Web Systems Programming

CS Tools Documentation

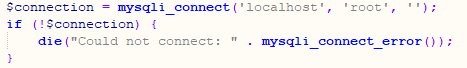
Shaun Mueller - 13171256

# Database creation (createDB.php)

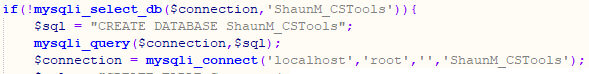
The database is created via a php file called creatDB.php.

This file first checks to see whether a connection to MySQL can be made using the default login with the username ‘root’ and no password.

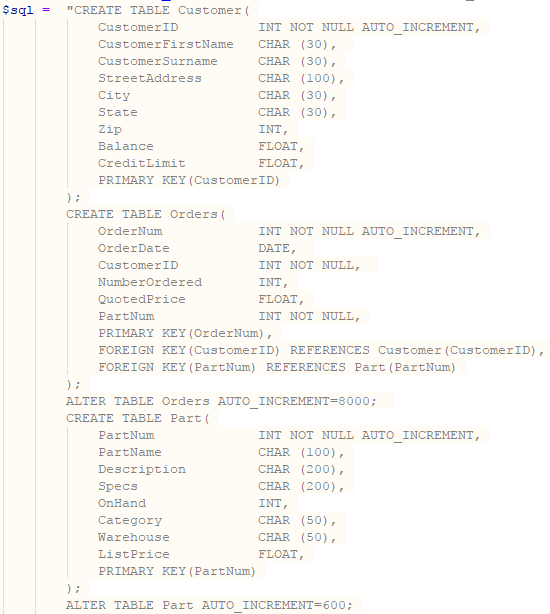
If it cannot connect to MySQL, then the connection error will be displayed and the remaining code will not be executed.



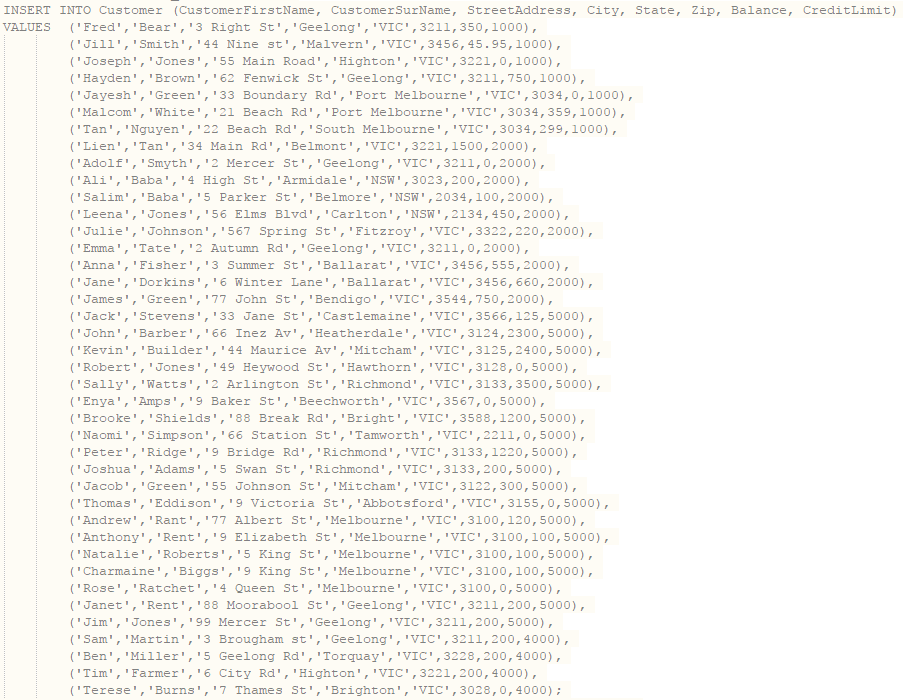
The creatDB.php file then checks to see if a database called ‘ShaunM\_CSTools’ exists. If the database does not exist, the php code will create a new database called ‘ShaunM\_CSTools’ and connect to it.



After that it will create the tables within the database, assigning data types to fields, setting the length of string fields, set auto increments, and set the primary and foreign keys.

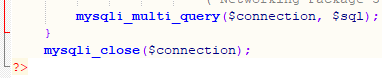


The code then populates the tables with data, which is hardcoded into the php.

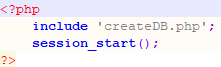


This is all done using the ‘mysqli\_multi\_query’ function.

When the tables have been created and filled with data, the connection to the database is closed and the code ends.

