Database Store Website

CS 405G Database Systems
University of Kentucky
Spring 2016

Authors: Carmon Zachary Almon, Dylan Wise

Table of contents:

I. Description of Programs ……. 2

II. Program functions ……. 7

III. Testing and Developer Experience ……. 15

IV. Finalized database design ……. 16

I. Description of Programs:

Project URL: http://www.cs.uky.edu/~czal222/CS405G/index.php

List of Programs:

1. index.php
2. products.php
3. add_to_basket.php
4. productSearch.php
5. customer.php
6. change_basket.php
7. show_order_contents.php
8. employee.php
9. register.php
10. login.php
11. logout.php
12. ECOMMERCE.sql
13. drop_tables.sql

List of Extra Files:

1. README.txt
2. FinalReport.pdf
3. /Images/*

Starting with the program files, we have an index page. This is the main landing of the website and includes a title as well as a navigation bar. This main navigation bar is on every page of this website. The main page is simple in design, with only a UKY College of Engineering graphic with the navigation bar. The navigation bar has buttons for the main page, products page, customer page, employee page, registration page, login page, and a logout page. The login link from this navigation bar, no matter which page, will show if you are logged in or not. If you are not logged in the bar will only have a link that says “Login”. If you are logged in, then this text changes to “You are logged in as: Username”. This will show users if they are logged in, and who they are logged in as.

Moving to the products page, this page is viewable by anyone, whether they are logged in or not, whether customer or staff member. At the top of the page is a short line of text stating that if the user isn’t logged in they can still view the page of products, but won’t be able to add to basket or order anything. There is a non-working search bar at the top. There wasn’t enough time in the semester to implement a full search function. As of right now the search bar does work and POST’s the search terms to the product search page, but the product search page does not work. Please see the description for the product search page for more information. The only other thing on the product page is a table full of products and information. The tables layout is as follows:

item picture, item name, item UPC number, quantity of the item in stock, if the item is on promotion or not, the item's cost and finally a column for the button to add to basket. The only thing that changes here is whether a button appears in the last column. The button here only shows up if you are currently logged in as a customer. No matter if you are not logged in, logged in as an employee, or somehow logged in elsewise, the button won't appear unless you are a customer. When you click on this button the link POST's information, about which item you are adding to the basket due to which line the item and button are on, to the add to basket page for processing. Please see that description for more information.

The product search page is not implemented. There is a chunk of code that is commented out. We did not have time to implement full measures as needed to request data from the database and display it. There is a navigation bar at the top of the page to take you back to any other page if you find yourself on the page, by trying to search for something from the product page. On the product page there is a warning about the search function not working.

The add to basket page is only called from the product page. The add to basket button POST's information to this page. The page detects when information hasn't been sent, and/or if the user is logged in. If either or both of these things are true, then nothing will happen except error messages and redirects will happen. The only time anything will happen is when a correctly logged in customer clicks the add to basket button and the information is POST'ed to this page. There is not user input except for the button click so no validation needs to occur. The button has encoded within it which item number is being added to the basket. The file detects the user who is logged in and gets that users unique user ID number from the database. Then the file adds the item to the basket table with the user's ID, the item's ID, and the quantity of 1. As of now the user is not able to change quantity added to the cart, but within the customer page they can change the quantity in their cart. So the action is doable, just not from this or the products page. When this query is completed then a notification pops up to the user and redirects them back to the products page. If something went wrong, an error message appears on the page for the user to go back and try again.

Next we have the customer page. The customer page has multiple items on it. The first thing that happens is when you load the page it detects whether you are a logged in as a customer or not. If you are not logged in as a customer, the only thing that displays is a short text notification that says you have to be a customer to view this page. Whether you are not logged in, logged in a staff member, or logged in as someone else somehow, this message will pop up. If you are logged in as a customer there are a few things you can view on this page. There are two tables of things. The first is a table of your current basket list. This table consists of similar things as the products table; the item picture, item name, number, cost, quantity, and finally a text box and button to change the quantity. This text box and button are within a form that POST's the information, user input of quantity and the item number, to the change basket file, please see that description for more information on change the quantity. Under this table is a total cost line and a button to place the order. If there is nothing in your basket this button is hidden, and the total cost line is $0. This button to place the order POST's information within

this same customer page. The POST is detected and validation occurs to make sure there is stock of each item in the quantity ordered by the user. If stock is less than the quantity being requested, then the file stops and a notification pops up to the user alerting them that the order couldn't go through and text is shown on the page that tells the customer which items can't be ordered at this time so they can change their basket. The next table is spaced under the place order button, or where it would be if hidden, and is a table of all the orders that have ever been placed by the customer. This table consists of an order number column, a shipped column, and a view entire order column. The order number column shows the order number that is stored in the database for that particular order. The shipped column tells the user if the order has been shipped yet or not. The view entire order column is simply a button. This button POST's the order number to the show order contents page so the user can view the items that were in that order. Please see that description for more information.

The change basket page receives POST information from the customer page. There is the same customer check as before. If the POST information is invalid/empty or the person logged in, or lack thereof, is not a customer then nothing happens but a notification pop up of the error and a redirect to the main index page happens. If there is POST information and it is a customer that is logged in, then the POST information is parsed. As this information contains data entered by the user, it checks for 5 digits and only digits. The customer can only enter numbers between 0-99999. If the validation passed, the new quantity is checked, if it is 0, then the item is deleted from the basket and the user is alerted that it was successfully deleted or if there was an error, the error message is shown and nothing further happens. The user must go back to try again. If the new quantity is anything between 1-99999, then the basket is updated with the new quantity with the user's ID, the item ID number and the new quantity. Again, if successful the user is alerted and redirected back to the customer back, if not then an error message comes up and nothing further happens. The user will have to go back to the customer page to try again.

The show contents page receives POST information from the customer page. There is again a customer and POST check as before. If both validation checks pass, then the file goes on with the order ID number as from the POST and gets all order information from the orders table and the inventory table for everything under that order and item numbers. This information is displayed in a table with columns: item picture, name, number, cost, quantity. At the top of the table is the order number and at the bottom of the table is the total cost for the order.

