By
Bhushan Chitre
Patrick Good

The Computer online Store

CS405g E-commerce Project

Table of Contents

ER Digram Included As Pdf

(As ER Diagram)

Database Schema Design:

Database Schema:

Users(username: varchar(25), password: varchar(25), email: varchar(25),
fname: varchar(25), lname: varchar(25), u_street: varchar(25), u_city: varchar(25),
u_state: varchar(25), u_zip: varchar(25), u_type: varchar(25));
Inventory(itemID: int(11), name: varchar(25), brand: varchar(25),
item_type: varchar(25), quantity: int(11), price: float, on_promotion: tinyint(1),
promo_price: float, image_path: varchar(25), description: varchar(255));
Cart(itemID: int(11), desired_quantity: int(11), username: varchar(25));
Foreign Keys: itemID, username
Orders(orderID: integer, username: varchar(25), status: tinyint(1), date: date,
shipping_adress_street: varchar(25), shipping_adress_state: varchar(25), shipping_adress_city:
varchar(25),shipping_adress_zip: varchar(25), total_price: float);
Foreign Keys: orderID, username
Order_Details(orderID: int(11), itemId: int(11), quantity: int(11), component_price: int(11));
Foreign Keys: orderID, itemID

All values are NOT NULL

Functional Dependencies And Highest Degree Normal Form:

Users(username: varchar(25), password: varchar(25), email: varchar(25),
fname: varchar(25), lname: varchar(25), u_street: varchar(25), u_city: varchar(25),
u_state: varchar(25), u_zip: varchar(25), u_type: varchar(25));

All non key attributes in this relation are functionally dependent on the Primary Key. This relation is in 1NF because the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain. This relation is in 2NF because every nonprimary attribute in the relation is not partially dependent on any key in this relation. This relation is in 3NF because all non-key attributes are mutually independent.

Inventory(itemID: int(11), name: varchar(25), brand: varchar(25),
item_type: varchar(25), quantity: int(11), price: float, on_promotion: tinyint(1),
promo_price: float, image_path: varchar(25), description: varchar(255));

All non key attributes in this relation are functionally dependent on the Primary Key.This relation is in 1NF because the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain. This relation is in 2NF because every nonprimary attribute in the relation is not partially dependent on any key in this relation. This relation is in 3NF because all non-key attributes are mutually independent.

Cart(itemID: int(11), desired_quantity: int(11), username: varchar(25));
Foreign Keys: itemID, username

All non key attributes in this relation are functionally dependent on the Primary Keys.This relation is in 1NF because the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain. This relation is in 2NF because every nonprimary attribute in the relation is not partially dependent on any key in this relation. This relation is in 3NF because all non-key attributes are mutually independent.

Orders(orderID: integer, username: varchar(25), status: tinyint(1), date: date,
shipping_adress_street: varchar(25), shipping_adress_state: varchar(25), shipping_adress_city: varchar(25),shipping_adress_zip: varchar(25), total_price: float);
Foreign Keys: orderID, username

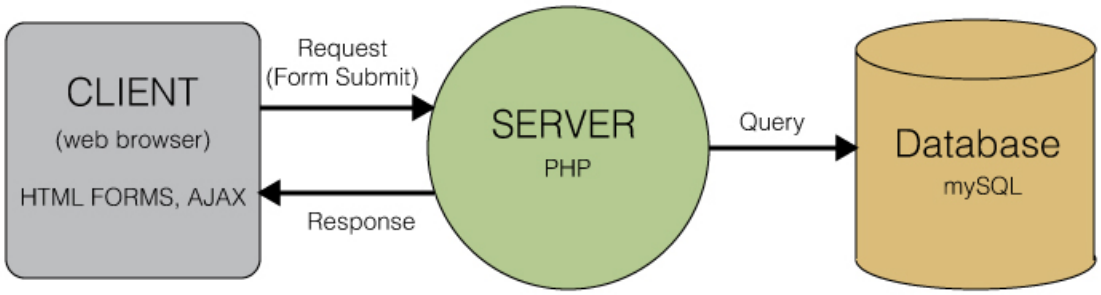
All non key attributes in this relation are functionally dependent on the Primary Key.This relation is in 1NF because the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain. This relation is in 2NF because every nonprimary attribute in the relation is not partially dependent on any key in this relation. This relation is in 3NF because all non-key attributes are mutually independent.

Order_Details(orderID: int(11), itemId: int(11),quantity: int(11), component_price: int(11));
Foreign Keys: orderID, itemID

All non key attributes in this relation are functionally dependent on the Primary Keys.This relation is in 1NF because the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain. This relation is in 2NF because every nonprimary attribute in the relation is not partially dependent on any key in this relation. This relation is in 3NF because all non-key attributes are mutually independent.

Therefore our relations in our database are in 3NF. Each database table in this design achieves Boyce-Codd Normal Form. All data within each of the tables depends only on the table's primary key, and not on any other field in the table. This is supported by the previously stated functional dependencies.

Program Flow Design



1. CLIENT represents the main user interface
    a. Contains all HTML forms, CSS/BootStrap, and JavaScript/JQuery files
    b. Makes requests to SERVER based on user input
2. SERVER represents the interface between the USER and DATABASE information written in PHP code
    a. SERVER accepts HTML form requests
    b. SERVER makes queries to the MySQL DATABASE
    c. SERVER returns returned data to the CLIENT (End User)

Data Structures:

The primary data structures used in this site are arrays used to return MySQL query results from the PHP files to JQuery AJAX post, where the data is then parsed and organized in Boostrap rows and columns on the corresponding HTML page

List of Files and short description

Everyone:

carousel.php //Main page for all users.

falsecart.php // If a non registered user tries to add items to cart a message will be displayed.

login.html // HTML page responsible for showing login page for customer, staff, and manager.

register.html// HTML page for users to register as either customer, staff, and manager.

change_password.php // Responsible for changing the password for all users.

change_user.php // Change user displays forms for users to change password.

connect_DB.php // Used to connect to database and included in other files.

login.php // php code to handle login requests.

logout.php// Handles logging out.

registerUser.php// Connects to database to see if username is taken or not. Handles registration.

render_nonuser.php // Renders the shop items for non registered users.

session.php // Creates session for username. Included in every file to ensure people are logged in , in order to connect to database they have to be logged in.

user_checker.php // Checks who the user is and then directs users to correct welcome page.

Customer:

addtocart.php // Connecs to database to add items to cart when button is clicked.

checkout.php // Check out the customers cart and creates orders and order_details in the database as well as removing the items from cart.

customer_history.php //Connects with database to retrieve appropriate information to be parsed by customer_order_history.php

customer_order_history.php// Contains HTML to display the customer history to the screen.

customer_welcome.php // Homepage for customers

deleteFromCart.php// Talks with database so that customer can remove item from cart.

render_shop.php// Renders dynamic arrays with shop item information that are to be parsed.

shop.php // Contains html tags to display to the customer shop items, etc.

viewcart.php// Pulls data from database to be used and parsed by viewcartpage.php

viewcartpage.php// Displays the customers cart and current items in their cart.

Manager and Staff:

addItem.php// An addtional feature created so that staff or managers can add a completely new item to store.

staff_welcome.php (staff only)//Homepage for staff.

list_inventory.php (Staff only)//Displays items in stock.

get_items_inventory.php// Gets the items in stock to then be parsed and used in a drop down list.

order_history.php // Connects with database to get all orders and stores them in array to be parsed by View_Inventory_Employee.php.

ship_order.php// Talks with database to Ship the selected order.

updateStock.php // Talks with database to update currents item in stock.

View_Inventory_Employee.php //Contains html tags to display order history of all orders made.

Manager:

list_manager_inventory.php// Lists items in shop to the page.

manager_welcome.php// Homepage ro manager.

sales_history.php// Connects with database to pull data about items concerning when and how many of them have been purchased in the last year, month, or week.

sales_history_manager.php/// Contains html tags to display the information parsed from sales_history.php

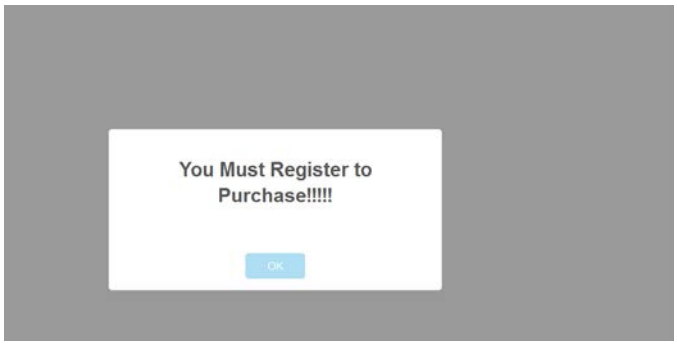
setPromo.php// Sets or removes items from being on promotion as well as their promo price. e.g. 75%, 50%, 25% off.

Program Functions:

Customer Functions:

Home Page:

Non Register customer can view the shop items. but once they Press Add to cart button then they get alerted with popup message stating that they must be register to purchase item with the store and then redirected to register.html page.

