YOUR AVERAGE JOE

E-Commerce Site

CSIT-355: Database Systems

Dr. Katherine Herbert

Spring 2016 Final Project

Ronald Mangiliman

Arthur Rozenberg

Gene Hernandez

Daniel Wiesel

Table of Contents

Contents

1. Concept………………………....………………………………………………4
2. ER Diagram…………………………………………………………………….6
3. Relational Table………………………………………………………………...8
4. Database Design………………....…………………………………………….19
5. Site Overview………………………………………………………………….21
6. Site Implementation…………………………………………………………...26
   1. Logging In………………………………………………………………....29
   2. Registering New Customers……………………………………………….35
   3. Resetting User Passwords………………………………………………....37
   4. Customer View……………………………………………………………40
   5. Changing User Passwords………………………………………………...44
   6. Managing Personal Information…………………………………………..46
   7. Displaying Categories, Brands and Products…………………….……….51
   8. Search Engine……………………………………………………………..56
   9. Product Profile and Adding to Cart……………………………………….56
   10. Checking Out……………………………………………………………...63
   11. Viewing Order History…………………………………………………....76
   12. Logging Out………………………………………………………………80
   13. Employee View……………………………………………………….......80
   14. Updating Quantity………………………………………………………...81
   15. Updating Order Statuses…………………………………………………..83
   16. Database Administrator View…………………………………………….86
   17. Adding a New Product…………………………………………………....87
   18. Edit Existing Product Details……………………………………………..89
   19. Owner View……………………………………………………………....90
   20. Creating New Employee Accounts……………………………………….92
   21. Update Existing Employee Accounts………………………………….....93
   22. Viewing Existing Employees…………………………………………….95
7. Analysis of Database Issues………………………………………………….98
8. Work Plan…………………………………………………………………….99
   1. Ron’s Scripts……………………………………………………………..99
   2. Arthur’s Scripts…………………………………………………………..99
   3. Gene’s Scripts……………………………………………………………99
   4. Dan’s Scripts……………………………………………………………..100
9. Security …………………………………………………………………….101
   1. Measures Against SQL Injections……………………………………...101
10. Appendix of Code…………………………………………………………..103
11. Conclusion………………………………………………………………….104
12. Citations…………………………………………………………………….105
13. Business Plan and Status Reports…………………………………………..109
14. Concept

The concept for our E-Commerce site is a coffee/beverage business that we called Your Average Joe. This business would sell coffee related products, as well other beverages one would be able to find when visiting a café or coffee shop, such as tea, hot chocolate, etc.

Our site at the time of this writing offers fifty-seven products spread across seven categories fifteen real-world brands. Our categories are coffee bags, accessories, tea, hot chocolate, K-cups, Coffee Machines, and other beverages. Coffee bags are self-explanatory. They are the bags of ground coffee beans one can buy at stores, cafes, coffee shops, etc. Accessories consists of items such as mugs, tumblers, and tea pots. Tea is another common beverage served at cafes, so it was included in the site as well. Our tea products consist of boxes of tea bags of varying flavors, brands, and counts. Our hot chocolate products are similar to our tea products. They are boxes of hot chocolate packets of different brands and flavors. K-cups have been increasing in popularity so we felt it was essential to have them on our site. We offer boxes of K-cups from many different brands. While they may not be something you’d be able to purchase directly from a café, having ground coffee or K-cups without having a coffee machine or Keurig would be of little point. Thus, we felt it was necessary to include them in our site’s inventory of products. We offer a few options that range from simple coffee machines to elaborate set ups. Finally, Other Beverages is just what it sounds like. Any beverage that one could find in a café that doesn’t fall into the aforementioned categories can be found in this one.

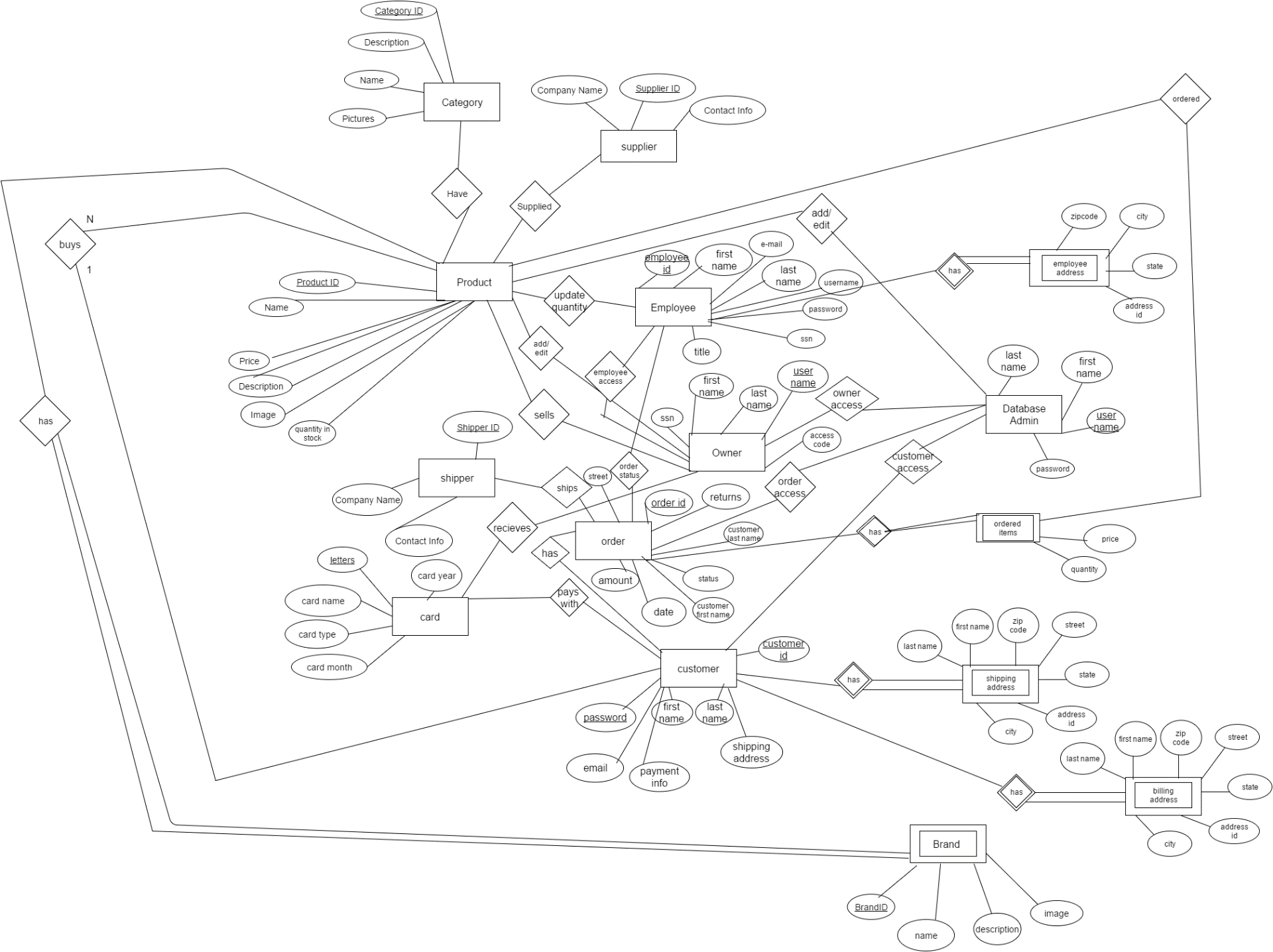
Our four user groups for our site are fairly simple and rather self-explanatory. They are customers, employees, database administrators, and owners. Customers will make up the majority of users visiting the site. They will be able to view the different brands and categories of our site and the associated products with them. The customers will be able to see brief descriptions and images of our brands and categories. For products they will be able to see much of the data associated with those products via our product profiles. These include the name of the product, description, price, an image of the product, and the amount in stock for that particular item. For this project’s purposes, our information for the products were taken primarily off of Amazon.com. While they are able to view much of this information, they will not be able to affect any of it, except by buying products, which affect the amount of products in stock. The customers also have the ability to view their own personal information that they have stored on the site, such as their name, associated email address, username, and any stored billing and shipping information that they use on the site. They can also update any of this information except for their username. We have also included a feature to reset a customer’s password if they forget it. The customers are also able to view their history of their past orders on the site. They are also the only user group that can actually view our cart and buy products.

Our employees will be able to view all the product information that our customers can see. Unlike the customer, however, we gave them the ability to update or edit the amount in stock of a certain item. While employees cannot see their order history (because they cannot buy anything from the site), they do have the ability to view all the orders ever made on the site and update the order status.

Our database administrators can do anything an employee can do. They also have the ability to add new products to the database as well as being able to alter existing product data.

Our final user group is the owners of the site. Because the site is being made for them, they have the most control over the data in the site. They have all the abilities of employees and database administrators and can also create new employee accounts. They can also view and update all employee information.

1. ER Diagram

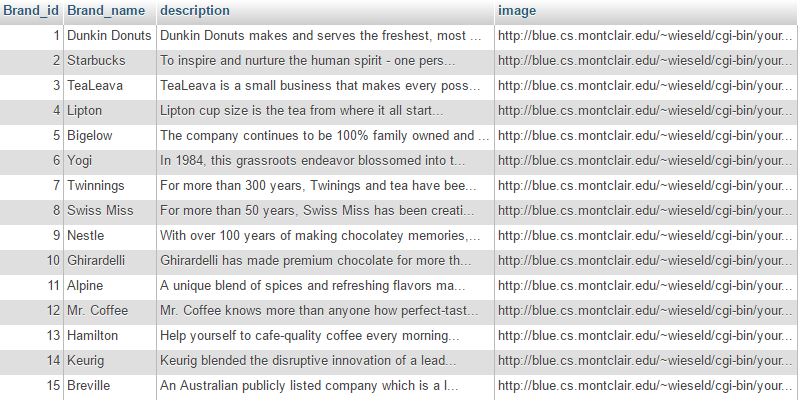


1. Relational Table

Customer Table



Brand Table



Card Table



b\_address Table



s\_address Table



Customer Table



DB\_Admin Table



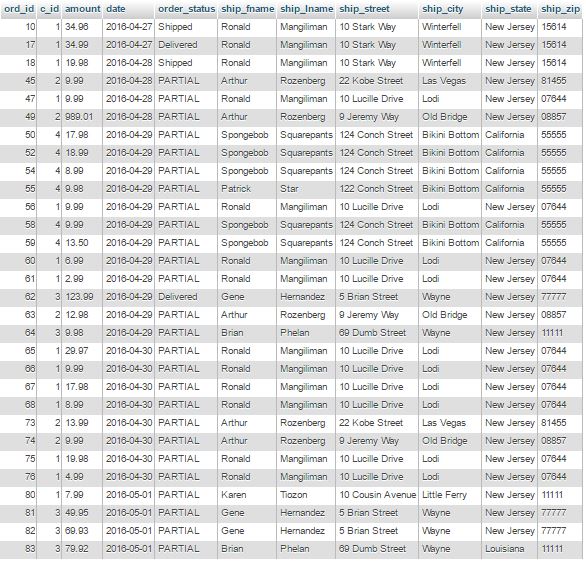
employee\_address Table



Ordered\_items Table



Orders Table

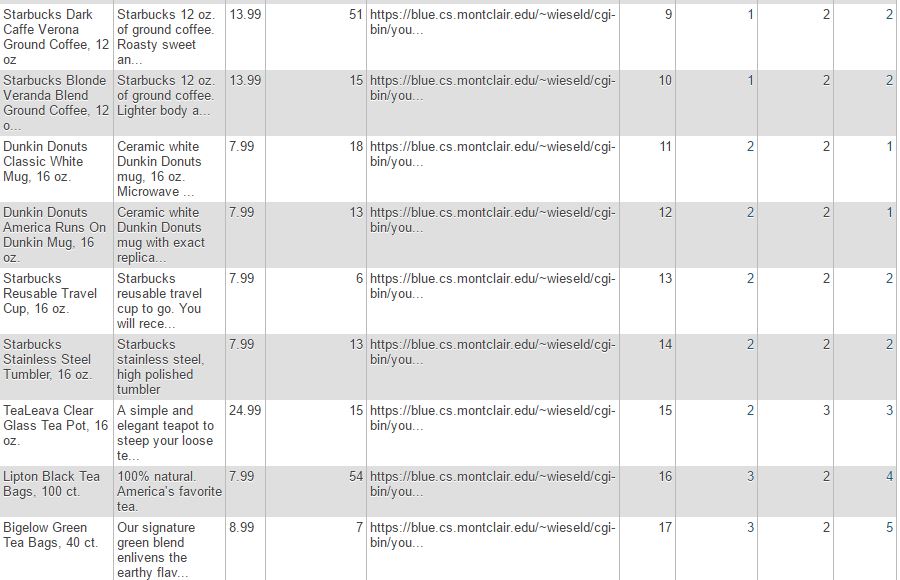


Owner Table



Product Table





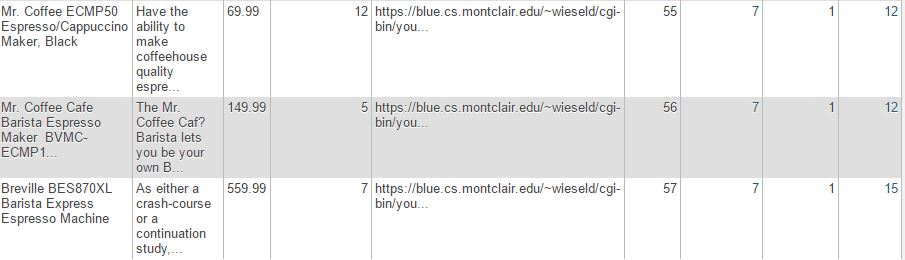




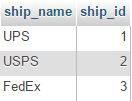




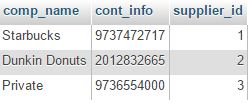




Shipper Table



Supplier Table



1. Database Design

Our database consists of 15 tables. The first table we created is Products. Of course it’s obvious why this table is needed. You can’t have an Ecommerce site without products. The product table has many attributes from name to price. Its primary key is Product ID to uniquely identify each product with a unique number. It has three foreign keys Category ID, Brand ID, and Shipper ID to give each product a unique number for shipper, category and brand to distinguish it from another product.

The next three tables Category, brand and shipper were made because we wanted to give these tables other attributes including name, description etc. It would be a lot easier to make them all separate tables instead of cluttering them into the products table. Each of these tables has a unique ID which is a primary key and is passed to the products table as a foreign key.

The next table orders, was made to keep track when the customer purchases a product. Orders have basic attributes like the amount of an order. Its primary key is a unique order id, so each order cannot have the same id as another order. It has a foreign key of customer id so each order can tracked by the customer that bought it.

The next table ordered items, is a weak entity because it has no primary keys and has 2 foreign keys. This table was made to keep track of the actual items that are ordered. To do this it needs a foreign key of product id, so it can actually get the product that is ordered. It also needs a foreign key of order id, so each product can have an order id.

The next table customer, is needed because we need to keep track of the person that is buying products of our website. Customer has basic attributes like Name, username, etc. Each customer has a unique primary key, customer id, so no customers have the same data.

The next two tables shipping & billing address are weak entities and are needed for the customer to know where to ship their products. We decided to make them separate tables so we can have separated attributes for city, street and state. They have a foreign key of customer id, so each address can be identified to a certain customer.

The next table card, is necessary for customer to pay for their products. Some attributes that exist are card Name, card type, etc. Its primary key is the letters of the card (which only takes alphabetic characters) to uniquely identify each card. Of course, its foreign key is customer id so each card belongs to a customer.

The next table employee, is necessary for our employees that work for our website. We need to keep a record of all of our employees. They have basic attributes like name. Its primary key is employee id, so no two employees have the same data.

The next table employee address is a weak entity, (similar to shipping and billing addresses) and is needed for the employees. We wanted the address to have various attributes like city, state etc. so making it a separate table was the way to go. It has a foreign key of employee id, so each address belongs to a certain employee.

The last two tables, Owner & DB Admin is needed because we need to give the owner and DB admins more priority than customers and employees so they can access more information. Both the owner and DB admin have their own unique primary usernames. They also have access to create employee accounts, so they have a foreign key of the employee id.

V. Site Overview

For our site, there are four views: one for each of our user groups. The site has our login page as our landing platform. Anyone accessing the site must do so by logging in through the login page. Anyone trying to access parts of the site without logging in will only see a page saying that they are not logged in. After five seconds, they will be redirected to the login page.

The general flow map of the site’s login is as follows:



Figure 1. The general flow chart of the site’s login, which shows the landing pad of the site and the different views.

Figure 1 shows the link between the login page and the different views that can be reached by logging in. Besides the views, the registration page and forgot password can also be reached from login. Those three pages are those only pages of the site that can be accessed without being logged in.



Figure 2. The customer view of the site.

Figure 2 shows the customer view for the site. The customer is the only user group that can access the shopping cart and buy products. Consequently, they are also the only ones able to view the order history page. They can also change their own personal information, including their password.



Figure 3. The Employee view of the site.

The employee view is similar in basic design as the customer view. The most glaring difference is the inability to access the shopping cart. Instead, the employee can update quantity for products and change an order’s status.



Figure 4. The database administrator view.

The database admin view of the site is similar to that of the employee. They have the added abilities of being able to add new products and edit existing product details.



Figure 5. The owner view

The owner view has the most access to the data. They are able to do anything a database admin can do, but are also able to view and edit existing employee information and create new employee accounts.

1. Site Implementation

For our implementation, we used to HTML5 and CSS3 for our front-end. We decided to forgo JavaScript for this project as the majority of our group does not know it. This raised a few problems throughout the development of the site, but we managed to use get it done. For back-end, we used PHP and MySQL for database queries.

