Shopping Cart Capstone Experience

Professor Don Colton, BYU Hawaii

April 19, 2007

The Shopping Cart lab sequence is the **capstone experience** for this course. It is a substantial part of your learning and your grade. Everything before was to prepare you for this achievement. Earlier labs introduced you to the tools you need. In this lab, you apply them to build a simplified but working eCommerce web site.

Here are the specifications (what your program must do and how your database tables must be). Use your judgment and creativity in naming your store and designing the appearance of your screens, so long as they function in the way required.

Four Steps: There are four major steps in this lab, each composed of several smaller steps. We will discuss and work on these steps in class, but it is expected that everyone (except maybe an Einstein) will need to spend substantial time outside of class to finish. These are *lab* assignments. Start early before the tutors are overwhelmed.

Friendliness: When your programs can not do what the user requests, show a helpful message.

Grading: These labs are due the last day of class (by midnight or soon after) and are graded by the teacher the next morning. Partial credit is given based on functions working. You can request earlier grading if you are present to receive feedback.

TableViewer: These labs must inter-operate with your Table Viewer to see your inventory and shopping cart tables. (Use the same database.)

1 BuyOne

This preliminary program is designed for use by customers of your store. There are three steps.

(1) Table: Name your table `inventory`. Include the following columns: an inventory ID number, a product description, a quantity on hand, a cost to acquire new items, and a price at which we sell items. You must be able to handle a cost and price of $999.99 or more. You may include other columns if you wish. I recommend that you use by-hand methods to create the inventory table and insert a few items into it. The table must have at least three items.

(2) Customer View of Inventory: Name your CGI program `buyone`. It has only one screen. Display an HTML table of the appropriate information for all products in the inventory table. You must include the product description, price, and quantity on hand. You must not include the product cost because it is confidential. Other fields are optional. On each row (each product) should be a **Buy One** button, one button per inventory item.

(3) Actually Reduce Inventory: When the **Buy One** button is pressed the quantity **in the inventory table** should be update correctly. Then redraw the screen.

2 Manager

The `manager` program is intended for use by employees of the store. It allows them to display and modify the inventory. There are three steps.

(1) Display: Display an HTML table listing ALL information from the inventory table. On each row (each product) place an **Update** button, one button per inventory item. At the top or bottom there should be a button to **Add** new items.

(2) Add: When the **Add** button is pressed, bring up a screen with a blank input field for each column of the inventory table. At the bottom of the screen put two buttons. The **Back** button should return the user to the (updated) inventory screen. The **Add** button should add the new item to inventory and remain on this screen so that more items can be added.

(3) Update: When an **Update** button is pressed, bring up a screen like your add screen, with an input field for each changeable column of the inventory table, filled in with current information from the table. (The product ID itself should not be changeable.) At the bottom of the screen put three buttons. The **Back**

button should return the user to the (unchanged) inventory screen. The **Update** button should update the item in inventory and return to the (updated) inventory screen. The **Delete** button has two actions. If the quantity is zero or negative, the **Delete** button removes the item shown and returns to the inventory screen. If the quantity is positive, the **Delete** button changes nothing. It redraws the update screen with an appropriate warning message.

Testing: Should two products have the same ID?

3 ShopCart

The `shopcart` program is like your preliminary `buyone` program. Instead of a **Buy One** button, it has a quantity input field and an **Add to Cart** button on each line. It also shows the contents of your shopping cart. There are three steps.

(1) Screen Layout: The screen has two `<h1>` sections. The first section is called **The Store** and lists all items currently for sale. The second section is called **Shopping Cart** and lists all items currently in your shopping cart.

When the program starts, generate a random number to be the shopping cart ID. Pass this number to the screen in a hidden field. As items are added to the shopping cart, insert them (cart ID, product ID, and quantity) into your (newly created) shopping cart table.

The **Shopping Cart** section begins empty, showing a message stating that your shopping cart is currently empty. When one or more items are in the cart, it shows an HTML table that lists item code, description, quantity (as a modifiable field), price each, and total price (quantity times price). At the end of the HTML table is a grand total price and several submit buttons. One button says **Recalculate** and is useful when quantities have been changed. Another button says **Check Out** and leads to the checkout program (screen) that you will write later.

The **The Store** section displays an HTML table of the appropriate inventory information for all products in the inventory table. As with `buyone`, you must include the product description, price, and quantity on hand, not product cost, other fields optional. Next to each row (each product) with positive inventory should be a **buy** input field with a default value of "1" (changeable by the customer) and a **Add to Cart** button, one button per inventory item.

(2) Add to Cart: When pressed, the quantity in the shopping cart should be updated. The screen should then be redrawn. Do *not* modify the inventory quantities until the customer finalizes his/her purchase.

(3) Recalculate: When pressed, the quantity in the shopping cart should be updated. If the new quantity for any item is zero, that item should be deleted from the shopping cart.

Testing: What should happen if you add zero or negative of something? What if you put more widgets into your cart than exist in the store? What if you have two people shopping at the same time?

4 Checkout

When the customer presses the **Check Out** button on the `shopcart` screen, this program should run. At your option, you can make this a separate program or combine it into the `shopcart` program.

(1) Screen Layout: The screen has two `<h1>` sections. The first section is called **Shopping Cart** and lists all items currently in your shopping cart. The second section is called **Check Out** and finalizes the purchase.

The **Shopping Cart** section looks similar to `shopcart`, but quantities cannot be changed. The **Recalculate** button is gone. The **Check Out** button is replaced by a **Continue Shopping** button that takes you back to `shopcart`.

The **Check Out** section displays sales tax, shipping and handling, and final total. You may calculate these in any way that seems reasonable to you. Provide input fields for credit card number, expiration date, and card holder name. At your option, you can also provide appropriate input fields for ship-to address. Provide a **Finalize Purchase** button.

(2) Finalize Purchase: When pressed, verify that the credit card information is reasonable. (Use Business::CreditCard to validate the number.) If not, print a warning and redraw the screen. Next, all quantities in the shopping cart are deducted from store inventory and removed from the shopping cart table. A Thank You screen should then be drawn giving appropriate links of your choice.

Testing: What if the card is expired? What if the card number is invalid? What if the shopping cart has more of an item than the quantity in stock?