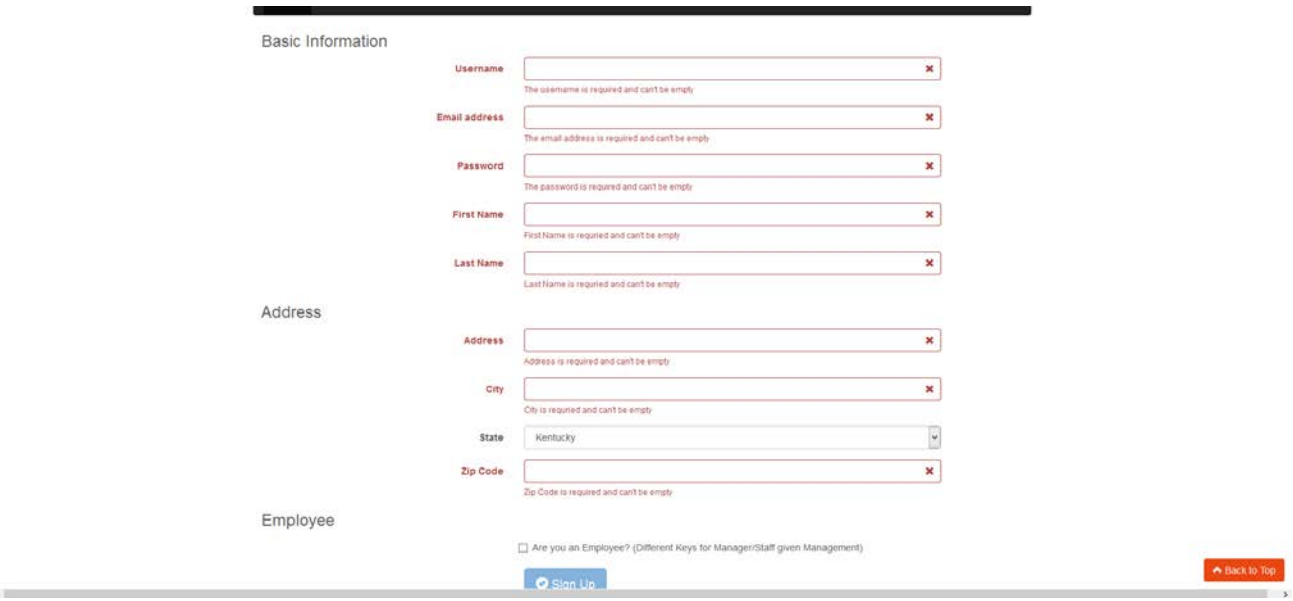
Add New 

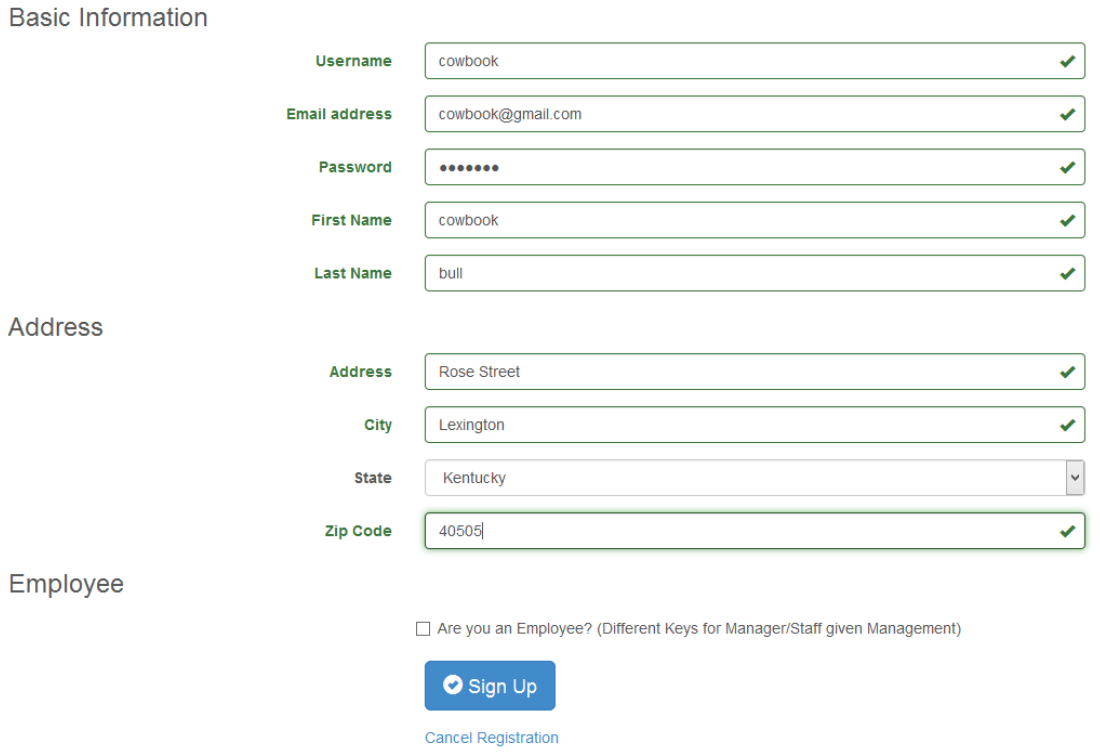
Customer, Staff, and Manager.

(Staff and manager are provided with store password and depending on password given they get assign their roles as staff or manager.)

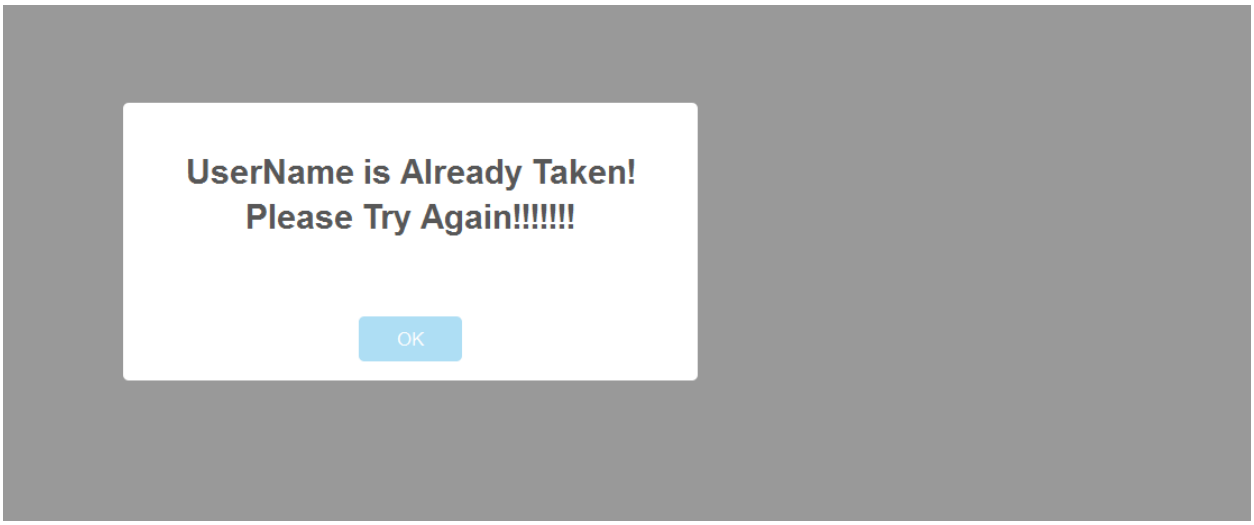
Customers leave (Are you employee field empty) to sign up as customer.

While filling up the register form, user goes through steps of validator in order to make sure that we store correct data in database using JavaScript and BootStrap Validators.





Once the proper information is filled up in correct input places, Customer gets validations using Javascript just showing them that their input is valid to enter. After that Sign up. It goes through database checks whether user name is already in Use or not and alerts user with pop up message saying it's already taken and re-directs them to register page.



Once user gets successfully register then he gets greeted with message. Thank you for registering with us and re-directs page to login.html

Login Screen:

Login Screen is same for all users (Customer, Staff or Manager)

Depending on their User Type, Once clicked on Login, they go to different Pages depending on their types.

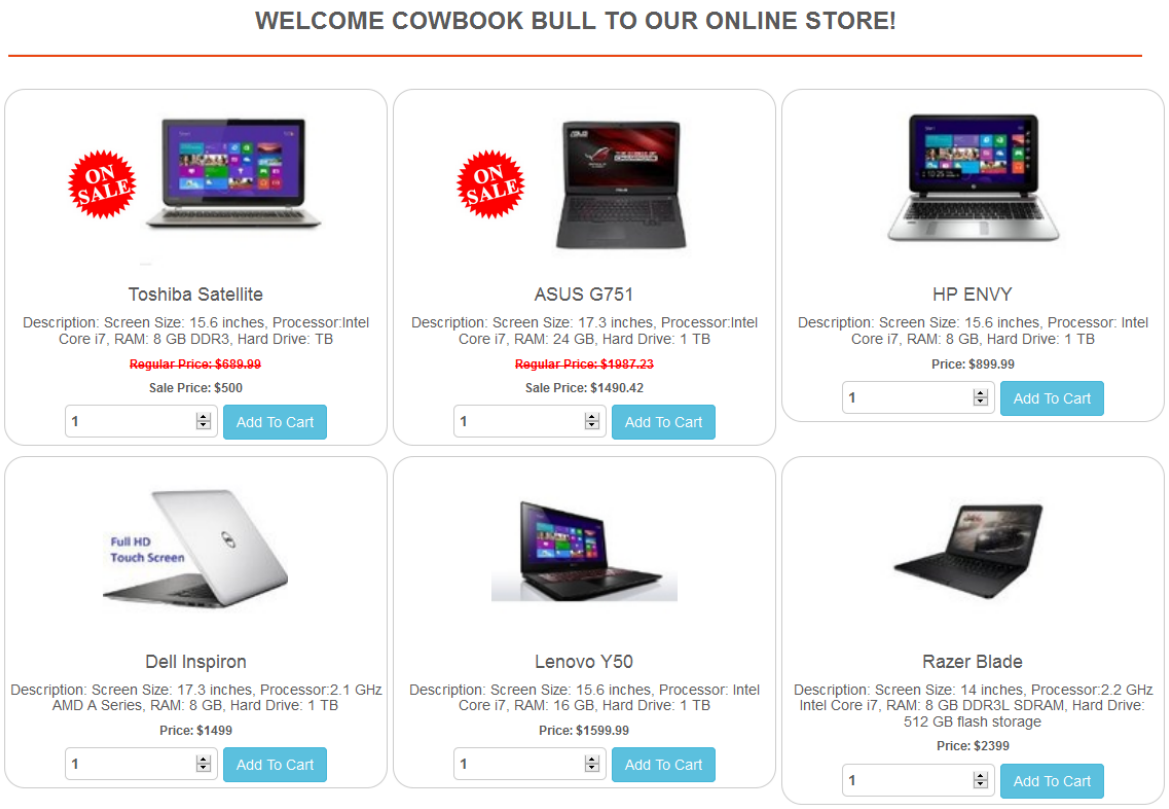
After successful login using new credentials User can now login using login.html creates the secure session start on the web browser and holds their session.

Customer now logins to their main page.

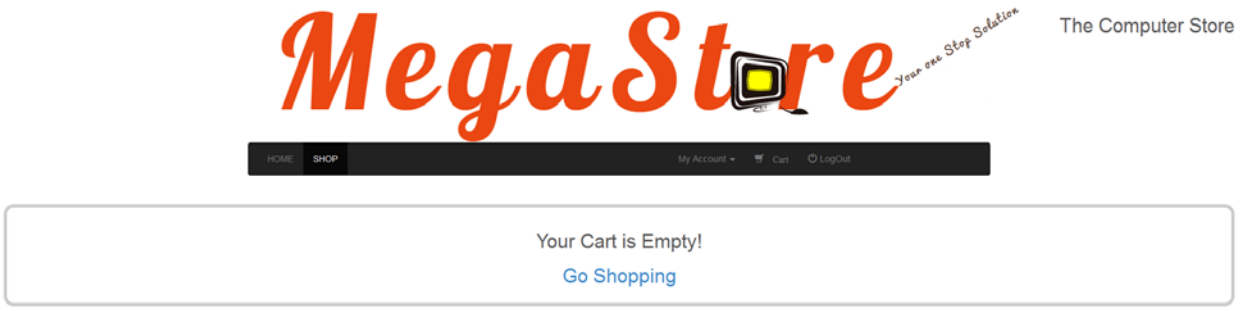
On this page, Customer has access to view shopping items though SHOP tab.