Now we have the employee page. Like before, with the customer checks this page checks to see if you are logged in as an employee. If the user is not logged in as an employee the page only displays a message that the user cannot view that page as it is only for employee's. If the user passes the employee check, then what is displayed depends if the user logged in is a manager or a regular employee. Both types of employees are shown just two tables. The first table, a table of all the products, is different for each type of employee. For normal employees the table shown consists of: item pictures, names, numbers, costs, quantity's and finally the last column contains a text box and submit button so that employees can change quantities of items. Manager's product table will show: item picture, name, number, cost, a text box and submit

button to change cost, the quantity, a text box and submit button to change quantity, sales for that item that week, sales for that month, sales for that year, if the item is on sale or not, the sale price, and finally a new text box, drop down menu, and submit button to change the sale promotion information. So starting off with the quantity text box and button, this works the same for both managers and regular employees. The file POST's the information submitted to itself to do the work. The file detects whether a POST is empty or not and if it is empty it notifies the user with an error alert and reloads the page with no work done. The valid range of numbers is 0-99999. The program detects what type of command is POST'ed, and then validates the text box entry from the user. If the input is valid then the program goes on to update the quantity within the inventory table based on the information POST'ed, the new quantity and the item number. If the validation checks do not pass, a notification pops up alerting the user of what went wrong and then the page is reloaded with no work being done. The input is for managers only, changing the item's regular cost. The file POST's the information submitted to itself to do the work. The program detects what type of command is POST'ed, and then validates the text box entry from the user. The valid range of numbers is 0-99.99. As the price can contain a '.' the program can only detect this number range. The program has a more difficult check for the price. If the user's input validates then the program updates the price within the database. If the input doesn't validate then a notification is shown to the user and the page is reloaded with no work being done. The last thing a manager can do that a regular employee can't is to set the sale and sale price. This column has a text box for user input on price, same as the regular price, and a drop down menu. This drop down menu must have something selected. The two options are to set the item not on sale or on sale. One of these two options must be selected for every price change submission. If it is not selected an error message is shown to the user. The text box is parsed the same as previous, and with whichever drop down is selected the item in the inventory is updated with whatever information is submitted by the user. Finally, the last table on the employee tab. This table is shown to both types of users and is not different between types. This table shows all pending orders from the customer order table. The table has columns for: customer username, customer ID, order number ID, has it shipped, and finally a column for a button to ship the order. Like the other forms, this button POST's information to this same file to do the work. The POST information on this is only the information about the order. This POST information has no user input other than the button clicking so it simply checks the POST data is correct from the button and then updates the customer orders table within the database to show the order has now been shipped. And that concludes the employee tab.

The next page is the registration tab. This page is for new users to register with the website and database so that they may order products. There is a single form on this page. Above the form is a check to see if the user is logged in. If the user is currently logged in as a staff or customer the form will not submit. Nothing will happen. Users cannot register with the site when they are already logged in. When not logged in, users can enter user information about themselves. The only things required is a username they desire and a password. There are inputs for name and addresses but there are no checks done on them because they are not recorded in the database or kept. The username and password have checks done on them to verify they are valid. The username and password are also required to be under 20 characters. There are also

checks to make sure the entered username is not already in the employee or customer tables, if it already exists then an error message is sent back. If any error is detected the error message is displayed next to the username and/or password box and no work is done on the database. Once the checks are passed then the user is added to the customer table with the desired username and password and a randomly generated unique user ID number that is generated on the fly from the program itself. If something went wrong, then the program stops and lets the user know something went wrong and to try again.

Next is the login page. If the user is already logged in, an alert is shown to the user and redirected back to the main index page. This is page has a simple form that accepts a username and password. The page POST's this information to itself and does the checks. The username and password have checks done on them to verify they are valid. The username and password are also required to be under 20 characters. If any error is detected the error message is displayed next to the username and/or password box and no work is done. If the username and password match up with what is in the database, then the user is logged in by storing a cookie in the user's browser. This is how the program knows whether a user is logged in and who the user is. Once logged in the page will redirect you to the main index page.

The logout page is simple, the only thing done is all cookies with our webpage's cookie name are deleted. The user is alerted to the fact that they have been logged out, or if they are not logged in the program alerts them that they must be logged in to log out. In either case the user is redirected to the main index page. This page is not meant to be stayed on at any time length, although it does contain the navigation bar.

The Ecommerce SQL file is simple, it uses the database coded and then creates the tables needed. Then the tables are filled with values. This data was generated on a semi-random basis and will at times not make sense. See the database design portion for more information.

The drop tables SQL file is very simple, using the database, it drops the tables for our database in the correct order needed to. This file was made to speed testing and creation of the website. We are including it in case it is ever needed. See the database design portion for more information.

II. Program functions:

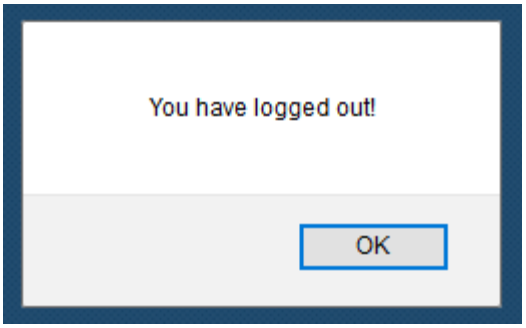
Figure 1.

This figure shows the main page with no one logged in.

Figure 2.

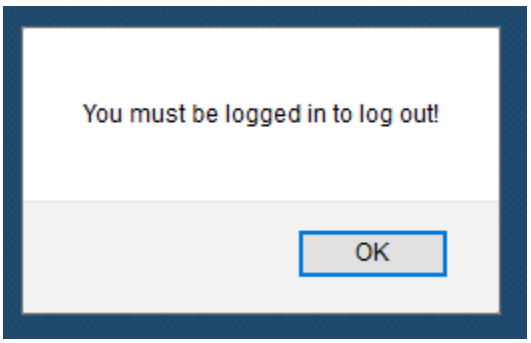
This figure shows the main page with someone logged in.