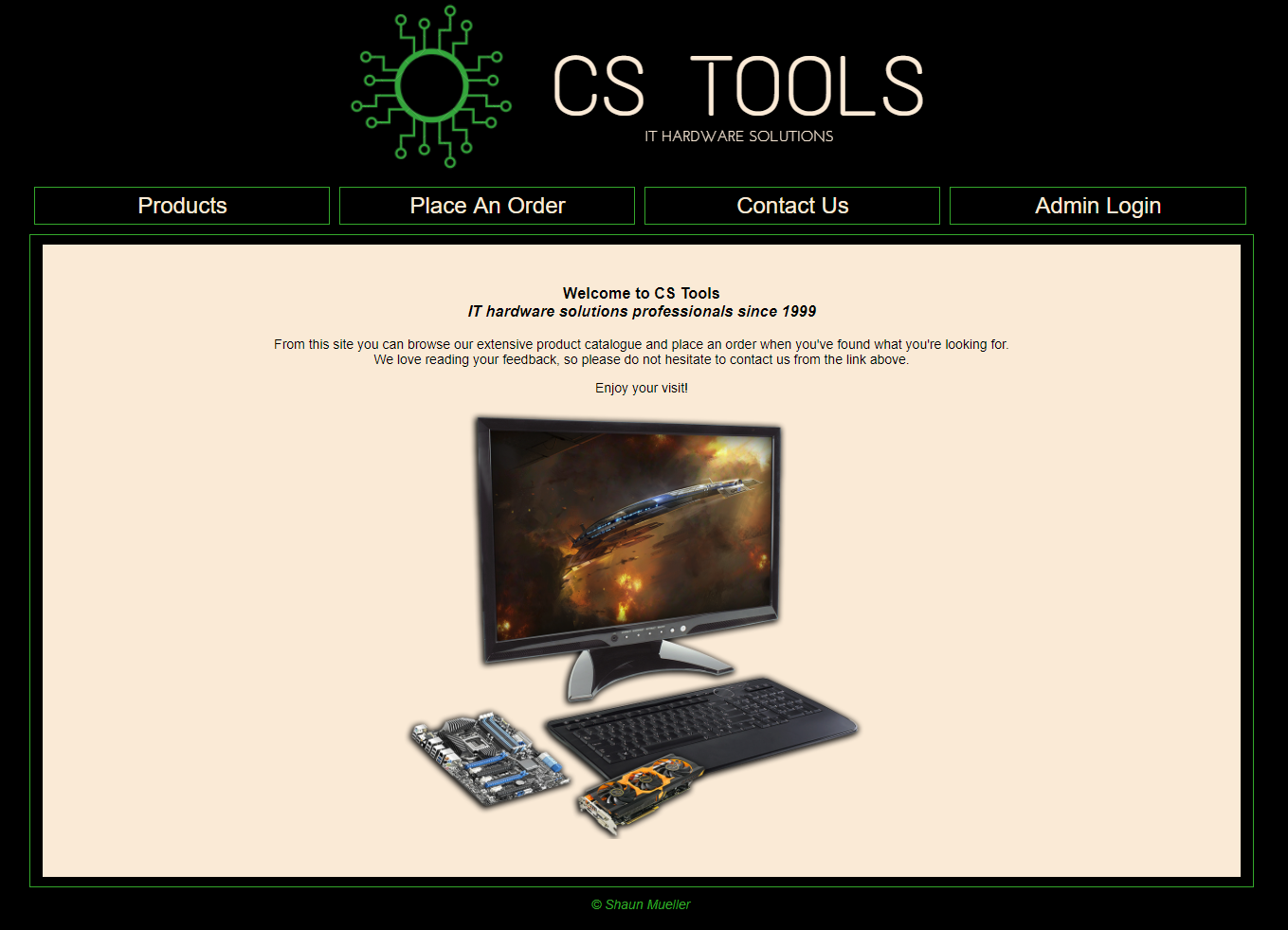


To ensure that the creatDB.php file is run when the site is opened, I used the ‘include’ function at the start of the index site. This means that when the index site is run, it will also run the createDB.php file.

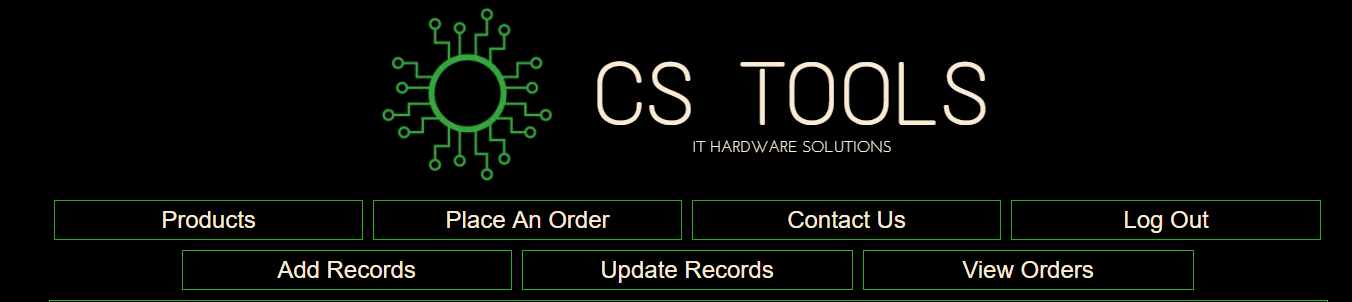


# Index Page (index.php)

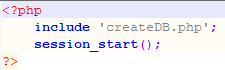
The index page is pretty straight-forward and acts as a template for each of the other pages, with only the inner content changing.



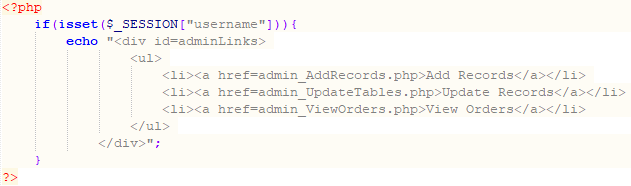
The navigation links across the top of the screen will change depending on if an administrator has logged in. The picture above shows the links with no administrator logged in. After an administrator has logged in, the links will appear as below.



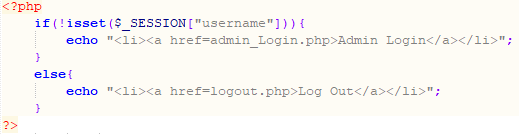
This is achieved by resuming any existing sessions previously created through the login process using the ‘session\_start()’ function at the start of the code.



Then the code can check whether a session exists and display the administrator links if one does.



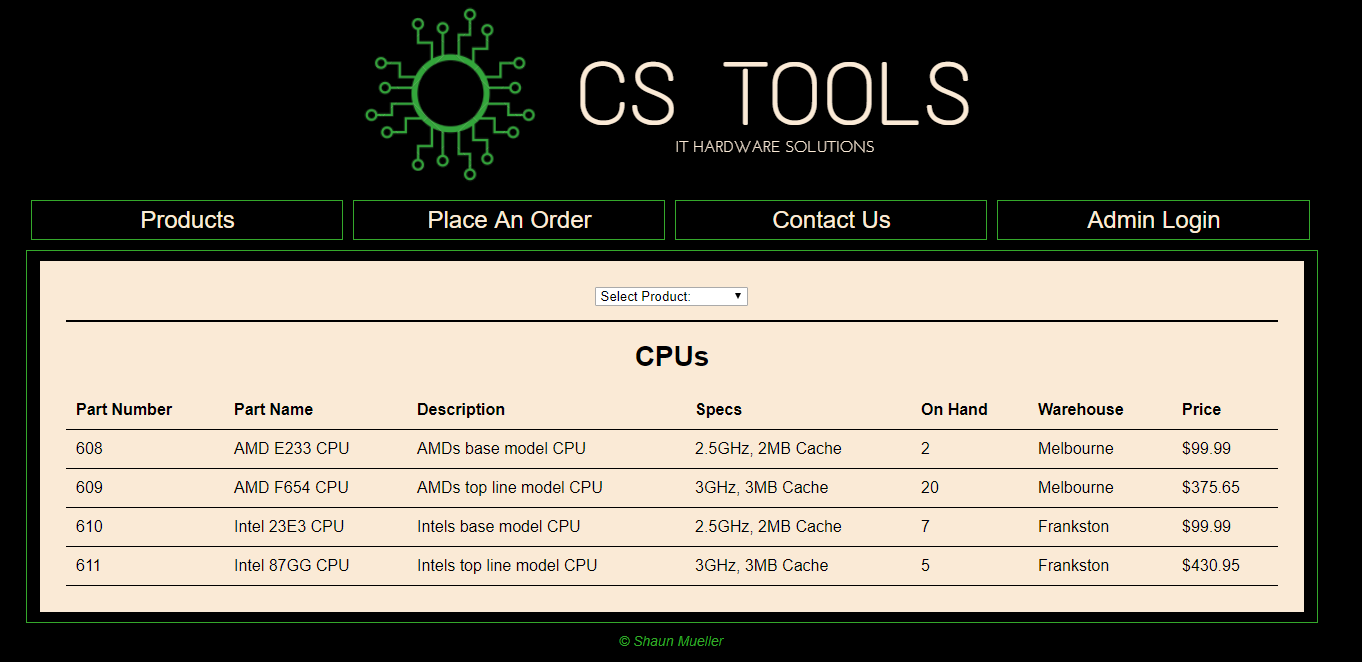
The code also checks the session to determine whether to display the login or logout link.



