining and describing the functions and specifications of the Book_maza is the primary goal of this Software Requirements Specification (SRS).
This Software Requirements Specification illustrates, in clear terms, the system's primary uses and required functionality as specified by our customer.

1.2 Scope

The software system being produced is called Book E-Commerce System or BECS. It is being produced for a customer interested in selling books via the Internet. This system is designed to "provide automation support" [2] for the process of placing books for sale on the Internet and facilitating the actual sale. This system is largely cross-platform and is available to anyone using the Computer Science Department's provided computer resources in the MSU Engineering Building. The system will be run on a central server with each user having a remote user interface through a web browser to interact with it.

The Book E-Commerce System will allow any user to create an account to become a customer. The customer, through the process of account creation, will have the option to become a member of the site. The system will allow customers to browse, search, select, and add books to a shopping cart. Then, provided they have books in their shopping cart, check out books in shopping cart and decrement the stock that the inventory the system maintains. The BECS also allows a manager to manage the inventory with full create, retrieve, update and delete (CRUD) functionality with regards to books in the system. It will also allow, on an inventory wide basis, customers and managers to interact with a promotion system that handles percentage-off promotions that can be applied to member's orders. This interaction includes the creation (by managers) and the application to orders (by customers) of the promotions. The BECS has full email capabilities; the automated email functionality will be used to send promotions to members of the system as well as provide the managers with low-stock notifications.

The BECS will have numerous constraints on what it can do. The system will not have full credit-card processing capabilities. It will not allow managers to be customers. The manager will be a hard-coded user and only a single manager will exist. There will be no actual book ordering and order completion, however the system will provide the customer with a receipt and it will log the transaction details. The system will not allow multiple promotions to be added to a single shopping cart nor will it allow a customer to add more than one of each item to their cart. The system also will not allow users to retrieve passwords or edit their user details.

1.3 Definitions, acronyms, and abbreviations

BECS	Book Maza Online Book Store
Barcode	A unique identifier assigned to single items
Book	An instance of an Item that has these additional attributes: Title, Author
Button	A user interface element that allows a User to click and inform the system to take an action
Checkbox	A user interface element that allows a User to inform the system that he/she selected a particular item
Checkout	The process a Customer goes through to purchase an Item
CRUD	Create, Retrieve, Update, Delete
Customer	A person that is a user of the system but has created an account
Inventory	An object that holds items available for purchase by the Customer
Item	An individual entity in the inventory which has several descriptive attributes:

	Barcode, Price, Reorder Threshold, Stock
Manager	A single person that has the ability to create, retrieve, update and delete items in the store. This person cannot simultaneously act as a Customer and Manager.
Member	A person that is a customer of the system and has requested to be sent promotions
Promotion	An item-wide percentage-off price discount applied to a Member's shopping cart
Reorder	The system process that automatically orders new stock of an item
Reorder Threshold	The numeric value of an item's stock that must be reached before the system will order additional quantities of the item
Session	The time which a User is actively using the system
Shopping Cart	An object that lists a Customer's selected Items, their applied promotions and gives them an option to check out
SRS	Software Requirements Specification
Stock	The quantity of any particular item the inventory has on hand
Text Box	A user interface element that allows a User to input text to the system
Transaction	The information related to a customer's purchase that is logged
User	The person who operate the software product.

1.4 Organization

This Software Requirements Specification document is divided in to multiple subsections. The first section includes explanations of the Purpose, Scope and Organization of the document. The first section also handles the description of project-specific words, acronyms and abbreviations that will be used in the document. The second section of the document is separated into the following five different sections, each detailing specific details of system uses and their corresponding actions: Product Perspective, Product Functions, User Characteristics, Constraints, Assumptions and Dependencies, Apportioning of Requirements. The third section is an enumerated listing of all of the requirements described for this system. The fourth section encompasses all of the Use-case, Sequence, State and Class diagrams that model the system. In the fifth section there exists a Prototype of the system along with a sample scenario that graphically describes the use of the system. The sixth section contains a listing of all related reference materials used in this document. The seventh and final subsection is dedicated to providing a point of contact for any viewer of this document.

2 Overall Description

This section includes details about what is and is not expected of the BECS system in addition to which cases are intentionally unsupported and assumptions that will be used in the creation of the BECS system.

2.1 Product Perspective

BECS is an online bookstore website which supports a number of functions for both the consumer and store's management.

The website must be available to anyone using the Computer Science Department's provided computer resources in the MSU Engineering Building and as such must work correctly in both Internet Explorer and Mozilla Firefox. As stated by the customer, there are no hardware or software requirements beyond these including, but not limited to, memory or specific software packages that need to be utilized nor software packages that need not be utilized.

2.2 Product Functions

BECS will provide a number of functions; each is listed below.

- Maintain data associated with the inventory (a collection of books)
  - A book has a title, author and price
  - The inventory also keep track of the stock/quantity of each book
- Maintain records for many customers
  - A customer can be either a member or non-member.
  - A customer has a username (unique across all users), password (no restrictions), email address (no restrictions), and postal address (unverified.)
  - Anyone may sign up for a customer account.
- Allow any customer to become a member.
- Show a listing of available books
  - Books are to be displayed in ascending alphabetical order by title.
  - Each book will list the following from left to right
    - Title
    - Author
    - Price
- Allow customers and managers to log in and out of the system.
  - Users (both customers and the manager) will be logged out if inactive for 30 minutes.
- Shopping cart
  - Anyone is able to add one or more books to the shopping cart.
    - The shopping cart does not need to allow multiple copies of any book.
- Checkout
  - Checkout is only available to logged-in customers. A user that is not logged in as a customer is given a chance to log in.
  - Member customers may enter a promotion code.
  - Only one promotion code may be used per purchase
    - The promotion is a fixed percentage discount that is to be applied to an entire order.
    - The discount is specified by the manager at the time of the promotion's creation or most recent update/edit.
  - Collect a 16-digit credit card number from the customer
  - Log/record the transaction
- Allow manager to specify a stop-order for a book
  - Each book has its own stop-order status – either on or off. Details of its use are involved in the following feature.
- Notify manager when books need to be reordered
  - When the quantity a book falls below a threshold, the manager is notified that the book

needs to be reordered.
        - One exception is if the manager has already specified a stop-order for this book.
            - Every book must either have stop-order enabled or disabled
- Allow manager to update stock quantities
        - Allow manager to change any book's price
        - Allow manager to view transaction logs
    - Allow manager to create promotions
        - A promotion is a percentage discount that can be applied to an entire order
            - Promotions may only be used by member customers
            - A promotion has an expiration date specified by the manager

When a promotion is created, it is emailed to all member customers via the email address on record

2.3 User Characteristics

The typical BECS user is simply anyone that has access to the Internet and a web browser in the computer science department at Michigan State University. It is assumed that the user is familiar enough with a computer to operate the browser, keyboard and mouse and is capable of browsing to, from and within simple websites [1].

2.4 Constraints

As stated by the customer, security is not a concern for this system. The database may store passwords in plain text and there doesn't need to be a password recovery feature nor lockout after numerous invalid login attempts. As such, the system may not work correctly in cases when security is a concern. These cases include those listed above in addition to lack of an encrypted connection when sending credit card information and forcing users to use “strong” passwords. A strong password is a password that meets a number of conditions that are set in place so that user's passwords cannot be easily guessed by an attacker. Generally, these rules include ensuring that the password contains a sufficient number of characters and contains not only lowercase letters but also capitals, numbers, and in some cases, symbols.

