Functional Specification

by Timothy Broxson and Jose Perea

Table of Contents

Introduction:

This document describes the user requirements and functional specification of a shopping cart application, like a significantly simplified version of amazon.com and other such websites. The application shows a login window as it begins, and it performs different functions depending on who logs in (either a customer or a seller).

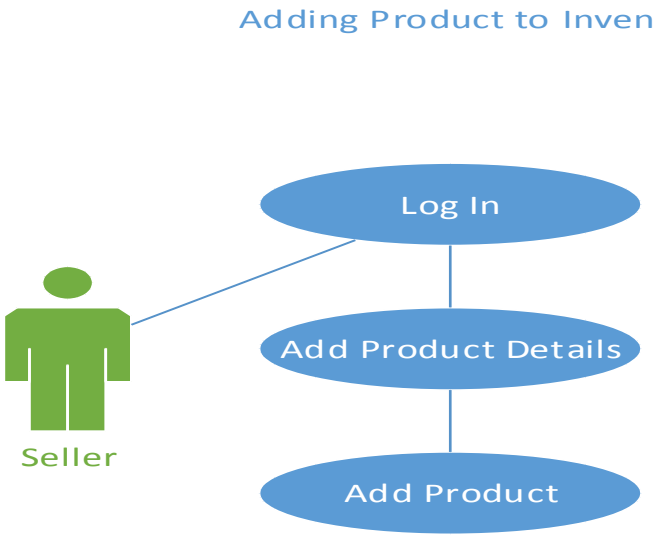
General Functional Specification:

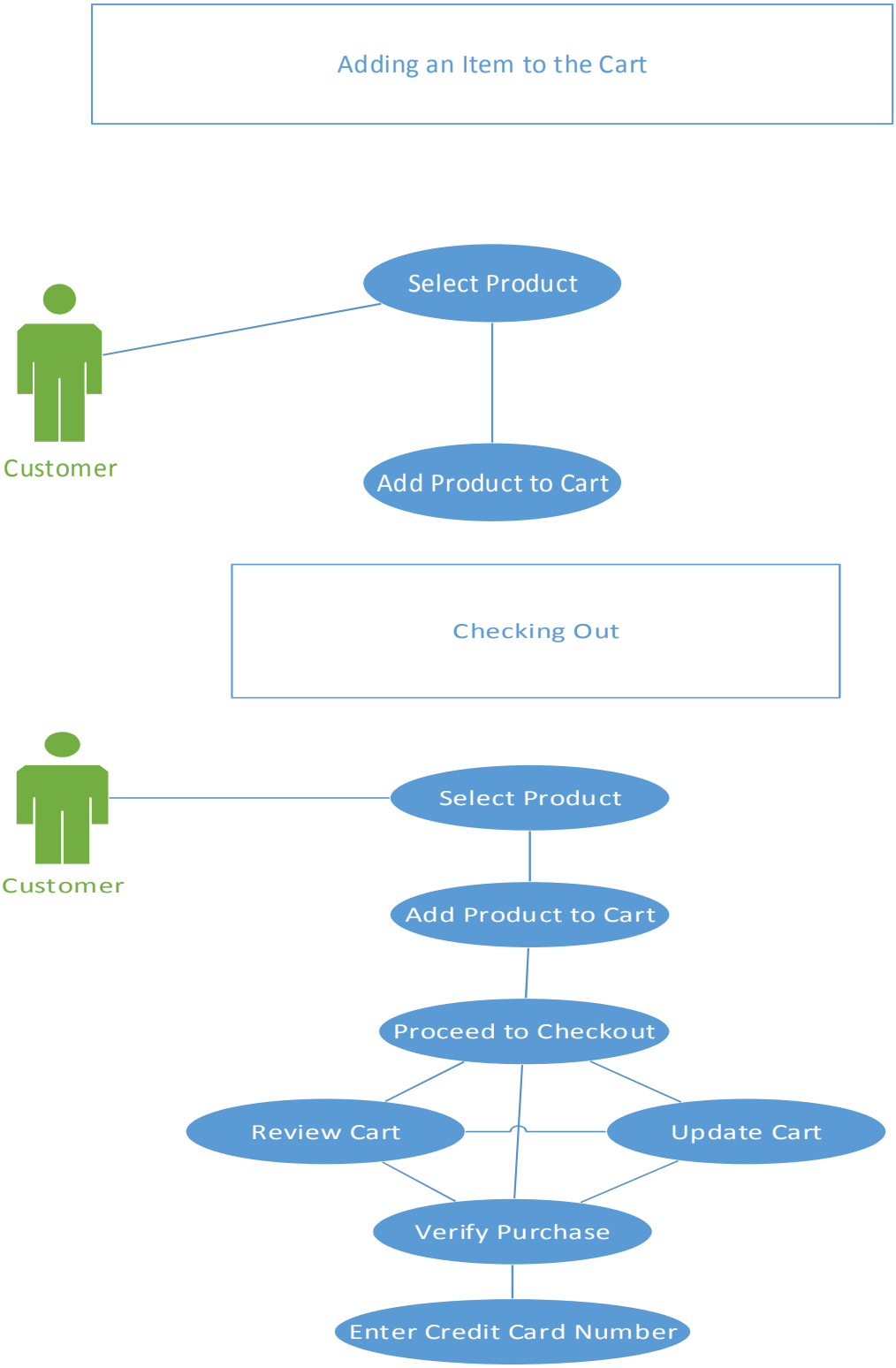
After logging in with a user name and a password, a window will open displaying a list of available products, including the products name, quantity, and price. The customer can then select products from the window and either add them to their shopping cart or get a full product description, pricing, and availability in a separate pop-up window. Whether or not the customer can add a product to the shopping cart depends on the availability of the product, the quantity.