List of Files

|  |  |  |
| --- | --- | --- |
| Filename | View | Brief Description |
| index.php | All | Login for site |
| joe\_add\_product.php | DB Admin, Owner | Form for adding product |
| joe\_add\_product\_results.php | DB Admin, Owner | Adds new products |
| joe\_all\_brands.php | All | Displays all brands |
| joe\_all\_category.php | All | Displays all categories |
| joe\_average\_fns.php | All | Container for all site functions |
| joe\_change\_info\_form.php | All | Form to manage personal information |
| joe\_change\_orderstatus.php | Employee, DB Admin, Owner | Allows an order’s status to be updated |
| joe\_change\_password.php | All | Allows a user to change their password |
| joe\_change\_password\_form.php | All | Form for changing password |
| joe\_checkout.php | Customer | Displays cart and checkout form |
| joe\_create\_employee.php | Owner | Allows owner to create new employee accounts |
| joe\_create\_employee\_id.php | Owner | Allows owner to create new employee ID |
| joe\_create\_or\_update\_employee\_form.php | Owner | Form for creating or updating employee accounts |
| joe\_data\_valid\_fns.php | All | Container for data validity functions |
| joe\_db\_fns.php | All | Container for functions that interact with database |
| joe\_edit\_product\_details.php | DB Admin, Owner | Form for editing product data |
| joe\_edit\_product\_details\_results.php | DB Admin, Owner | Allows editing of product data |
| joe\_employee\_form.php | Owner | Form for editing employee data |
| joe\_footer.php | All | Website footer |
| joe\_forgot\_password.php | All | Allows user to reset password |
| joe\_forgot\_password\_form.php | All | Form for resetting password |
| joe\_front\_page.php | All | Front page of site |
| joe\_information\_results.php | All | Allows changes to user personal info |
| joe\_login\_footer.php | All | Footer for pages without a menu |
| joe\_login\_header.php | All | Header for pages without menu |
| joe\_logout.php | All | Logs user out |
| joe\_member.php | All | Logs user in |
| joe\_menus.php | All | Menus for the site |
| joe\_new\_employee.php | Owner | Creates new employee |
| joe\_not\_proper\_length.php | All | Form for when user enters a password of improper length |
| joe\_not\_same\_pwd.php | All | Form for when user attempts to reset password and new passwords do not match |
| joe\_order\_history.php | Customer | Displays customer order history |
| joe\_output\_fns.php | All | Container for output functions |
| joe\_process.php | Customer | Prints out order recipt and inserts card info into database |
| joe\_product\_fns.php | All | Container for fucntions that relate to products |
| joe\_product\_profile.php | All | Displays product details |
| joe\_profilepage.php | All | Profile page for users |
| joe\_purchase.php | Customer | Inserts shipping/billing info into database. Prompts for payment info. |
| joe\_quant\_update\_results.php | Employee, DB Admin, Owner | Allows permitted users to update stock of item |
| joe\_quantity\_update.php | Employee, DB Admin, Owner | Form for updating stock |
| joe\_register\_new.php | Customer | Registers new customers |
| joe\_registration\_form.php | Customer | Form for registration |
| joe\_search\_results.php | All | Basic search of products |
| joe\_show\_brand.php | All | Shows all products of specific brand |
| joe\_show\_cart.php | Customer | Displays cart |
| joe\_show\_cat.php | All | Displays all products of certain category |
| joe\_show\_product\_all.php | All | Displays all products |
| joe\_update\_employee.php | Owner | Updates employee |
| joe\_update\_employee\_form.php | Owner | Form for updating employee data |
| joe\_update\_orders.php | Employee, DB Admin, Owner | Changes order status |
| joe\_user\_auth\_fns.php | All | Container of functions for authenticating users |
| joe\_view\_employees.php | Owner | Displays all employees and their info |
| joe\_wrong\_login.php | All | Form displayed when user enters invalid login info |
| joestyle.css | All | CSS stylesheet for site |

There were many files to this site. Looking back, the number of files could have definitely been lessened, as a few features of the site could have been implemented better. Due to time constraints however, this is what we have.

Logging In

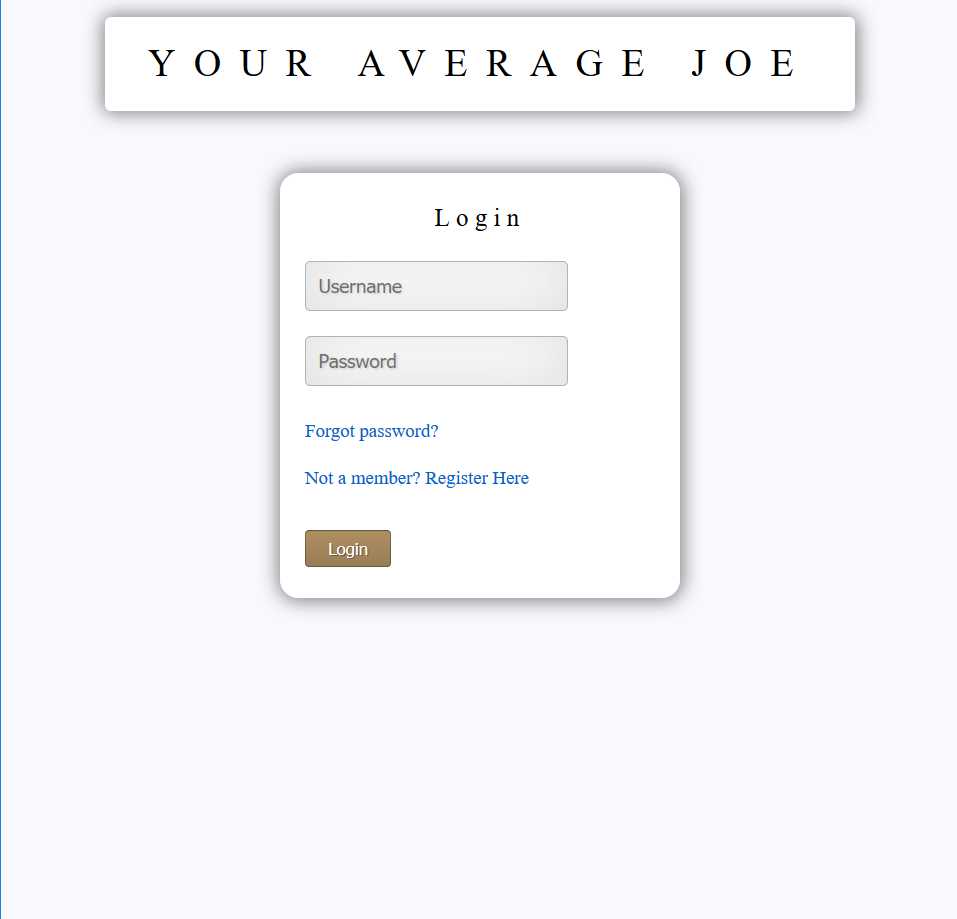


Figure 6. The site login

The first page of the site that every user must go through is the site login. No user can access the site without going through login. If they somehow try to access a page on the site without logging in, then they will receive a message saying that they are not logged in and be redirected to the login page after five seconds. The message is a simple “You are not logged in to view this page. You will be redirected to login in 5 seconds.”

From the site login, users can click on “Forgot password?” to reset their passwords (if they are an existing user) or register as a new customer on the site.

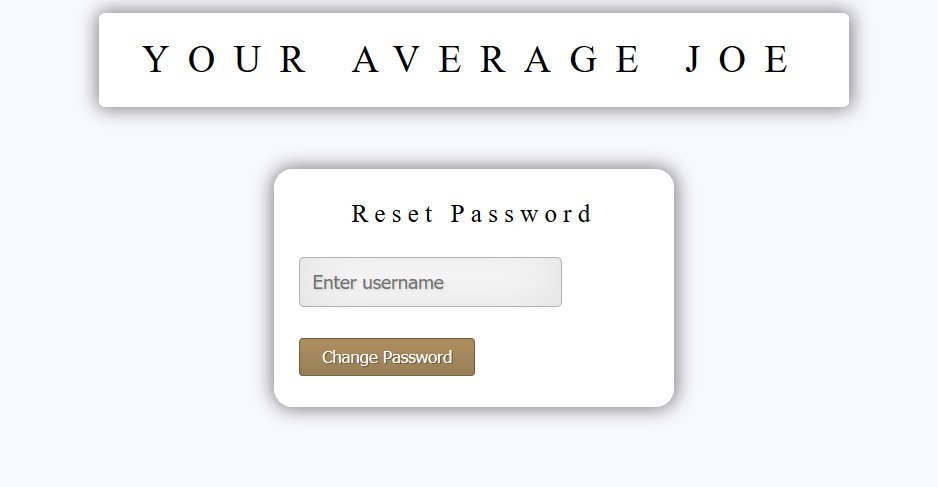


Figure 7. The rest password page

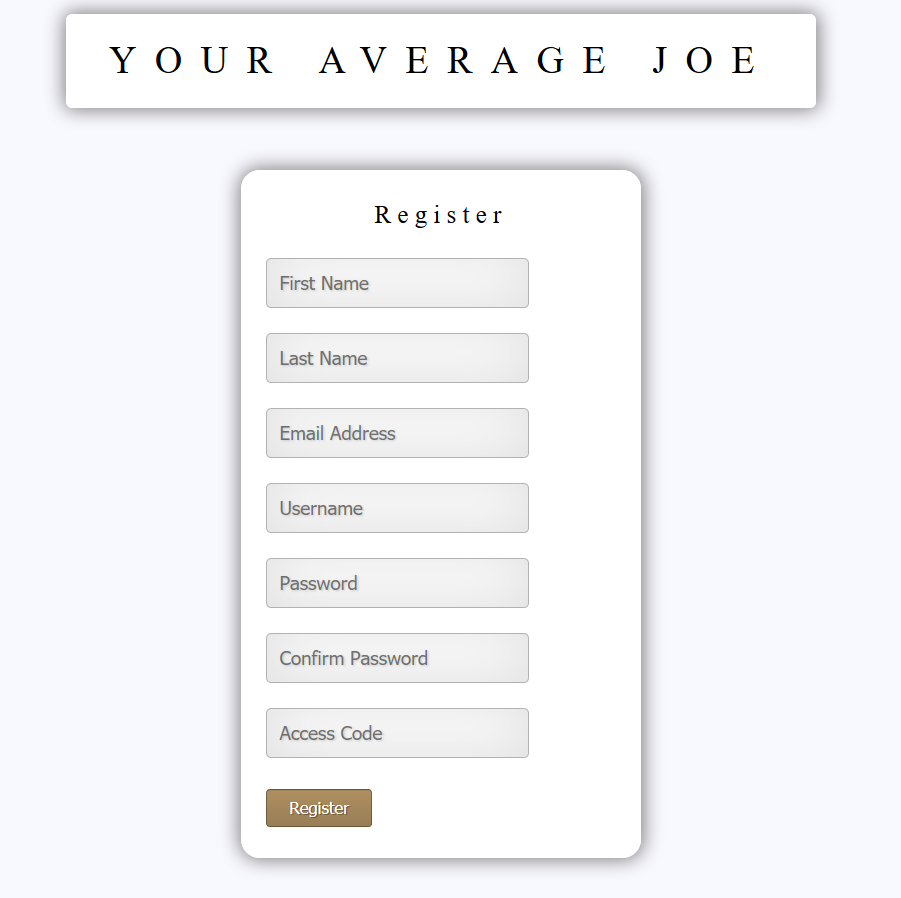


Figure 8. The registration form.

The reset password page in Figure 7 is quite simple. It has a single text field where users can enter their username to have their password reset to a randomly generated string. An email is then sent to the user containing their new password.



Figure 9. index.php

The code for the login is very simple. It is an HTML form with two text fields for username and password and a button. Something of note is the pattern attribute used for the inputs. This was our way of validating input by only accepting a certain type of expression. As an example, for username, the input only accepts usernames consisting of letters and numbers and must be 6-25 characters in length.

When a user fills in the form and the correct type of input is put in, then pressing the button will execute the joe\_member.php script.



Figure 10. joe\_member.php

The joe\_member.php is the script that logs users into the site. It takes the input from the user and calls a function login() to log the user in. If login() returns true, then a session variable is created and is set to the user’s username. They are then presented to the front page of the site. If not, then the user will be redirected back to another login page, which is the same as the regular login page, except that it has a message telling the user that they entered invalid user information.



Figure 11. The login() function from joe\_user\_auth\_fns.php

While the login() function seems long, it is a rather simple function. It takes in two parameters, the username and password inputted by the user and tries to select to find a customer with those login credentials. If it does not find one, then it will execute another query to trying to match the credentials with row in the Owner table. If not, then the Employee table, and if not that, then the DB\_Admin table. If the credentials inputted by the user cannot be found in the database, then it will throw an exception.

Something of note in this function is the sha1() function that encases the user’s password. This function calculates the sha1 hash of a string and uses that to compare the passwords stored in the database. This is because all passwords stored in our database are passed through this function. This is used for security measures.

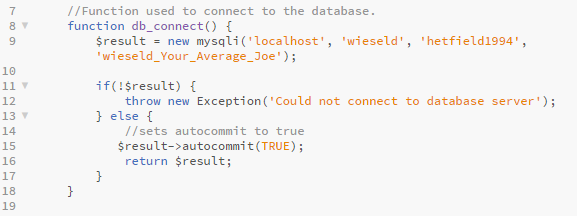


Figure 12. db\_connect() function from joe\_db\_fns.php

The first line of the login() function calls a function called db\_connect(). This a simple function that is used to connect to the database. It will be used many times throughout the site.

Registering New Customers

The register form is very similar to the form in index.php. Therefore, we will not include it here, but it can be found in the appendix at the back of the documentation. The register form has seven text fields for first name, last name, email address, desired username, password, confirm password, and access code. All of these are very self-explanatory. The only one of note is access code. This was a decision by the group to include an access code as a requirement to register for the site. This is solely for the purpose of keeping out regular individuals from accessing, and to avoid any possible copyright issues with the images that we used.

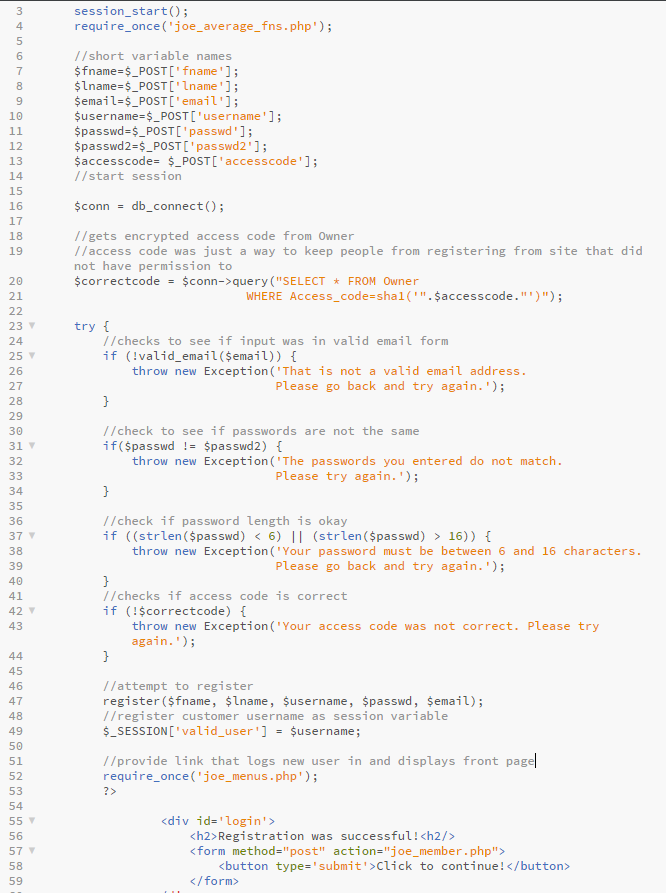


Figure 13. Part of joe\_register\_new.php

This script is primarily checks to validate the user’s input. If they pass through all of them, then the register() function is called and passes the user’s input as its arguments. If it is successful, then the valid\_user session variable is created and set to the username chosen by the user. They are then presented with a simple screen with a message telling them registration was successful and a button that logs them into the site. Else, exceptions are thrown.

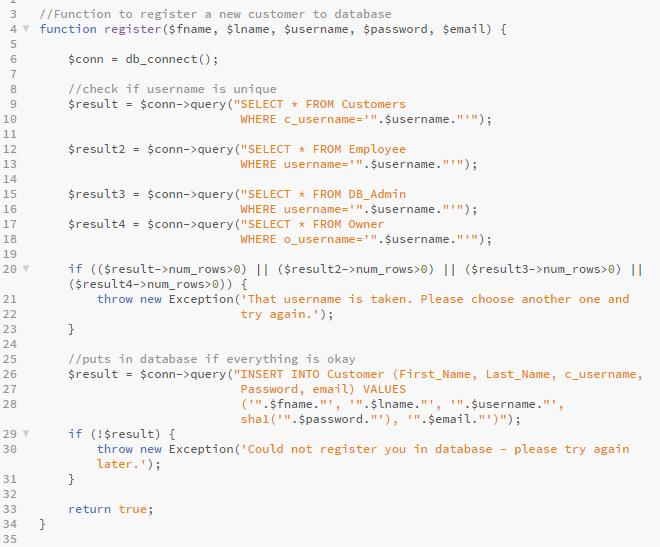


Figure 14. The register() function from joe\_user\_auth\_fns.php

The register() function is a simple function that checks to see if the username inputted by the user is unique. If so, then the user’s info is inserted into the Customer table in the database. It achieves this by executing four separate queries to select any row that has a matching username. If all of them return zero rows, then the insertion is executed.

Resetting User Passwords

The site allows users to reset their passwords by entering their username. If their username is valid, then they will receive an email with their new randomly generated password.



Figure 15. joe\_forgot\_password.php

This script creates a new password using the reset\_password() function from user\_auth\_fns.php. If it succeeds, then the notify\_password() function sends the email to the user with their new password.

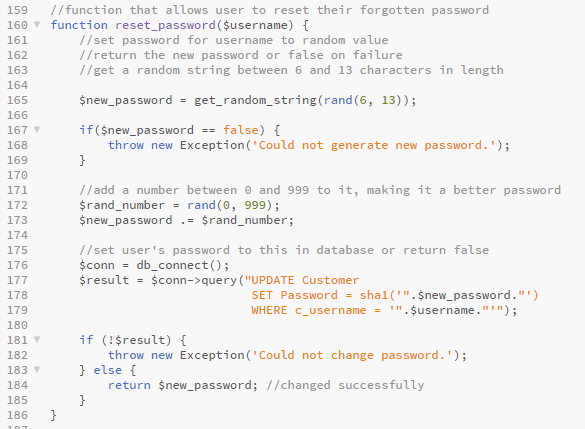


Figure 16. reset\_password() function from joe\_user\_auth\_fns.php

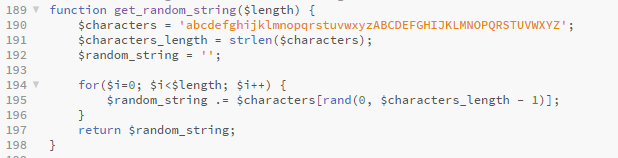


Figure 17. get\_random\_string() function from joe\_user\_auth\_fns.php

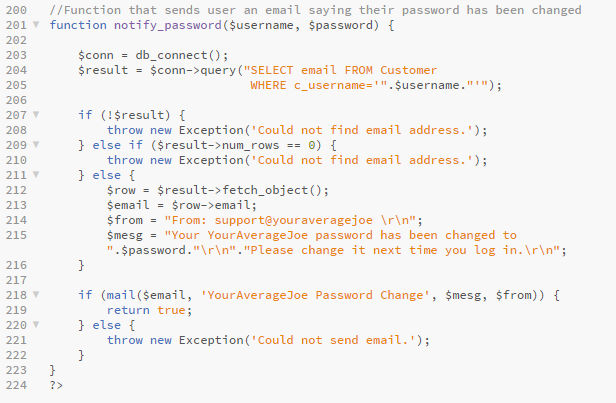


Figure 18. notify\_password() function from joe\_user\_auth\_fns.php

The reset\_password() function uses get\_random\_string() to get a random string consisting of letters of length six to thirteen. It then uses the rand() function to concatenate a number from 0 to 999 to the end of that string. This will end up being the user’s new password. The new password is then passed as an argument of the notify\_password() function, which uses PHP’s mail() function to send an email to the user (which it obtains by executing a query to select it from the database by matching the username inputted by the user).