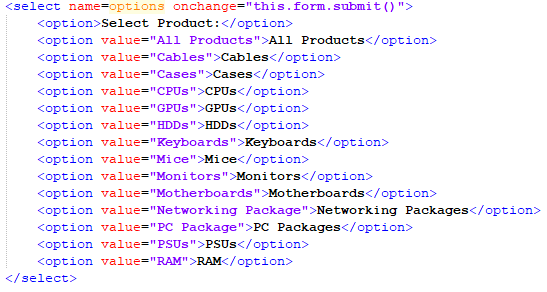
This code is included on all the pages to keep a uniform browsing appearance.

# Products Page (products.php)

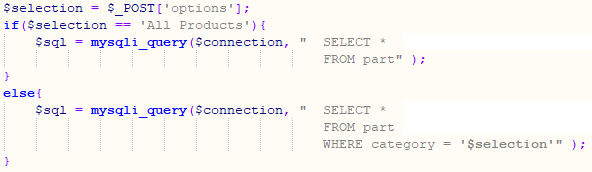
The products.php page allows customers and administrators to browse the products offered by CS Tools. It can display all the products at once, or display only the products of a specified category. Product categories are selected via a drop-down list and the results are displayed in a table.



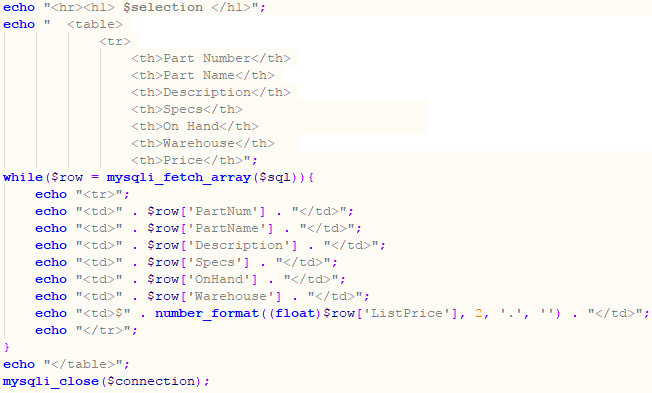
The drop-down list’s html:



The code stores the value of the selection into a variable and queries the database based on that variable.

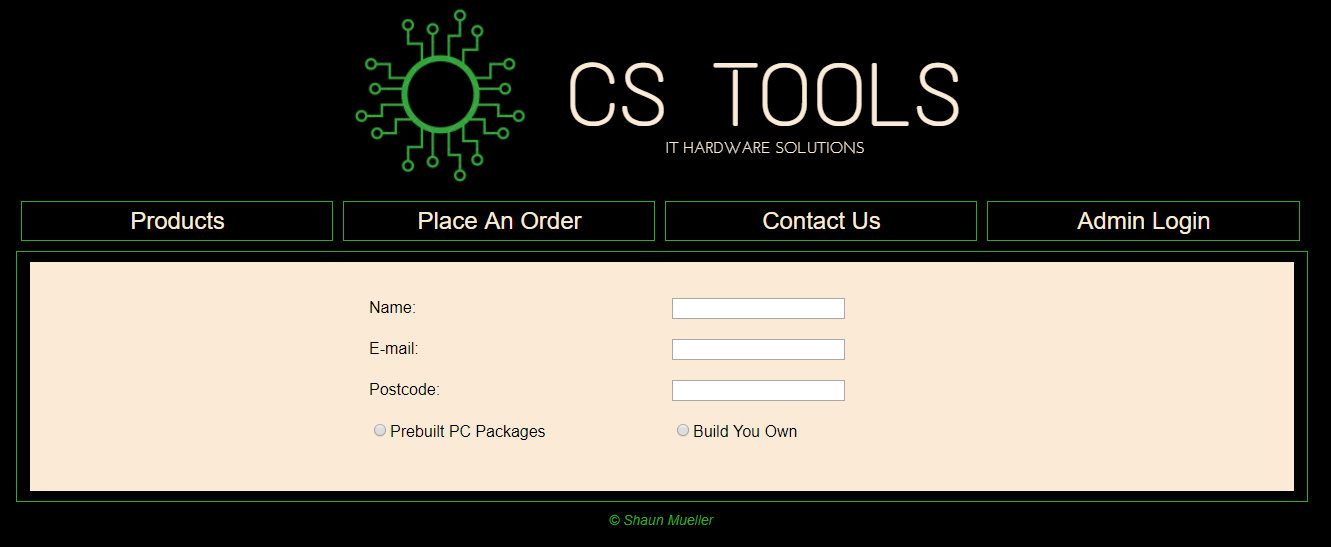


The php then extracts the part table’s data and displays it in a table before closing the database connection.

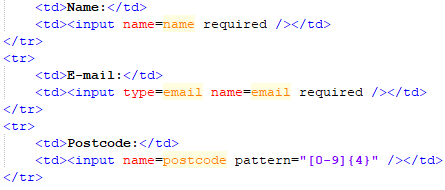


# Place An Order Page (customerRequest.php)

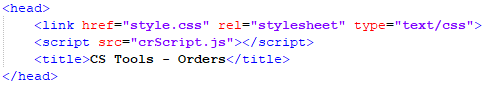
The Place An Order page allows customers to fill out a form to express their interest in making a purchase.



The name, email, and postcode fields use data validation to gather the correct data.

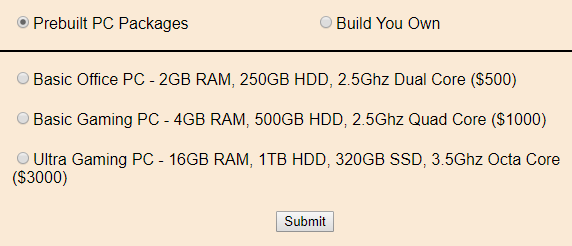


This page uses a JavaScript file called crScript.js, therefore the html needs to be made aware of it.