Figure 3.



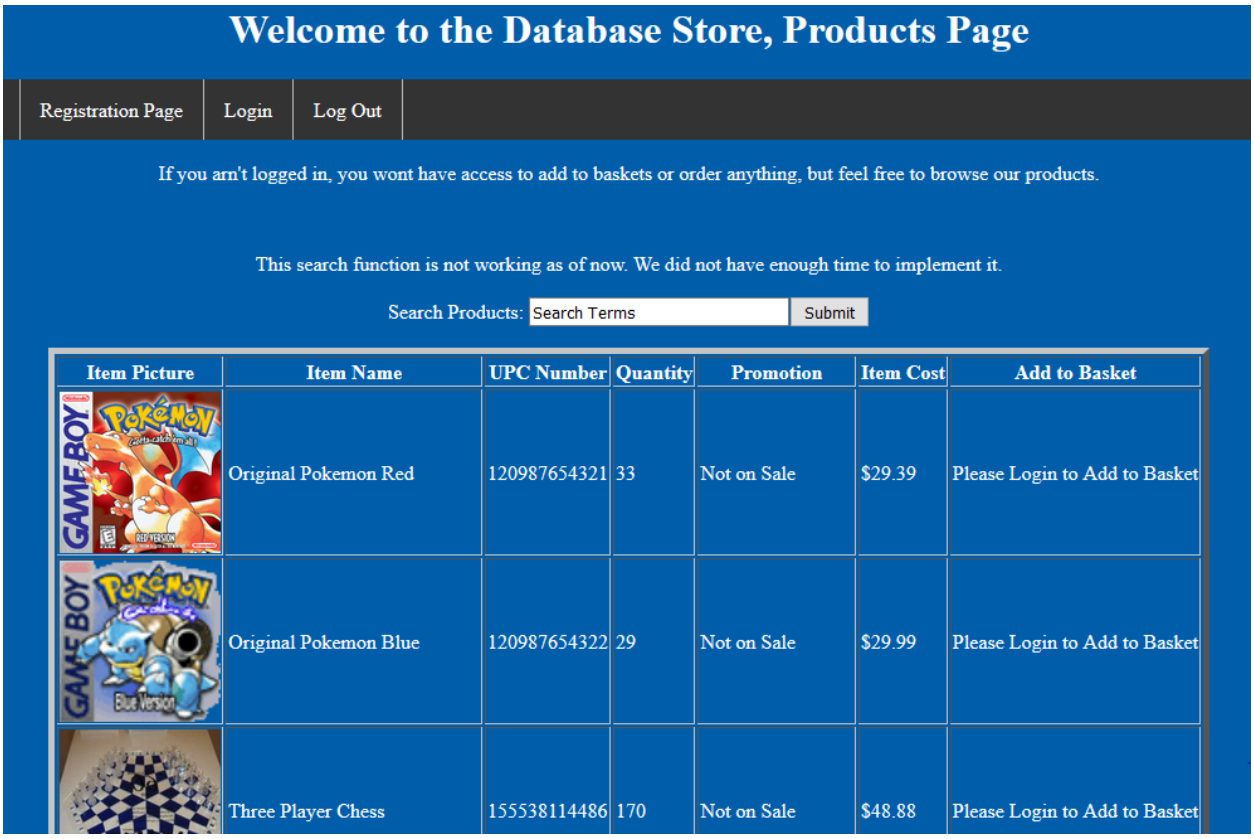
This figure shows the alert when the user logs out.

Figure 4.



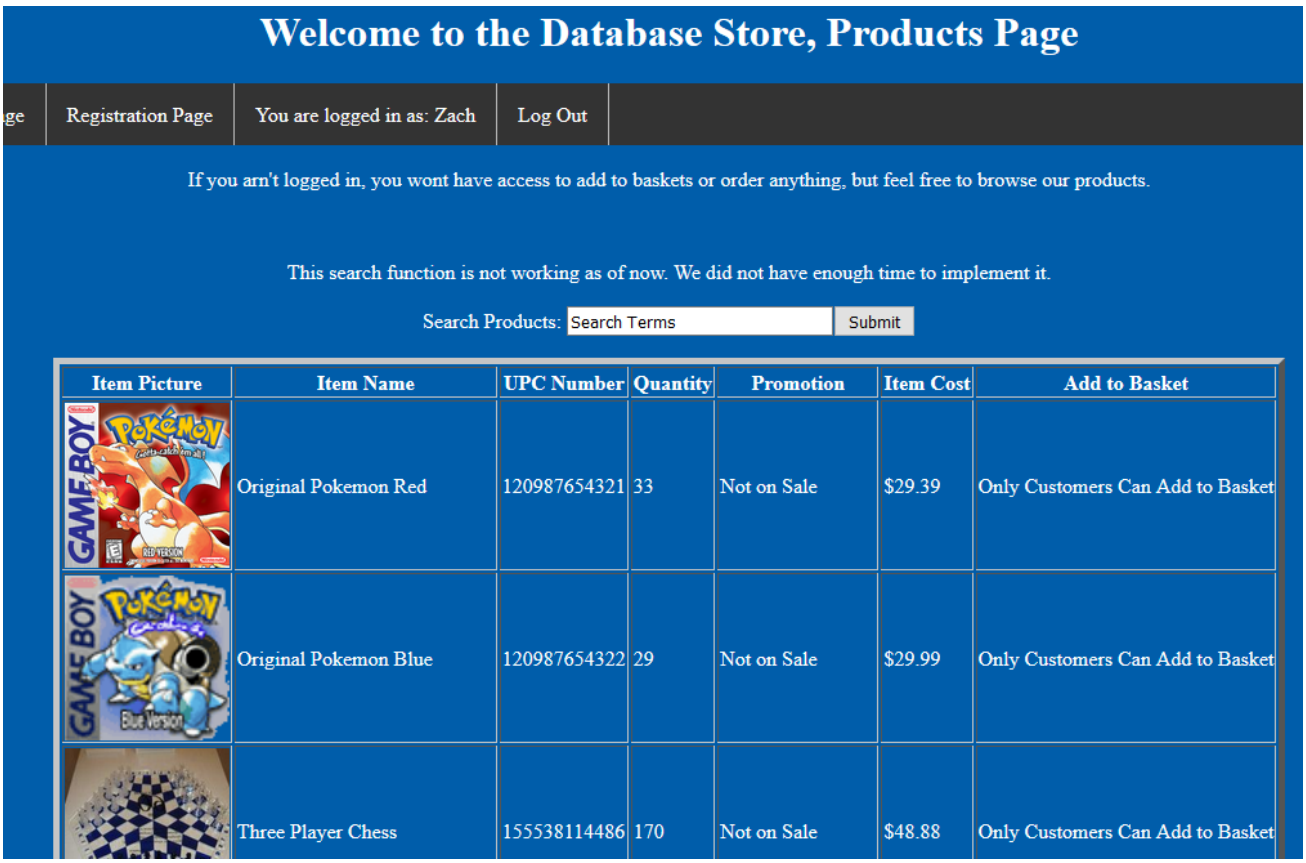
This figure shows the alert that shows up when the user is not logged in and tries to log out.