My Account tab gives them access to change their account password or settings and also they can view their previous orders.

Shopping: Shopping page gives varieties of items where user can select the desired quantity and add item to their shopping cart. On shopping page, customers also get to see whether the item is on promotion or not.

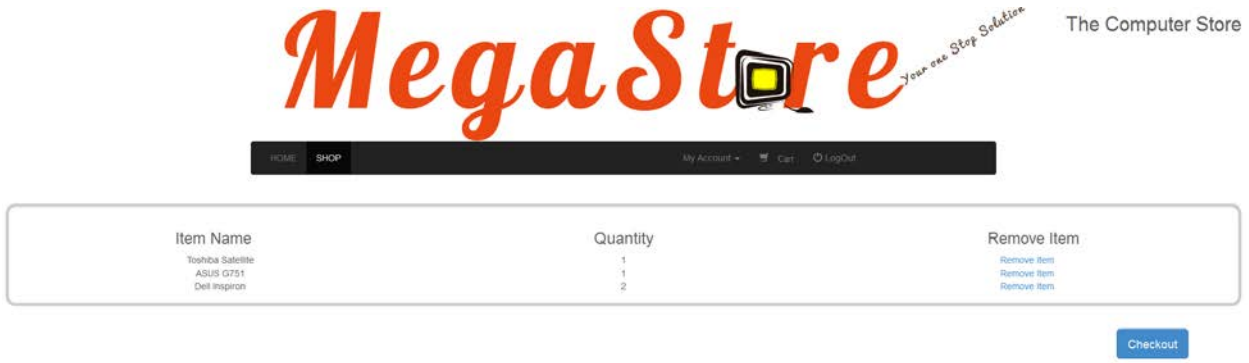


Currently I don’t have any items in cart so it should show user cart empty.



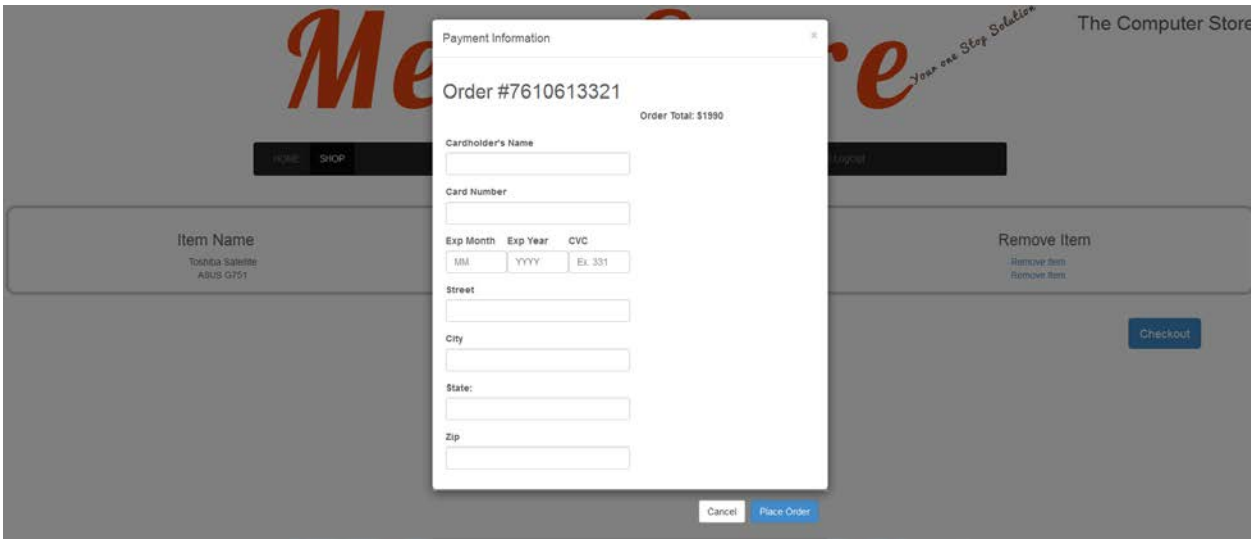
Now I will add Toshiba Satellite and Dell Inspiron to my shopping cart.

Now, I can either go back to shopping or update my cart by adding more items

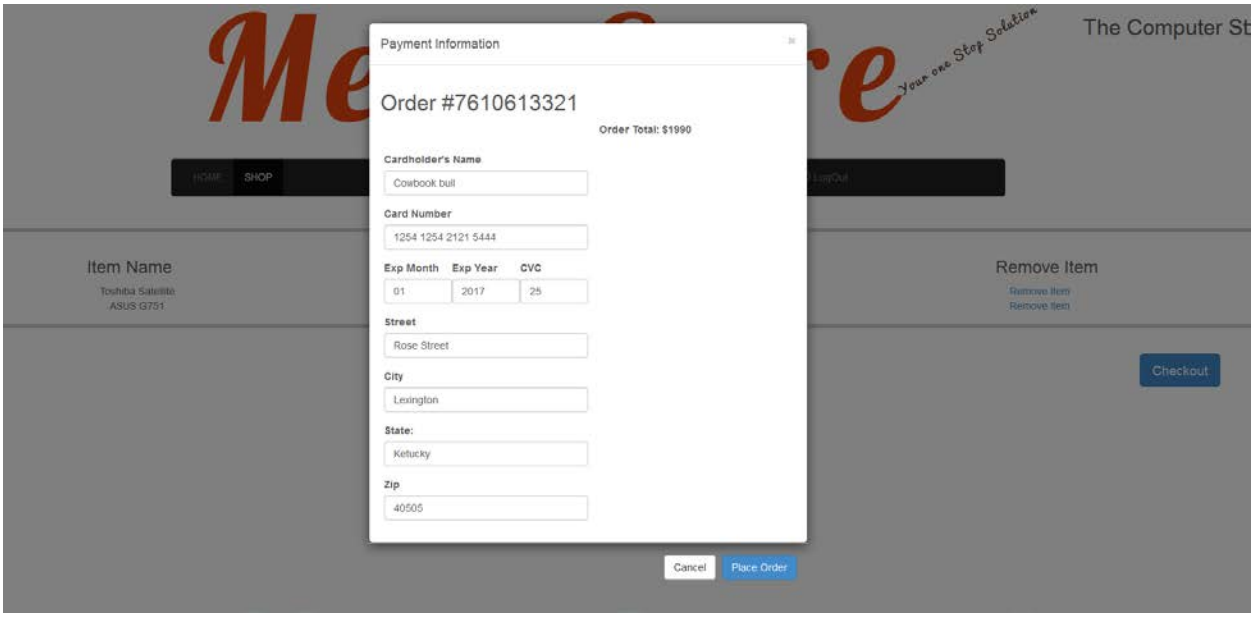


Or I can Remove item for example I am going to remove Dell

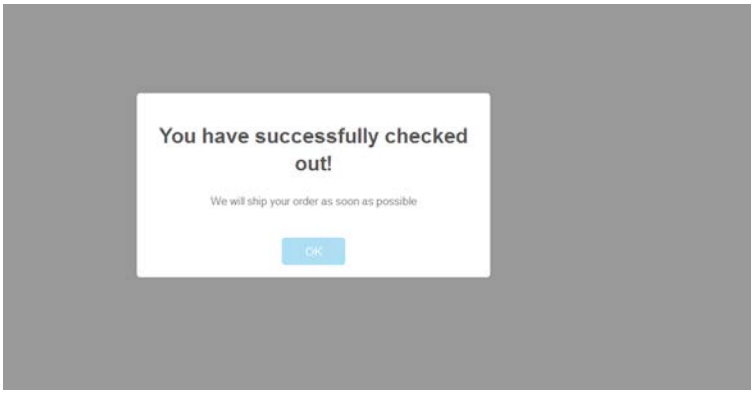
Next Step Check Out, Once Pressed on Checkout



It Generates the total price for the order, Order Id and Asks for the user payment and shipping information.



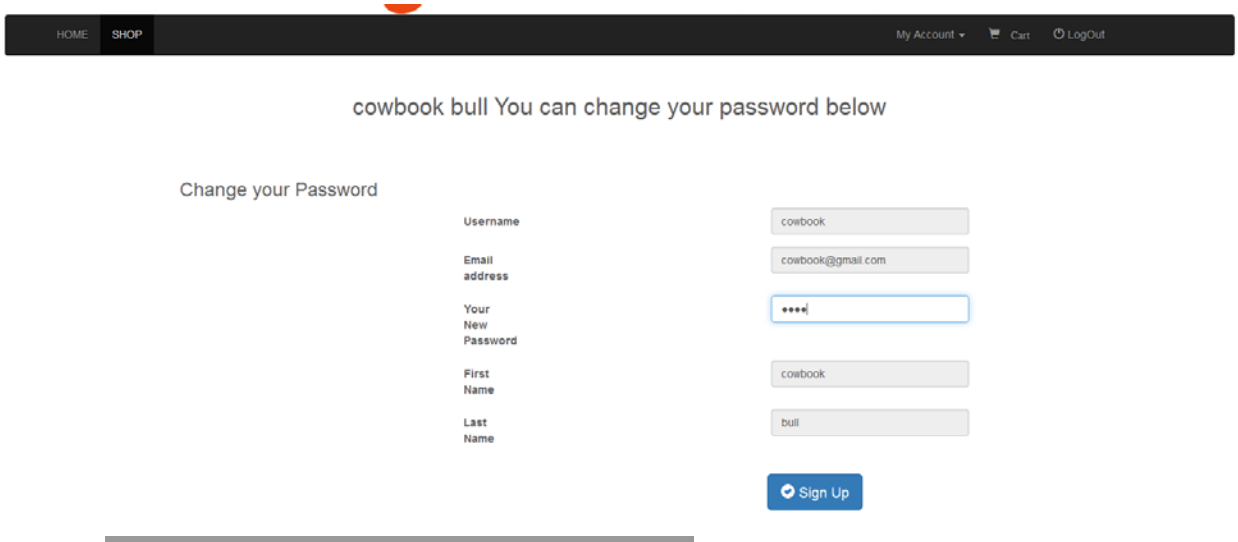
After the valid information received from the user, now I can Place the order. And It gets removed from cart.



Order History

Now Order shows Status Pending on it until Staff / Manager Ships their items and then status Changes to Shipped instead of pending.