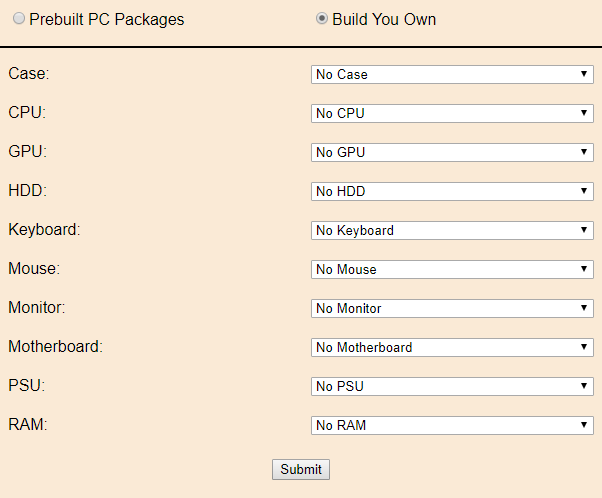


The java script hides the remainder of the form until one of the radio buttons is selected.

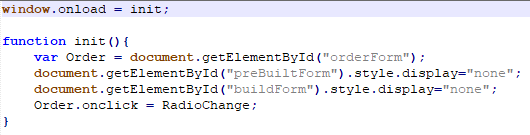
When ‘Prebuilt PC Packages’ is selected:



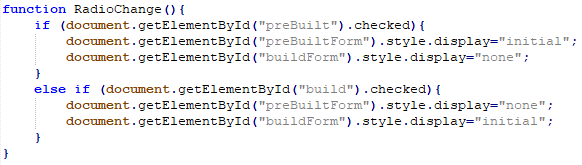
When ‘Build Your Own’ is selected:



The JavaScript file will initially hide the form when the page loads.

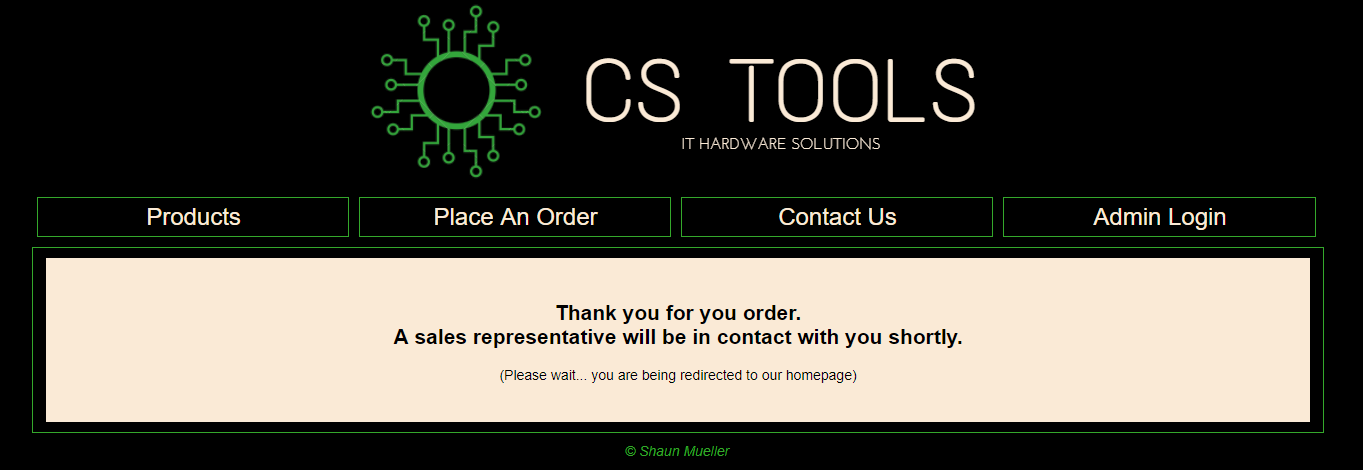


When a radio button is checked, the JavaScript will display the appropriate form.



When the user clicks the submit button, they are forwarded to a thank you page (orderThankyou.php).



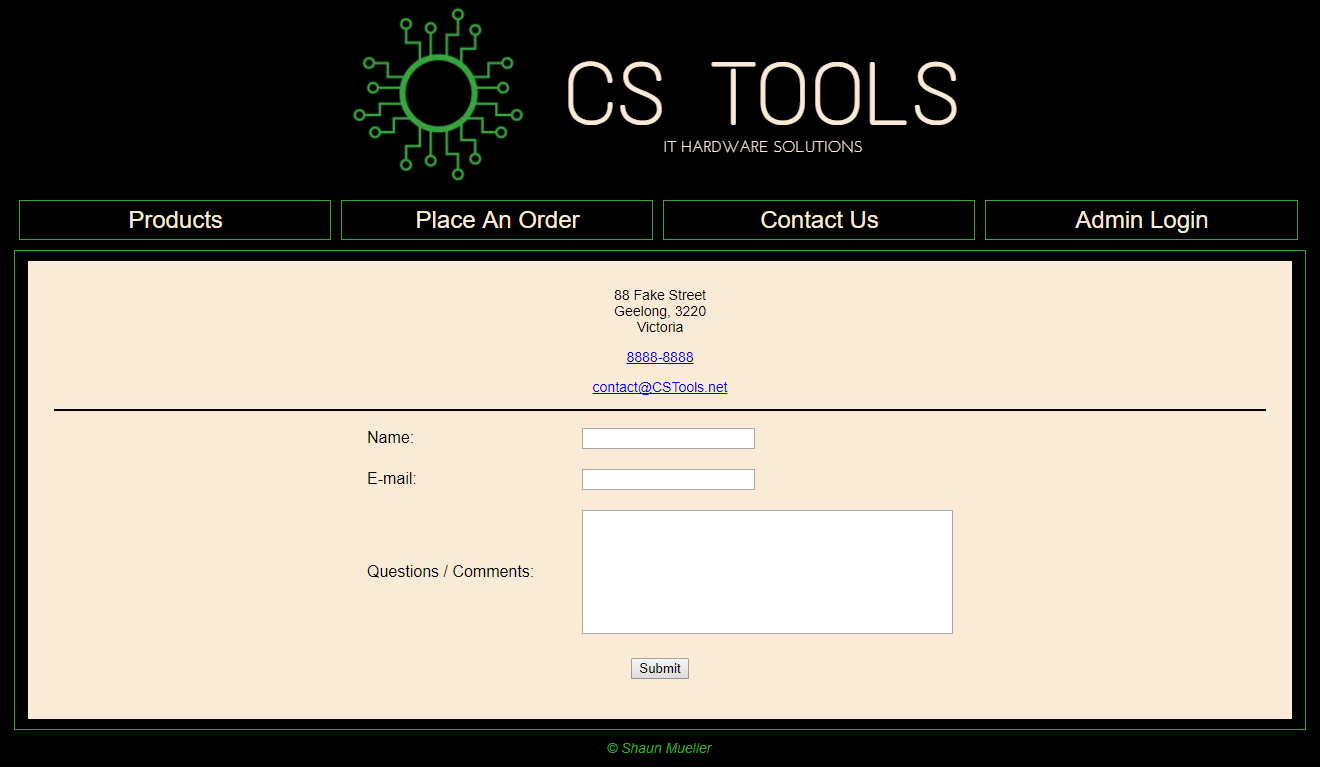


The user will sit on the thankyou page for 5 seconds, before being redirected to the index page.