The system may not behave correctly when used with Internet browsers other than Firefox and Internet Explorer.

SCR	"Software Cost Reduction (SCR) is a set of techniques for designing software systems developed by David Parnas and researchers from the U.S. Naval Research Laboratory beginning in the late 1970s." [7]
Mode Class	"A mode class is a finite state machine, with states called system modes" [8]
System State	The current state or mode that the system is in. The system must be in exactly one state at any moment in time.
Event	An event is any action that can trigger an action within the software system. Examples include but are not limited to changing values of variables or user-triggered events.
Event Notation	We may need to refer to both the old and new value of a

	variable: Used primed values to denote values after the event @T(c) ≡ ¬c ^ c' e.g. @T(y=1) ≡ y<>1 ^ y'=1 @F(c) ≡ c ^ ¬c' A conditioned event is an event with a predicate @T(c) WHEN d ≡ ¬c ^ c' ^ d [8]
Controlled Variable	A variable whose value can change throughout the lifetime of the system and whose value is critical and must be maintained correctly.
Mode Transition	When the mode (state) changes from one mode (described as the old mode) to a new mode.
Mode Class Table	Table consisting of a list of modes that the system can be in, modes that can be transitioned to, and the conditions required for the transition to occur. The table if formatted such that the first column lists the current mode (old mode) and the last column lists the new mode. The columns between the first and last columns each describe a specific event.
Event Table	An event table illustrates how an input event can affect a controlled variable. The first column shows modes and the last row shows the values that the controlled variable will be set to. The remaining cells are conditions required for a mode to affect the value of a controlled variable.
Condition Table	A condition table shows the conditions (one of which must be met) in order for a controlled variable to be set to some specified value. The first column lists modes and the last row names the controlled variable and the values it is set to.

Mode Class						
Old Mode	**Checkout**	**CheckoutFinished**	**AddPromotion**	**IsMember**	**AddPromotionFinished**	**New Mode**
CustomerLoggedIn	@T	-	-	-	-	**CheckingOut**
CheckingOut	-	@T	-	-	-	**CustomerLoggedIn**
	-	-	@T	t	-	**AddingPromotion**
AddingPromotion	-	-	-	-	@T	**CheckingOut**

Event Table		
AddPromotion	@T(AddingPromotion() == TRUE)	@T(AddingPromotion() == FALSE)
ShoppingCart::PromotionAdded	**TRUE**	FALSE

Condition Table

Mode	**Conditions**	
AddPromotion	TRUE	FALSE
PromotionAdded	**TRUE**	**FALSE**

Mode Class

Old Mode	**Login**	**IsManager**	**IsCustomer**	**Logout**	**New Mode**
UserLoggedOut	@T	t	-	-	**ManagerLoggedIn**

	@T	-	t	-	**CustomerLoggedIn**
ManagerLoggedIn	-	-	-	@T	**UserLoggedOut**
CustomerLoggedIn	-	-	-	@T	**UserLoggedOut**

Event Table

UserLoggingIn	@T(User::Login() == TRUE)	never
UserLoggingOut	never	@T(Logout() == TRUE)
User::LoggedIn	**TRUE**	**FALSE**

Condition Table

Mode	**Conditions**	
UserLoggedOut	FALSE	TRUE
ManagerLoggedIn	TRUE	FALSE
CustomerLoggedIn	TRUE	FALSE
UserLoggedIn	**TRUE**	**FALSE**

2.5 Assumptions and Dependencies

Client:
We have assumed that all of the computer systems in the Engineering building labs are in proper working condition and that the user is capable of operating these system's basic functions including but not limited to being able to power on the system, login and open either Internet Explorer or Mozilla Firefox, and navigate the browser to the address of this BECS website.

Provider:
We have assumed that the BECS will be running on a properly working web server and database system with an Internet connection that allows this system to perform all communications with clients.

Assumptions:

- There is no need for anyone to be able to order more than a single copy of a book (or any item) in a single transaction.
- The manager account's username and password maybe hard coded.
- The manager cannot be a customer.
- Any user cannot edit their account information.

2.6 Approportioning of Requirements

As stated by the customer, security is not a concern of this project. As such, it is beyond the scope of this system to encrypt personal user data, encrypt credit card information, prevent unauthorized login attempts, or any other concern of this nature. Additionally, the system is not responsible for the following:

- Verifying that credit card information is valid

- Verifying the email address provided by a user
- Storing additional information about a book beyond simply the title, name of author, and price
- Allowing users to edit their account details (username, password, mailing address, etc)
- Allowing customers to order multiple copies of a book in a single order
- Providing individual product pages (one page for every item in the inventory)
- Allowing the manager to update login credentials or other information about the manager

Additionally, the system may need to later be extended to provide additional functions. One such example is added support for visually impaired users. In many cases a screen-reading program is used and ensuring that page-layout reads from top-left to bottom-right in a logical manner would be required.

3 Specific Requirements

1. Restrictions
   - 1.1. User Side
     - 1.1.1. Software
       - 1.1.1.1. Internet Explorer or Mozilla Firefox
     - 1.1.2. Hardware
       - 1.1.2.1. Computer Science Department Laboratory Terminal
   - 1.2. System Side
     - 1.2.1. Software
       - 1.2.1.1. Web-based application
       - 1.2.1.2. Database information storage system
2. Data Structure
   - 2.1. Book has these attributes
     - 2.1.1. Unique ID (auto-increment starting at 1)
     - 2.1.2. Title
     - 2.1.3. Author
     - 2.1.4. Price
     - 2.1.5. Reorder Threshold
     - 2.1.6. Stop-order Boolean value
     - 2.1.7. Stock
   - 2.2. Customer has these attributes
     - 2.2.1. Unique Username
     - 2.2.2. Password
     - 2.2.3. Name
     - 2.2.4. Email Address
     - 2.2.5. Postal Address
     - 2.2.6. Member/Not Member Boolean value
   - 2.3. Manager has these attributes
     - 2.3.1. Username
     - 2.3.2. Password
     - 2.3.3. Email address
   - 2.4. Order log entries have these attributes:
     - 2.4.1. Unique ID (auto generated)

2.4.2. Time transaction took place
2.4.3. Date transaction took place
2.4.4. Username of customer
2.4.5. Listing of the contents in customer's shopping cart