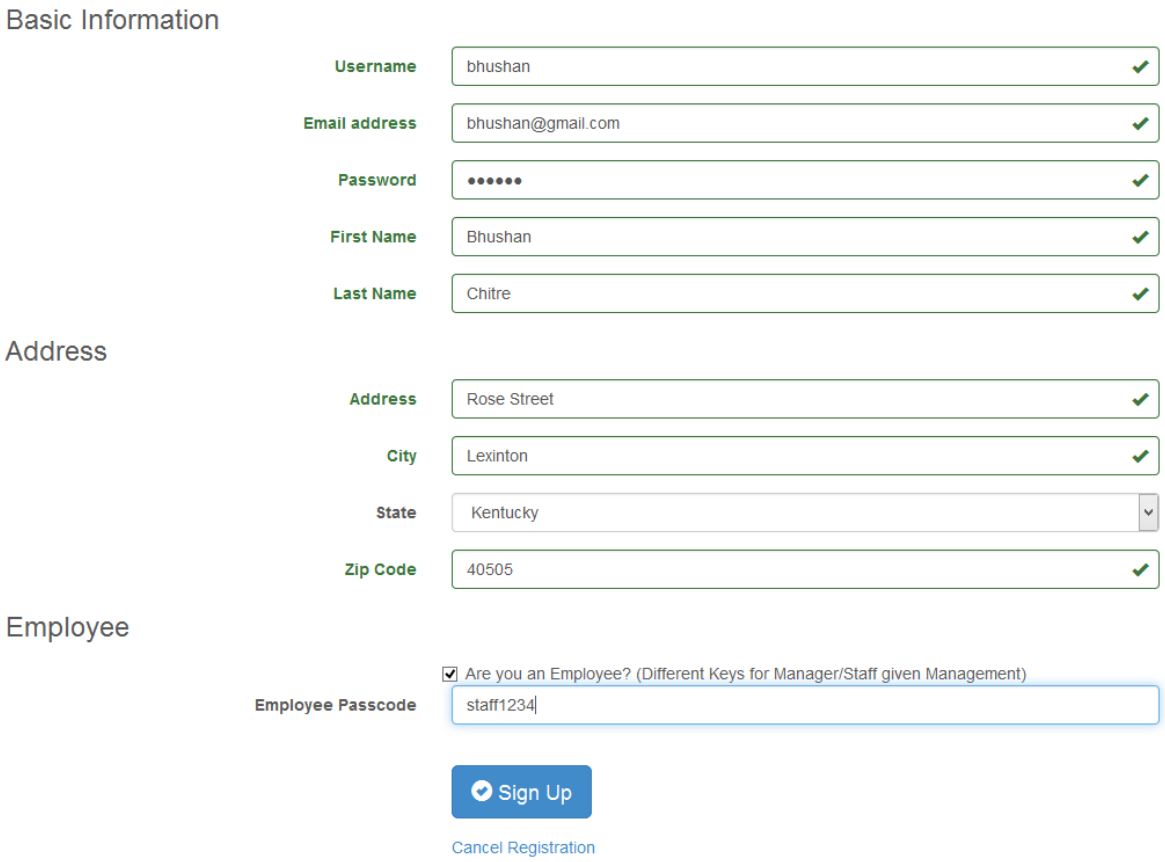
User Account setting to remind user their account information and they can update their information if they wish.



Logout, 

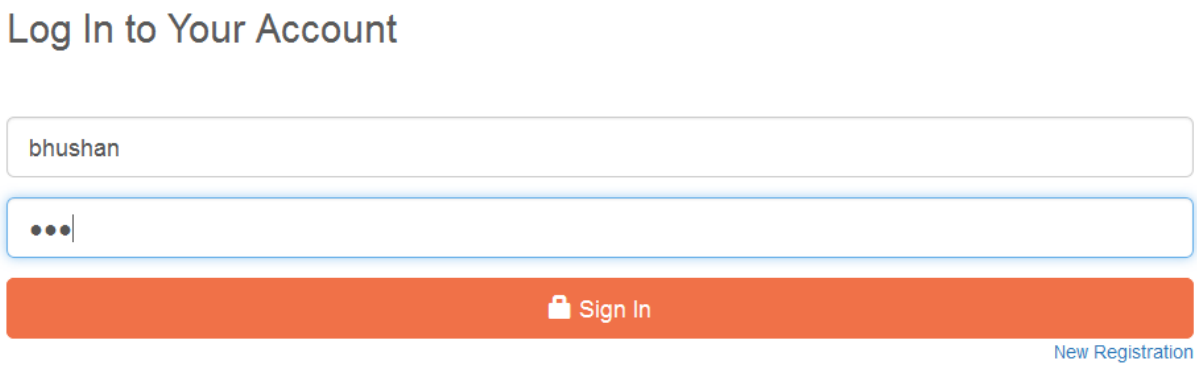
Staff Functions:

Staff can also create new account once they receive store password and then they can follow the same method as Customer except they will enter secrete key in place and click on checkbox for are you employee. Depending on their secrete key they will get access as Staff or Manager.



Following by the same validators, staff is now register and ready to login.

Depending on user type after user login, they get logged in to proper pages.

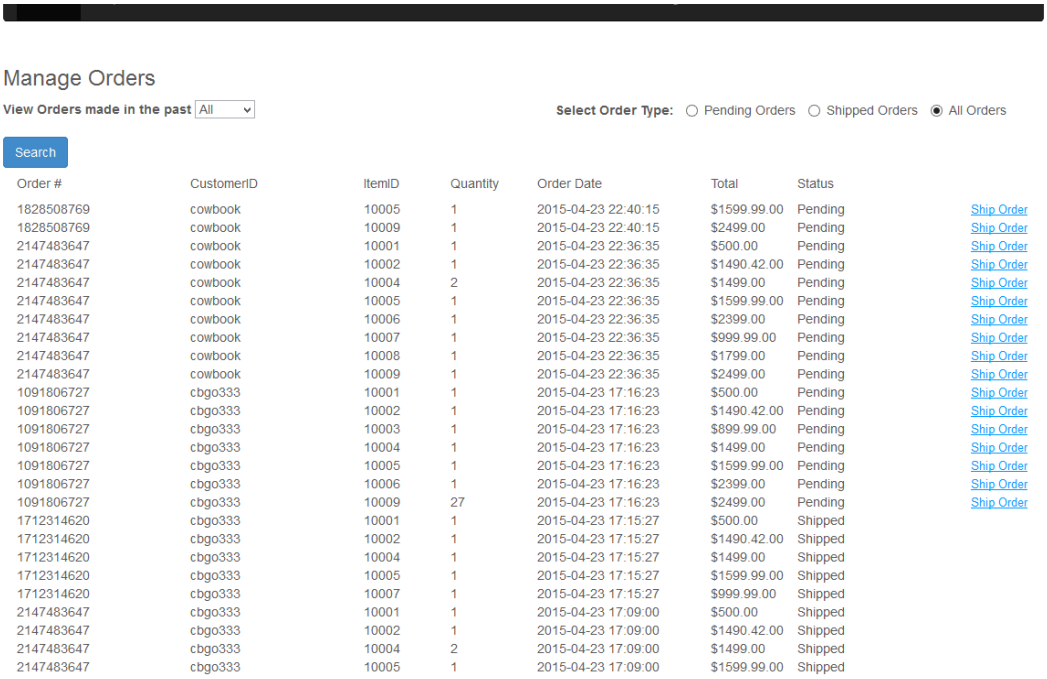


Once that’s complete

Staff is greeted with their staff page with appropriate access tabs.

Now, Staff can Ship Pending Order, Change their Account Settings, Restock and Add new Item to Inventory.

Staff Ship and View Orders.



Manage Orders

View Orders made in the past All

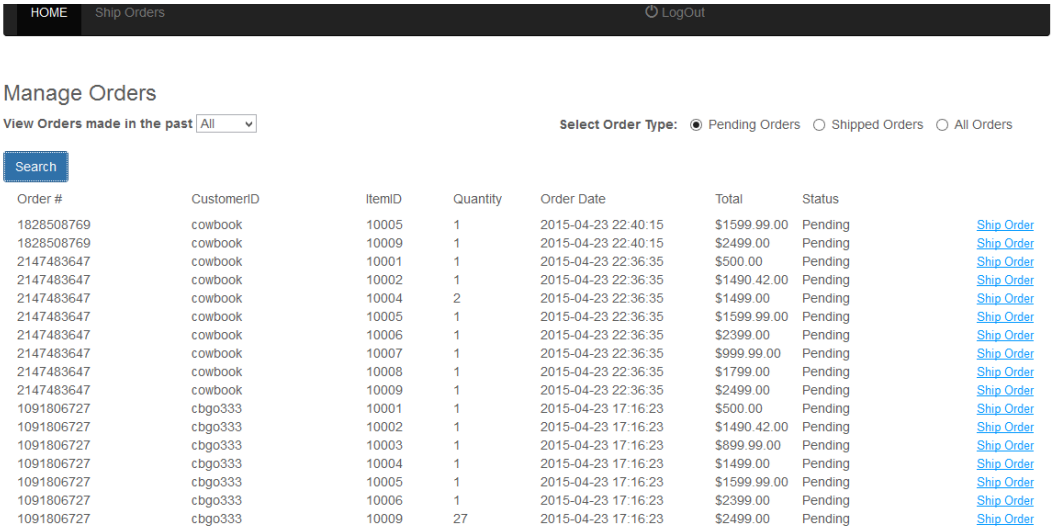
Select Order Type: Pending Orders Shipped Orders All Orders