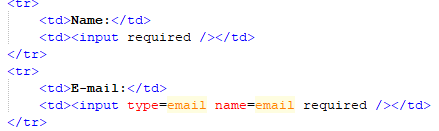
# Contact Us Page (contact.php)

The ‘Contact Us’ page allows the user to fill in a form with a question or comment.



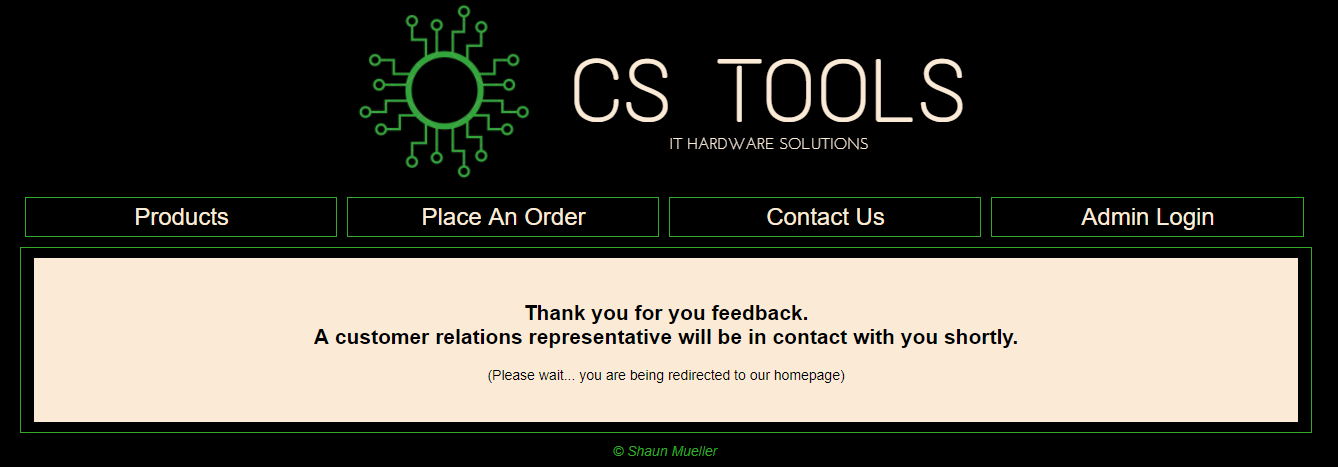
This page is quite similar to the ‘Place An Order’ page.

The name and email fields use data validation:



When the user clicks the submit button, they are forwarded to a thank you page (contactThankyou.php).





The user will sit on the thankyou page for 5 seconds, before being redirected to the index page.

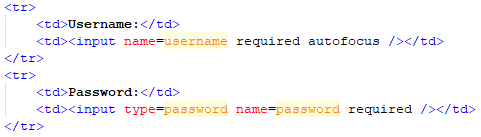


# Admin Login Page (admin\_Login.php)

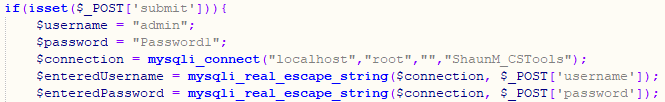
This page allows administrators with the correct credentials to log in to the site and access additional features.



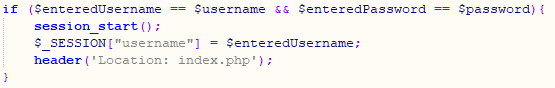
The username and password fields are both required and the password field will not display the inputs in clear text.



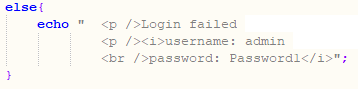
When the log in button is clicked, the input is stored into variables.



If the input matched the username and password, a session is started and the user is redirected to the index page.



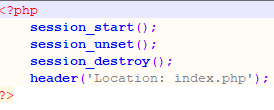
If the input does not match the username and password, then an error message is displayed.



# Logging out (logout.php)

There is no html page for logging out, clicking the ‘Log Out’ link simply runs the logout.php code.

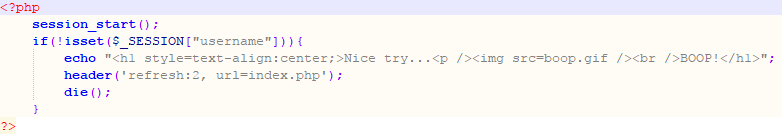
This piece of code is very simple. It resumes the current session, frees all session variables, destroys all data registered to the session, and then redirects the user to the index page.



# Restricting Access To Admin Pages

Each admin page (except the login) includes some code to ensure a user cannot gain access to it without going through the login procedure first.