The customer has the option of proceeding to checkout at any given time. The shopping cart can be updated within the checkout window by increasing or decreasing the item count for each product in the cart. When checking out, the customer must verify the shopping cart and then must pay for the goods by supplying a credit card number. The application, however, does not handle shipping.

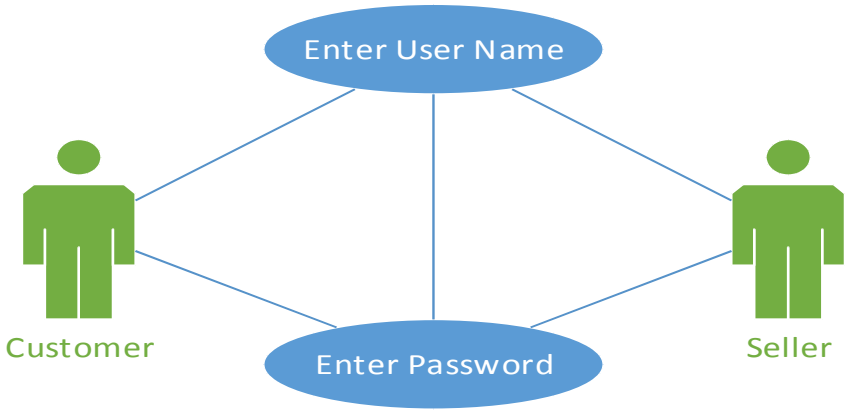
If a seller logs in, a window showing the current state of the inventory opens. The seller can update the inventory by adding products. The seller must specify the product name, invoice price, sell price, and available quantity. Each product is represented internally by ID, type, quantity, invoice price, and selling price. The application keeps track of all costs, revenues, and profits. The seller has the ability to access this information straight from the application's user interface.

Use Cases:

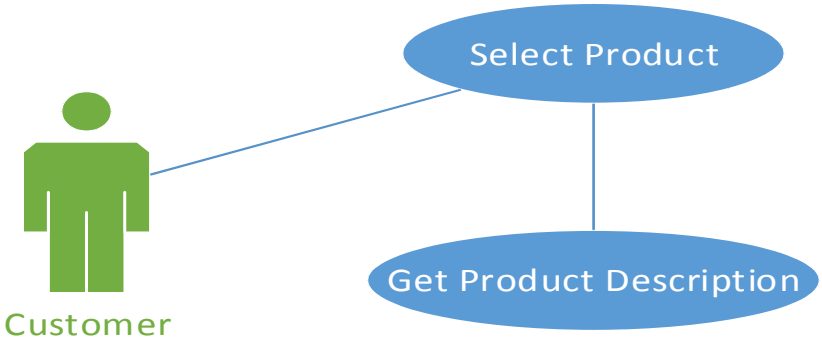



Adding an Item to the Cart
Select Product
Customer
Add Product to Cart
Checking Out
Customer
Select Product
Add Product to Cart
Proceed to Checkout
Review Cart
Update Cart
Verify Purchase
Enter Credit Card Number