3. System
   3.1. Browse Inventory
      3.1.1. Organization
         3.1.1.1. Items Listed on single page
         3.1.1.2. Items shown in tabular format
         3.1.1.3. Each Item listing contains
            3.1.1.3.1. Title
            3.1.1.3.2. Name of Author
            3.1.1.3.3. Price
         3.1.1.4. Listing sorted by Ascending item Title
         3.1.1.5. No individual Item pages
      3.1.2. Interaction
         3.1.2.1. Each Item has checkbox to mark selection
         3.1.2.2. Single button to add all selected items to Shopping Cart
   3.2. Search Inventory
      3.2.1. Search available only by Title of book
      3.2.2. Search is exact-match only
   3.3. Create, Update and Destroy (CRUD) Functionality
      3.3.1. Only managers are allowed to modify inventory
      3.3.2. Managers have an interface to:
         3.3.2.1. Create a book entry
         3.3.2.2. Update a book entry
         3.3.2.3. Update the stock/quantity of a particular book
         3.3.2.4. Create a new promotion
         3.3.2.5. Review current inventory
            3.3.2.5.1. Using the same interface to browse inventory as described in section 3.1, the manager has an additional "Edit Item" option for each book.
               3.3.2.5.1.1. Manager has full CRUD capabilities on each book.
      3.3.3. Managers may delete items from the inventory
   3.4. Shopping Cart
      3.4.1. Logged In
         3.4.1.1. Can add items to cart
            3.4.1.1.1. If Item is not in stock, message displayed informing user to try again later
            3.4.1.1.2. Customer can only purchase one of each item (no quantities associated with orders)
            3.4.1.1.3.
         3.4.1.2. If shopping cart not empty, a user may begin Checkout procedure
      3.4.2. Not Logged In
         3.4.2.1. Can add items to cart
         3.4.2.2. User required to login before they may begin Checkout procedure
   3.5. Checkout procedure
      3.5.1. User must successfully use shopping cart before beginning this procedure
      3.5.2. Checkout page consists of
         3.5.2.1. A text box for promotion entering
         3.5.2.2. An overview of the purchase

3.5.2.3. A text box to hold the credit card number
3.5.2.4. A button to complete the order
3.5.3. Order details sent via email after the checkout has completed
3.5.4. On order completion the inventory is decremented based on items purchased by user
3.6. Authentication System
3.6.1. User Levels
3.6.1.1. Manager (single, hardcoded user, no orders)
3.6.1.2. Customer (unlimited, open creation, unlimited orders)
3.6.2. Account Creation
3.6.2.1. Everyone is allowed to create an account
3.6.2.2. Required Information
3.6.2.2.1. Listed in section **2.2**
3.6.3. Account Modification
3.6.3.1. Users are not able to modify any aspect of their account after creation ("it would be nice but not needed")
3.6.4. Login and Logout
3.6.4.1. There is no lost-password recovery
3.6.4.2. Logging in allows one to logout
3.6.4.3. Logging in allows checkout
3.6.4.4. There is a 30-minute session time out after which a logged in user will be logged out automatically.
3.7. Promotions
3.7.1. Specifications
3.7.1.1. Applies to entire order
3.7.1.2. Percentage-off type promotion (x% off entire order)
3.7.1.3. Expiration occurs at manager specified date
3.7.1.4. Multiple coupons cannot be applied to same order
3.7.1.5. Non-member users cannot apply promotions to order
3.7.2. Creation
3.7.2.1. Promotion created by manager
3.7.2.2. Each promotion has a unique identifying number (can be auto generated)
3.7.2.3. Email containing promotion sent to all member users of the BECS system
3.7.2.4.
3.7.3. Deletion
3.7.3.1. Promotions are auto-deleted when the expiration date has passed
3.8. Automated Reorder
3.8.1. Specifications
3.8.1.1. Manager sets reorder threshold on a per-item basis
3.8.1.2. If item reaches the reorder threshold, an email is sent informing the manager of the item's status and the system automatically reorders the item
3.8.1.2.1. If the item has a stop-order applied to it, it will not automatically reorder until the manager removes it.
3.8.1.3. A manager may increase the stock of an item using the manager's account
3.9. Order Logging
3.9.1. Specifications
3.9.1.1. Required Information:
3.9.1.1.1. Listed in section **2.4**
3.9.1.2. A manager can view all past transactions from all users

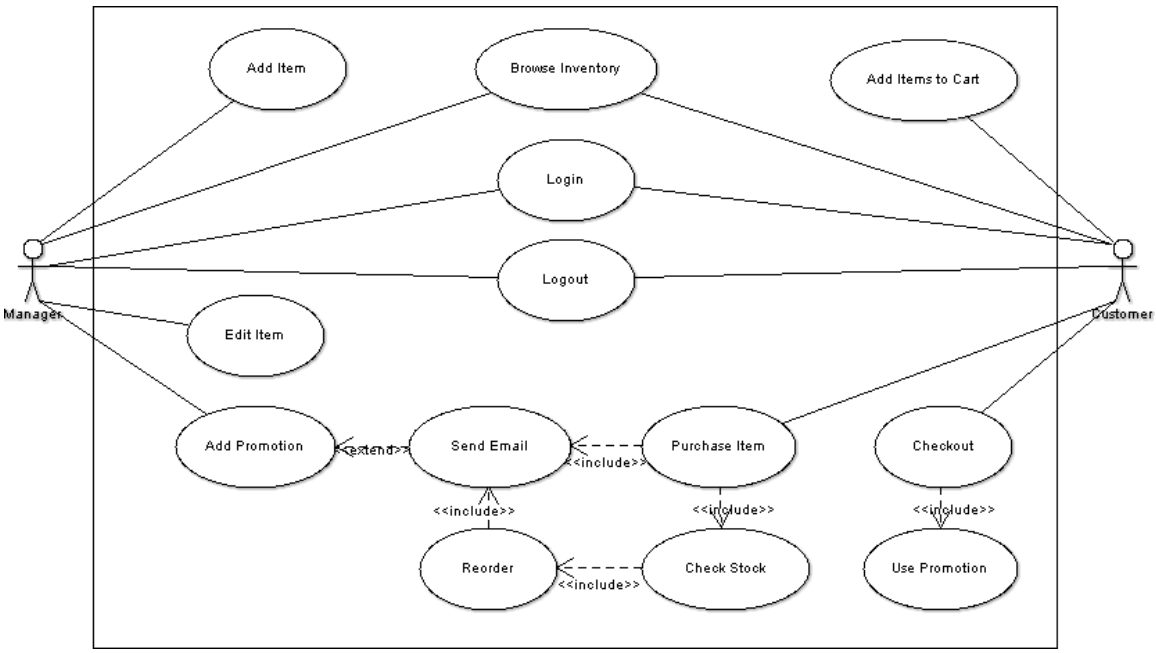
Order log entries are generated when a user successfully checks out their shopping cart

4 Modeling Requirements

Use Case Diagram

The purpose of this diagram is to demonstrate how objects will interact with BECS and map out the basic functionality of the system. Below is a list of the elements that you will see in the diagram on the next page as well what is included in the use case templates that follow.

Actors	Shown in the diagram as stick figures with a name underneath. They represent elements that will be directly interacting with the system.
Use Cases	Oval shapes that have their names in the center. These represent direct functionality within the system that must be implemented.
Interactions	Lines that connect the actors with the different Use Cases. These show that there is some form of direct interaction between the actor and that specific functionality.
Includes	Dotted lines labeled "<<include>>" that connect two use cases and have an arrow pointing towards one. This means that the use case without the arrow calls on the functionality of the use case with the arrow.
Extends	Dotted lines labeled "<<extend>>" that connect two use cases and have an arrow pointing towards one. This means that the use case without the arrow takes all of the functionality of the use case with the arrow and adds extra functionality.
The System Boundary	The large rectangle that contains the Use Cases. Everything within the rectangle is what the system is responsible for implementing
Use Case Template	Describes the basic functionality and features of each use case and the can be found in the pages following the use case diagram.
Type	A field in the use case template that states whether or not the use case is directly interacted with by an actor (Primary) or not (Secondary) as well as whether or not it is essential to having a functioning system.
Cross Ref	A field in the use case templates that states which one of the original requirements that particular use case satisfies.
Use-Cases	A field in the use case templates that state which other use cases must be executed prior to that particular use case.