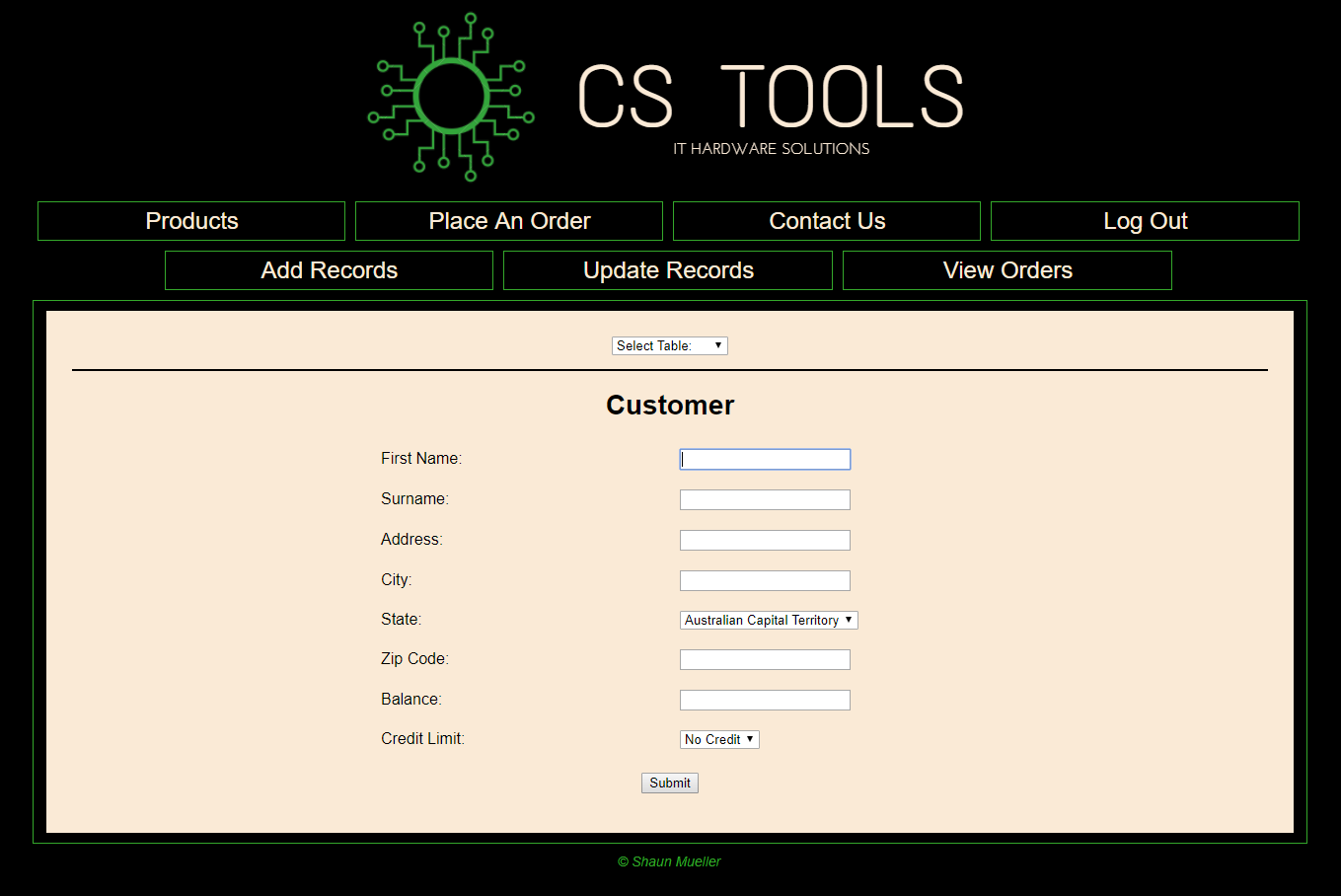




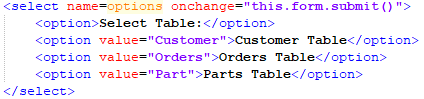
The code resumes any existing sessions, then if there is no existing session a short html page is displayed for 2 seconds, before the user is redirected to the index page. The remaining code on the page will not be executed.

# Add Records Page (admin\_AddRecords.php)

This page allows an administrator to add entries into the database tables via a web interface.



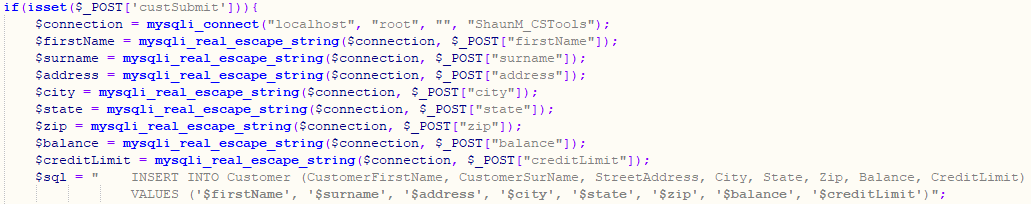
The administrator selects which table to add the entries to from a drop-down list.



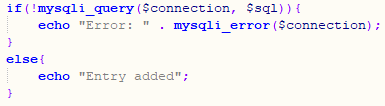
When this selection is made, the appropriate from will be displayed.



When the submit button is pressed, a connection to the database is opened, all inputs are stored into variables, and the data within those variables is added into the database table.

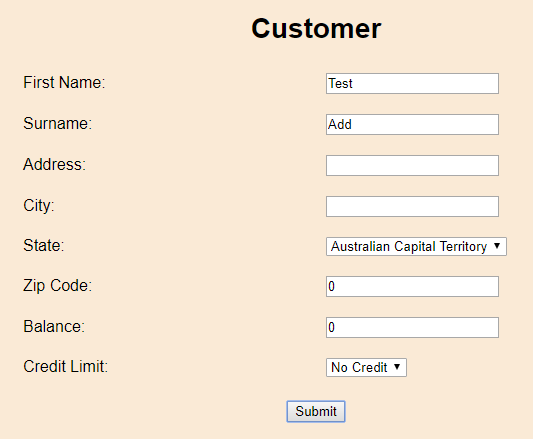


If the data cannot be entered into the database table, an error message will be displayed, otherwise it will display “Entry added”.

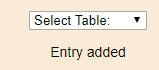


**VERIFYING**

Let’s test the functionality by adding a new customer named “Test Add” to the ‘Customer’ table.



After pressing the submit button:



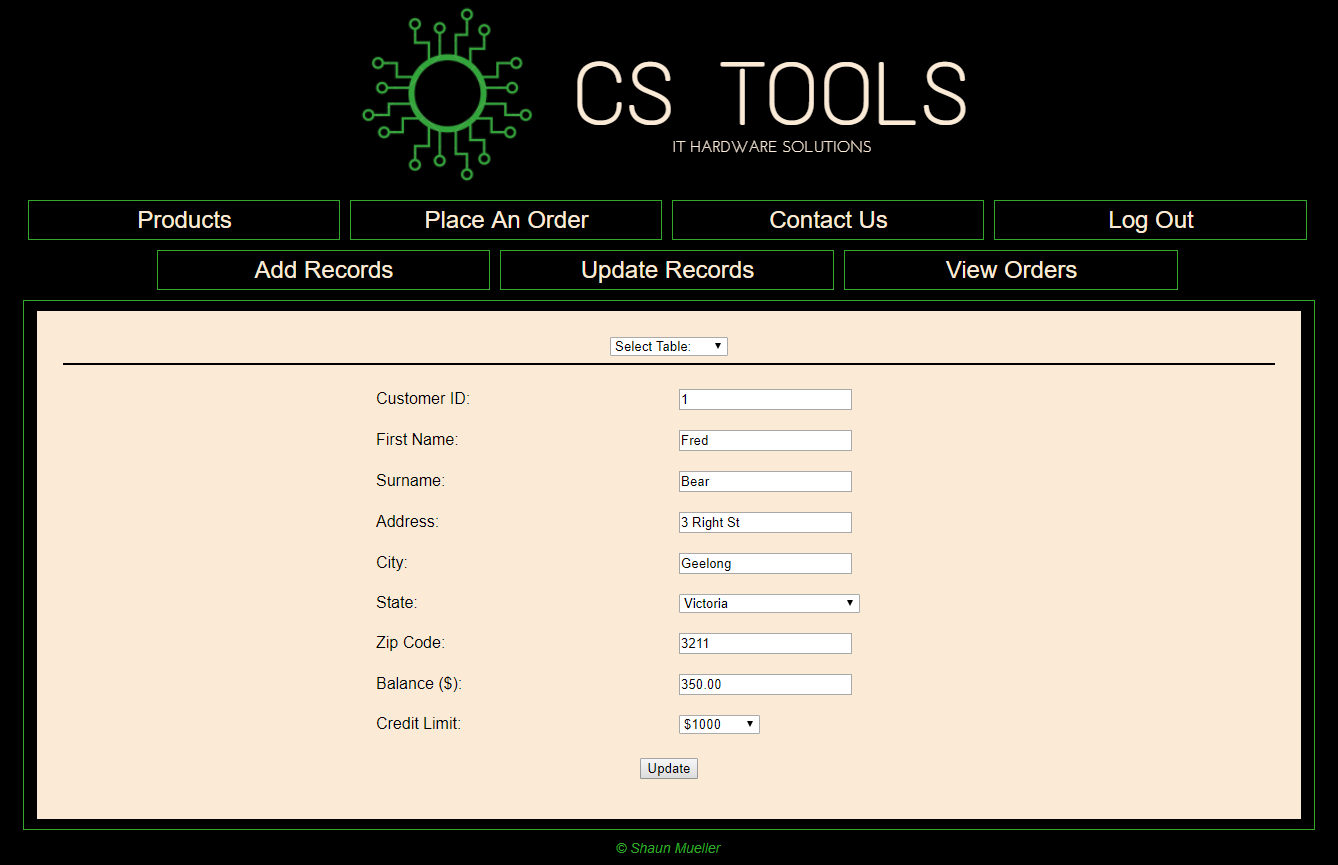
phpMyAdmin:



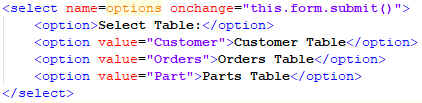
The entry was successfully added to the ‘Customer’ table in the database.

# Update Records (admin\_UpdateTables.php)

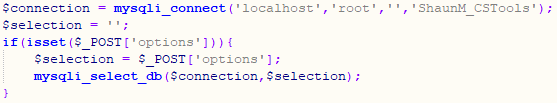
This page allows an administrator to update an entry of the database tables via a web interface.



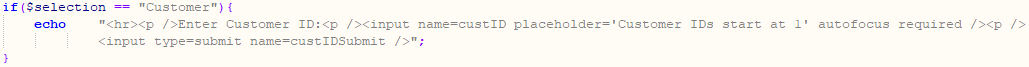
The administrator will first choose which table they want to update the entry of from a drop-down list.



When the administrator chooses a table from the drop-down list, the code will connect to the database and select the specified table.

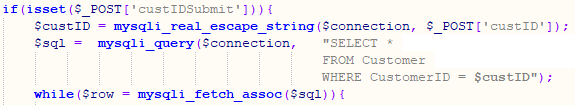
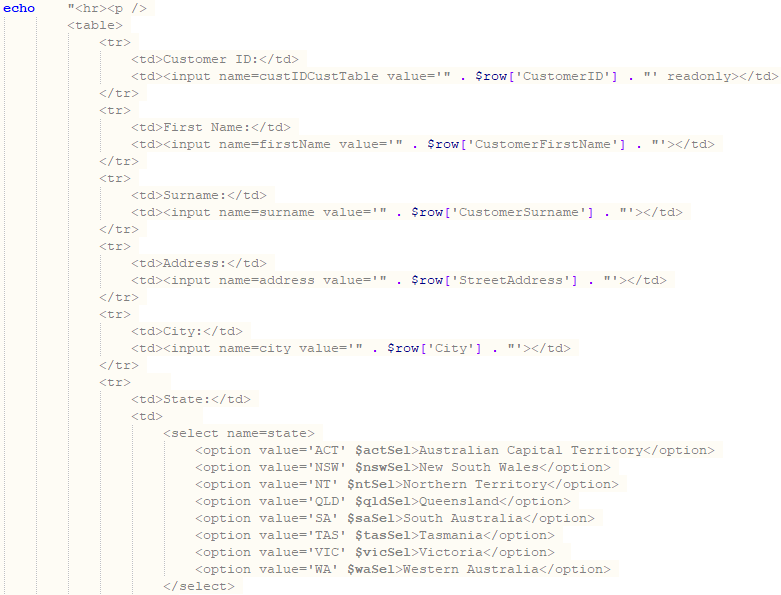


The administrator will be prompted to enter a value unique to the specified table, for example the Customer ID the unique value from the customer table (these are the primary keys).

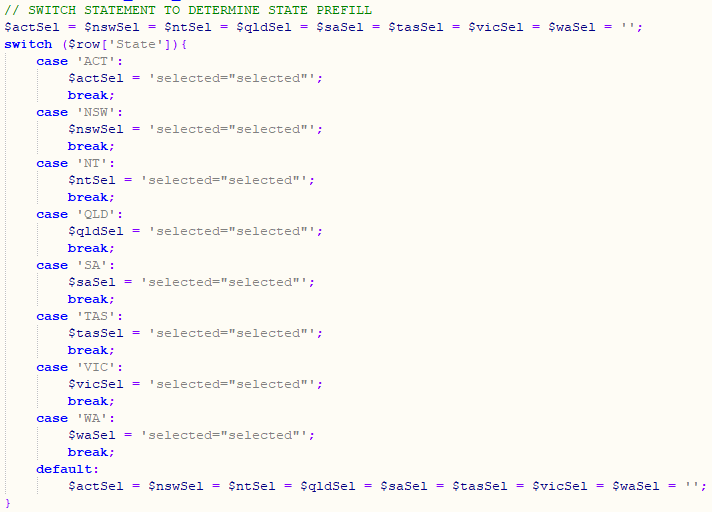




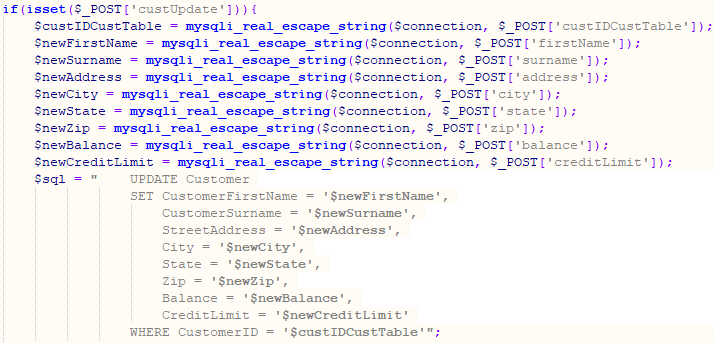
When the unique value is input and the submit button pressed, a new form will be displayed with all the entry’s data prefilled.