Login



Reviewing Product Details



CRC Cards:

Buyer	
Creates Buyer type for User super class	

Seller	
Inherits from User; builds Seller class	Inventory

CheckOut	
Displays buyer's checkout window	Inventory Item UPDatabase

AddScreen	
Works with SellerUI Tells model to add item to Inventory	Inventory InventoryGUI UPDatabase

DeleteScreen	
Displays screen that allows user to remove item from inventory	UPDatabase Inventory InventoryGUI

ItemDescription	
Dsiplays window with information on selected item	

Profit	
Displays results of Business class	Inventory

SellerUI	
Template for screens seller uses to update Inventory	Inventory

UpdateScreen	
Provides screen for seller to update an item in Inventory	UPDatabase Inventory InventoryGUI

ShoppingCart	
Main class	UPDatabase Inventory SignInScreen

UPDatabase	
Stores user information	User

User	
Super class for User types	

SignInScreen	
Builds GUI for sign in	UPDatabase CreateUserGUI BuyerInventory CreateUserGUI Inventory

Inventory	
Stores the items for purchase	Business Item

Business
Calculates profits, costs, and revenue.

Item
Creates items to be stored in inventory

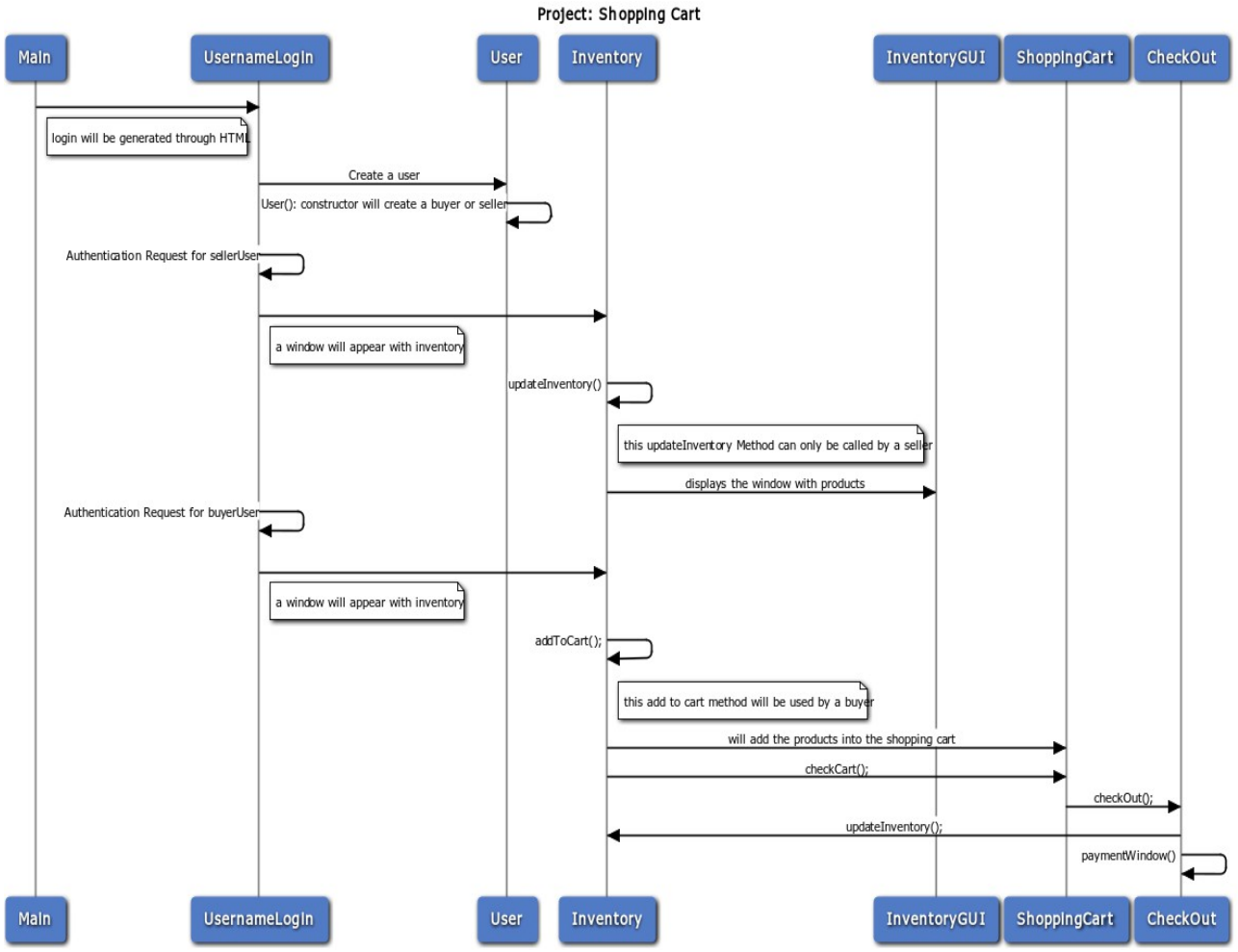
InventoryGUI	
Dsiplays the inventory to the seller, which the seller can then update	UPDatabase
	DeleteScreen
	Item
	AddScreen
	UpdateScreen
	Inventory