Login
Actors: Manager, Customer
Type: Primary and essential
Description: Initiated when a user attempts an action that is restricted. The user is then prompted to enter in their username and password in order to proceed.
Includes: None
Extends: None
Cross Ref: Required for 2
Use-Cases: None

Use Case: **Logout**
Actors: Manager, Customer, System
Type: Primary and essential
Description: The customer or manager will have the option to logout and if that user is inactive for a given amount of time then that user should be logged out by the system automatically.
Includes: None
Extends: None
Cross Ref: Required for 3.61
Use-Cases: User must have completed the Log In use case.

Use Case: **Browse Inventory**
Actors: Manager, Customer
Type: Primary and Essential
Description: All the books in the inventory are listed on a single page with each book including its title, name of author, and price. List should be sorted by title.
Includes: None

Extends: None
Cross Ref: Required for 4
Use-Cases: None

Use Case: **Add Items to Cart**
Actors: Customer
Type: Primary and Essential
Description: Allows the Customer to place items selected in the Browse Inventory screen to their shopping cart for later purchase.
Includes: None
Extends: None
Cross Ref: Required for Elicitation Meeting
Use-Cases: Customer must have completed the Log In use case.

Use Case: **Add Item**
Actors: Manager
Type: Primary and Essential
Description: Allows the Manager to add an additional book to the inventory that should include the books price, title, number in stock, stop-order, and reordering threshold.
Includes: None
Extends: None
Cross Ref: Required for 1
Use-Cases: Manager must have completed the Log In use case

Use Case: **Edit Item**
Actors: Manager
Type: Primary
Description: Lets the Manager edit all of the attributes of a particular item in the inventory.
Includes: None
Extends: None
Cross Ref: Required for 1 and 5
Use-Cases: Manager must have completed the Log In use case

Use Case: **Add Promotion**
Actors: Manager
Type: Primary
Description: This allows the manager to add a special promotion such as a certain percentage off for members. This will email all customers who are members to inform them of the new promotion.
Includes: None
Extends: None
Cross Ref: Required for 5
Use-Cases: Manager must have completed the Log In use case

Use Case: **Checkout**
Actors: Customer

Type: Primary and Essential
Description: This takes the items in the customers shopping cart and processes them for a purchase.
Includes: Use Promotion
Extends: None
Cross Ref: Required for 3
Use-Cases: Customer must have completed the Log In use case

Use Case: **Use Promotion**
Actors: Customer
Type: Primary
Description: If the user is a member they are presented with the option to enter in a promotion code that will take off a percentage from the total.
Includes: None
Extends: None
Cross Ref: Required for 6
Use-Cases: Customer must have completed the Log In and Checkout use cases

Use Case: **Purchase Item**
Actors: Customer
Type: Secondary
Description: Acted on when the user presses the finalize order button in checkout. This decrements the inventory of all items within the order, email the user, create a log of the transaction, and check stock to see if a reorder needs to take place.
Includes: Send Email, Check Stock
Extends: None
Cross Ref: Required for 3 and 7
Use-Cases: Customer must have completed the Log In and Checkout use cases

Use Case: **Check Stock**
Actors: System
Type: Secondary
Description: Checks to see if stop-order is on for a particular item and if it is checks to see if the amount in stock is below the reorder amount. If it is then it will reorder.
Includes: Send Email, Reorder
Extends: None
Cross Ref: Required for 5
Use-Cases: None

Use Case: **Reorder**
Actors: System
Type: Secondary
Description: Reorders a particular item and emails the manager.
Includes: None
Extends: None
Cross Ref: Required for 5
Use-Cases: None

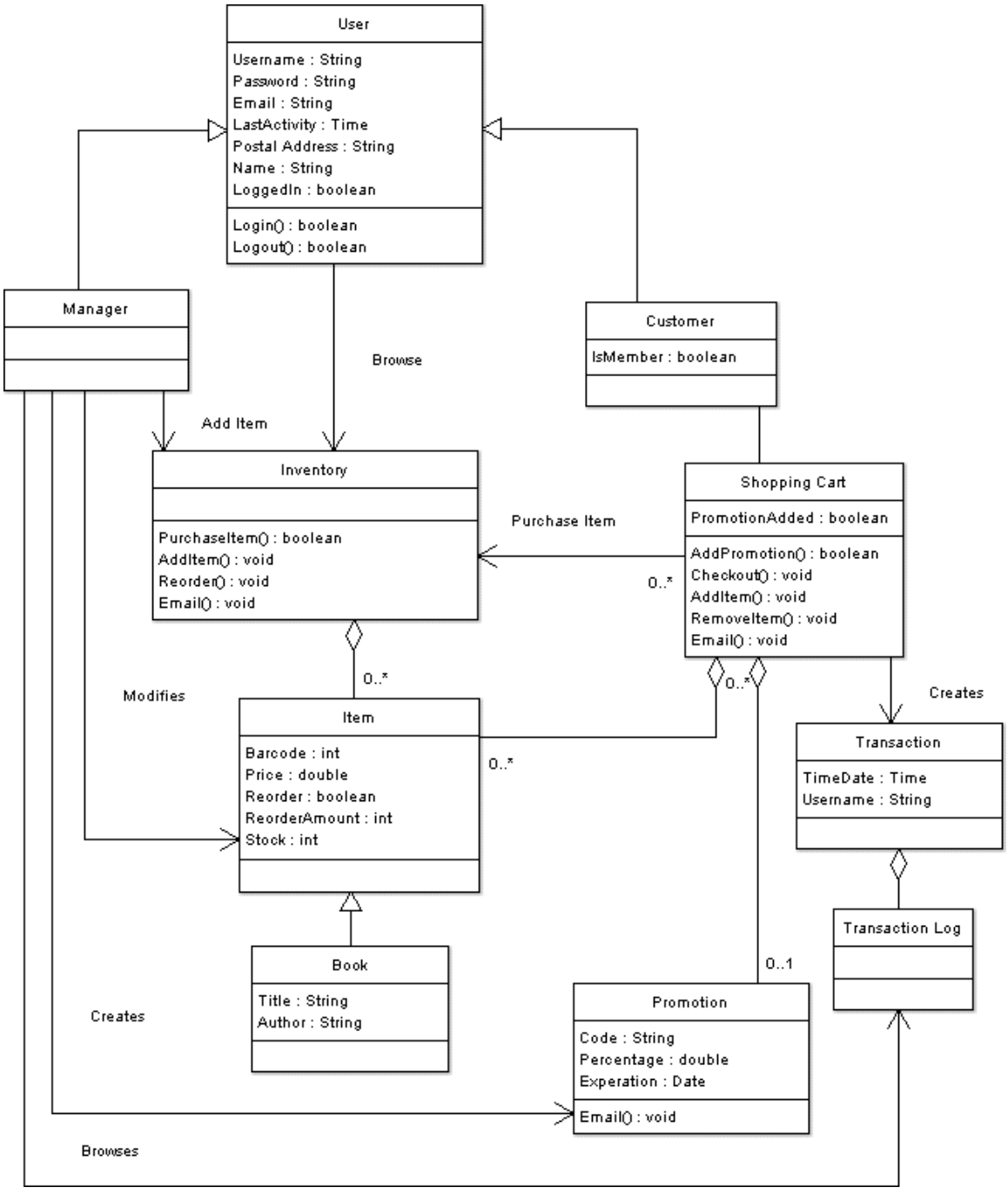
Use Case: **Send Email**
Actors: System
Type: Secondary
Description: This is called by a variety of other use cases whenever an email needs to be sent.
Includes: None
Extends: Add Promotion
Cross Ref: Required for 5, 6, and Elicitation Meeting
Use-Cases: None