Figure 5.



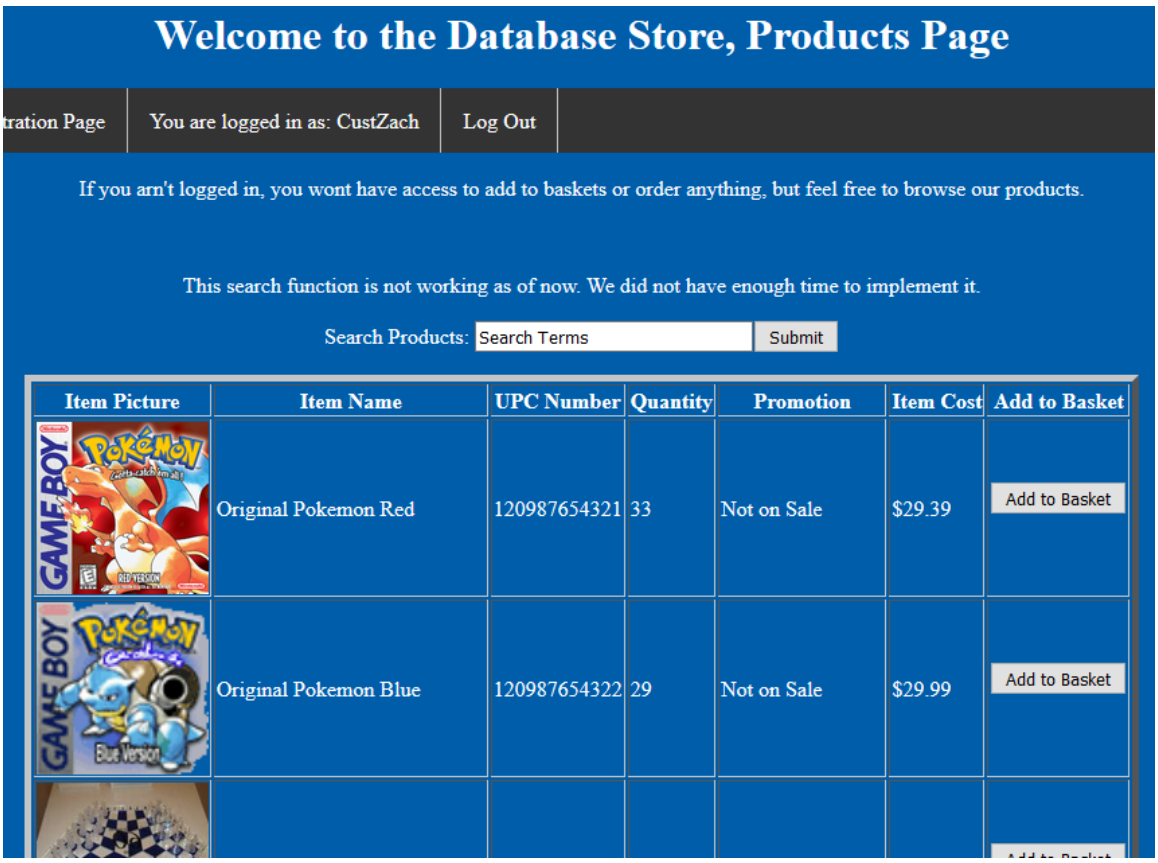
This figure shows the products page as a user sees when they are not logged in.

Figure 6.



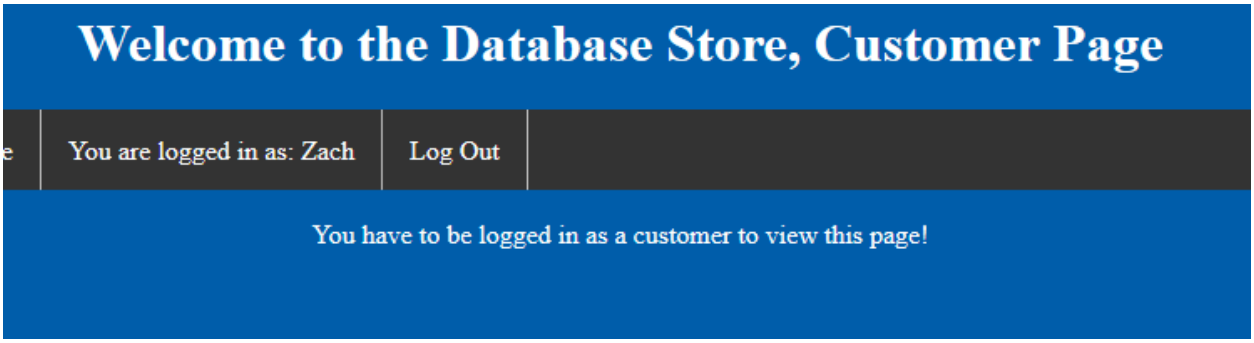
This figure shows the product page when you are not logged in as a customer.