CreateUserGUI	
Creates GUI for making a user account	UPDatabase
	SignInScreen
	Inventory
	Buyer
	Seller

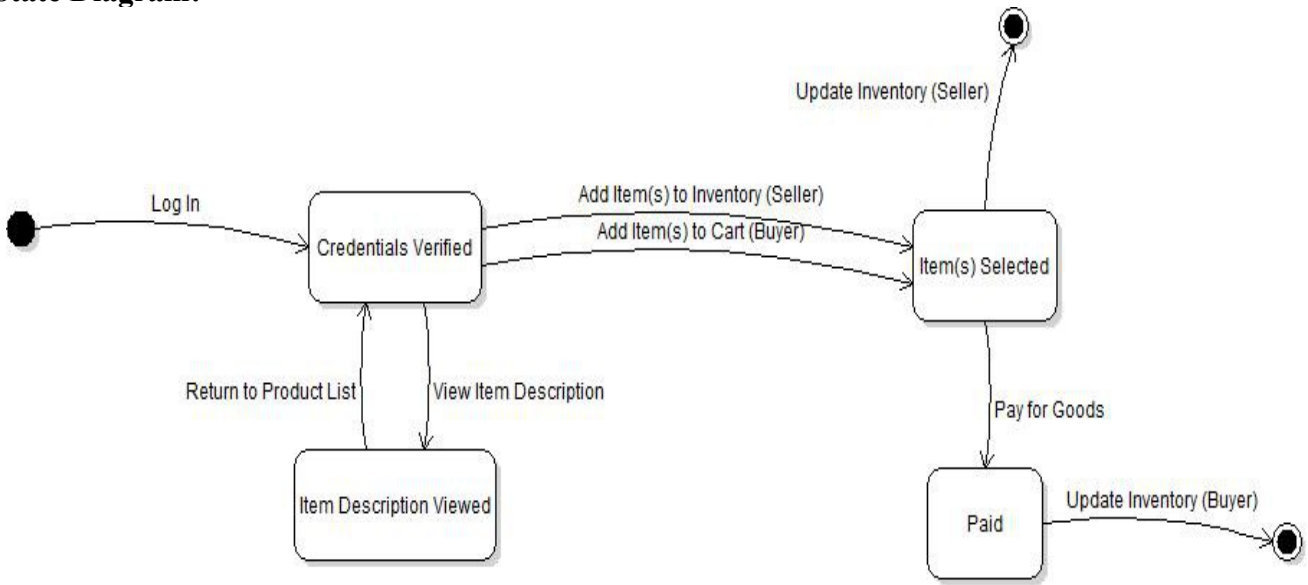
BuyerInventory	
View for Inventory	ItemDescription
	UPDatabase
	Inventory
	YourCart

YourCart	
Displays items that are in the shopping cart	UPDatabse
	BuyerInventory
	CheckOut
	Inventory
	Item