Class Diagram

The purpose of this diagram is to show how objects within the BECS system will interact with each other in order to achieve the functionality required by the Use Case diagram. Below is a list of what you will see in the diagram itself as well as the class descriptions that follow.

Classes	Rectangles in the diagram that are split into three parts. The top section is the name of the class, the middle section is the list of variables that are stored in the class and the bottom section is the list of functions in the class. These rectangles represent objects within the system.
Variables	These have a name followed by a semicolon and then a type. The type denotes what kind of data can be stored in the variable.
Functions	These have a name followed by a list of any variable that the function receives in-between the parenthesis "()". After that there is a semicolon and any variables that the function may return, if none it will be void.
Generalizations	Shown using a line from one object to the other with an unfilled triangle on one end. The object without the triangle inherits the functionality and variables from the object that has the triangle pointing towards it.
Aggregations	Lines that have an unfilled diamond on one end. This means the object with the diamond contains the object(s) without the diamond. This may have numbers on the ends (multiplicities).
Associations	Lines connecting two classes that can have a name beside it, may point in one direction, and may have numbers at the ends (multiplicities). These designate some relationship between the objects. Arrows are simply there to assist you in recognizing which direction the name of the association is read.
Multiplicities	Numbers that may be on the ends of Aggregations and Associations. They state how many of the one object can be

	related to the other. The first number is the minimum and the second number is the maximum. An asterisk '*' means many, so "1..*" can be read as 1 to many. If no number exists it is assumed to be 1.



Class	

User	***This is a base class in which manager and customer extend. This class provides the login ability that is shared between the two as well as some shared variables.***	
	Public: Yes	
	Relationships	Associations: None
		Aggregations: None
		Generalization: None
	Variables - Username:String, Password:String, Email:String, LastActivity:Time, PostalAddress:String, Name:String, LoggedIn:Boolean	
	Functions - Login(), Logout()	

Class		
Manager	***There is only one instance of this class because there is only one manager. This manager has the additional ability of adding items, creating promotions, and modifying books.***	
	Public: No	
	Relationships	Associations: Inventory, Book, Promotion
		Aggregations: None
		Generalization: User
	Variables - None	
	Functions - None	

Class		
Customer	***There are many customers in the system and they all have their own shopping cart. A customer is a member if IsMember is set to True.***	
	Public: No	
	Relationships	Associations: Shopping Cart
		Aggregations: None
		Generalization: User
	Variables - IsMember:Boolean	
	Functions - None	

Class		
Inventory	***This class contains a list of items and all the functions that are required to be acted on those items. The functions AddItem(), SetPrice(), and SetReorder() are only accessible by the Manager. PurchaseItem() only by a Shopping Cart. Browse() can be accessed by any User.***	
	Public: No	
	Relationships	Associations: User, Manager, Shopping Cart
		Aggregations: None
		Generalization: None
	Variables - Items:Vector	

	Functions - PurchaseItem(), AddItem, Reorder(), SetPrice(), SetReorder, Browse(), Email()

Class		
Item	***This is a single item in which there are many in the system and they are all part of the inventory. They can also be a part of shopping carts and a transaction.***	
	Public: No	
	Relationships	Associations: None
		Aggregations: Inventory, Transaction, Shopping Cart
		Generalization: None
	Variables - Barcode:Int, Price:Double, Reorder:Boolean, ReorderAmount:Int, Stock:Int	
	Functions - CheckStock()	

Class		
Book	***This is a specific type of Item that has the additional values of Title and Author and it can be modified by a manager.***	
	Public: No	
	Relationships	Associations: Manager
		Aggregations: None
		Generalization: Item
	Variables - Title:String, Author:String	
	Functions - None	

Class		
Promotion	***The manager can create a promotion and when it is created an email is sent to all customers who are members to inform them of the new promotion. The promotion is for a percentage off of an order and they have unique codes and expiration dates. The promotion can be added to a customer's shopping cart.***	
	Public: No	
	Relationships	Associations: Manager
		Aggregations: Shopping Cart
		Generalization: None
	Variables - Code:String, Percentage:Double, Experation:Date	
	Functions - Email()	

Class	
Transaction	***This is created when an order is processed and it is the record of the order which includes who placed the order, the time and date it was placed, and the list of items that were purchased.***
	Public: No

	Relationships	Associations: Shopping Cart
		Aggregations: None
		Generalization: None
	Variables - TimeDate:Time, Username:String, Items:Vector	
	Functions - None	

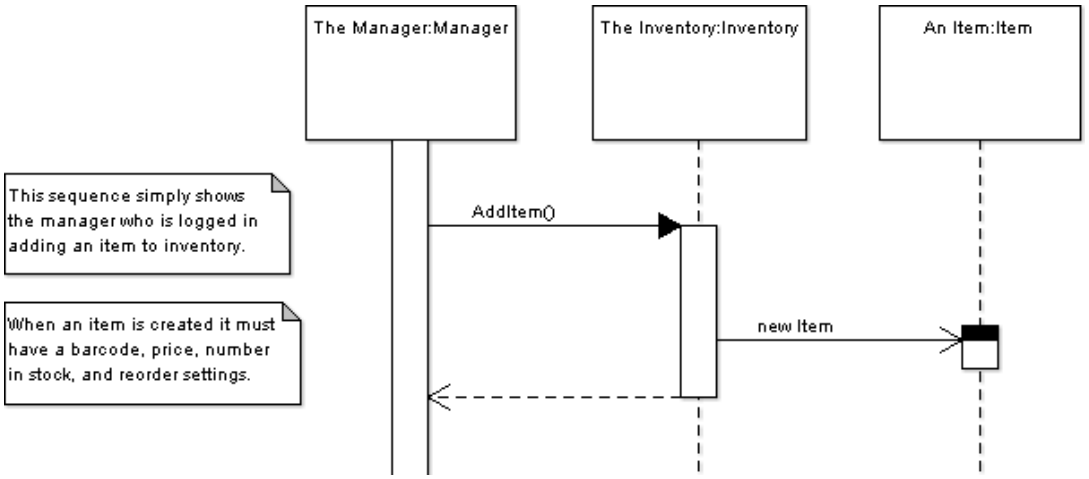
Class		
Shopping Cart	***Each customer has its own shopping cart and it stores items and is responsible for processing the order for the customer.***	
	Public: No	
	Relationships	Associations: Customer, Inventory
		Aggregations: None
		Generalization: None
	Variables - Items:Vector, PromotionAdded:Boolean	
	Functions - AddPromotion(), Checkout(), AddItem(), RemoveItem(), Email(), BrowseCart()	