Figure 7.



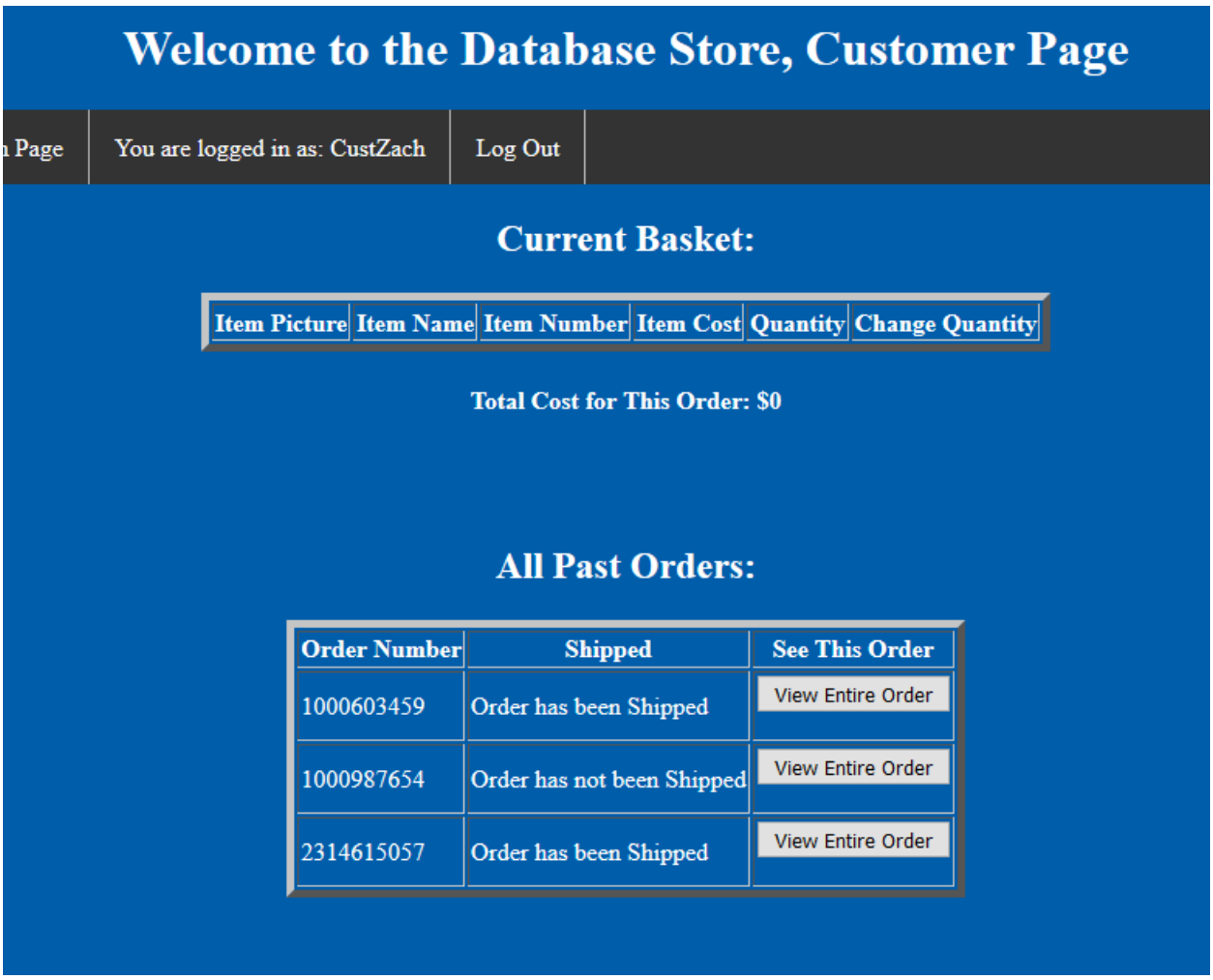
This figure shows the product page when you are logged in as a customer.

Figure 8.



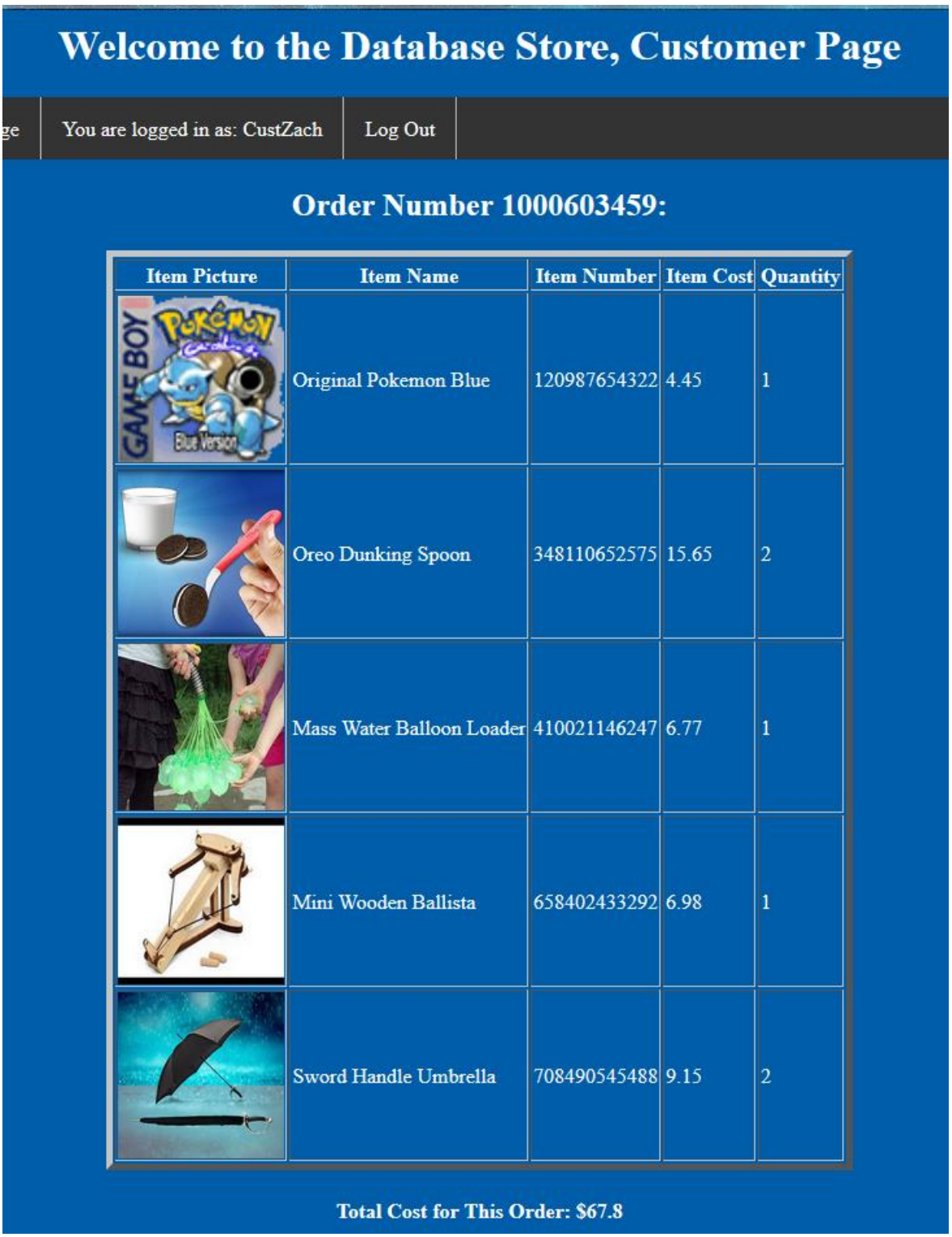
This figure shows the customer page when you are not a customer.