Customer View

After logging in, the customer is sent to the front page of the website, shown below in Figure 19. The Customer’s view of the front page is slightly different from the rest of the user groups’ views. For customer, they have a tab that lets them view their shopping cart (if there are items inside of it). The other two tabs are drop down menus. One is a product tab that displays the categories and brands of the products. Hovering over these will display the brands and categories that the site currently has (This was achieved using nested loops and CSS. The CSS stylesheet will not be displayed here, but will be included at the back of the documentation). The other tab is the profile tab. For the customer, this allows them to view their order history, manage their personal info, change their password, and logout. There is also a search bar at the top of the site that allows the user to browse for products using a word or phrase they enter.

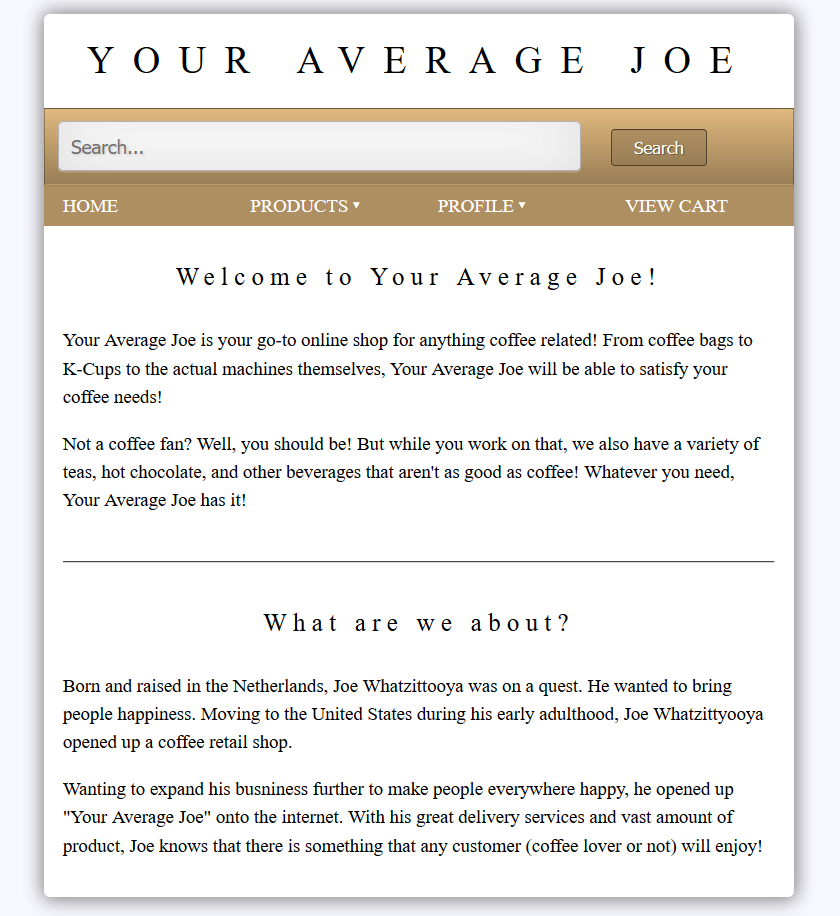


Figure 19. The Front Page of Your Average Joe (Customer View)

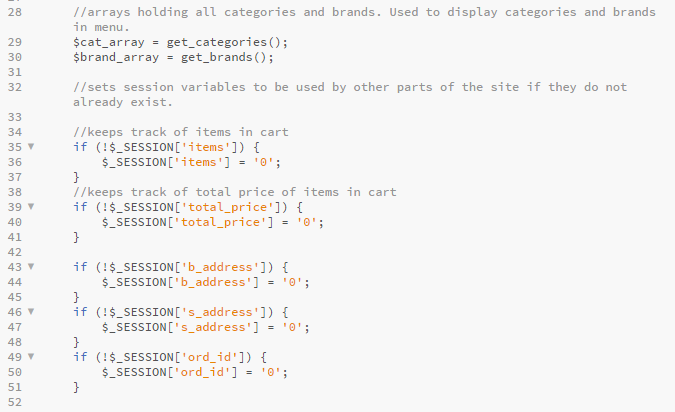


Figure 20. Part of joe\_menus.php showing the calls to the get\_categories() and

get\_brands() functions to fill their respective arrays. Also shows the

creation of certain session varibles used for other features of the site.

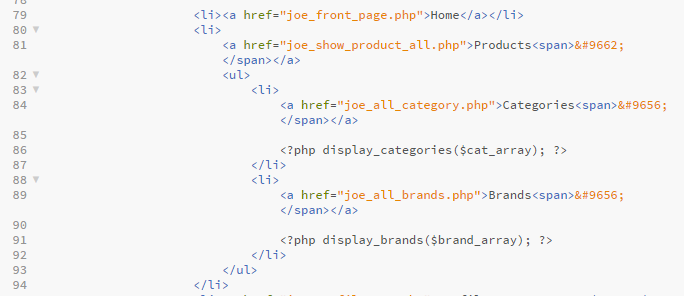


Figure 21. Part of joe\_menus.php showing the display\_categories() and

display\_brands() functions.

Much of the code for the menu is HTML. There are a few thing of note here, however. One is are the session variables being created. These will be used later on when the customer makes an order. The second are two array of categories and brands near the top. They use two functions to get the categories and brands from the database and then the display functions are used to print out HTML to create lists. Using CSS, we achieve the dropdown menu effect.

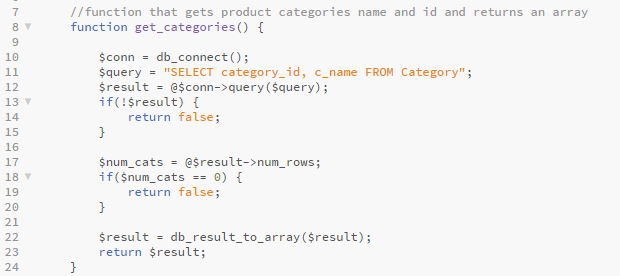


Figure 22. get\_categories() function in joe\_product\_fns.php

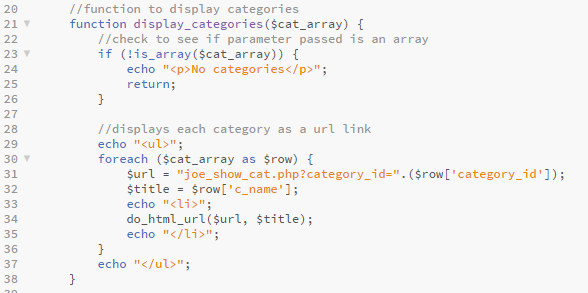


Figure 23. display\_categories() function in joe\_output\_fns.php

The above two functions aren’t too difficult to grasp. One gets categories and their information from the database and returns it as an array, while the other displays that information. The functions get\_brands() and display\_brands() are nearly identical. The only difference is that they get and display brands instead of categories.

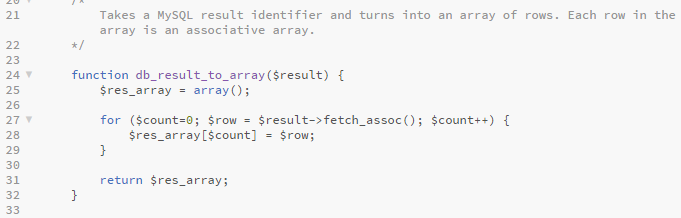


Figure 24. Function db\_result\_to\_array() in joe\_db\_fns.php

The db\_result\_to\_array() function does exactly what the function name implies. It takes a MySQL result identifier and turns it into an array of rows, with each row being an associative array. This function is used throughout the site.

Changing User Passwords

Clicking on the change password option in profile for any type of user will bring up the change password form in Figure 20. This prompts the user to enter in their old password and enter in their new password twice. If their old password is correct and their new password is valid and the two inputs are matching, then the user’s password will change.

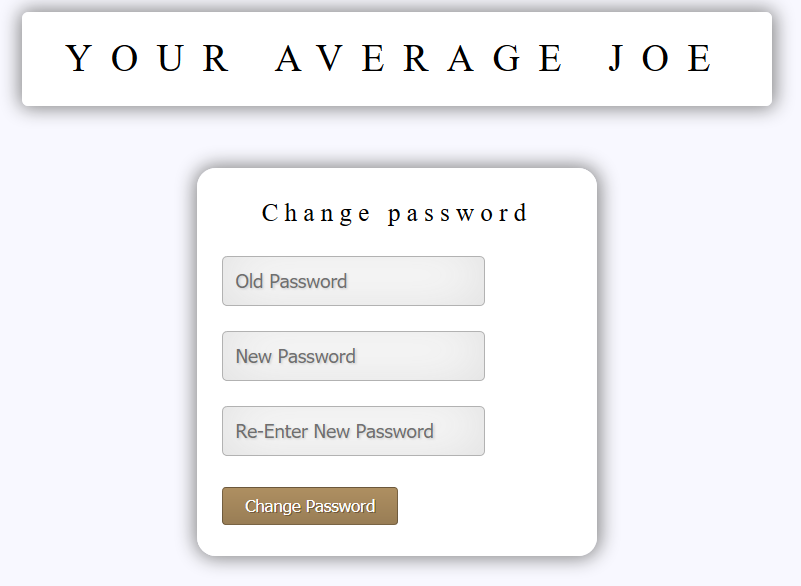


Figure 25. The change password screen.



Figure 26. joe\_change\_password.php

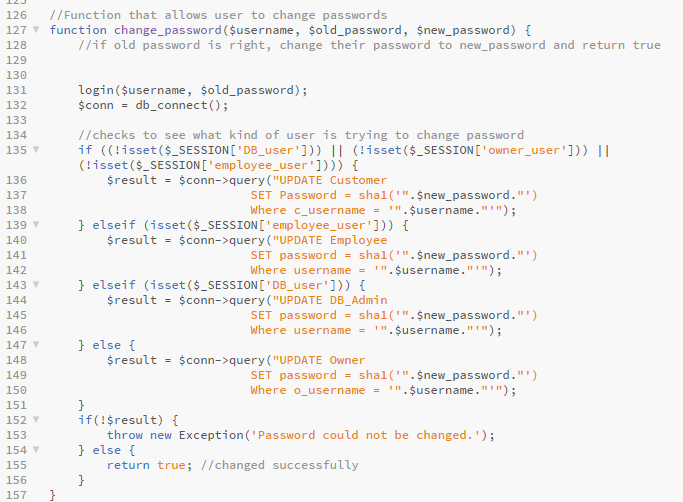


Figure 27. change\_password() function in joe\_user\_auth\_fns.php

There is nothing too interesting happening in joe\_change\_password.php. After some checks, it calls the function change\_password(). This function has checks to determine what kind of user is trying to change their password. It does this by seeing checking to see if the session variables associated with the different types of user are set using the isset() PHP function. It then executes the correct query, updating the table that corresponds with the type of user. If it does not get anything from the query at the end, then it throws an exception.

Managing Personal Information

Managing personal information is one of the things every type of user can do on the site. However, the customer view of the change information page is the most extensive. While all the user can edit their name, employee and customers can edit their email addresses. Customers, however, can also store billing and shipping addresses on their account that they can edit in the change information page.

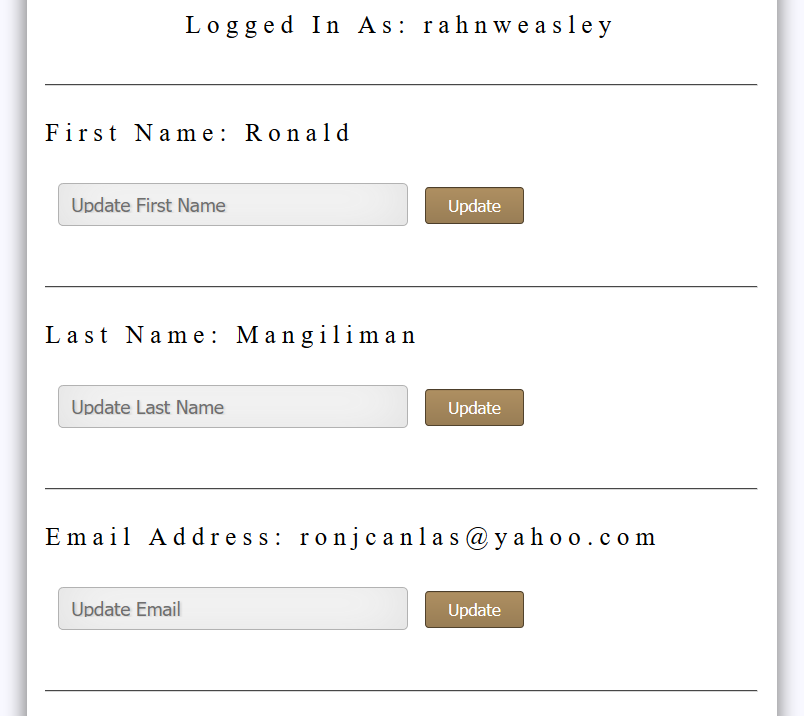


Figure 28. Customer view of the change information page

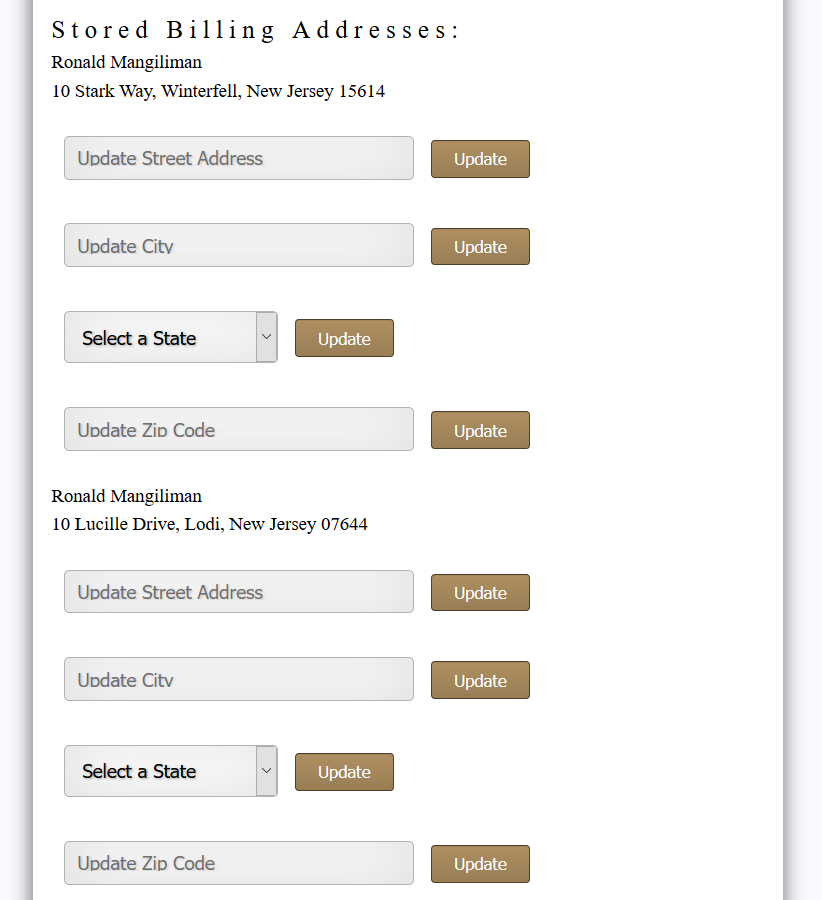


Figure 29. Another view of the change information page. This time, showing the ability to update stored billing addresses.

The query to obtain the first and last name and email address of the customer are very simple and straightforward, so they will not be shown here. Getting the stored billing and shipping addresses and allowing changes to them was a bit more complicated. First, there is a check to see what kind of user is logged into the page and it displays the respective change information page. It then executes the proper query (depending on the user type). For customers, a two other queries are executed, one to obtain all the billing addresses associated with customer’s customer id (stored as c\_id in the database) and the other does the same thing, but with shipping address. We choose to have two separate tables for billing and shipping address, as they can be different. After we store the query result and convert it into an array using db\_result\_to\_array(), we use a foreach loop to display the information for each address.



Figure 30. Part of joe\_change\_info\_form.php that showcases how the addresses are

displayed.

You can see from Figure 30 that each form will take in that address’s id. Using $\_GET in joe\_information\_results.php, we are able to change the correct address using that id, as each shipping and billing address has a unique ID associated with it.



Figure 31. Part of joe\_information\_results.php, showing how if a user submitted a

to their personal information, it would execute a query to change it.

The above example shows how this works with addresses.

Displaying Categories, Brands, and Products

When a user clicks on the products tab, it will display every single product in the database with its name, price, picture, and a description of the product when hovering over its name. While this is nice, its it much easier if you could view products based on their category and brand, or even through a search. When pressing on the categories or brands tab, it will display each category and brand in the database with a picture, description, and a link that will display all products of that category/brand. This is easily done by using query to get the categories or brands from the database, and then looping through them to display.