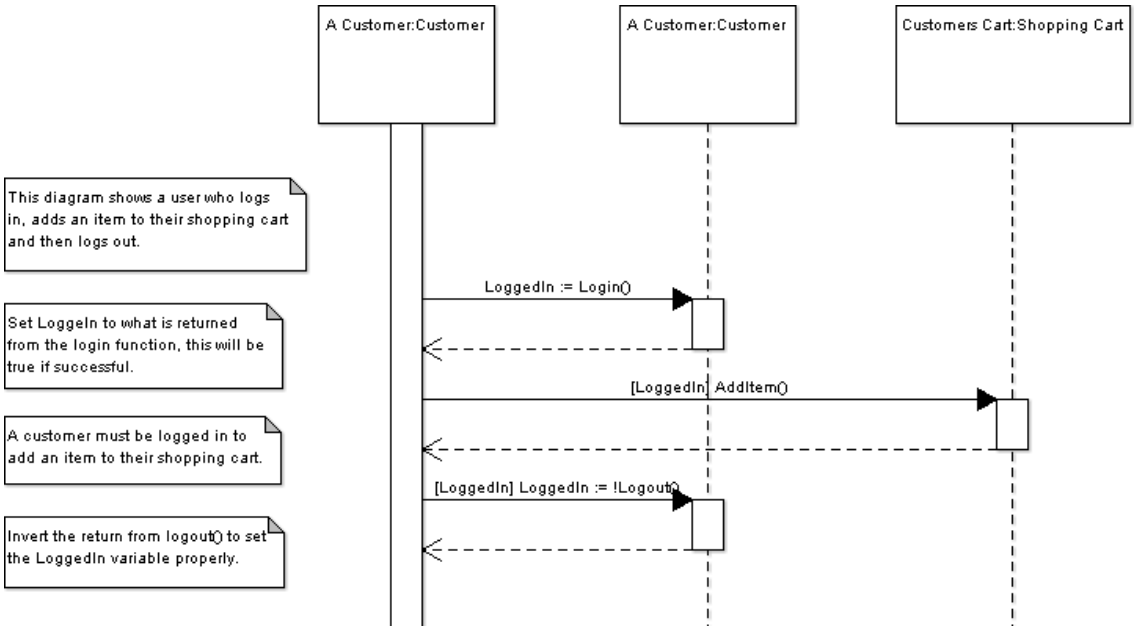
Sequence Diagrams

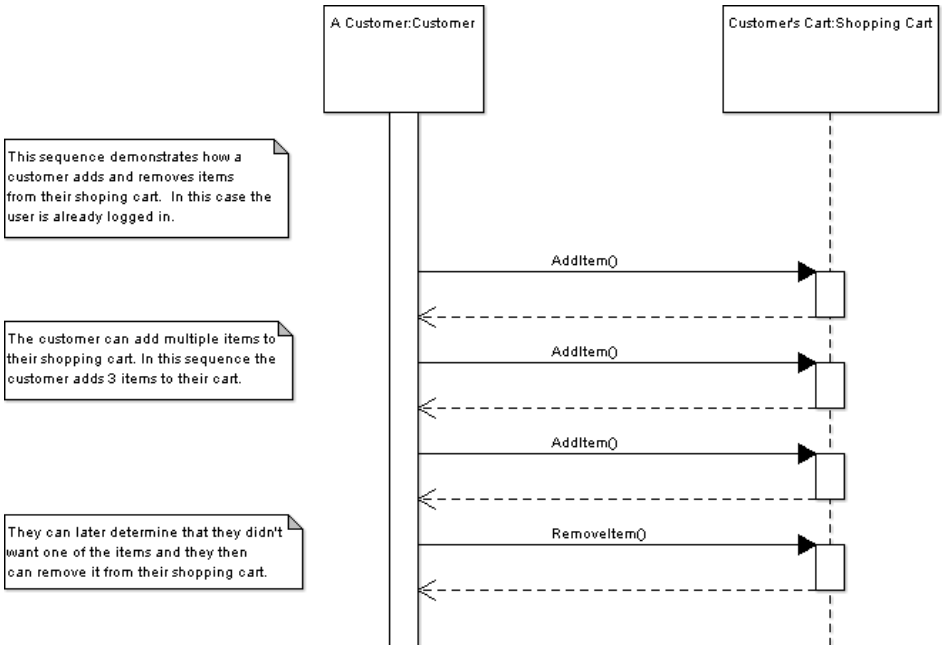
The sequence diagrams use the class diagram and demonstrate specific sequences of actions in the system. The purpose is to ensure that the BECS system runs in an expected way and that the class structure is sufficient to accomplish the tasks needed. Below is a list of the items that you will see in the diagram and their definitions.

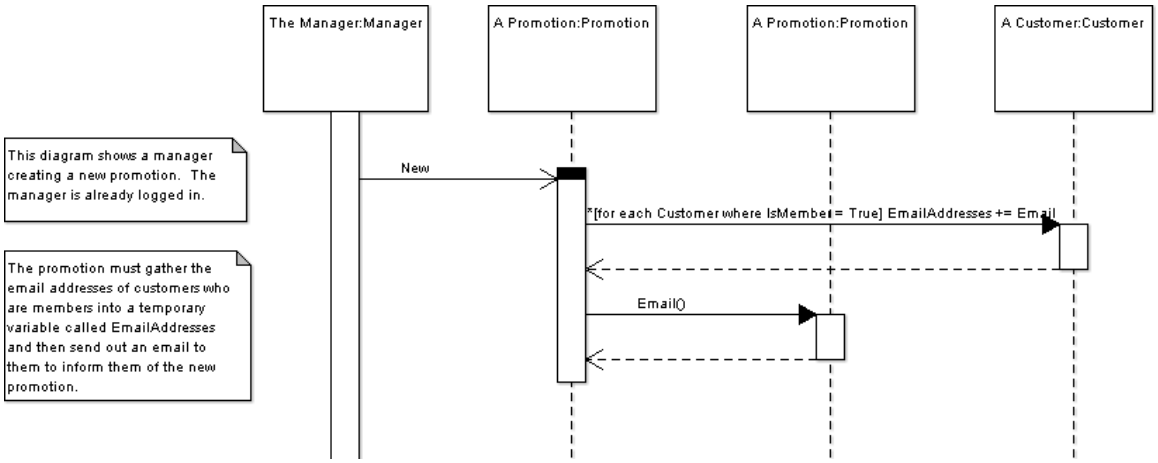
Axis's	The x-axis identifies movement between objects and the y-axis identifies time.
Comments	These are the boxes that are along the left side. These explain the actions that are occurring in the sequence and other helpful information.
Instances	Solid boxes along the top that have dotted lines that stretch vertically below them. These are specific instances of an object. The first part of the title is the name of that specific instance and the object it is an instance of follows it. (Special Note: If there are multiple instances that have the same title then they are actually the same instance and are only there to diagram calls onto themselves.)
Calls	Lines that have filled triangles at the ends. These are transitions from one instance to another and have a label above them that is a function call, a variable being set, or both. It can also have a guard statement that precedes it.
Variable being set	Have a variable name followed by either a ':=', '+=', or '-=' and then another variable name with the first being set to the