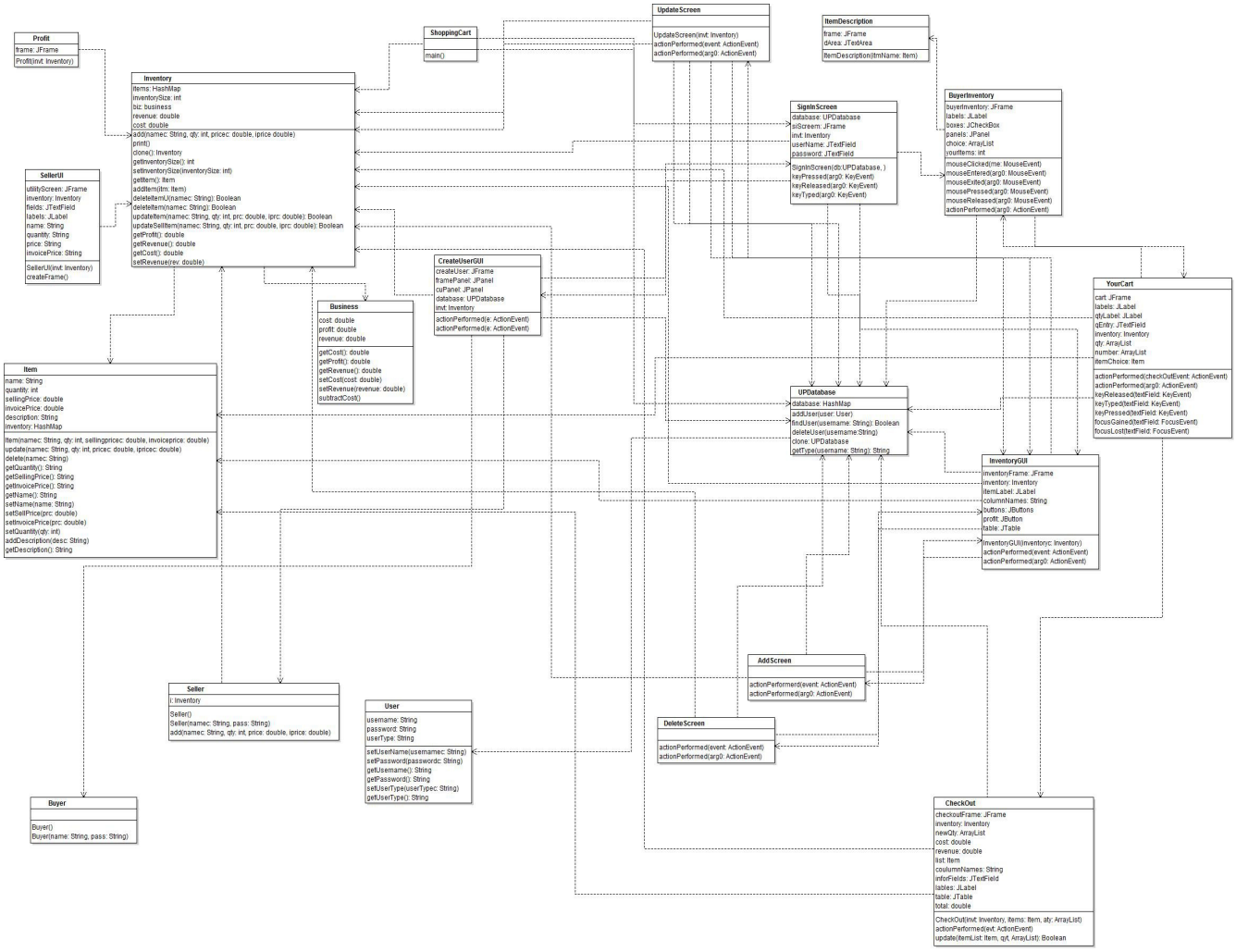
Sequence Diagram:



State Diagram:



Class Diagram:



Implementation:

The application is implemented with Java Swing GUI libraries. The design patterns used are as follows:

- Model-View-Controller: Implemented using Swing library methods.
- Strategy: Implemented using layout managers to handle placing the components correctly in the frame.
- Composite: Implemented by using JPanels to hold components. The JPanels are then added to a frame, treating each JPanel as an individual component.
- Decorator: The decorator pattern is used in two places. The first is the JTable displaying the inventory. Here, scroll bars appear when many items are added to the inventory. The second implementation of this pattern occurs when data is serialized. When a class is serialized, the class needs to stream I/O readers to read data out and then read data in.