Search

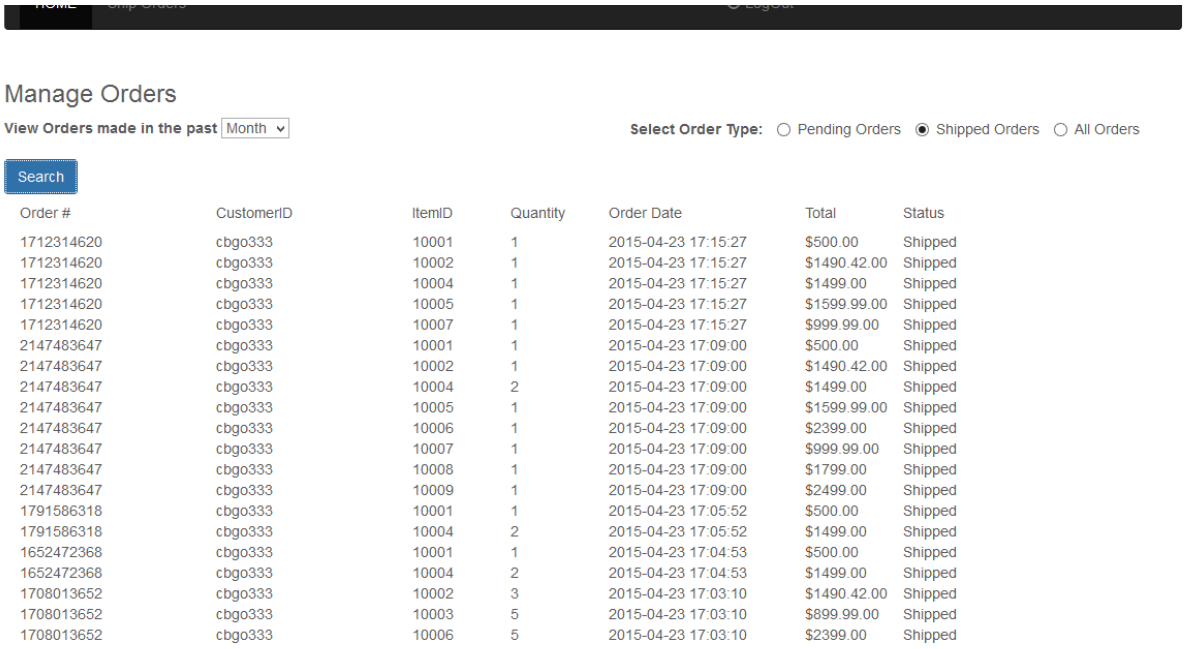
Order #	CustomerID	ItemID	Quantity	Order Date	Total	Status	
1828508769	cowbook	10005	1	2015-04-23 22:40:15	$1599.99.00	Pending	Ship Order
1828508769	cowbook	10009	1	2015-04-23 22:40:15	$2499.00	Pending	Ship Order
2147483647	cowbook	10001	1	2015-04-23 22:36:35	$500.00	Pending	Ship Order
2147483647	cowbook	10002	1	2015-04-23 22:36:35	$1490.42.00	Pending	Ship Order
2147483647	cowbook	10004	2	2015-04-23 22:36:35	$1499.00	Pending	Ship Order
2147483647	cowbook	10005	1	2015-04-23 22:36:35	$1599.99.00	Pending	Ship Order
2147483647	cowbook	10006	1	2015-04-23 22:36:35	$2399.00	Pending	Ship Order
2147483647	cowbook	10007	1	2015-04-23 22:36:35	$999.99.00	Pending	Ship Order
2147483647	cowbook	10008	1	2015-04-23 22:36:35	$1799.00	Pending	Ship Order
2147483647	cowbook	10009	1	2015-04-23 22:36:35	$2499.00	Pending	Ship Order
1091806727	cbgo333	10001	1	2015-04-23 17:16:23	$500.00	Pending	Ship Order
1091806727	cbgo333	10002	1	2015-04-23 17:16:23	$1490.42.00	Pending	Ship Order
1091806727	cbgo333	10003	1	2015-04-23 17:16:23	$899.99.00	Pending	Ship Order
1091806727	cbgo333	10004	1	2015-04-23 17:16:23	$1499.00	Pending	Ship Order
1091806727	cbgo333	10005	1	2015-04-23 17:16:23	$1599.99.00	Pending	Ship Order
1091806727	cbgo333	10006	1	2015-04-23 17:16:23	$2399.00	Pending	Ship Order
1091806727	cbgo333	10009	27	2015-04-23 17:16:23	$2499.00	Pending	Ship Order
1712314620	cbgo333	10001	1	2015-04-23 17:15:27	$500.00	Shipped	
1712314620	cbgo333	10002	1	2015-04-23 17:15:27	$1490.42.00	Shipped	
1712314620	cbgo333	10004	1	2015-04-23 17:15:27	$1499.00	Shipped	
1712314620	cbgo333	10005	1	2015-04-23 17:15:27	$1599.99.00	Shipped	
1712314620	cbgo333	10007	1	2015-04-23 17:15:27	$999.99.00	Shipped	
2147483647	cbgo333	10001	1	2015-04-23 17:09:00	$500.00	Shipped	
2147483647	cbgo333	10002	1	2015-04-23 17:09:00	$1490.42.00	Shipped	
2147483647	cbgo333	10004	2	2015-04-23 17:09:00	$1499.00	Shipped	
2147483647	cbgo333	10005	1	2015-04-23 17:09:00	$1599.99.00	Shipped	

Where they can Sort the Orders, Per Year, Month, Week basis And also per their Status such as All pendings and all shipped orders.

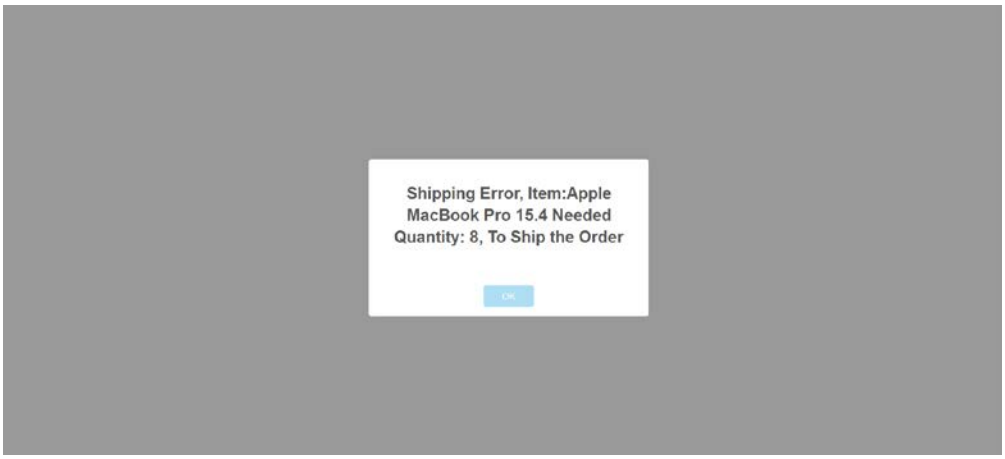
All Pendings



All Shipped Orders per month

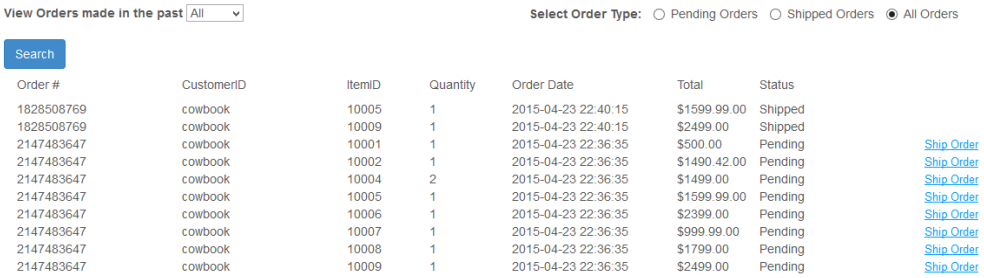


IN this staff can ship pending order and then it quantity gets decrease from the inventory. They can't ship orders until all required quantity in stock to fulfil the order.

Shipping error. 

Saying it can’t ship order until staff restocks 8 new MacBook Pro 15.4 in Inventory.