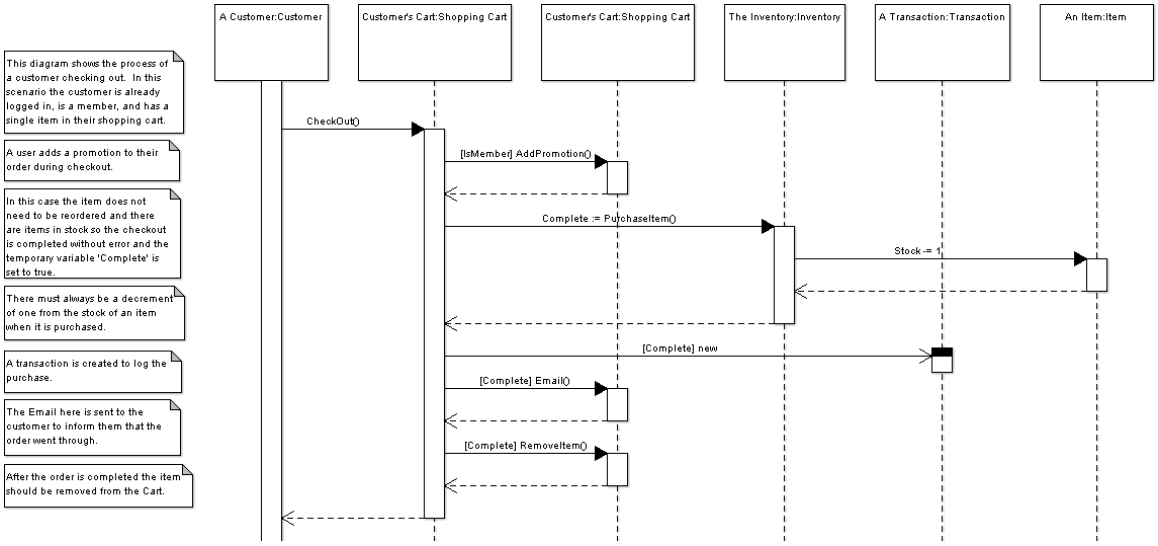
	later.
:=	means that its being set directly to the variable that follows.
+=	means that the variable that follows is being added to the current value.
-=	means that the variable that follows is being subtracted from the current value.
Guard Statements	These are located between brackets ([])and come before the function in a call. This means that the condition must be true in order to make that call.
Returns	These are represented by dotted lines with an arrow at the end. These simply represent a return from a function call.
Object Execution Time	This is shown with the solid white boxes that run vertically along the dotted lines. These simply represent the execution time for the objects. (Special Note: the first object has a solid box all the way down this is a special case and should not be there and is due to the application used to create the diagram).

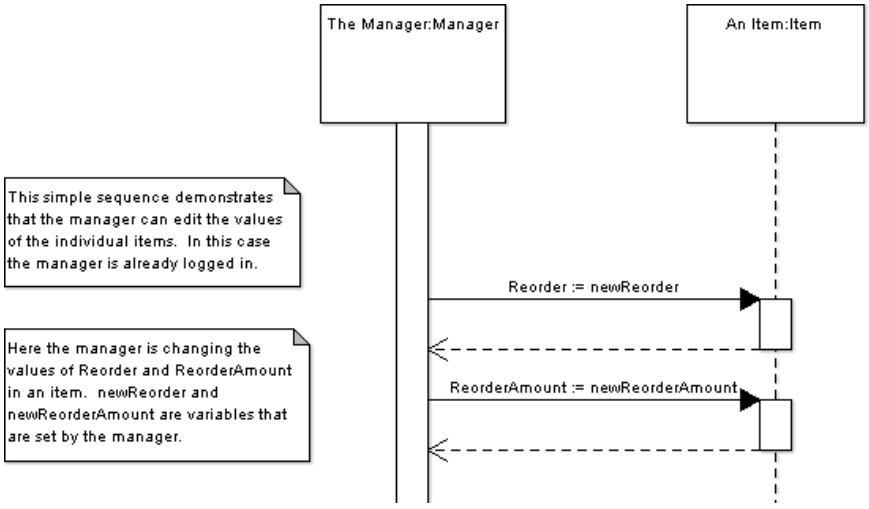



A Customer:Customer
A Customer:Customer
Customers Cart:Shopping Cart
This diagram shows a user who logs in, adds an item to their shopping cart and then logs out.
LoggedIn := Login()
Set LoggeIn to what is returned from the login function, this will be true if successful.
[LoggedIn] AddItem()
A customer must be logged in to add an item to their shopping cart.
[LoggedIn] LoggedIn := !Logout()
Invert the return from logout() to set the LoggedIn variable properly.

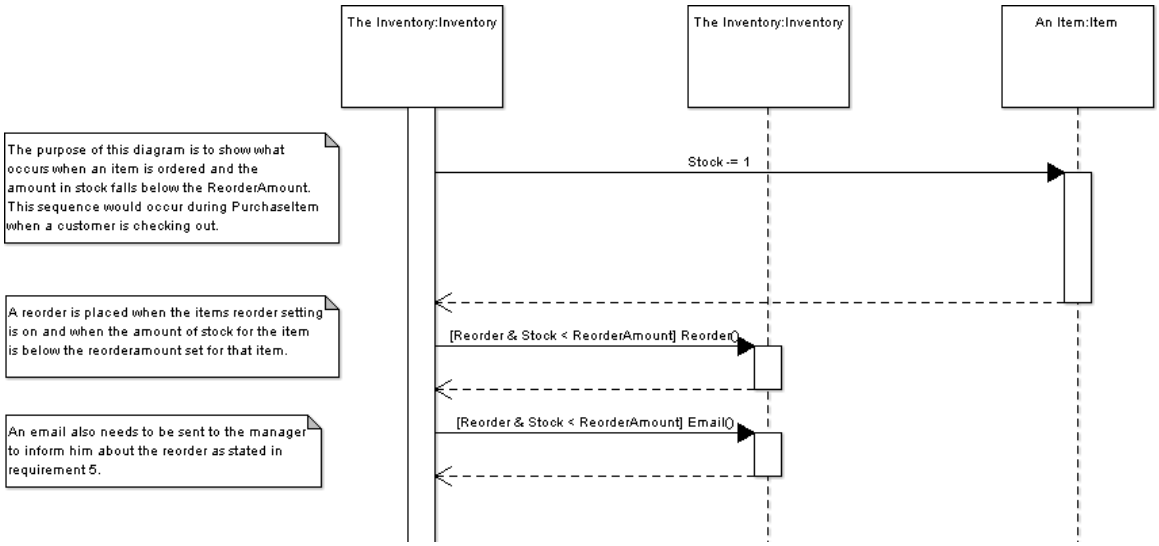

A Customer:Customer
Customer's Cart:Shopping Cart
This sequence demonstrates how a customer adds and removes items from their shoping cart. In this case the user is already logged in.
AddItem()
The customer can add multiple items to their shopping cart. In this sequence the customer adds 3 items to their cart.
AddItem()
AddItem()
They can later determine that they didn't want one of the items and they then can remove it from their shopping cart.
RemoveItem()