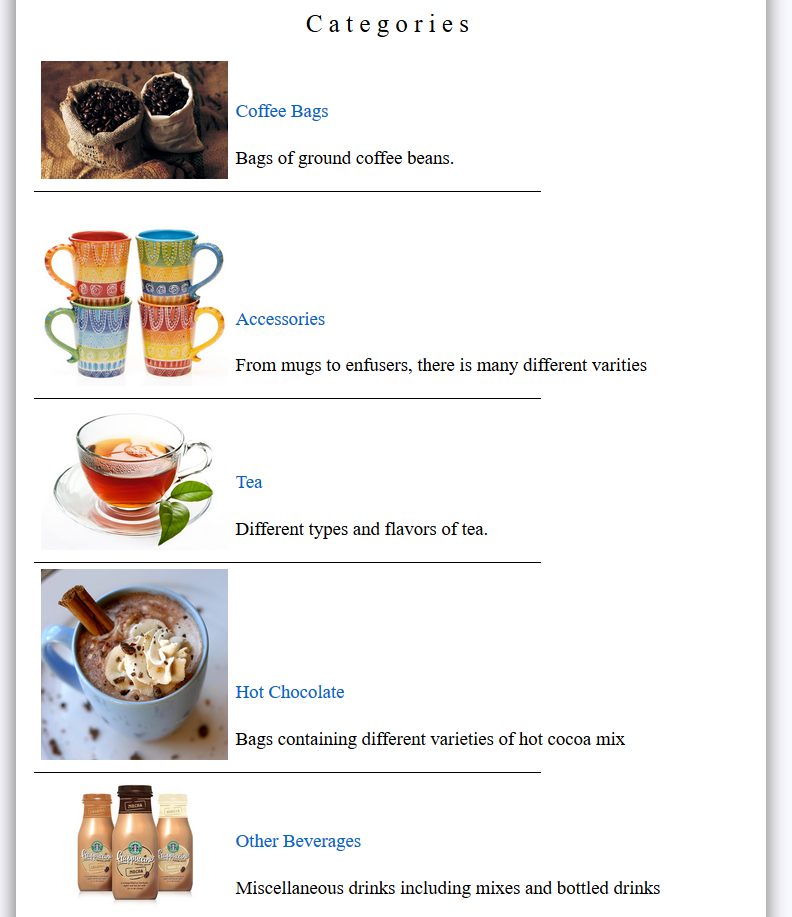


Figure 32. Result of clicking on the categories tab.

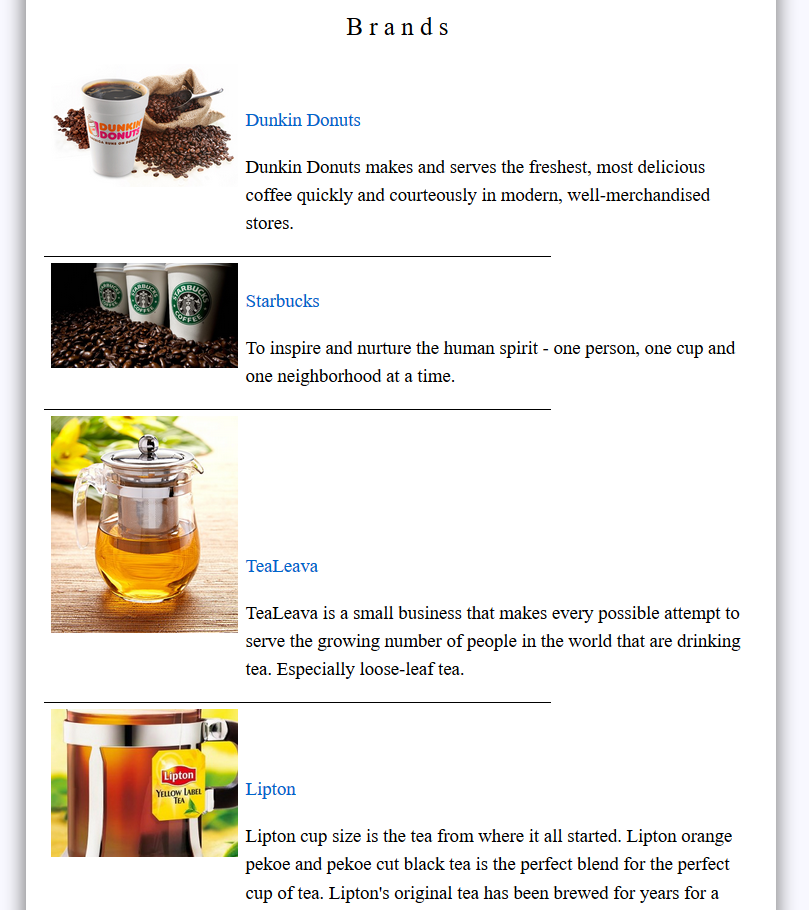


Figure 33. Result of clicking on the Brands tab.

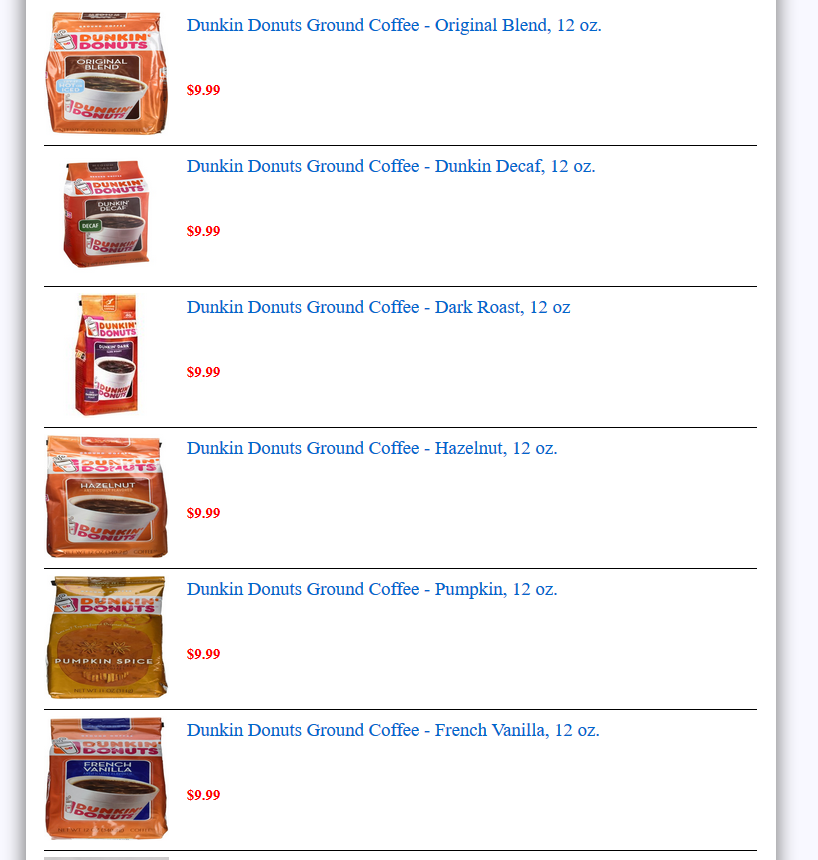


Figure 34. Result of clicking on Dunkin Donuts in brands.



Figure 32. Part of the code for joe\_all\_category.php

Figure 32 shows the code for displaying all the categories from the database. The code for joe\_all\_brands.php and joe\_show\_product\_all.php is very similar. We create links to joe\_show\_cat.php with a variable similar to how we displayed and updated billing and shipping addresses in the Customer view of the change information page.



Figure 33. Part of the code for joe\_show\_cat.php

The script joe\_show\_cat takes the variable passed using $\_GET, gets the category name using the function get\_category\_name() and then uses the get\_prods\_cat() function and display\_prods() function to get the information of products and then display them. These functions are very similar to the functions get\_categories() and display\_categories() discussed earlier.

Search Engine

The search engine for the site is very basic. It takes a key word or phrase from user input and then goes through all the names and descriptions of the products in the database, looks for a pattern where that word or phrase exists and returns those products. We achieve this using the percent sign wildcard and LIKE keywords. It then displays them using a loop. This process should seem very familiar by now.

.

Figure 34. Part of joe\_search\_results.php

Product Profile and Adding to Cart

When clicking the name of any product, it will transfer you to that product’s profile page. This page is a more detailed look at the product’s details. It displays the name, product description, image of the product, amount of that item left in stock, and also gives customers the ability to add that product to the cart (assuming it is in stock, otherwise the add to cart button is replaced by a disabled OUT OF STOCK button).

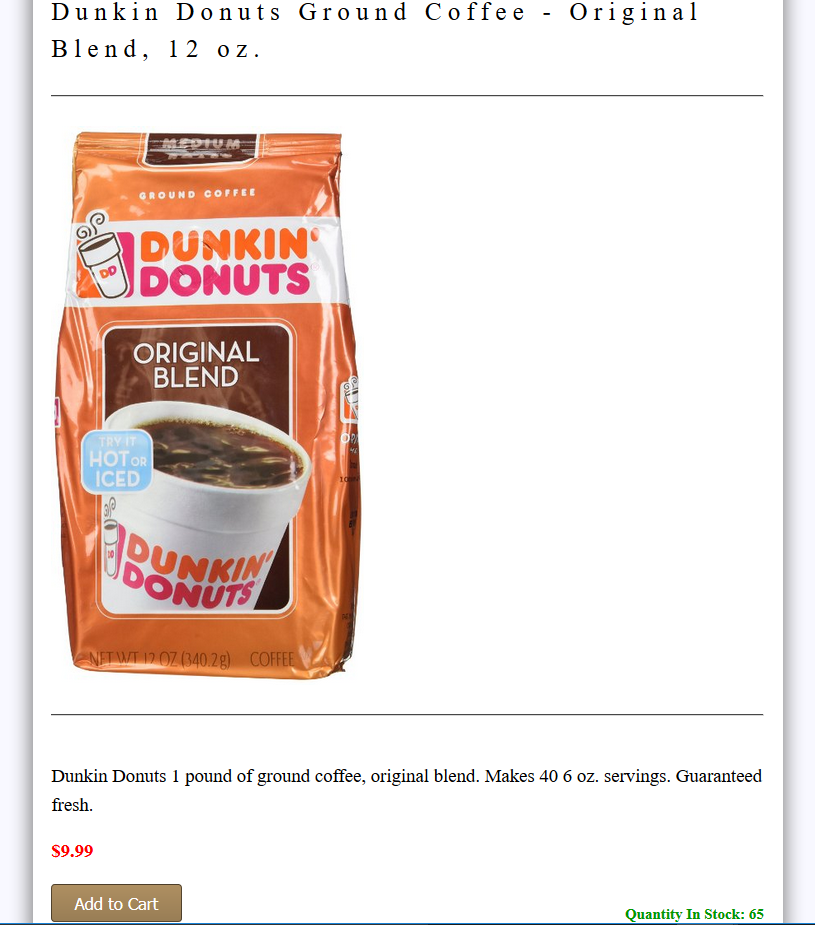


Figure 35. Example of a product profile.

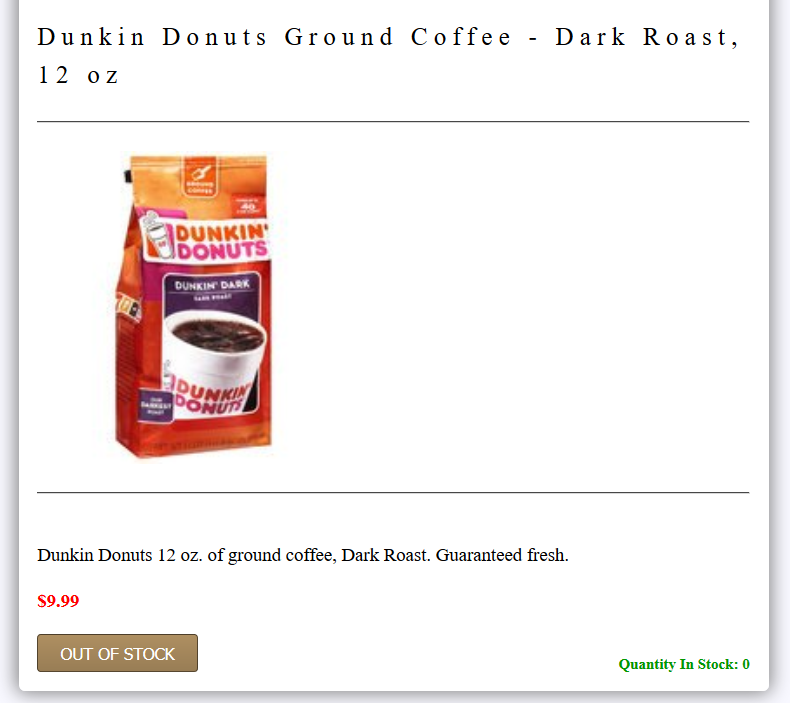


Figure 36. Product out of stock



Figure 37. Part of joe\_product\_profile.php

Once again, we see a similar pattern on how to get information and display it. The get\_prods() function gets the product information and display\_product\_details() displays it.

The get\_prods() function is not very interesting at this point, but there are a few interesting happening with display\_product\_details(). You can see from Figure 38 that there are checks in place to hide certain elements of the page from the customer. Another thing of note is how the Add to Cart button appears only to customers and is replaced by the product’s ID number or the OUT OF STOCK button for every other type of user.



Figure 38. The display\_product\_details() function from joe\_output\_fns.php

Clicking on the add to cart button will take the user to their shopping cart. Here, they can change the quantity of each individual product in their cart. This is limited to how much is left in stock. A customer may not purchase more than the amount in stock. Attempting to update the cart with a quantity larger than the amount in stock will reveal a little bubble that tells the user that they cannot do that. This is achieved by using the HTML attribute max for input type number. We set this attribute equal to the amount in stock. The total price for each type of product also increases when changing the quantity and updating the cart. The total price and total amount of items can also be seen by hovering over the View Cart tab in the upper right of the site. That was achieved using session variables that were declared earlier. This can be seen all the way back in the code for joe\_menus.php. There are also button that allow the customer to continue shopping or checkout.





Figure 39. The display\_cart() function from joe\_output\_fns.php

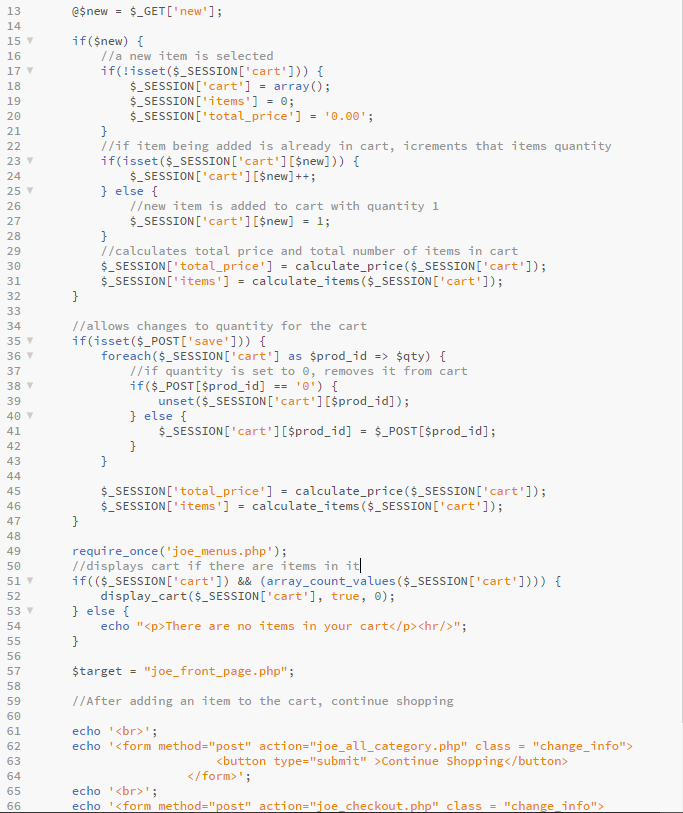


Figure 40. Part of joe\_show\_cart.php

The function display\_cart() may look long and complicated, but a closer look shows that it is very similar in approach to the other display functions shown in this documentation. Things of note include the ability to disable the changing of quantities. This option is useful, as we want to show the shopping cart and its contents during checkout, but do not want the customer to be able to change quantity at that point. There is also an option for images that was not used throughout the site at the time of this writing. The images displayed were small and looked odd, and the group felt that it was better if they were not present. Otherwise, the rest of the function is your basic output function. It uses the function get\_product\_details() to get the information of each product and display it in an HTML table. The foreach loop that displays the cart is slightly different from the other loops we’ve encountered. Here, the current element is assigned to a value, which is the quantity of that item in the cart. This is used to calculate the total price of each different item in the cart and the total number of items in the cart.

The joe\_show\_cart.php script goes in Figure 40 goes into more detail on how the actual cart works. The cart is actually a session variable. Every time the add to cart button is pressed in the product profile page, that product is either added to the cart or the quantity is incremented by one. If it is set to 0, that product is unset from the cart. If there are no items in the cart, then the cart table will not be displayed. The calculate\_price() and calculate\_items() functions does as their names imply.

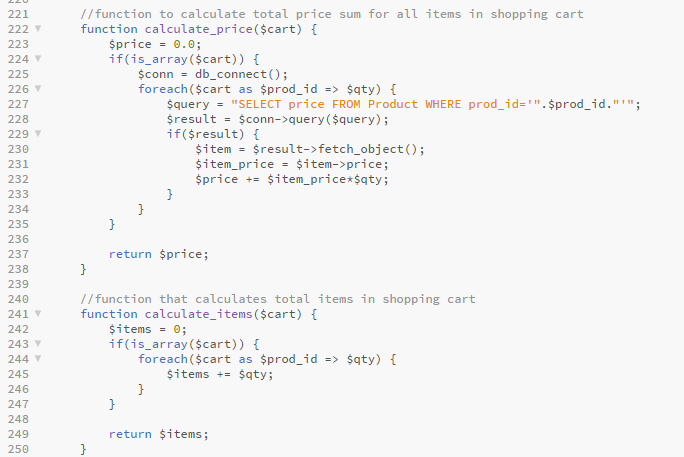


Figure 41. The functions calculate\_price() and calculate\_items() in joe\_product\_fns.php

Checking Out

From the cart, a customer can click on the checkout button to begin the checkout process. Doing so will take them to the checkout form. Here they can see the contents of their cart (now with the ability to change quantity disabled) and are prompted to choose one of their stored billing and shipping addresses to use for the order or enter a new one.