Figure 9.



This figure shows the customer page, with no items in the basket.

Figure 10.

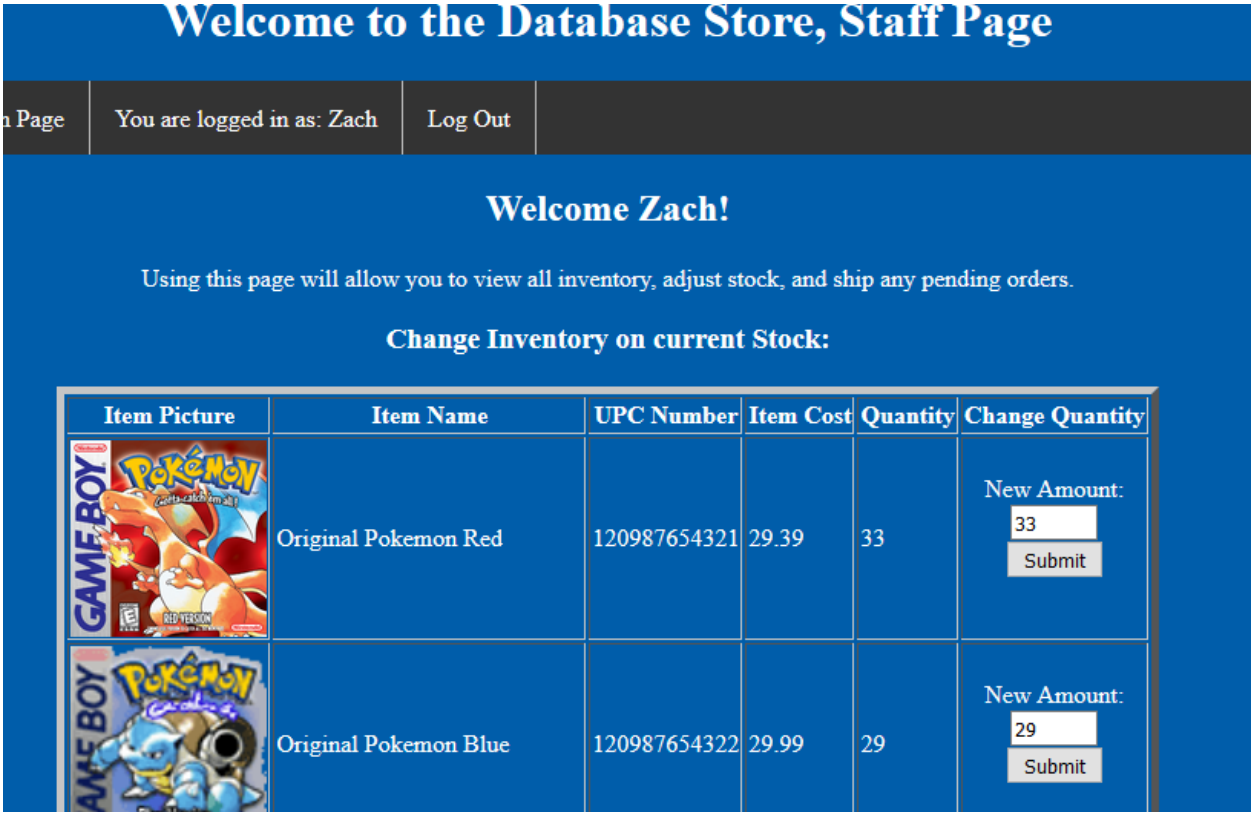


This figure shows the customer order contents when viewing entire orders.

Figure 11.