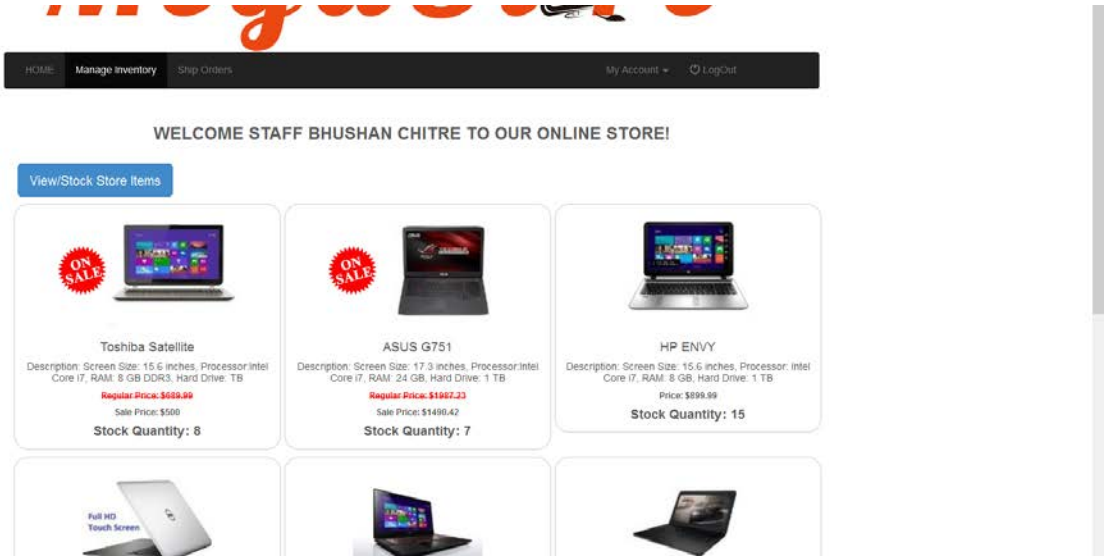
Just Shipped user cowbook orders



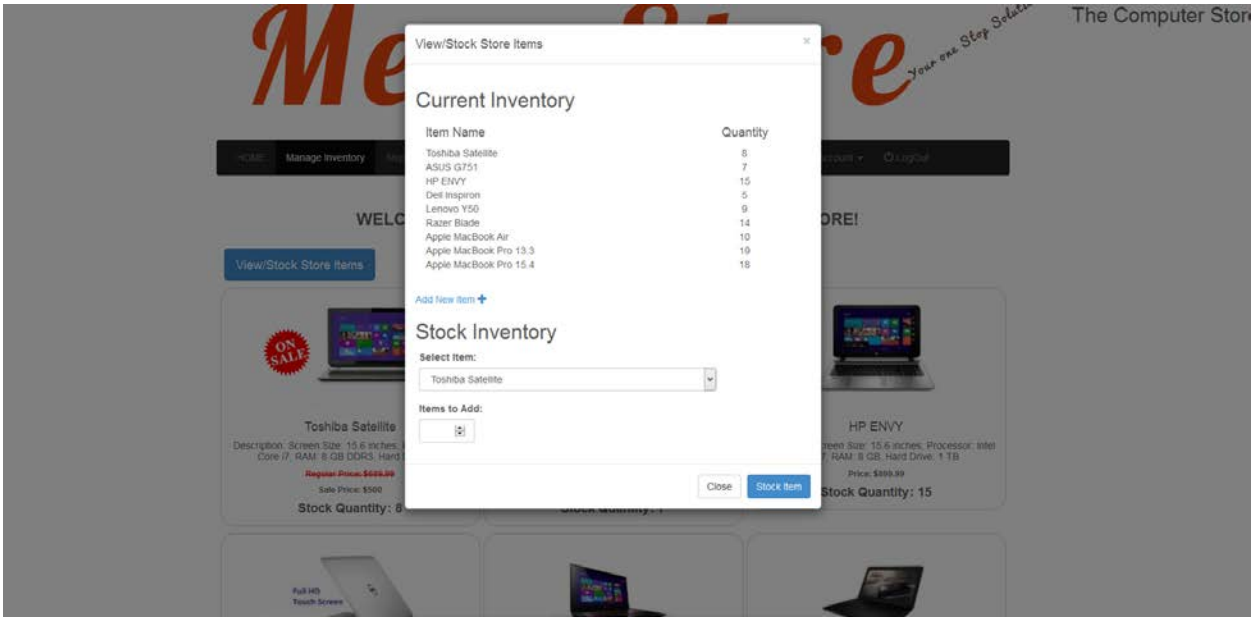
Order #	CustomerID	ItemID	Quantity	Order Date	Total	Status	
1828508769	cowbook	10005	1	2015-04-23 22:40:15	$1599.99.00	Shipped	
1828508769	cowbook	10009	1	2015-04-23 22:40:15	$2499.00	Shipped	
2147483647	cowbook	10001	1	2015-04-23 22:36:35	$500.00	Pending	Ship Order
2147483647	cowbook	10002	1	2015-04-23 22:36:35	$1490.42.00	Pending	Ship Order
2147483647	cowbook	10004	2	2015-04-23 22:36:35	$1499.00	Pending	Ship Order
2147483647	cowbook	10005	1	2015-04-23 22:36:35	$1599.99.00	Pending	Ship Order
2147483647	cowbook	10006	1	2015-04-23 22:36:35	$2399.00	Pending	Ship Order
2147483647	cowbook	10007	1	2015-04-23 22:36:35	$999.99.00	Pending	Ship Order
2147483647	cowbook	10008	1	2015-04-23 22:36:35	$1799.00	Pending	Ship Order
2147483647	cowbook	10009	1	2015-04-23 22:36:35	$2499.00	Pending	Ship Order

And status changes to shipped.

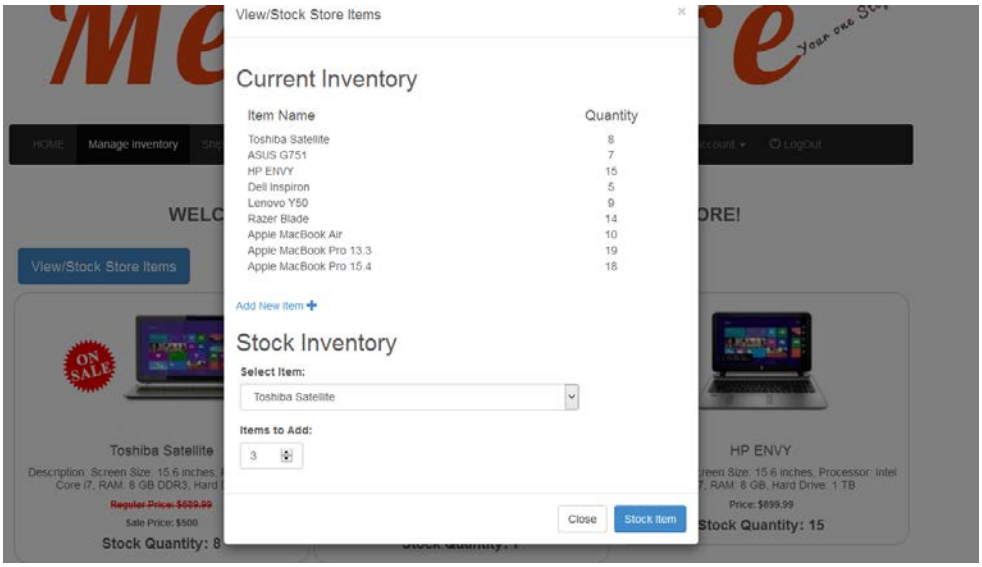
Manage Inventory, where staff can view the item, ReStock and Add new Item to inventory.



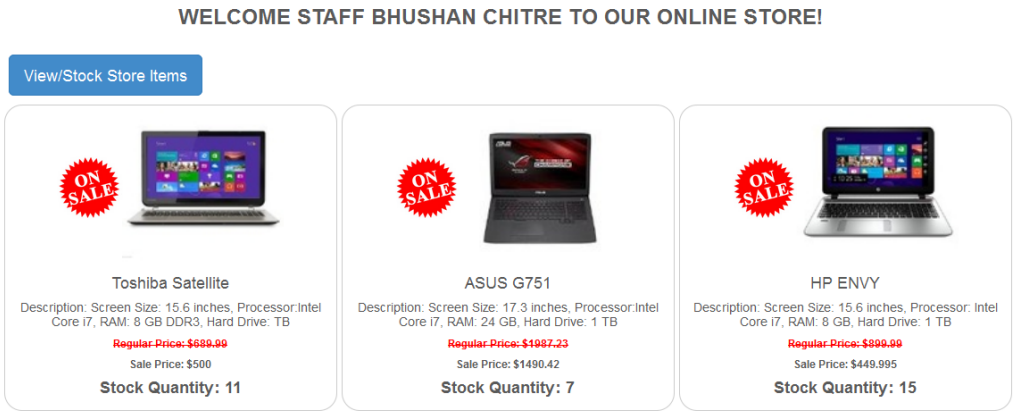
View Stock items and new Item



Now I will restock item for

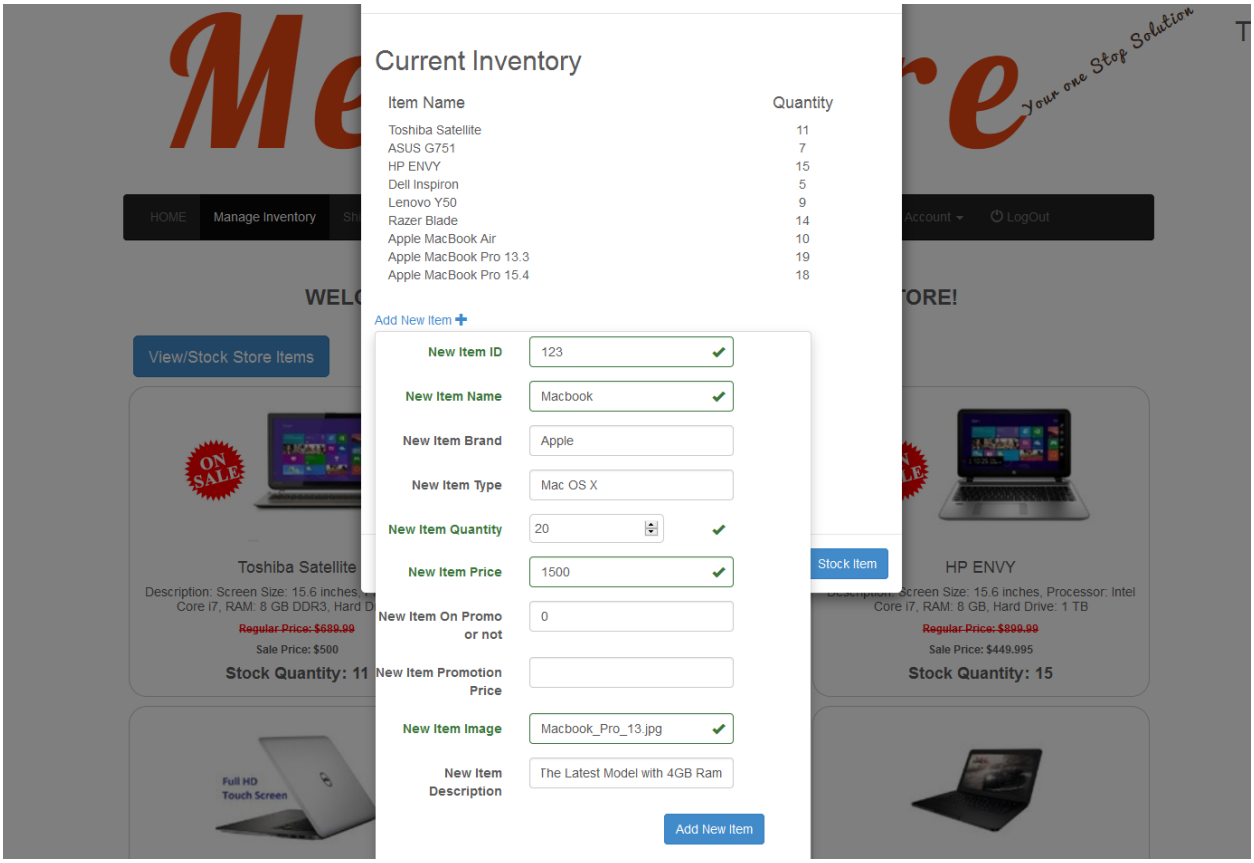


After restocking 3 items

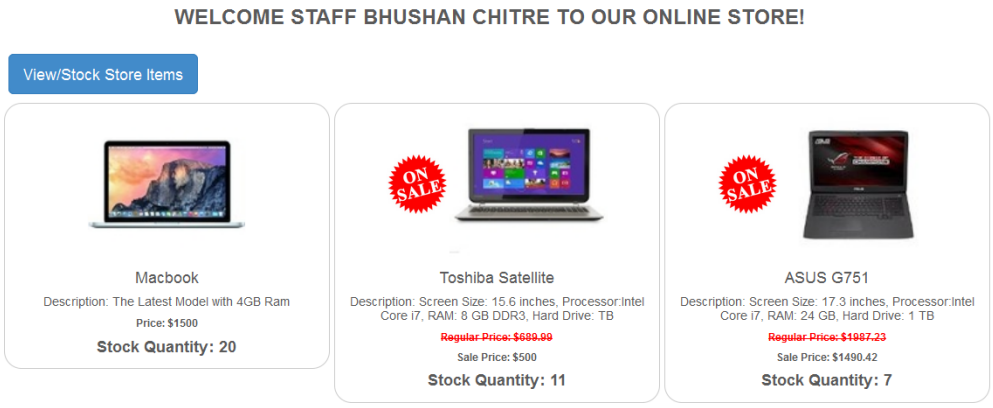


Now Toshiba has 11 items in stock.