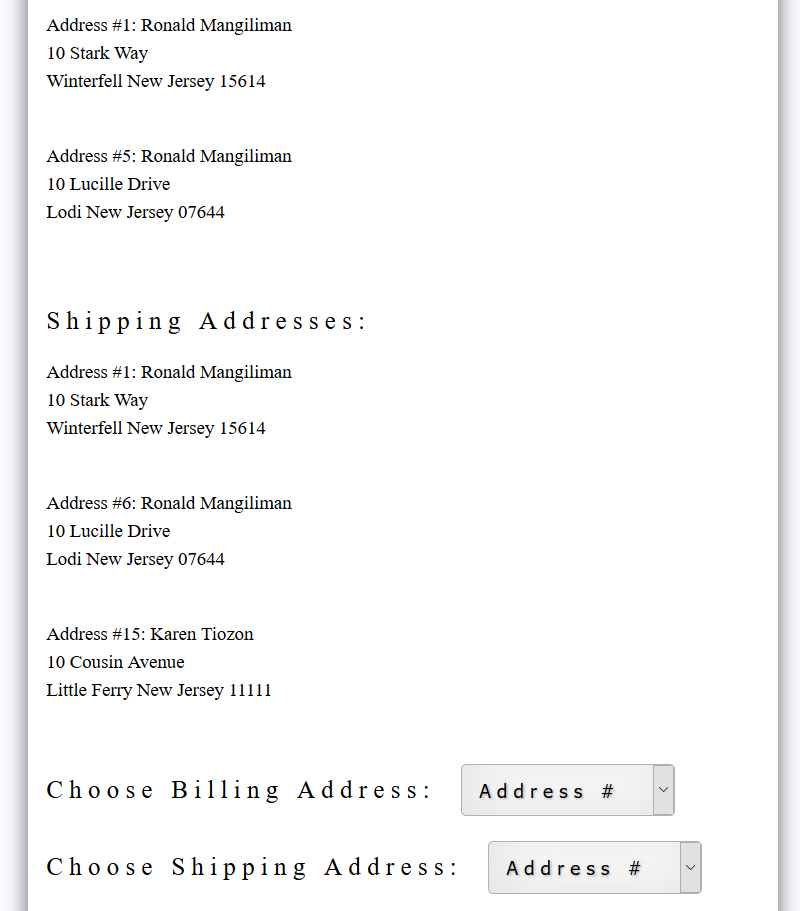


Figure 42. Choosing existing addresses.

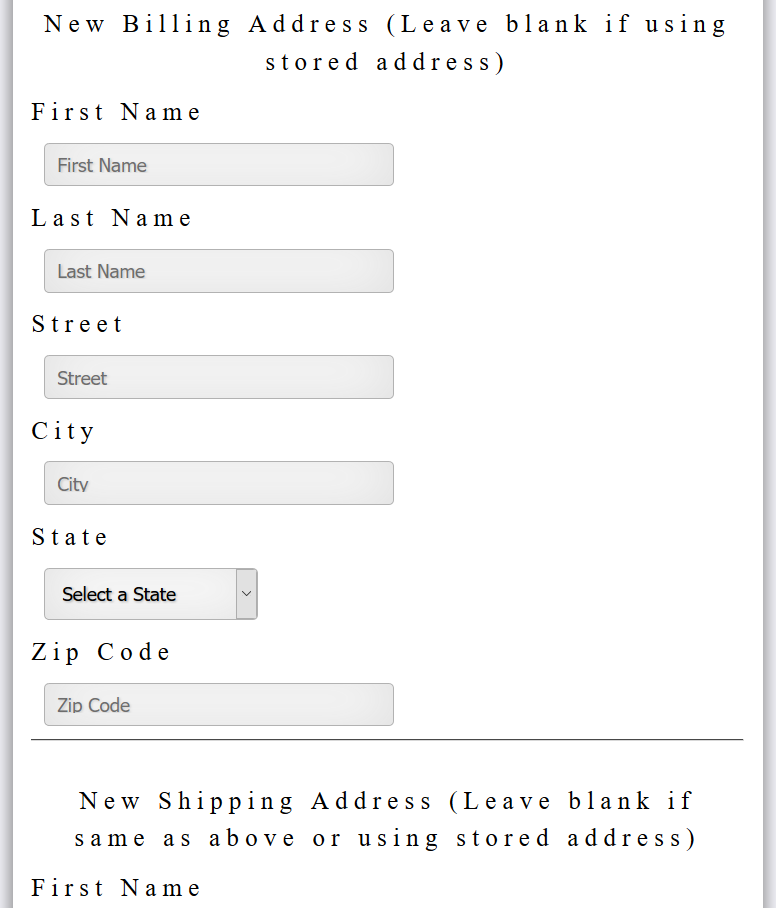


Figure 43. Entering new addresses

Something of note is that choosing an existing billing address does not mean you have to use an existing shipping address or vice versa. It can be any combination of new or existing billing or shipping addresses.

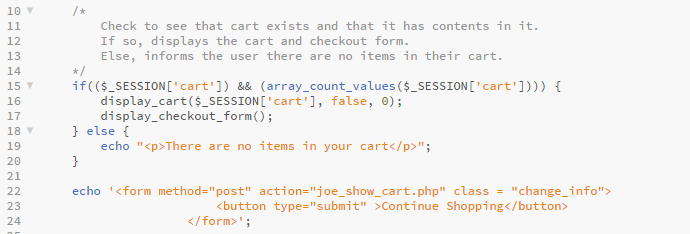
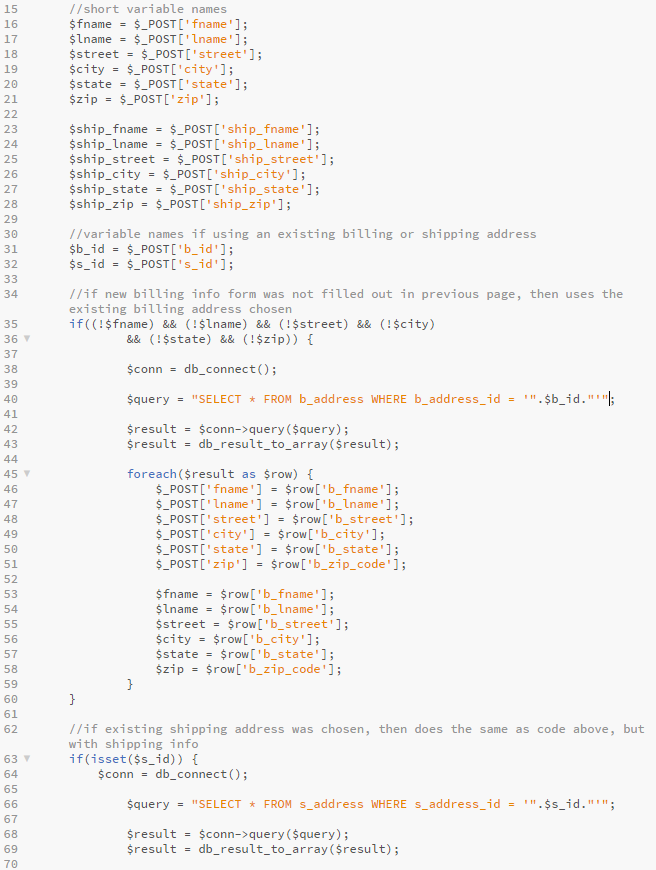


Figure 44. Part of joe\_checkout.php

The script for the first step of checking out is simple. It checks to see if the cart has contents in it, and if it does, it displays the cart and displays the checkout form. The function display\_checkout\_form() simply displays the stored addresses of the customer and also displays the form for entering new ones. It is a long function, but similar in approach to how we displayed addresses in the profile page of the customer.

After entering billing and shipping details, the site gets the user’s address information and depending on whether or not it was a new address, saves it to the database with that customer’s customer ID. It then inserts the order into the database and the decrements the quantity in stock for every item the customer bought. This is all done through the insert\_order() function. If the insert\_order() function is able to do all of this, then the site prompts the user for their credit card details. For this site, an actual payment system has not been implemented, so we used a bit of a dummy system. The user can choose from four different made-up credit card companies and enter their 16-Lettter Card Sequence (as opposed to numbers), the expiration date, and the name on the card.



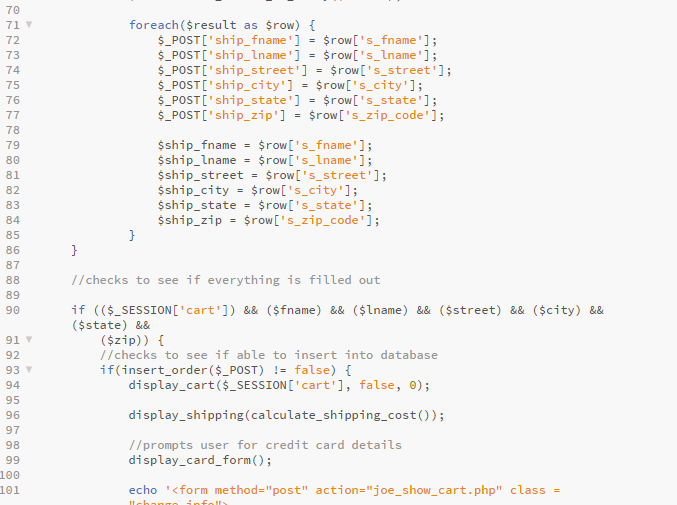
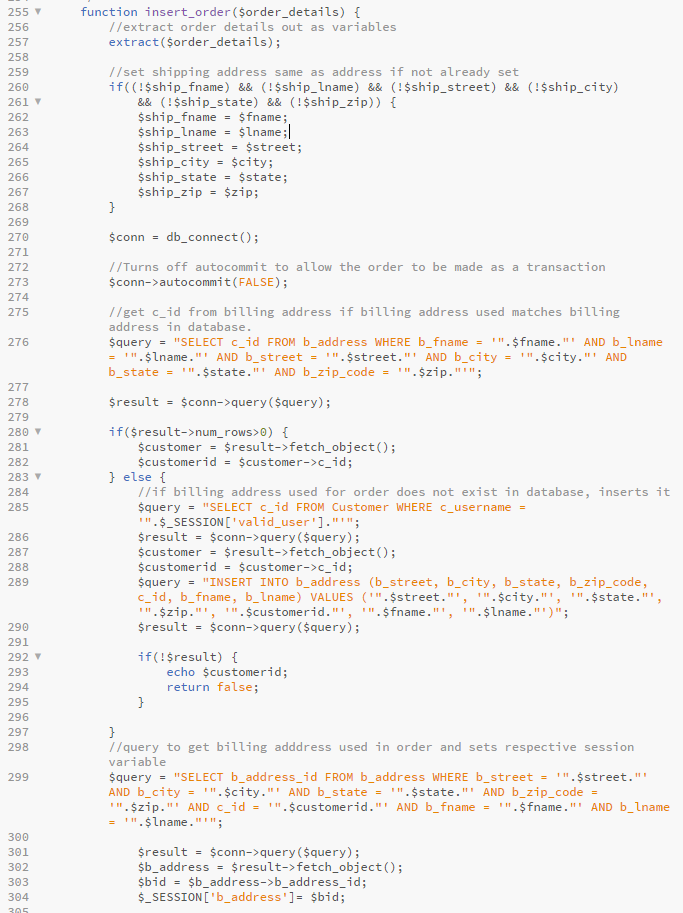


Figure 45. joe\_purchase.php



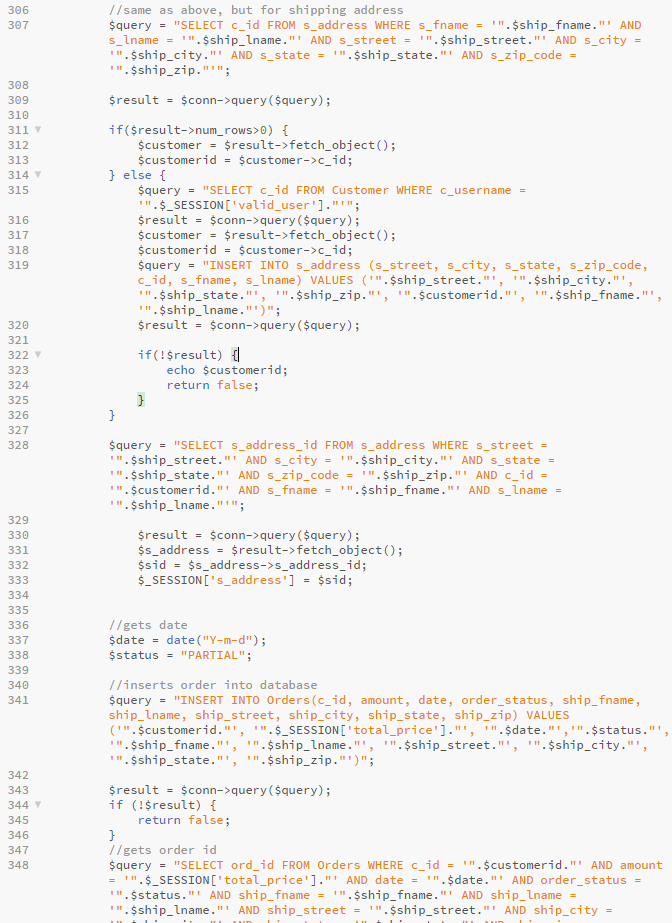




Figure 46. The insert\_order() function from joe\_product\_fns.php

The insert order function is very long, as it does quite a few things and has to make a quite a few checks as well. First, we want to make a transaction, so we turn autocommit off. Then we make checks to see if the addresses that the user entered are already in the database. If not, they are stored. It then takes the details of the order and saves those into the database. The ID’s of the addresses used are stored as session variables. These will be used later on to print out the receipt. After successfully inserting the order information in the database, it enters every item that was in the cart at the time of the order, its quantity and the total price of that item type. This is then inserted into the Ordered\_items table. The last thing the function does is update the stock of every item bought that was part of the order.

If all of this is done the user is prompted for their card information. If the card information entered is new, then it is saved to the database.

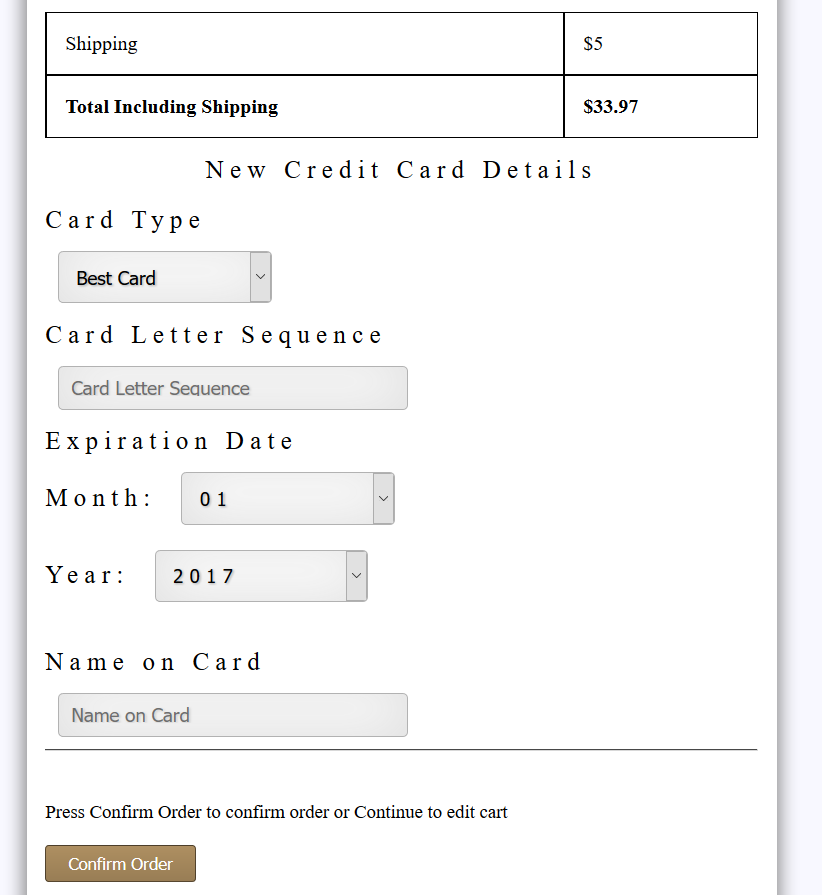
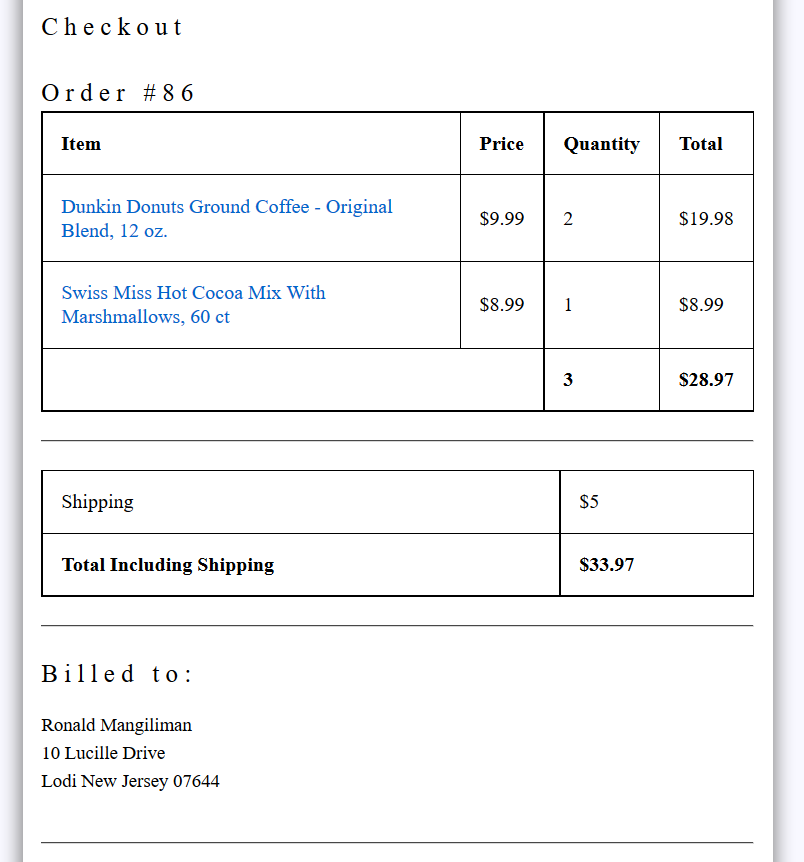


Figure 47. The form for inserting card details



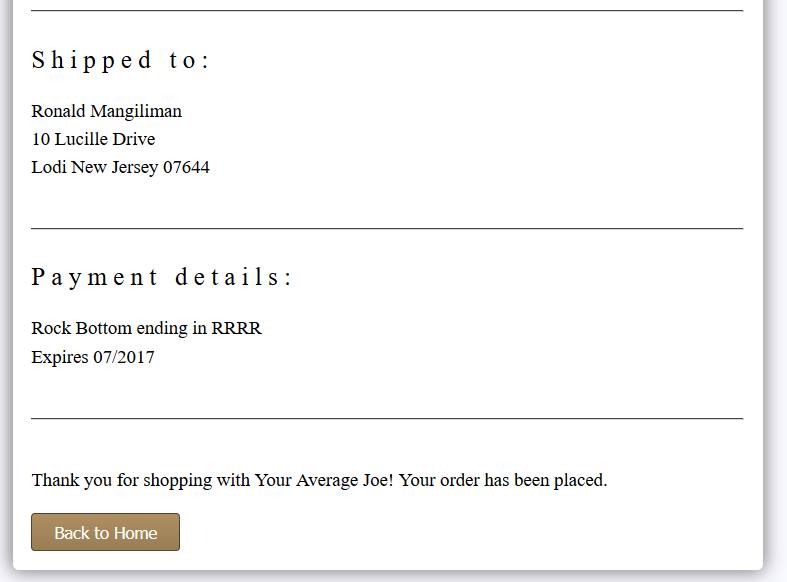
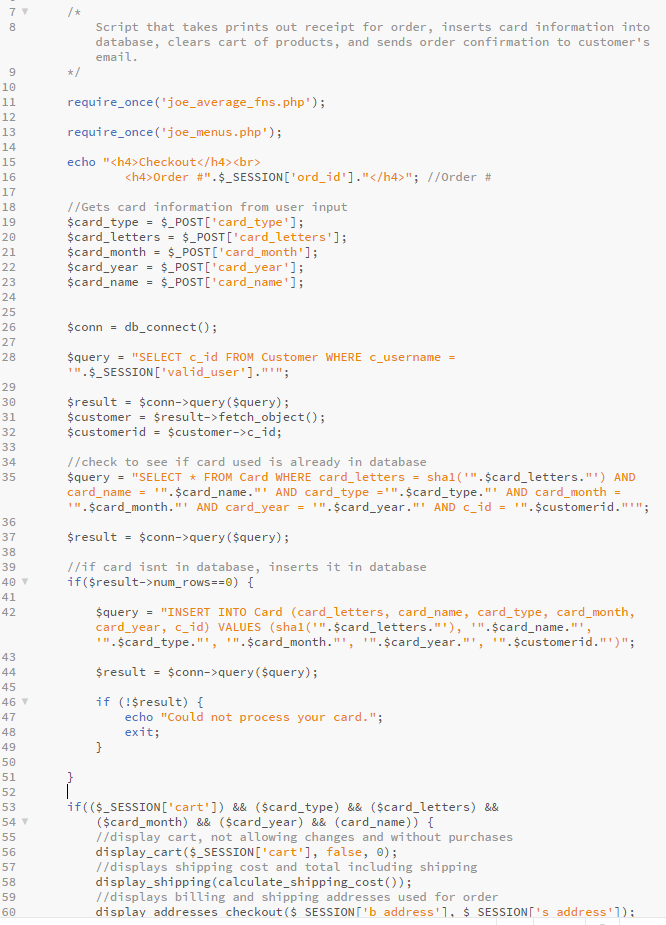


Figure 48. The receipt for the order is displayed to the customer.