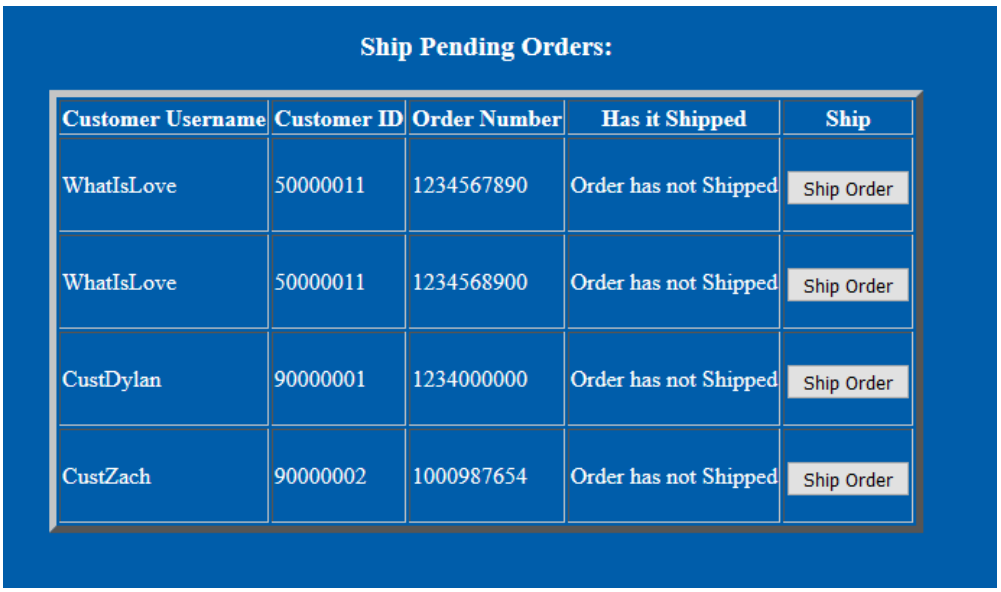
This figure shows the employee page when not logged in as an employee.

Figure 12.



This figure shows the employee page as a normal employee.

Figure 13.



Ship Pending Orders:

Customer Username	Customer ID	Order Number	Has it Shipped	Ship
WhatIsLove	50000011	1234567890	Order has not Shipped	Ship Order
WhatIsLove	50000011	1234568900	Order has not Shipped	Ship Order
CustDylan	90000001	1234000000	Order has not Shipped	Ship Order
CustZach	90000002	1000987654	Order has not Shipped	Ship Order

This figure shows the employee screen, pending orders table as shown to both types of employees.

Figure 14.

Welcome to the Database Store, Staff Page

Home | Products Page | Customer Page | Employee Page | Registration Page | You are logged in as: Dylan | Log Out

Welcome Dylan!

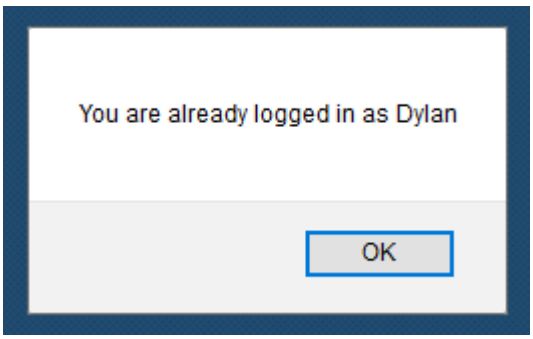
Using this page will allow you to view all inventory, adjust stock, and ship any pending orders. You may also see the sales statistics for all inventory and set sales promotions and prices.

Change Inventory on current Stock:

Item Picture	Item Name	UPC Number	Item Cost	Change Item Cost	Quantity	Change Quantity	Sales This Week	Sales This Month	Sales This Year	Sale	Sale Price	Change Sale Information
	Original Pokemon Red	120987654321	29.39	New Amount: 29.39 Submit	33	New Amount: 33 Submit	39	69	1102	Not On Sale Currently	35.00	New Amount: 35.00 Select Here Submit
	Original Pokemon Blue	120987654322	29.99	New Amount: 29.99 Submit	29	New Amount: 29 Submit	26	70	1201	Not On Sale Currently	19.90	New Amount: 19.90 Select Here Submit
	Three Player Chess	155538114486	48.88	New Amount: 48.88 Submit	170	New Amount: 170 Submit	28	143	804	Not On Sale Currently	38.71	New Amount: 38.71 Select Here Submit
	Call of Duty: Black Ops	209070799546	98.09	New Amount: 98.09 Submit	112	New Amount: 112 Submit	14	130	360	On Sale Currently	39.73	New Amount: 39.73 Select Here Submit

This figure shows the employee page as a manager. There are more tables and forms for changes to be made.

Figure 15.



This figure shows what happens when you try to re-log in after you are already logged in.

III. Testing and Developer Experience:

As we built the program we tested along the way. This may not be a good method, nor may it be efficient but we did not have many test cases between what we tested for within the program itself as described in the program description. See the program description for what we tested per webpage. Everything works in the program except for the search bar on the products page, and the search page. We did not end up having enough time to implement this function/feature.