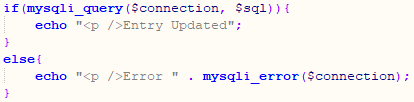
Having the drop-down lists and radio buttons prefill require using switch statements to select the correct option.



When the update button is pressed, the code stores all the values from the fields into variables, and updates the entry with those variables.

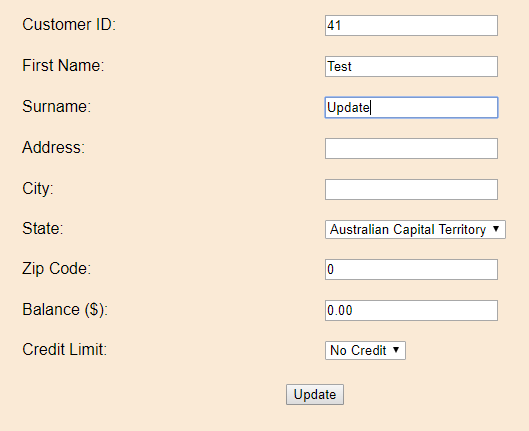


If the data cannot be entered into the database table, an error message will be displayed, otherwise it will display “Entry Updated”.



**VERIFYING**

Let’s test the functionality by updating the previously added customer (ID #41). Let’s rename them from “Test Add” to “Test Update”.



After pressing the update button:



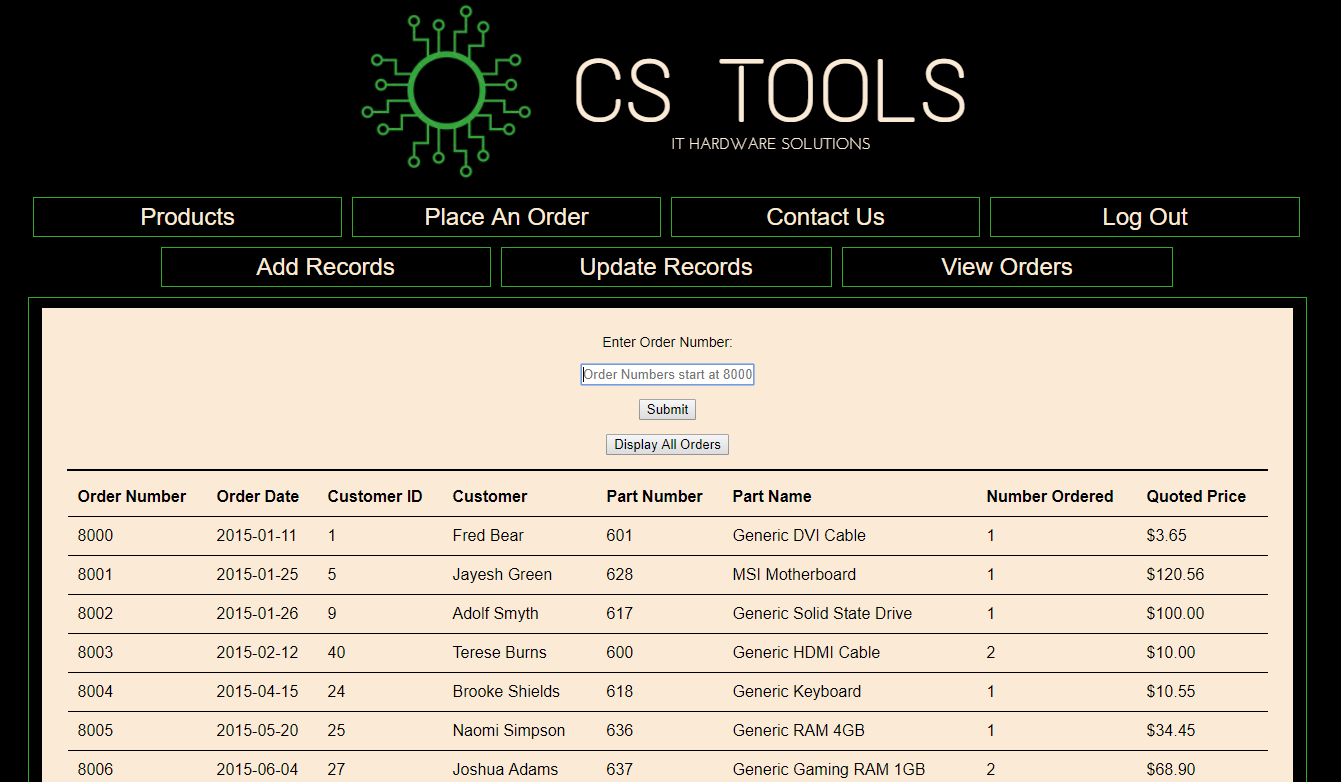
phpMyAdmin:



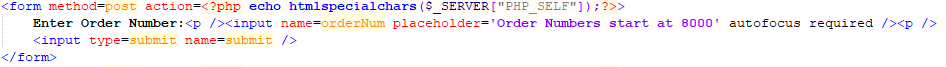
The entry was successfully updated in the database.

# View Orders Page (admin\_ViewOrders.php)

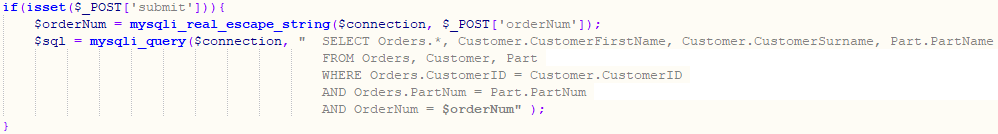
This page allows an administrator to view orders in a table. The data comes from all three tables of the database (Customer, Part, and Orders). Administrators can enter the order number to view a specific order.



This page features a form which asks for an order number.



When the submit button is pressed, the code connects to and queries the database based on the entered order number.



The data is then displayed in a table and the connection to the database is closed.