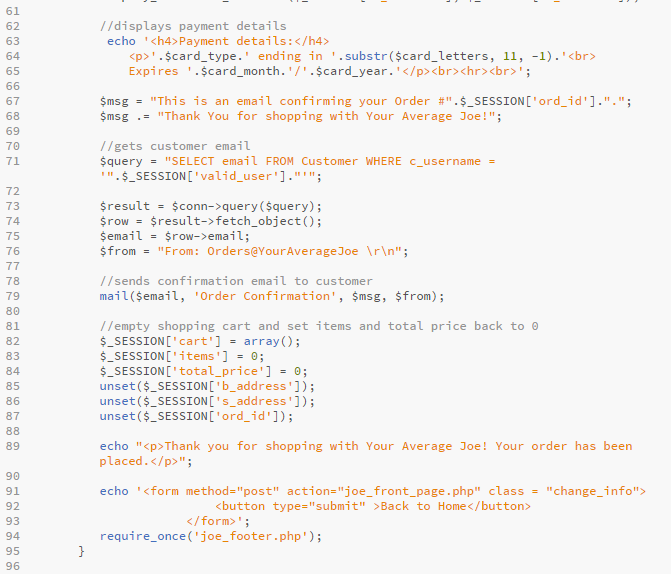


Figure 49. joe\_process.php

After displaying the order receipt, the rest of joe\_process.php empties the cart and unsets the session variables associated with it. It also sends a small confirmation email to the customer.

Viewing Order History

Since customers have the ability to make orders, it only makes sense for them to view orders they’ve made in the past. Since we stored their order information using the insert\_order() function when they checkout, we can view it.



Figure 50. Viewing order history

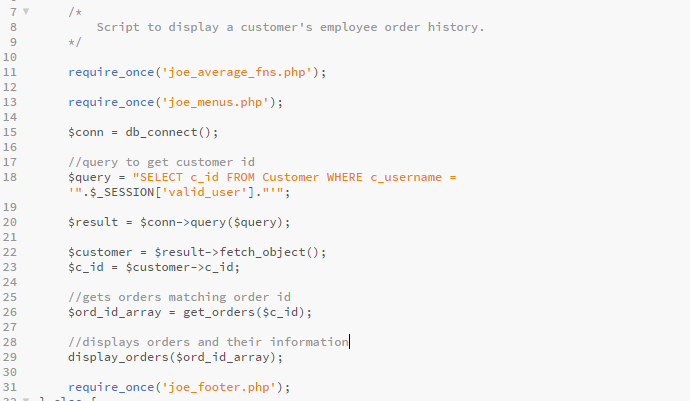


Figure 51. joe\_order\_history.php



Figure 52. The display\_orders() function from joe\_output\_fns.php

To view the order history of a customer, we used the familiar approach of getting information and displaying it. However, here it is a little different. Instead of having one foreach loop, there is also an inner foreach loop. The first displays the information of the order and then gets the details of the items that were part of that order using the function get\_order\_details(). Then the inner foreach loop displays each item that was purchased in that order, with its quantity and total price.

Logging Out

The final thing a customer can do is logout of the site. This is a simple script, but a crucial one, as it destroys the session and unsets any session variables related to it. If this script was not present, there is the chance that if an employee or database admin uses the site, exits out without logging in, and a customer logs in with their account, they would be able to see and access parts of the site that they are not allowed to see because the session variables from the previous user of a different type would still be set. Thus, all the checks in our site can be bypassed, or even cause errors.



Figure 53. joe\_logout.php

Employee View

Now that we have gotten the customer view out of the way, we will now discuss the employee view of the site. Logging in as an employee has a few distinct difference from being a customer. The most noticeable one is the fact that there is no way to access the cart as an employee and thus, no add to cart button on the product profiles. What replaces them is a disabled button with the product’s product ID number.

There are however, two new things an employee can do that customers cannot. One is updating the quantity of a certain product. The other is updating the status of an order.

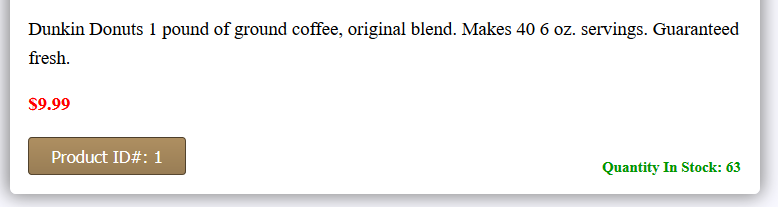


Figure 54. The add to cart button is gone for any non-customer users.

Updating Quantity



Figure 55. The update quantity showing up in employee view. Notice how the cart has been replaced with a logout button.

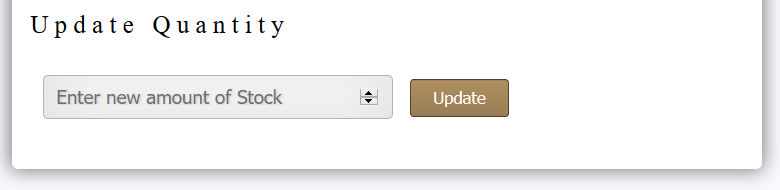


Figure 56. The update quantity page.

The update quantity page is simple. It allows the employee to alter the amount of items in stock by simply typing in a number and pressing update. There are two files that are used to achieve this. One is the form and the other is the script that takes the employee’s input and updates the Product table where the product ID matches that of the product profile used to access the update quantity page.



Figure 57. Part of joe\_quantity\_update.php Displays the form that takes in employee

Input

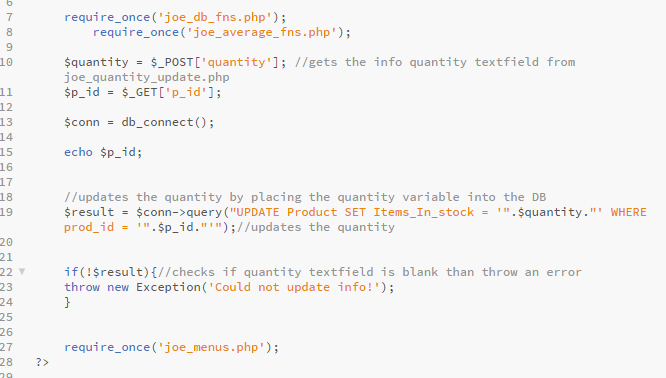


Figure 58. Part of joe\_quant\_update\_results.php

Updating Order Statuses

An employee has the ability to view every single order ever made by a customer on the site and can also alter the order status of that order to Shipped or Delivered.

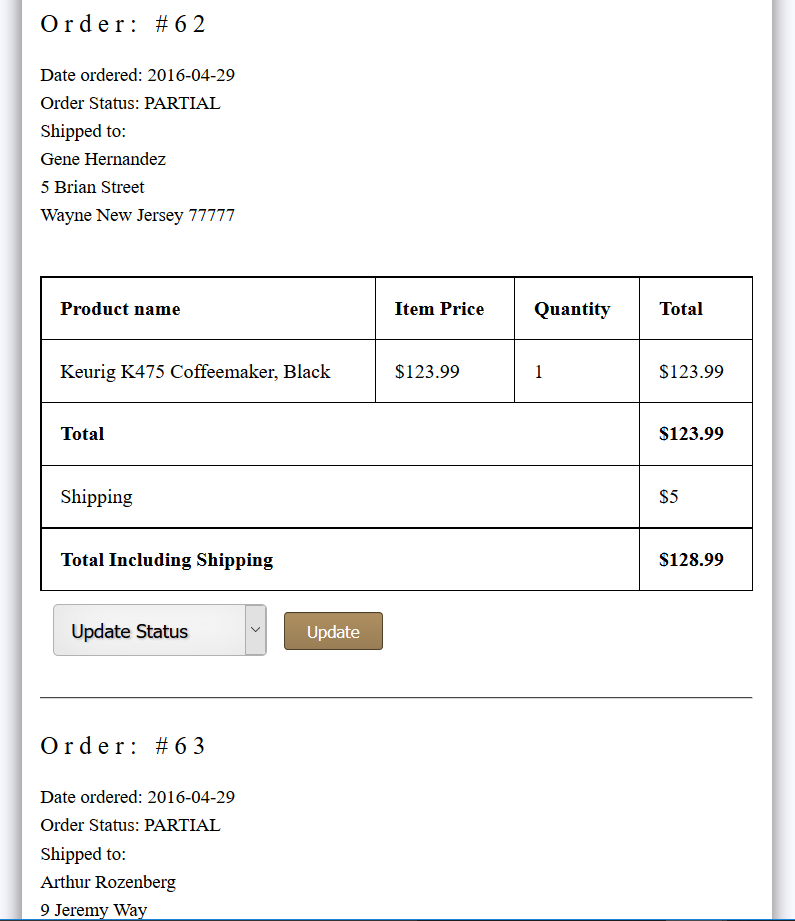


Figure 59. View of the update order status page before being updated.

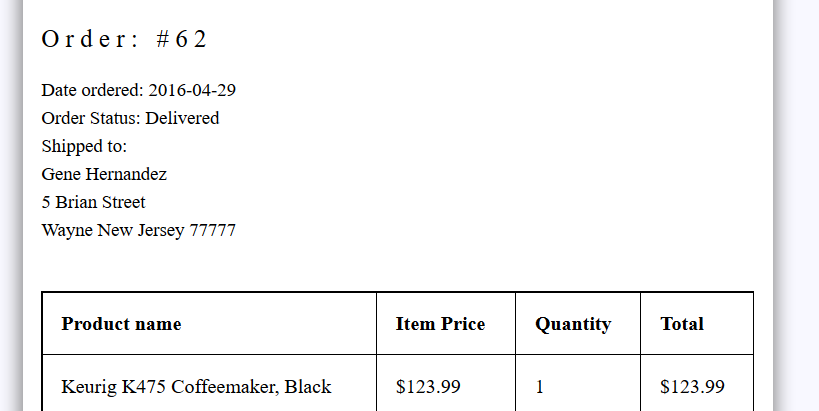


Figure 60. The same order after being updated to delivered.

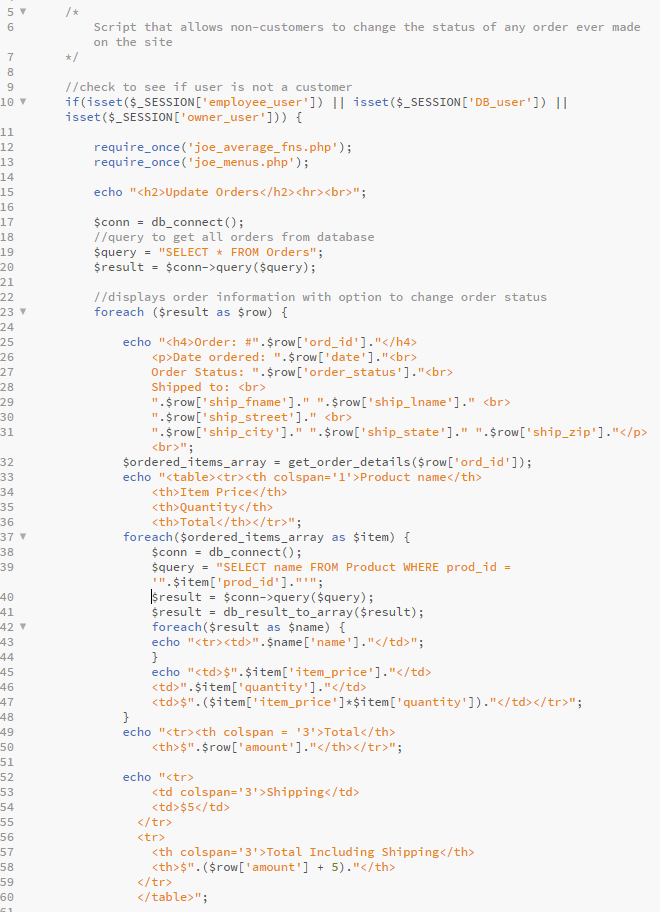




Figure 61. joe\_update\_orders.php

If the code for updating orders looks familiar, that is because it is very similar to the code that allows customers to view their online order history. The only difference here is that instead of viewing the orders made by one customer, you are viewing every order that has ever been made by a customer on the site. There is also a dropdown and button that allows for the actual changing of the order status via the script joe\_change\_order\_status.php.

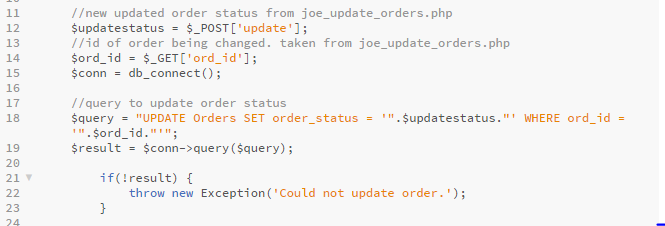


Figure 62. Part of joe\_change\_order\_status.php

Database Administrator View

Moving on to our Database Administrator view, we introduce two new things that this type of user is able to do on the site. The first one is the ability to add a new product to the database. The second is the ability to change the details of an existing product.



Figure 63. The option to add new products, visible in the Database Admin view



Figure 64. The option to edit an existing product’s details. Notice that the update quantity option is also available.

Adding a New Product

The option to add a new product can be seen at the top of any page that lists products. Clicking on it will transfer the user to the Add a Product page. This page prompts the user for a product name, description, price, image, category ID that it belongs to, brand ID, shipper ID, and supplier ID. All these fields must be filled for the product to be added into the database.

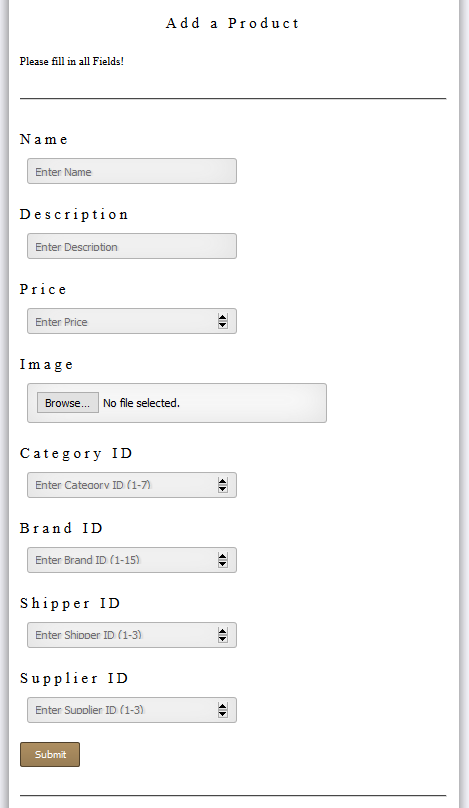


Figure 65. The add a new product page

The code for this form is straightforward and nothing we have not seen before, except for the new input type file that is used to upload a new image for the product. In Figure 66, you can see how we upload the image to the images folder on the server and get the actual image using $\_FILES. The image URL is then created by concatenating the image folder path to the basename of the uploaded image. The rest of the script is self-explanatory. The query takes in all the user inputted data and inserts it into the database as a new product.



Figure 66. Part of joe\_add\_product\_results.php.

Editing Existing Product Details

The ability to edit product details is very similar to changing personal information for the customer. The user is taken to a page where that displays the product’s information and can edit each piece of information individually. There is also an option to delete the product from the database.

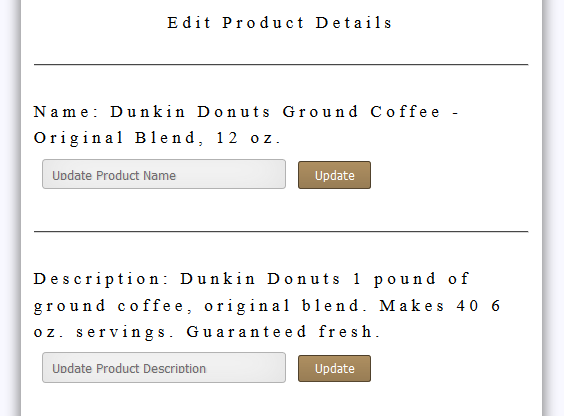


Figure 67. Part of the edit product details page. Notice the similarity to

change information page for the customer.