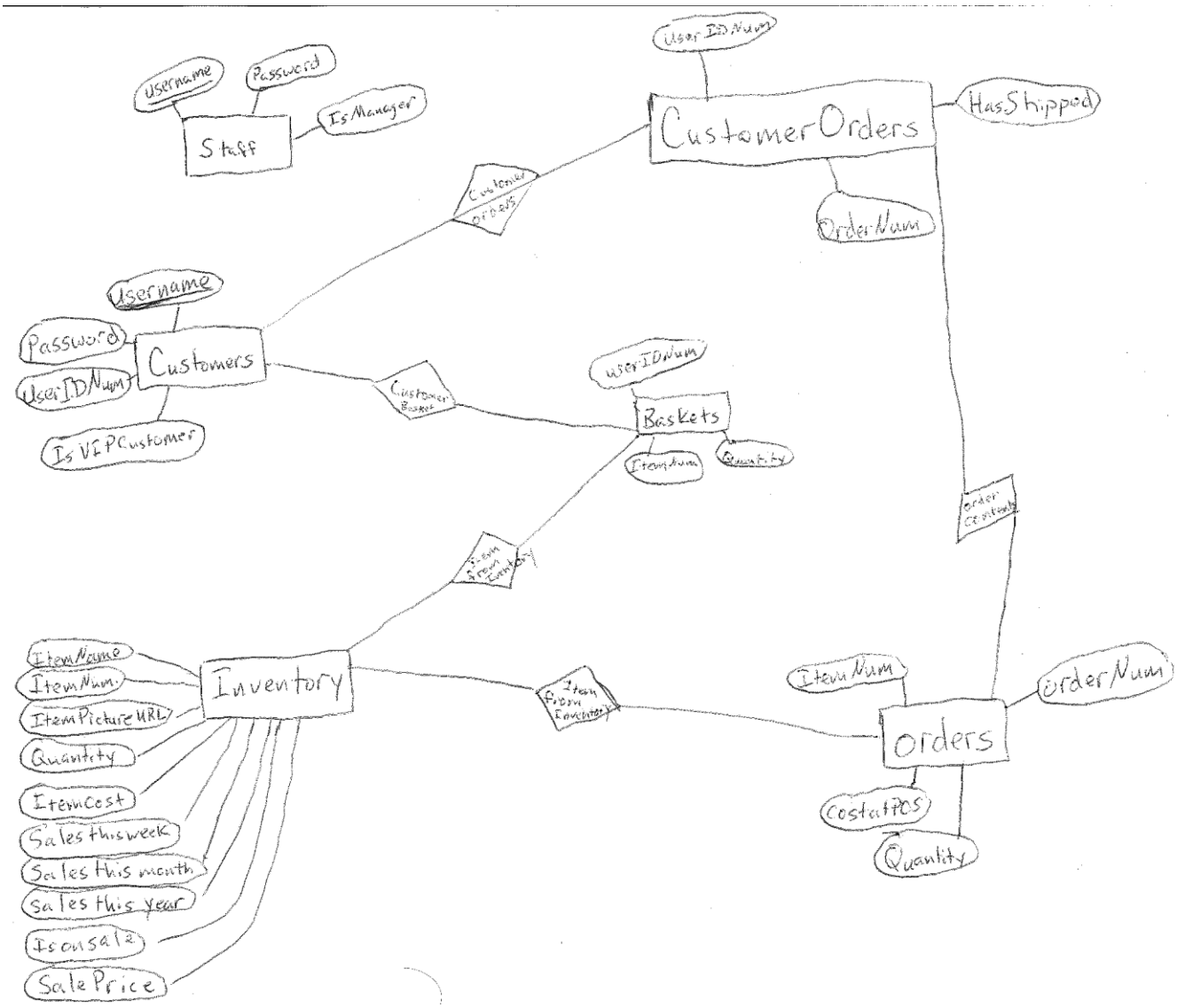
Program / Project Experience:

We each worked on portions of every file. We did all the testing on Zach's multilab and MySQL database. The files are still on Zach's multilab under the URL provided. We each wrote portions of each file, asking the other for help when we were stuck or had something else to attend to. As evidence by multiple ways of coding, we didn't have a defined structure to keep to. We also got better at coding in PHP as we went along, for Dylan it was his first experience with MySQL and PHP, while Zach had experience with PHP and minor experience in MySQL. The work was cut evenly in half.

Individual points for a 2-person group = 26
Zach's portion = 13
Dylan's portion = 13

IV. Finalized Database Design:

ER Diagram:



Database Schema:

We start off with 3 main tables that hold information. The staff, customer, and inventory tables.

The staff table holds the usernames, passwords, and a Boolean variable that tells the program if a staff member is a manager or not. The username variable is unique to this table. The username is also a primary key. In the PHP program, when a new customer registers, the program checks against this table for uniqueness of the desired username as well as the customer table. So in practice, the username column of the customer table and the staff table column of usernames are unique. This means that a username used in one table cannot be used in that table or the other table as well.

The second table holds the information for customers. The customer table holds the usernames, passwords, user ID numbers, and a Boolean variable for if the customer is a VIP customer or not. This variable for the VIP customer is not used as the extra credit portion was not implemented. The username and user ID number are unique to this table. The username is a primary key. In the PHP program, when a new customer registers, the program checks against this table for uniqueness of the desired username as well as the staff table. So in practice, the username column of the customer table and the staff table column of usernames are unique. This means that a username used in one table cannot be used in that table or the other table as well. The user ID number is randomly generated and checked for uniqueness within the PHP program. It will never try to give the database a non-unique number, but the database also has the check.

The third basic table is the table for items. The inventory table holds item names, numbers, picture URL's, quantity, cost, sales information for the past week, month and year, a Boolean variable on whether the item is on sale or not, and the sale price. The item number is a unique variable. The PHP program handles giving only certain information to certain types of users, such as only managers can see the sale information and can change the prices of items and the promotion of them. All employees have access to change quantities of items. All error handling is done by the PHP program and should never send erroneous data to the database.

These three tables are the basic tables of information for the store, and the last 3 tables are built off them. The next three tables are the basket, customer orders, and orders table.

Starting with the basket table, it holds the user ID number, item number and quantity for every customer. The user ID number references the user ID number from the customers table. The item number references the item number from the inventory table. There is a unique constraint of user ID number to item number. This means no two rows will have the same user ID number **AND** item number. This table could contain multiple rows for one user ID number for multiple different items in the basket. The quantity controls how many of the item the user wants in the basket.

The next table is the customer orders table. It holds the user ID number, order number, and has shipped Boolean variable that tells if the order has been shipped or not. The user ID number references the user ID number from the customers table. The order number references the order number from the orders table. There is a unique constraint of user ID number to order number. This means no two rows will have the same user ID number **AND** order number. There could also be another/instead a unique constraint solely on the order number, but as before the PHP program checks newly generated order numbers against the current order numbers within the database and only ever sets a new order number when it is unique.

Finally, the last table is the orders table which holds the multi-value items for each order. There is order number, item number, cost at point of sale (POS), and finally quantity. The item number references the item number from the inventory table. The order number references the order number from the orders table. There is a unique constraint of item number to order number. This means no two rows will have the same item number **AND** order number. This table holds all items in each order, meaning that item numbers can be duplicate, and order numbers can be duplicate, but never both at the same time.

When in practice, the staff table never references anything else. It only allows staff members to log in and PHP program allows them to change table information. The customer table refers to the basket and customer order tables via the user ID numbers. The customer order table refers to the orders table via order numbers. The orders table refers to the inventory table via item numbers.