Now Showing you Adding new Item to Inventory

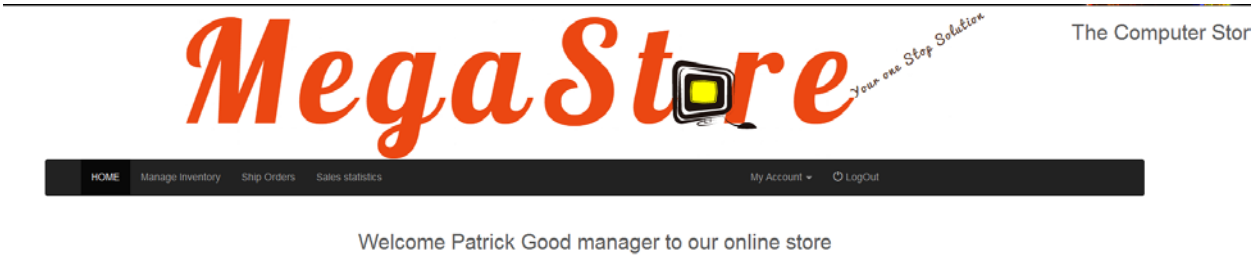


Now, New Macbook should be added in Inventory before Toshiba Satellite. With 20 quantity.



Manager Functions:

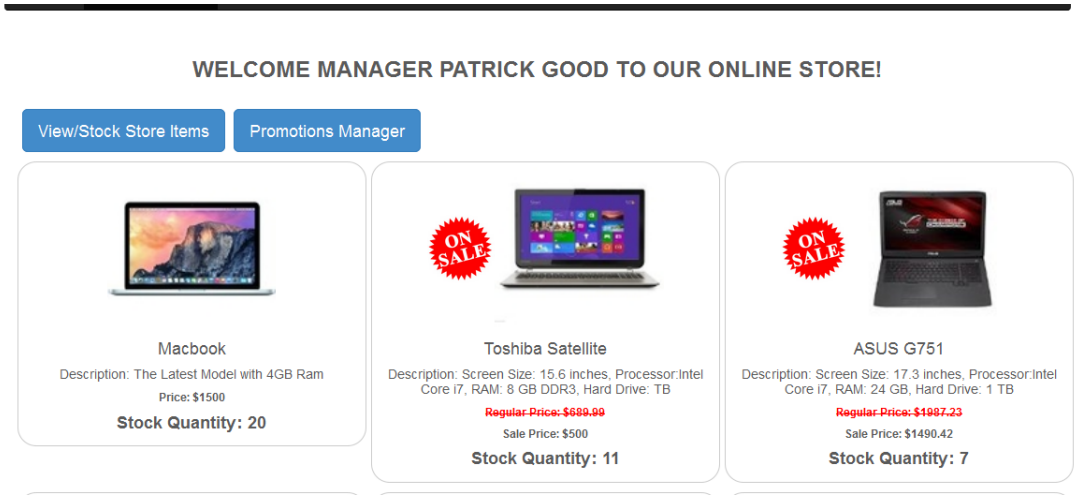
As similar to staff Manager Sign up for account if new to store and gets secrete key from store to register as Manager. Manager gets logged in depending on the user type. Once logged in to store gets welcome similar to staff except additional features like Put Items On promotional price and Sales History where he can see Items sale history past month, past week and past year.



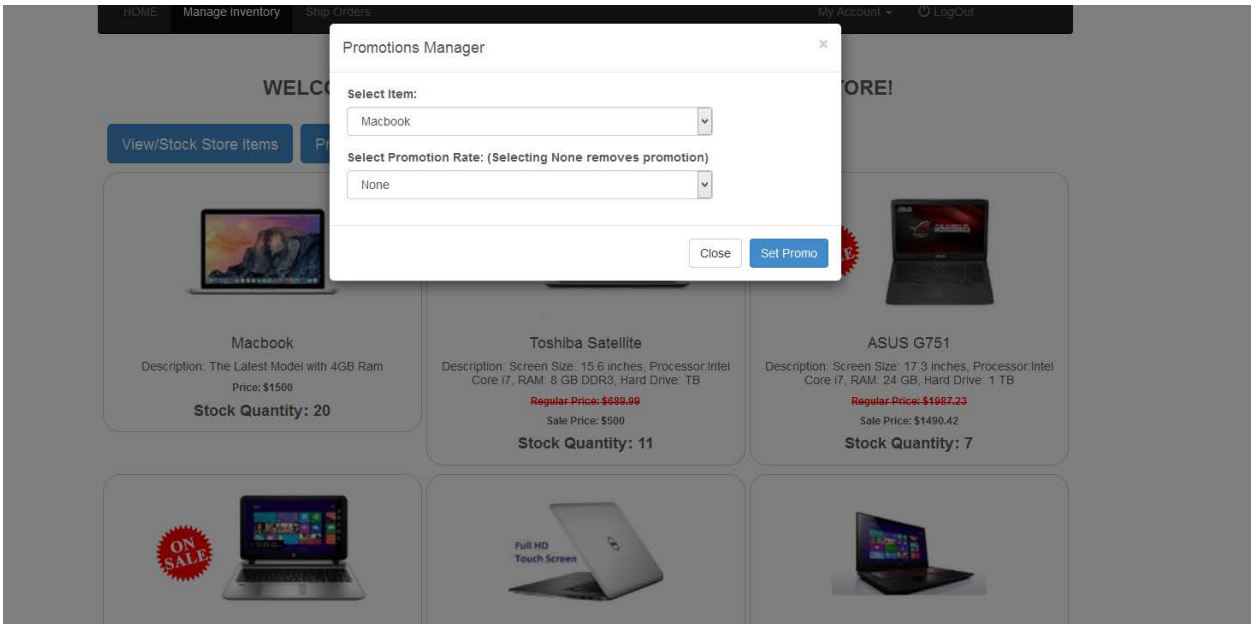
Similar to staff he can ship order and view stock, re-stock and add new item to inventory.

So I am going to show Add item on promotion or remove Item from promotion.

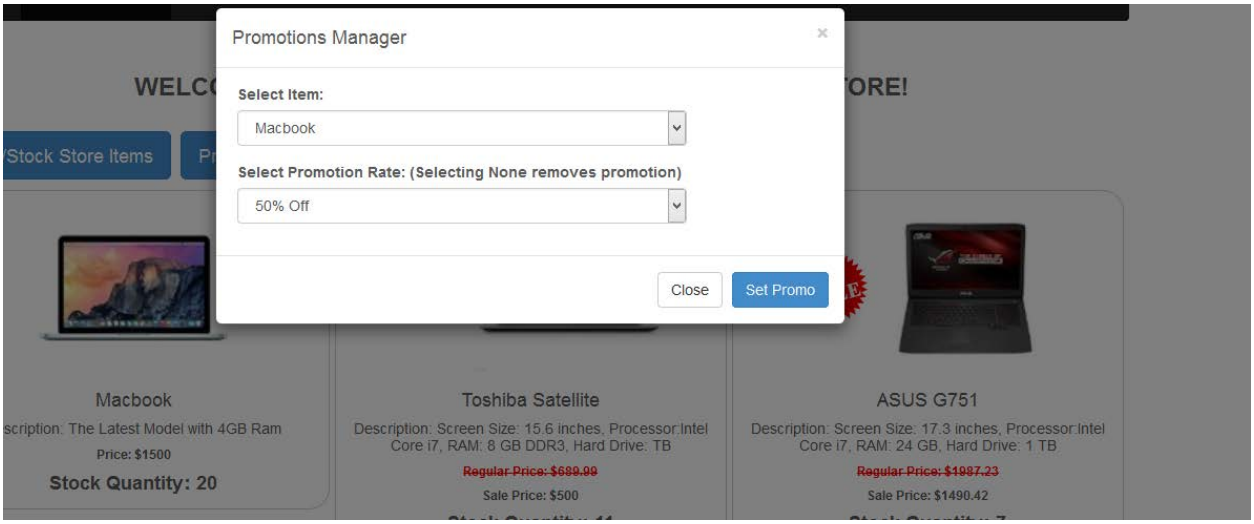
Shown Below,



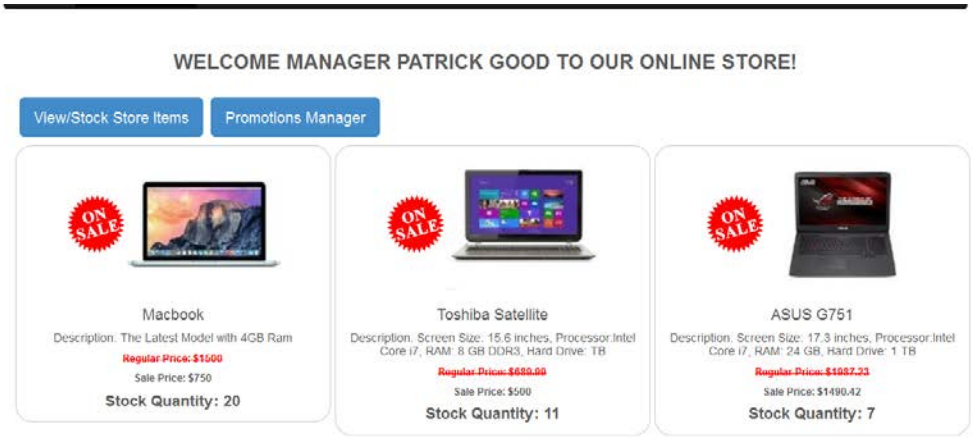
Toshiba is On promotion so I am going to remove Toshiba from Promotion and add Macbook on promotion. Promotional price is Based upon 20%, 50% and 75% sale so Manager doesn't have to calculate the price its gets done automatically for him.



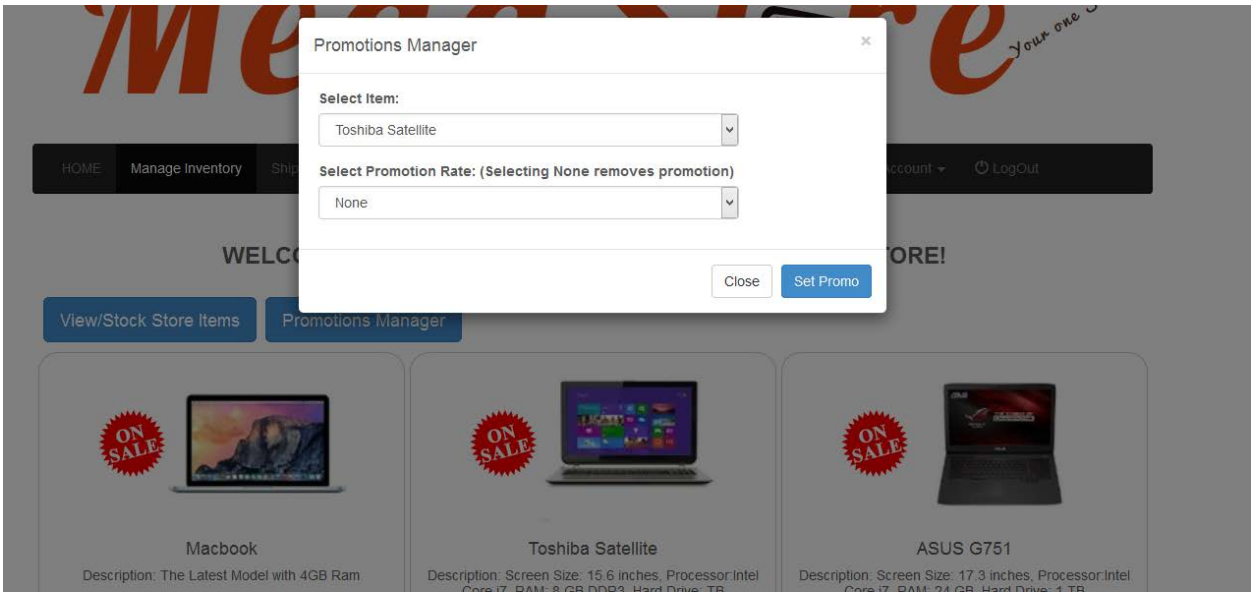
I am going to select Macbook and add 50% off on promotional Price.