The code for this follows the same basic thinking that previous updating pages have. Thus, we will not show the code. (It can still be seen at the back of the documentation). There is a form and a script that takes the user input from the form and uses that to manipulate the data in the database. There is one query in joe\_edit\_product\_details.php that we have not yet seen, however, and that is the query to delete a product from the database, shown in Figure 68.



Figure 68. Query found in joe\_edit\_product\_details.php that deletes a row from the

database.

Owner View

The Owner view is the last type of view for the site. This view introduces three new things to the site: the ability to create employee accounts, the ability to edit and update employee information, and the ability to view employee information. We thought allowing the owner as the only one able to view, manage and create employee data made sense. These new features can be accessed through the profile page of the owner.

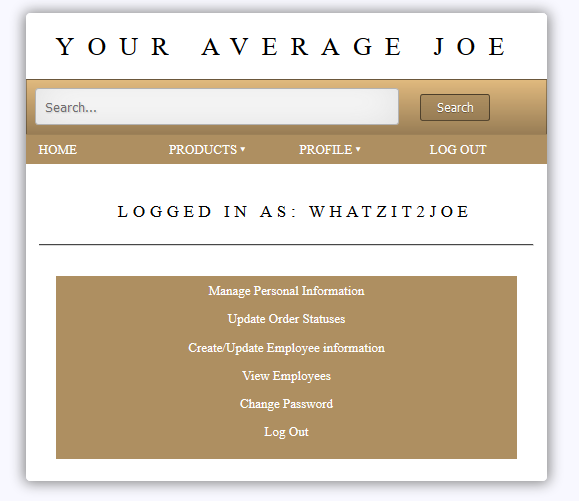


Figure 68. The Owner menu displaying options to create/edit employee

accounts and view employee information.

Clicking on the Create/Update Employee Information button will lead to two options. As one might expect, they are to create a new employee account or update an existing one.

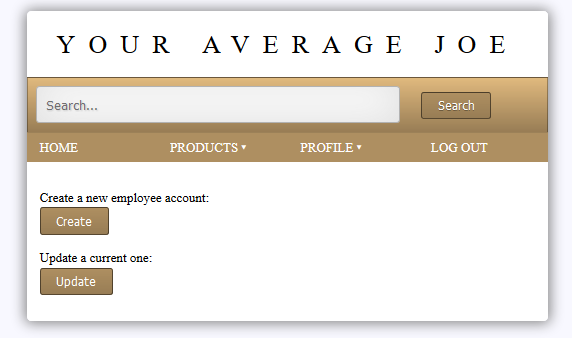


Figure 69. The Create/Update Employee Menu. So many options!

Pressing either option brings you to a page that allows you to either enter a new employee ID number or enter an existing one to edit.

Creating New Employee Accounts

The create employee page is very basic. Once a new employee ID has been entered and goes through the check to make sure that it does not already exist in the database, then the account is created. It then prompts the user for the following information to create a new employee account:

* Job Title
* First Name
* Last Name
* Employee Username
* Password
* Email Address
* Employee Address
  + Street
  + City
  + State
  + Zip Code
* Social Security Number

Each field will be updated separately, very similar to updating information for customers.

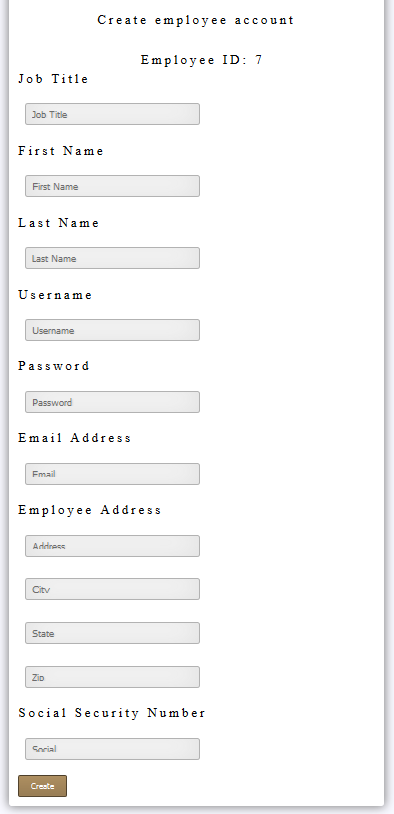


Figure 70. Create employee form

Updating Existing Employee Accounts

Updating existing employee accounts uses the same basic approach as creating a new employee. The difference here, is that the Update Employee Account page displays the current employee’s information (except for their Social Security Number and password for obvious reasons).

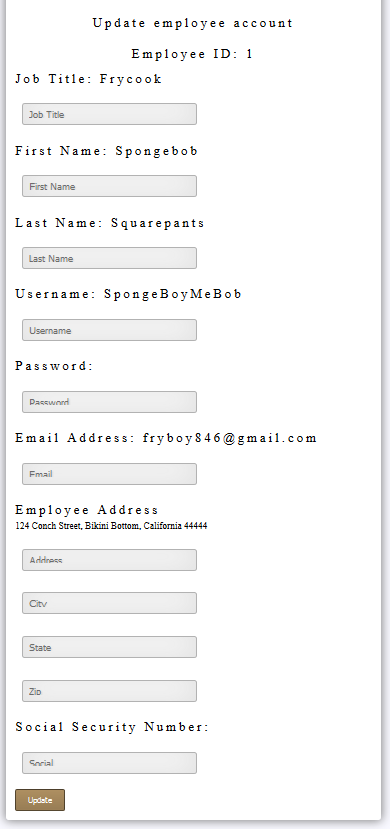


Figure 71. The Update Employee Page

The code for both of these is very straight forward.

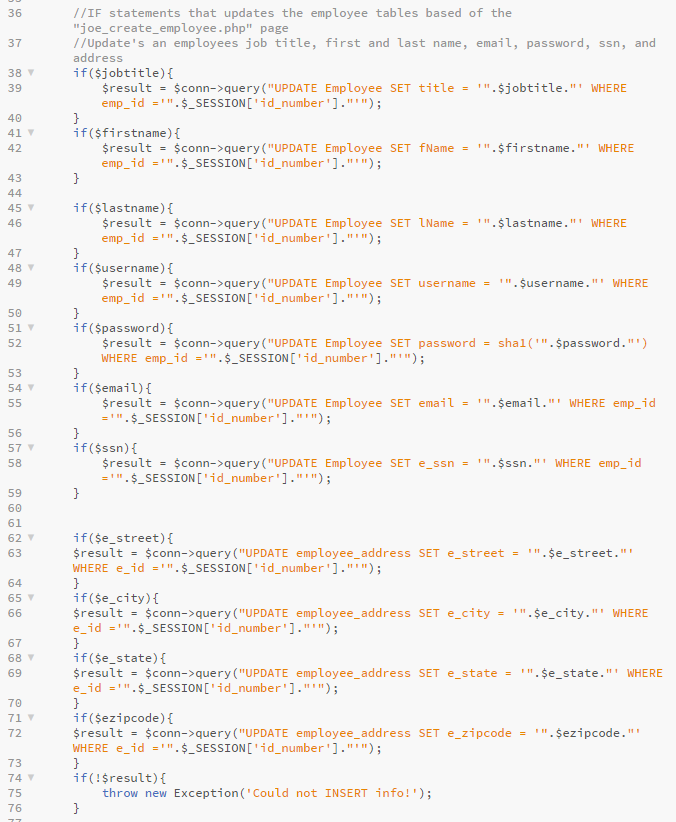


Figure 72. Part of joe\_new\_employee.php

Viewing Existing Employees

Since the owner of the site controls his employees, it only makes sense for his to have a compact view of his employee’s information. The view employees option in the Owner menu is exactly that. It gets employee information from the database and then displays onto the page. The one interesting thing about this is that because Employee and employee\_address are two separate tables in the database, we must use a join to get an employee’s full information.

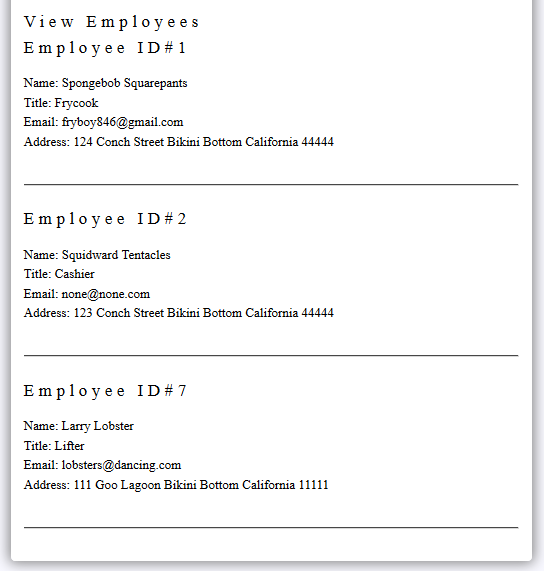


Figure 73. The Owner’s ability to see his all his employee’s info.

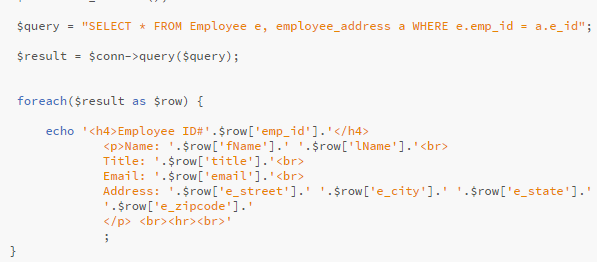


Figure 74. Part of joe\_view\_employees.php

In Figure 74, you can see how we used a join to combine the Employee and employee\_address table on their foreign key relation employee ID. After getting the information, it was just a matter of displaying the information using a foreach loop.

1. Analysis of Database Issues

Instead of making many tables for the addresses like employee address, billing address, and shipping address it would have been easier to just make one address table that stores everyone’s addresses in one table. The information might have been easier to access, and the written code could have been shorter.

Also in orders, we created a city, state zip etc. attributes. Of course this attribute already exists in the addresses table. We should have used the join operation with addresses to combine them to reduce redundancy. This was an oversight on our part, as the orders table was one of the last tables we implemented.

All data in every table is accessible in some form of the website. This means our data is redundancy is reduced to a minimum. For example, Employee table, all attributes that belong to employee are used in some way for our website. The first and last name attribute is used to identify the employee. The title is the job of the employee. Employee id is the primary key used to identify an employee. Username and password is what the employee uses to sign in. Finally email is needed if an employee resets their password.

Otherwise, the redundancy in our database is at a minimum as all tables have basic attributes and have a unique primary key to identify them. Some tables contain foreign keys to other tables.

1. Work Plan

For our work plan, we all agreed it would be in everyone’s best interest if everyone took part in every aspect of the project. We did end up splitting the actual coding amongst all of us by parts and features of the site, but there was a lot of overlap between everyone. So, what we decided was to have each person in charge of coding certain parts of the site. However, much of the coding happened in each other’s company (either in person or through Skype), we ended up helping each other out quite often. Once we had found a solution to the problem, we often used that person’s approach and applied it to similar problems another group member would run into. As a result, the approach to many of our site’s implementation was very similar, as indicated in the implementation section of this documentation. For example, our general approach to displaying data to the user was ‘get the data, and then display it’. This can be seen in much of our code. Much of it is the same basic structure, with minor alterations to fit our needs.

Ronald

The parts of the site Ronald was responsible for were user login, the shopping cart, and checkout. Some of the things this entailed was storing of addresses and storing orders and order details into the database and allowing users view their order history. He also took part in helping create the layout of the site, CSS, and took a very small part in editing user information (due to how addresses were stored) and creating the product profile with Arthur.

**Notable scripts written**: joe\_checkout.php, joe\_member.php, joe\_purchase.php, joe\_show\_cart.php, joe\_process.php, joe\_product\_profile.php, index.php, joe\_not\_proper\_length.php, joe\_not\_same\_pwd.php, joe\_change\_orderstatus.php, joe\_update\_orders.php

Arthur

The parts of the site Arthur was responsible for were things that had to do with products, category, and brands. This included displaying the categories, displaying the brands, and displaying products by brands. He also was responsible for allowing inserting products and altering product details through the various views for our site. He also took part in helping create the layout of the site, CSS, and worked with Ron to create the product profile page.

**Notable scripts written**: joe\_add\_product.php, joe\_add\_product\_results.php, joe\_all\_brands.php, joe\_all\_category.php, joe\_edit\_product\_details.php, joe\_edit\_product\_details\_results.php, joe\_product\_profile.php, joe\_quant\_update\_results.php, joe\_quantity\_update.php, joe\_show\_brand.php, joe\_show\_cat.php, joe\_show\_product\_all.php

Gene

Gene created most of the profile page and the features that were included in it. This consisted of displaying and allowing the editing of personal information for each type of user. He also implemented the feature to allow the Owner to create new employee accounts and update existing ones and view all the employees and their information. He also designed the layout of the site and did much of the CSS.

**Notable scripts written**: joe\_change\_info\_form.php, joe\_create\_employee.php, joe\_create\_employee\_id.php, joe\_create\_or\_update\_employee\_form.php, joe\_employee\_form.php, joe\_information\_results.php, joe\_new\_employee.php, joe\_profilepage.php, joe\_update\_employee.php, joe\_update\_employee\_form.php

Dan

Dan was responsible for allowing the user to register, change their password, reset their password, and logout. This entailed altering data to reflect these changes as well as inserting new customers. He also created the search engine that allowed the customer to browse products by entering a keyword or phrase into an input field. He took part in the layout and general design of the site, CSS, and worked with Gene to work out minor kinks in the profile page and editing personal information.

**Notable scripts written**: joe\_change\_password.php, joe\_change\_password\_form.php, joe\_forgot\_password.php, joe\_forgot\_password\_form.php, joe\_logout.php, joe\_register.php, joe\_registration\_form.php, joe\_search\_results.php

Any scripts not mentioned were either too general to assign credit for or were worked on by too many members to single out a single person. The database worked was split equally between the members. Any alterations to the database were discussed with the group before being made.

1. Security

For the security aspects of the site, we took certain measures to make sure only certain types of user could view or change data. The most obvious is that we have an if else statement on every file of the site except for the login, registration, and reset password page. This if else statement checks to make sure the session variable $\_SESSION[‘valid\_user’] is set. If it is, they can view the site. $\_SESSION[‘valid\_user’] is a session variable set by the login() function from joe\_user\_auth\_fns.php. Thus, to view the site, a user must be logged in. If they try to access the site without having $\_SESSION[‘valid\_user’] set (in other words, they accessed the site without logging in), then they will get a message saying they must log in to view the page and will be redirected after 5 seconds to the login in page.



Figure 75. Check to see if valid user.



Figure 76. If not, then they will be redirected.

We also did this for the views. We have checks to make sure that only a certain type of user can see it. The approach to implementing is very similar to the valid user check. The only difference is the checks are for $\_SESSION[‘DB\_user’], $\_SESSION[‘owner\_user’], and $\_SESSION[‘employee\_user’]. There is no session variable for customers. We just check to see whether all the others ones are unset.

Measures against SQL Injections

SQL injections are a way for users to input data into input fields that cause unexpected and oftentimes malicious actions to happen to the database. The input will usually be SQL code that will cause the query to return much more than the code author had intended. This causes unauthorized users to see or even manipulate data that they should not be able to.

To protect against this, we passed certain data through the sha1() PHP function to hash the input the user enters. This was only used for things such as passwords, social security numbers, and the card letter sequence for payment. This not only protects for malicious SQL code being entered into the input form, but also adds a layer of security as one cannot access the database and see a user’s password.

Another measure we took to protect against SQL injections was frontend for validation through HTML5 using the type, required, and pattern attribute. The simplest one of these is the required attribute. It is a simple Boolean attribute that forces the user to fill out an input field. These were used throughout the site to make sure users filled out certain important fields. The type attribute indicates how the user submits their input. We used these to make sure the user only inputs certain types of data. For example, when taking in states for addresses, we present a drop down menu of all states. This takes out the possibility of someone trying to input malicious code into a text field. Obviously, we cannot use this for all user input. When we do need to use text input, we used the pattern attribute to only accept certain characters or patterns. An example of this is first name, where we only allowed letters, both upper and lowercase from length 1-25. Another one is for zip code, where we only accepted a string of numbers that is five in length from zero to nine. Username is similar. For that one we used lower and uppercase letters with numbers. Length had to be six to sixteen.



Figure 77. Regular expression for our first name

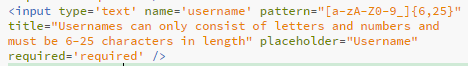


Figure 78. Username

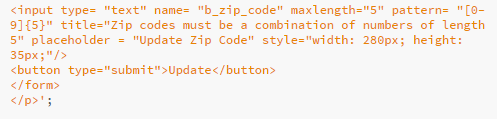


Figure 79. Zip Code

1. Appendix of Code
2. Conclusions

In conclusion, we learned a lot of important information in this course about database systems, how they work, and how important they are to our everyday lives. Before this class, we never really thought about how data is managed or stored. The term database was a common one we heard throughout our lives, but never really thought what they were. They were obviously used to store data, that is in the name itself. But how did they store it? How was the data retrieved or updated? These were questions that we never really thought about. After sixteen weeks of this class, we now know exactly how important databases are and how they play a role in our everyday lives, whether we realize it or not.

Things that we did almost every day without giving it much thought, such as entering a username and password to log into a site, now seem so different. Is the information we are inputting into the database secure? How are our passwords saved? Are they encrypted? (They better be!) How is security handled? Is the database protected from SQL injections? All these things are now things we think about as people in the computing field.

All the members of our group are also aspiring developers. Relational database management systems are an important part of the development stack. We’ve learned many important skills and concepts that we all hope to use to further our careers when we work in industry. Database systems aren’t the only thing we learned from this class. We got exposed to some web development, particularly the backend side with our use of PHP. At least two of our group members plan to use their experience with PHP to help them learn more about backend development. At least one of us plan to learn another backend language in the summer (Ruby on Rails).

This class (and particularly, this project) was a great learning experience. It definitely put stress levels to new heights and the amount of coffee we’ve drank while doing this is probably extremely unhealthy, but it pushed us as members of the computing field and gave us an invaluable experience that will be close to what we do in industry without many of the repercussions. We’ve definitely made mistakes during this project and we’ve learned from them. We learned to work as a team and develop good communication and leadership skills. We also definitely need to manage our time better for future projects. And while we are glad to finally turn this project in and have it over with, it will always be one of those moments we will look back on and reflect on how much it helped us improve our craft.