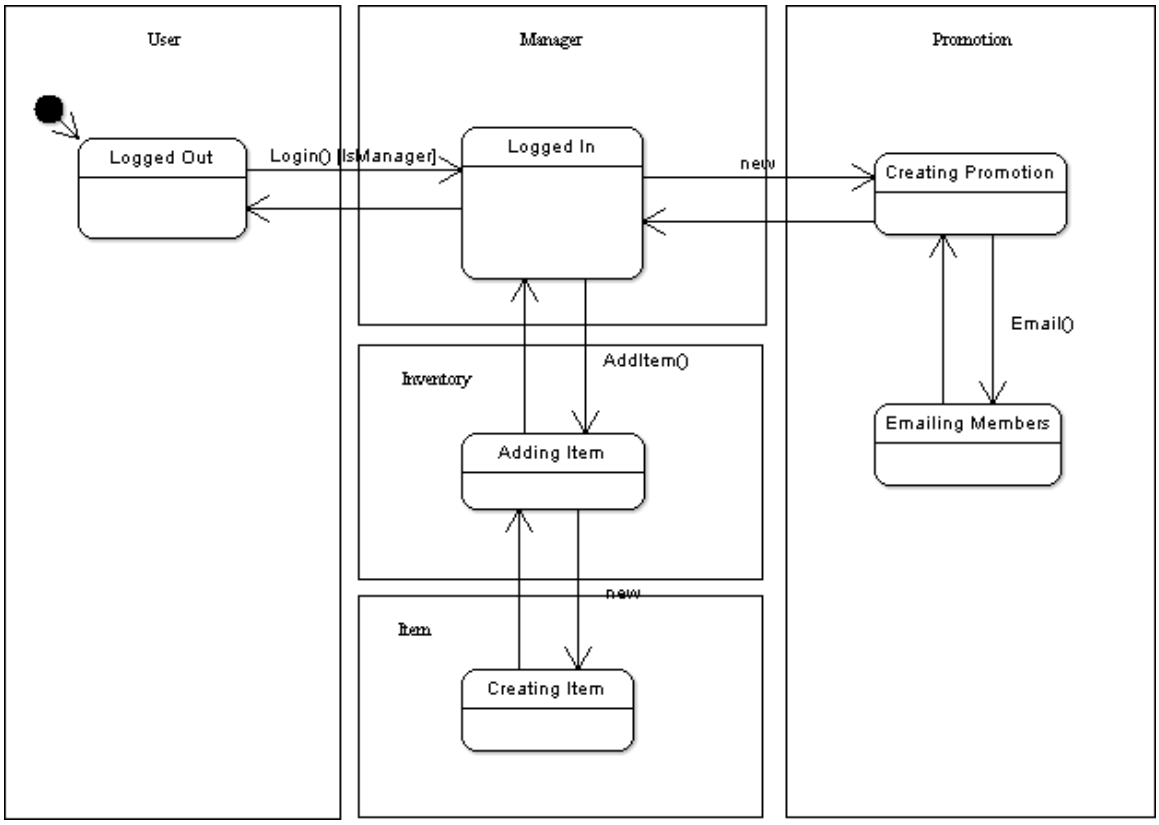
Template based on IEEE Std 830-1998 for SRS. Modifications (content and ordering of information) have

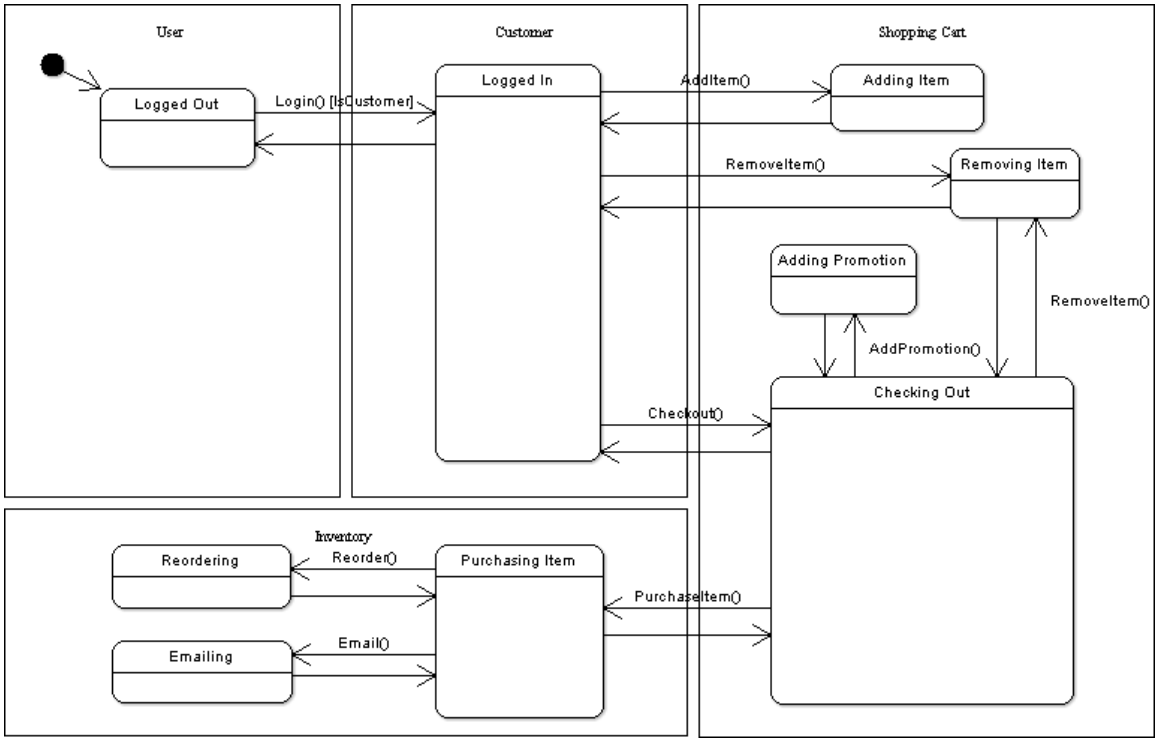


State Diagrams

The state diagrams take all of the functionality found in the previous diagrams and combines them together to demonstrate the possible changes in the state within BECS. These state diagrams contain all the possible scenarios shown in the sequence diagrams.

Starting Point	This is where the system starts at and it is represented with a filled circle that has an arrow that point towards the starting state.
States	These are represented in the diagram by the boxes that have the rounded edges. They are separated into two half's the top half is the title of the state and the bottom half can have additional states encapsulated within but these state diagrams do not contain any encapsulated states within states.
Transitions	The arrows that connect the states to each other and have a label of the event that occurs which triggers the transition. These triggers are normally functions but they can also be a new call that is the event of creating a new object; this is represented simply with the word "new". Some transitions do not have a label, these are returns from the current state to the previous state.
Objects	The larger boxes that can contain several states and transitions. They are classes within the state diagram. The text at the top of the box is the name of the class.





Template based on IEEE Std 830-1998 for SRS. Modifications (content and ordering of information) have