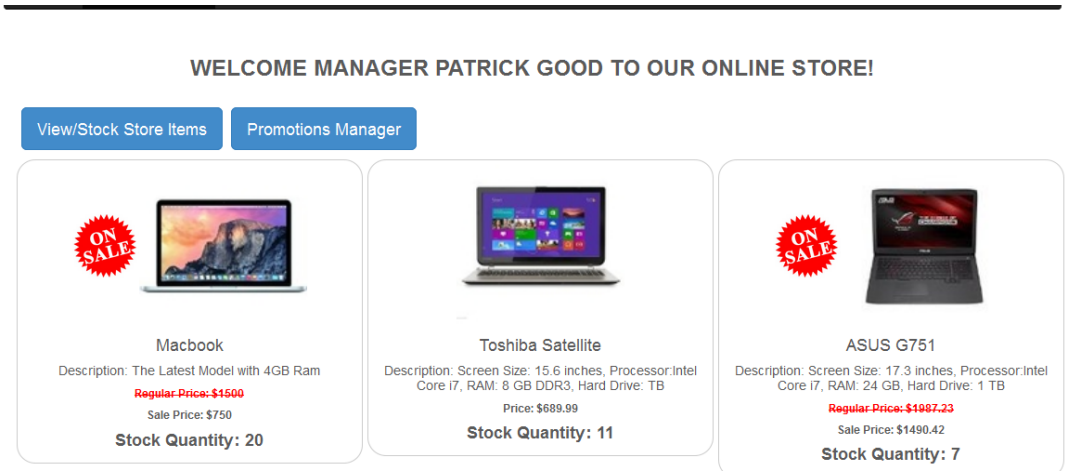
Now, Macbook is On sale 50% off



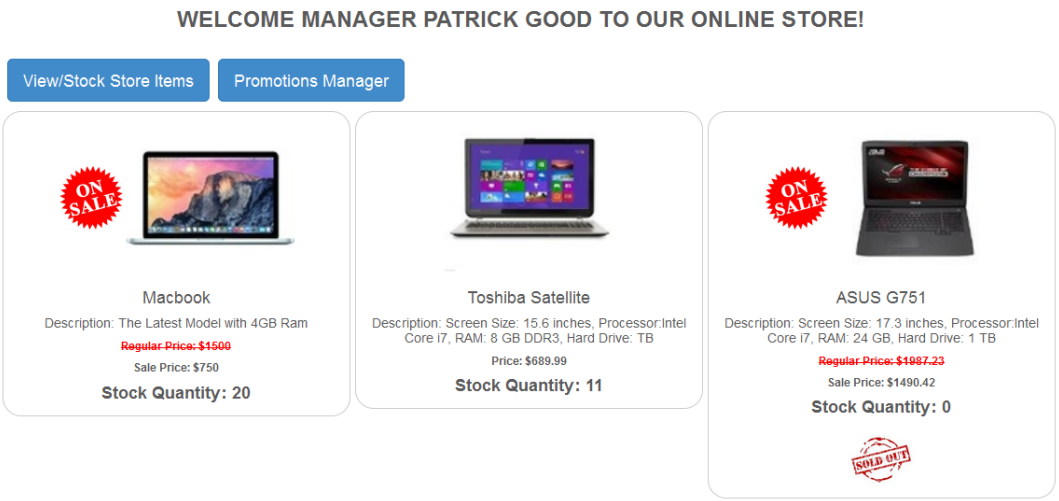
Remove Toshiba from Promotion.



Now, Toshiba Satellite is no longer on Promotion.

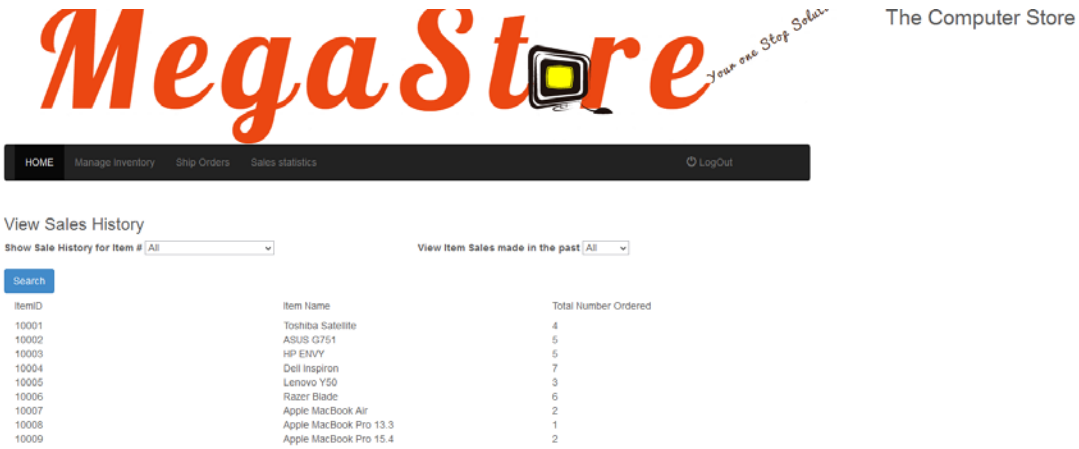


If Out of Stock



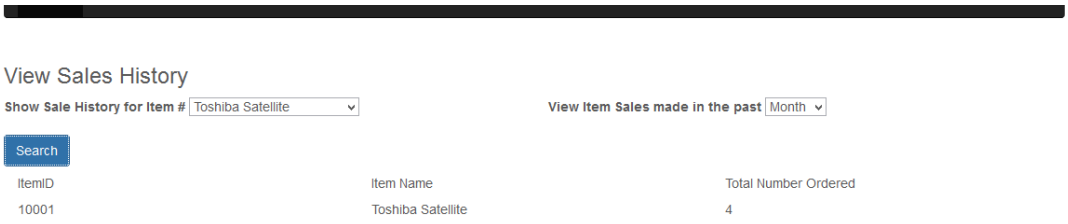
Sold out sign shows up.

Sales Statistics:



Where Manager can select item name and select how many of those items have sold in past week, past month and past year.

Showing Toshiba how many sold in past month



And so on, for each item for just all inventory detail in past month, year or week.

Testing:

We tested the program by doing unit testing at the beginning and then started adding units together to test the whole program. At first we did testing on the program to make sure that the main screen, registration, log in and log out worked. We tested each registration for each user type (customer, staff, and manager). We then tested login for each type as well. After we are able to login we made sure the correct pages were displayed for each user type. We then tested logging out for each user. We then tested to make sure that the session was created correctly for each user and then destroyed with the user logged out.

From here we moved on to testing the customer functions. So we started with the testing the customers' ability to go to the shopping page and then proceed to view and add various items to their cart. We tested by adding only a few items to cart and adding many items to the cart. From here we made sure that the cart pages were displaying correctly so that the user could view the cart and the items they added to it. We also made sure they could remove items from the cart. Next we ensured that the user could then check out their items and that the appropriate messages were shown. After the users placed orders we tested to make sure they could make multiple purchases for each order. We then tested to make sure that the user could then see previous orders that he placed and whether it has shown shipping or pending.

After testing for the customer user we moved onto testing the staff functions. So we made sure they could view the inventory, but not purchase anything. We then tested to make sure they could update the inventory. We also ensured that the staff could see and ship customer's orders which had been made. Then we tested an additional function to allow managers and staff to add new items to the store. We also tested to make sure that a message would be displayed if the staff tried to ship an order when the store didn't have the items available.

After that we began testing Manager Functions just to make sure they could do the same that the staff could do. So we made sure they could view items but not purchase. We made sure they could add an item and made sure they could update the stock like the staff can. After that we tested to make sure that managers could also view orders and ship them and that the message would appear if there wasn't sufficient items in stock. We then made sure that after shipping an order that the status would change like it does in staff. We then began to test to make sure that the manager could view sales statistics. So we tested it by adding orders to the inventory that older than 7 days and older than a month so that we could make sure that the customer could narrow the results. After this we tested sales history to make sure that Managers could put an item on sale and change the displayed price and the promotional price in the database. We also tested to make sure that they could revert an item to regular price again. After testing these individually we tested them in conjunction with each other. I logged in as one user and he logged in as another to insure that these functions worked when there are multiple users using the website.

In order to test specific php files we used the following methods. We often used echo to print values to the screen and then used these values to figure out why our code wasn't working.

This is what we used most of the time and we usually ended up hard coding values to determine why our sql queries weren't working. We then would change them back so that we could use POST and GET to send values to other pages. When working with ajax and Jquery we used functions such as document write to do the same thing as we did above.

Project Overall Experience:

From the beginning of the project, we wanted to create a store that could bring the user an immersive experience involving online shopping. Through our project we created MegaStore the computer Store website which gives user that experience of online shopping. We were able to achieve some great results of new styles and sleek design using PHP, CSS/BootStrap and JavaScript. Overall after few struggles with learning php and CSS/BootStrap framework we really learned a lot about Mysql.

- Bhushan Chitre

Overall I enjoyed this project. I enjoyed designing and developing an online store that resembled both the look and feel of an actual online store. I enjoyed designing the database especially it was very enjoyable to design how users are to interact with specific items and what attributes those item had. I am satisfied with what we were able to achieve by developing our online store. At times it was very frustrating, especially when debugging the files which got very repetitive, but I learned so much about databases, PHP, AJAX, and JQUERY. At first the project was a frustrating experience, but as my knowledge of using databases and developing websites increased I was able to write and debug code effectively. I also enjoyed working with my partner because it was nice working on the same project and being able to ask questions and get answers in a detailed manner. Having an extra set of eyes to debug the code was very helpful. Together we accomplished what we set out to do.

-Patrick Good.