1. Citations

**IMAGES FOR PRODUCT**

<http://www.amazon.com/Dunkin-Donuts-Original-Ground-Coffee/dp/B000URH0B0/ref=sr_1_1_a_it?ie=UTF8&qid=1462230160&sr=8-1&keywords=dunkin+donuts+12+oz+original+blend>

<http://www.amazon.com/Dunkin-Donuts-Decaffeinated-Medium-Ground/dp/B00KO8MUYW/ref=sr_1_2_a_it?ie=UTF8&qid=1462230578&sr=8-2&keywords=dunkin+donuts+12+oz+decaf>

<http://www.amazon.com/Dunkin-Donuts-Roast-Ground-Coffee/dp/B00JTI1LQ6/ref=sr_1_2_a_it?ie=UTF8&qid=1462230629&sr=8-2&keywords=dunkin+donuts+12+oz+dark+roast>

<http://www.amazon.com/Dunkin-Donuts-Hazelnut-Ground-Coffee/dp/B000YJKQ3I/ref=sr_1_1_a_it?ie=UTF8&qid=1462230674&sr=8-1&keywords=dunkin+donuts+12+oz+hazelnut>

<http://www.amazon.com/Dunkin-Donuts-Ground-Coffee-Pumpkin/dp/B00K97L36Y/ref=sr_1_fkmr0_2_a_it?ie=UTF8&qid=1462230734&sr=8-2-fkmr0&keywords=dunkin+donuts+12+oz+pumpkin>

<http://www.amazon.com/Dunkin-Donuts-French-Vanilla-Ground/dp/B000YJON4Q/ref=sr_1_1_a_it?ie=UTF8&qid=1462230886&sr=8-1&keywords=dunkin+donuts+12+oz+french+vanilla>

<http://www.amazon.com/Starbucks-Decaffeinated-Ground-Coffee-Medium/dp/B000RULTYY/ref=sr_1_1_a_it?ie=UTF8&qid=1462230924&sr=8-1&keywords=starbucks+medium+house+12+oz>

<http://www.amazon.com/Starbucks-Place%C2%AE-Roast-Ground-Coffee/dp/B002NBUNYM/ref=sr_1_1_a_it?ie=UTF8&qid=1462230955&sr=8-1&keywords=starbucks+pike+place+12+oz>

<http://www.amazon.com/Starbucks-Dark-Roast-Whole-Verona/dp/B019WMMZNG/ref=sr_1_1_a_it?ie=UTF8&qid=1462230981&sr=8-1&keywords=starbucks+dark+cafe+verona+12+oz>

<http://www.amazon.com/Starbucks-Blonde-Roast-Veranda-Blend/dp/B00D2AXM1Y/ref=sr_1_3_a_it?ie=UTF8&qid=1462231005&sr=8-3&keywords=starbucks+blonde+veranda+12+oz>

<http://www.amazon.com/Starbucks-Caffe-Verona-Ground-Coffee/dp/B003VDC3QW/ref=sr_1_sc_6_a_it?ie=UTF8&qid=1462231068&sr=8-6-spell&keywords=starbucks+dark+caffee+verona>

<http://www.amazon.com/gp/product/B00H9YU0XW/ref=pd_lpo_sbs_dp_ss_1?pf_rd_p=1944687522&pf_rd_s=lpo-top-stripe-1&pf_rd_t=201&pf_rd_i=B00G0RYZJY&pf_rd_m=ATVPDKIKX0DER&pf_rd_r=1NT5914YPM6GGF22RMV9>

<http://www.amazon.com/America-Dunkin-Donuts-Ceramic-Dishwasher/dp/B013PV11H6>

<http://www.amazon.com/Starbucks-Reusable-Travel-Coffee-Grande/dp/B00BE5L9OG/ref=sr_1_1?ie=UTF8&qid=1462231206&sr=8-1&keywords=starbucks+reusable+travel+16>

<http://www.amazon.com/Starbucks-Stainless-Steel-Tumbler-Handle/dp/B00H3VM3IQ>

<http://www.amazon.com/TeaLeava-Clear-Glass-Stainless-Infuser/dp/B01AV8E182/ref=sr_1_sc_1?ie=UTF8&qid=1462231531&sr=8-1-spell&keywords=tea+leava+glass+pot>

<http://www.amazon.com/Lipton-Black-Tea-Bags-100/dp/B00I8GPZ6W/ref=sr_1_1_a_it?ie=UTF8&qid=1462231552&sr=8-1&keywords=lipton+black+tea+bag+100>

<http://www.amazon.com/Bigelow-Green-Tea-Bags-40/dp/B00NAZF7I2/ref=sr_1_6_a_it?ie=UTF8&qid=1462231801&sr=8-6&keywords=bigelow+tea+green+40>

<http://www.amazon.com/Yogi-Honey-Lavender-Stress-Relief/dp/B007M8FRV0/ref=sr_1_1_a_it?ie=UTF8&qid=1462231824&sr=8-1&keywords=yogi+honey+lavender+stress+relief+16>

<http://www.amazon.com/Twinings-English-Breakfast-Count-7-05/dp/B001GM60LE/ref=sr_1_1_a_it?ie=UTF8&qid=1462231863&sr=8-1&keywords=twinnings+tea+english+100>

<http://www.amazon.com/Twinings-Tea-Earl-Grey-Count/dp/B009UQXD90/ref=sr_1_1_a_it?ie=UTF8&qid=1462231900&sr=8-1&keywords=twinnings+tea+earl+100>

<http://www.amazon.com/Twinings-Peppermint-Herbal-Ounce-Count/dp/B009L6Q2YW/ref=sr_1_1_a_it?ie=UTF8&qid=1462231924&sr=8-1&keywords=twinnings+peppermint+tea+20>

<http://www.amazon.com/Swiss-Miss-Hot-Cocoa-Mix/dp/B00S7NQQ0A/ref=sr_1_9?ie=UTF8&qid=1462231958&sr=8-9&keywords=swiss+miss+60>

<http://www.amazon.com/Swiss-Miss-Marshmallow-Cocoa-Ounce/dp/B00BJ9URYA/ref=sr_tnr_p_1_16318161_1_a_it?ie=UTF8&qid=1462231999&sr=8-1-fkmr1&keywords=swiss+miss+60+marshmallows>

<http://www.amazon.com/Starbucks-Classic-Hot-Cocoa-Ounce/dp/B00FKBUSW4/ref=sr_1_1_a_it?ie=UTF8&qid=1462232041&sr=8-1&keywords=starbucks+classic+30+cocoa>

<http://www.amazon.com/Ghirardelli-Premium-Chocolate-1-5-Ounce-Envelopes/dp/B0085SN7J0/ref=sr_1_1_a_it?ie=UTF8&qid=1462232078&sr=8-1&keywords=ghirardelli+hot+cocoa+double+15>

<http://www.amazon.com/Starbucks-Frappuccino-Vanilla-Light-Pack/dp/B00GH40N96/ref=sr_1_4_a_it?ie=UTF8&qid=1462232099&sr=8-4&keywords=starbucks+frappuccino+chilled+vanilla>

<http://www.amazon.com/Frappuccino-Chilled-Coffee-Drink-Flavor/dp/B00K5NPI18/ref=sr_1_4_a_it?ie=UTF8&qid=1462232134&sr=8-4&keywords=starbucks+frappuccino+chilled+mocha>

<http://www.amazon.com/Starbucks-Doubleshot-Espresso-Cream-Ounce/dp/B00DS4Q1IK/ref=sr_1_1_a_it?ie=UTF8&qid=1462232168&sr=8-1&keywords=starbucks+doubleshot+espresso>

<http://store.starbucks.co.uk/starbucks_logo_mug_16_fl_oz/011015684,en_GB,pd.html?start=1&navid=search>

<http://www.amazon.com/Alpine-Spiced-Apple-Instant-Original/dp/B002HQEMCW/ref=sr_1_fkmr0_3_a_it?ie=UTF8&qid=1462232341&sr=8-3-fkmr0&keywords=alpine+spiced+apple+cider+insta>

<http://www.amazon.com/Alpine-Cider-Caramel-10-Count-57-Ounce/dp/B00F94XR0W/ref=sr_1_3_a_it?ie=UTF8&qid=1462232382&sr=8-3&keywords=alpine+spiced+caramel+10>

[https://www.amazon.co.uk/ALPINE-SPICED-INSTANT-POUCHES-AMERICAN/dp/B007IXY3G4/278-9034419-2079542?ie=UTF8&\*Version\*=1&\*entries\*=0](https://www.amazon.co.uk/ALPINE-SPICED-INSTANT-POUCHES-AMERICAN/dp/B007IXY3G4/278-9034419-2079542?ie=UTF8&*Version*=1&*entries*=0)

<http://www.amazon.com/Dunkin-Donuts-K-Cups-Original-Flavor/dp/B00MUZ15HU/ref=sr_1_2_a_it?ie=UTF8&qid=1462232484&sr=8-2&keywords=dunkin+donuts+12+original>

<http://www.amazon.com/Dunkin-Donuts-K-cups-Decaf-Brewers/dp/B00O2BCCG8/ref=sr_1_fkmr1_1_a_it?ie=UTF8&qid=1462232508&sr=8-1-fkmr1&keywords=dunkin+donuts+12+original+decaf>

<http://goprodrone.org/product/B00MUZ2IWG/Dunkin-Donuts-K-Cups-Pumpkin-Flavor-Box-of-12-Kcups-for-use-in-Keurig-Coffee-Brewers/>

<http://www.amazon.com/French-Vanilla-Keurig-Coffee-Brewers/dp/B00NZJ11IS/ref=sr_1_3_a_it?ie=UTF8&qid=1462232658&sr=8-3&keywords=dunkin+donuts+12+french+vanilla+k-cup>

<http://www.amazon.com/Dunkin-Donuts-Coffee-Cups-12/dp/B00T8NI3MM/ref=sr_1_3_a_it?ie=UTF8&qid=1462232678&sr=8-3&keywords=dunkin+donuts+12+hazelnut+k-cup>

<http://www.amazon.com/Dunkin-Donuts-Chocolate-Cocoa-K-cups/dp/B00OQDD5ZY/ref=sr_1_1_a_it?ie=UTF8&qid=1462232698&sr=8-1&keywords=dunkin+donuts+12+hot+cocoa+k-cup>

<http://www.amazon.com/Starbucks-House-Blend-Keurig-Brewers/dp/B005XB5K0G/ref=sr_1_1_a_it?ie=UTF8&qid=1462232748&sr=8-1&keywords=starbucks+16+k-cups+house>

<http://www.amazon.com/Starbucks-NA-Caramel-16-ct/dp/B00NCBYNKM/ref=sr_1_1_a_it?ie=UTF8&qid=1462232765&sr=8-1&keywords=starbucks+16+caramel>

<http://www.amazon.com/Keurig-Starbucks-Sumatra-16-Count-K-Cups/dp/B005XB3U26/ref=sr_1_2_a_it?ie=UTF8&qid=1462232781&sr=8-2&keywords=starbucks+16+sumatra>

<http://www.amazon.com/Starbucks-French-Portion-Brewers-16-Count/dp/B005XB67F8/ref=sr_1_1_a_it?ie=UTF8&qid=1462232824&sr=8-1&keywords=starbucks+16+french+roast>

<http://www.amazon.com/Starbucks-Mocha-Cup-35-cups/dp/B00LKUHCX2/ref=sr_1_1_a_it?ie=UTF8&qid=1462232834&sr=8-1&keywords=starbucks+16+mocha>

<http://www.amazon.com/Starbucks-Veranda-Blonde-Keurig-Brewers/dp/B006XYUS5O/ref=sr_1_2_a_it?ie=UTF8&qid=1462232848&sr=8-2&keywords=starbucks+16+veranda>

<http://www.amazon.com/Starbucks-Breakfast-K-Cups-Keurig-Brewers/dp/B0065BM7Q8/ref=sr_1_4_a_it?ie=UTF8&qid=1462232875&sr=8-4&keywords=starbucks+16+breakfast>

<http://www.amazon.com/Mr-Coffee-BVMC-SJX33GT-Programmable-Coffeemaker/dp/B0047Y0UQO/ref=sr_1_1?ie=UTF8&qid=1462232907&sr=8-1&keywords=mr+coffee+bvmc-sjx33gt>

<http://www.amazon.com/Mr-Coffee-4-Cup-Programmable-Coffeemaker/dp/B0008JIW8U/ref=sr_1_1?ie=UTF8&qid=1462232927&sr=8-1&keywords=mr+coffee+drx5>

<http://www.amazon.com/Hamilton-Beach-Programmable-Brewstation-48464/dp/B001K66LPQ/ref=sr_1_1?ie=UTF8&qid=1462232952&sr=8-1&keywords=hamilton+beach+brewstation+12>

<http://www.amazon.com/Keurig-K15-Coffee-Maker-Packaging/dp/B018UQ56IG/ref=sr_1_1?ie=UTF8&qid=1462232970&sr=8-1&keywords=keurig+k15>

<http://www.amazon.com/Keurig-K250-Brewing-System-Black/dp/B00WUUNZIW/ref=sr_1_1?ie=UTF8&qid=1462233008&sr=8-1&keywords=keurig+k250>

<http://www.amazon.com/Mr-Coffee-ECMP50-Espresso-Cappuccino/dp/B001EU9UTM/ref=sr_1_1?ie=UTF8&qid=1462233027&sr=8-1&keywords=mr+coffee+ecmp50>

<http://www.amazon.com/Mr-Coffee-Espresso-Automatic-BVMC-ECMP1000/dp/B007K9OIMU/ref=sr_1_1?ie=UTF8&qid=1462233059&sr=8-1&keywords=mr+coffee+bvmc-ecmp1000>

<http://www.amazon.com/Breville-BES870XL-Barista-Express-Espresso/dp/B00CH9QWOU/ref=sr_1_1?ie=UTF8&qid=1462233113&sr=8-1&keywords=breville+870xl>

**IMAGES FOR CATEGORIES**

[**http://www.types.coffee/coffee-types**](http://www.types.coffee/coffee-types)

[**http://www.wayfair.com/Certified-International-Tapas-18-oz.-Mug-63053SET-4-CEI2827.html**](http://www.wayfair.com/Certified-International-Tapas-18-oz.-Mug-63053SET-4-CEI2827.html)

[**http://beforeitsnews.com/alternative/2014/04/top-10-favourite-beverages-of-all-times-2943524.html**](http://beforeitsnews.com/alternative/2014/04/top-10-favourite-beverages-of-all-times-2943524.html)

[**http://www.babble.com/best-recipes/wishing-you-a-warm-creamy-weekend-mexican-hot-chocolate-with-dulce-de-leche-spiced-cream/**](http://www.babble.com/best-recipes/wishing-you-a-warm-creamy-weekend-mexican-hot-chocolate-with-dulce-de-leche-spiced-cream/)

[**http://www.coroflot.com/wrightni/Frappuccino**](http://www.coroflot.com/wrightni/Frappuccino)

[**https://www.soap.com/p/productname-282365**](https://www.soap.com/p/productname-282365)

[**http://www.expertmarket.com/How-To-Descale-A-Commercial-Coffee-Machine**](http://www.expertmarket.com/How-To-Descale-A-Commercial-Coffee-Machine)

**IMAGES FOR BRANDS**

<http://www.isciencetimes.com/articles/4091/20121127/dunkin-donuts-coffee-claim-trademark-best-america.htm>

<http://fifteenseconds.co/loyalty-program-changes-starbucks-selbstverschuldeter-image-crash/>

<http://www.tealeava.com/products/tealeava-glass-tea-pot-with-stainless-steel-infuser-for-loose-leaf-tea-16oz-tea-maker>

<http://gulfmarketingreview.com/company-brand/lipton-yellow-label-relaunches>

<http://www.webstaurantstore.com/bigelow-green-tea-variety-tray-pack-8-flavors-64-bags/11030568.html>

<http://www.harisingh.com/YogiTeaByYogiBhajan.htm>

<http://coffeeandtea.co.za/tea/twinings-tea-boxes>

<http://www.cw-usa.com/ghirardelli-hot-cocoa-mix-zip-pouch-double-chocolate.html>

<http://www.canadianfavourites.com/Nestle_Carnation_Hot_Chocolate_10x28g_p/nestle118.htm>

<http://www.chocolates-and-chocolate-candy.com/hot-chocolate-gifts.html>

<http://www.costcobusinessdelivery.com/drink-mixes.html>

<https://poshmark.com/listing/Final-SaleMr-Coffee-wit-Keurig-Coffee-Makr-5691e7ea6ba9e661e9046adb>

[http://www1.macys.com/shop/product/keurig-2.0-k250-brewer?ID=2457525&pla\_country=US&CAGPSPN=pla&CAWELAID=120156340002767919&CAAGID=17571657436&CATCI=pla-160052279836&catargetid=120156340005911038&cadevice=c&cm\_mmc=Google\_PLA\_Home\_Electrics-\_-Electrics+-+Coffee+Makers+-+GS\_Keurig-\_-55806077836\_-\_-\_mkwid\_gbUhAoq7|dc\_55806077836%7C-%7CgbUhAoq7&gclid=CLr0pIjQvMwCFYIfhgodXE4OBw](http://www1.macys.com/shop/product/keurig-2.0-k250-brewer?ID=2457525&pla_country=US&CAGPSPN=pla&CAWELAID=120156340002767919&CAAGID=17571657436&CATCI=pla-160052279836&catargetid=120156340005911038&cadevice=c&cm_mmc=Google_PLA_Home_Electrics-_-Electrics+-+Coffee+Makers+-+GS_Keurig-_-55806077836_-_-_mkwid_gbUhAoq7%7Cdc_55806077836%7C-%7CgbUhAoq7&gclid=CLr0pIjQvMwCFYIfhgodXE4OBw)

<http://www.meijne.eu/2016/04/03/guilty-pleasures-1/>

**List of Brands Used:**

**Dunkin Donuts, Starbucks, TeaLeava, Lipton, Bigelow, Yogi, Twinnings, Swiss Miss, Nestle, Ghirardeli , Alpine, Mr. Coffee, Hamilton, Keurig, Breville. Certified Internation Corp.**

**Books:**

Elmasri, Ramez, and Sham Navathe. *Fundamentals of Database Systems*. Boston: Addison-Wesley, 2011. Print.

Welling, Luke, and Laura Thomson. *PHP and MySQL Web Development, Fourth Edition*. Upper Saddle River: Addison-Wesley, 2009. Print.

1. Business Plan and Status Reports