Simple Web Store

A Report on the PHP Coursework

By Nida Farooqui | H00166409 | naf31@hw.ac.uk

Table of Contents

Introduction 2
Requirements' Checklist 2
Design Considerations 3
User Guide 4
Developer Guide 9
Testing 14
Conclusions 15

Introduction

The purpose of this report is to demonstrate an online simple web store and explain its functionality. This web store allows a user to register an account, login, search for books, view book details, and add books to a shopping cart. This coursework was implemented in PHP and HTML with a little bit of CSS. The online link to this coursework is http://www2.macs.hw.ac.uk/~naf31/naf31/.

Requirements' Checklist

Requirements	
Home: This is the front page of the site. For this page, the user should specify whether he/she is a customer or not. If a customer, a customer id and password are required to be passed for this page. The returned HTML document should contain customer details and a link to the Search Request page.	✓
Search Request: This page should allow the user to enter a search criterion (search by name, by author, by topic) and a value in a form.	✓
Search Result: The reply of the search request should be provided by a search result page, which requires a search criterion and a value. The HTML document returned should contain links to Search Request and Product Detail.	✓
Product Detail: This web page returns the details of a particular book. An item id parameter is required for this interaction. The HTML document returned should contain links to Search Request and Shopping Cart.	✓
Shopping Cart: A cart should be associated with each user session. In this web page, such a cart is updated. If no cart is associated with user session, one should be created in this interaction. New items can be added to the cart or existing items updated. The function interface to the cart should be designed in such a way that a list of (i_id, i_qty) pairs is passed to the main function updating the contents of the cart, where the first component of the pair is the identity of the item to add/update and the second component is the quantity to add/update. Additionally, the function interface should provide a parameter add flag, specifying whether the quantity should be added or updated in the shopping cart. The HTML document returned contains the updated cart and includes links to Shopping Cart and Search Request.	All requirements are met in this except that the shopping cart does not add the items dynamically

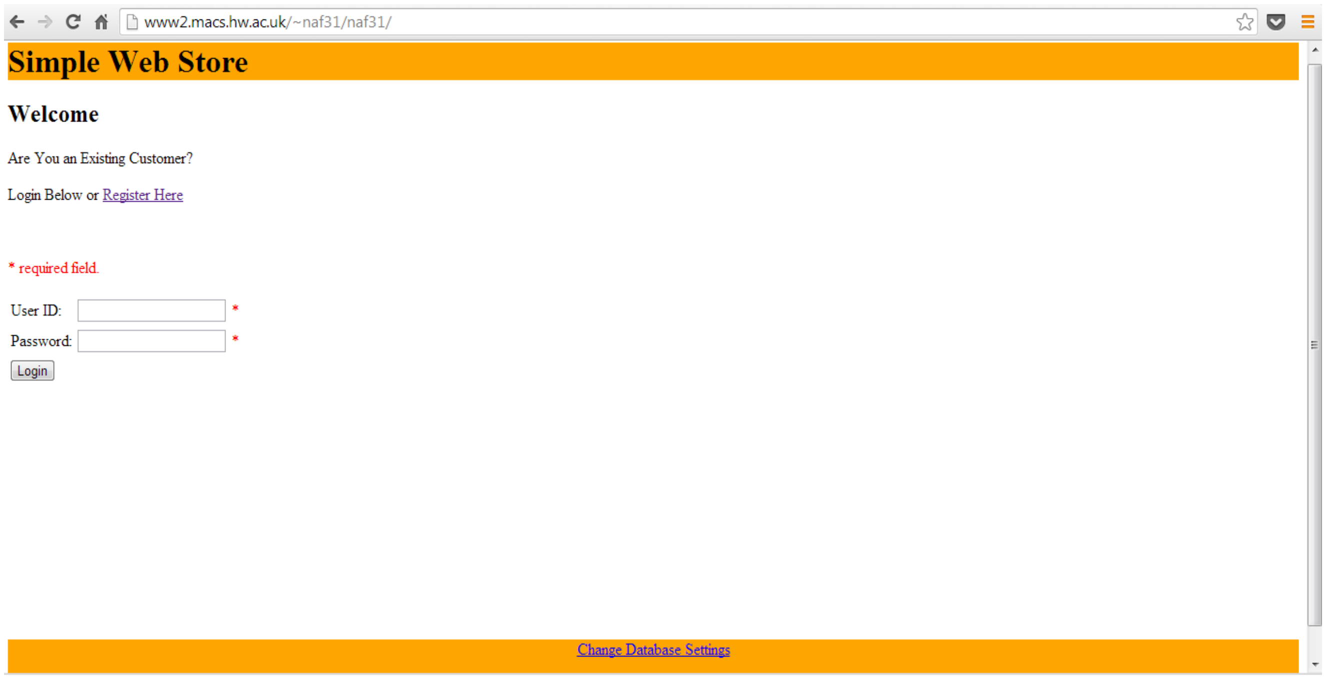
DB interface: The application should provide an interface to change the settings for the database connection (server, user-name, db-name, table-names).	✓

Design Considerations

There are 12 PHP files created for this coursework. To make the application more usable, CSS styling has been added. The search bar is placed in the center when searching for a book so it is more visible and the use can write large search queries. Form validation has been checked everywhere, so if a user enters a wrong user id or password or if he/she exceeds the characters or even if the search is not found a message is displayed. Also, this form validation has in some cases tried to prevent SQL injection. A header and a footer is given in every page, where the footer always contains a link for the user to change the database settings if he needs to. An '*' mark is used in places where the form cannot be submitted without the user entering data. Also, in the database settings, to make it easy for the user, the current database settings is displayed to the user on screen but he/she can edit it if he/she wants to change the settings. A separate file 'conntodb.php' is used and a function conntostore() is written to it which gets called every time the database connection is needed rather than writing the connection code each time in every file. That file also contains a function that closes the connection and that is only called once the user clicks on log out. A Log Out link is given in every page so the user can log out whenever he/she wants to. Also the log out page after closing the connection to the database and destroying the user session redirects the page to the login page. There is also a message telling the user has logged in once the login is successful. The password field is hidden to the user and is not saved in any session variable in the PHP code for security purposes. All text fields are also neatly aligned. The 'Change Database Settings' link was chosen to be placed at the bottom on every page to avoid less clutter on the web page and make it look neater. Also in some places the tables were given a background colour and shaded to make them look presentable. Also the sql query for the search bar was written in such a way to be able to search for items by not writing the complete name but this did not work.

User Guide

To access the online book store, click on http://www2.macs.hw.ac.uk/~naf31/naf31/. You will see the login page below.


www2.macs.hw.ac.uk/~naf31/naf31/

Simple Web Store

Welcome

Are You an Existing Customer?

Login Below or Register Here

* required field.

User ID: *

Password: *

Login

Change Database Settings

There are currently 2 users in the database.

UserID = 1

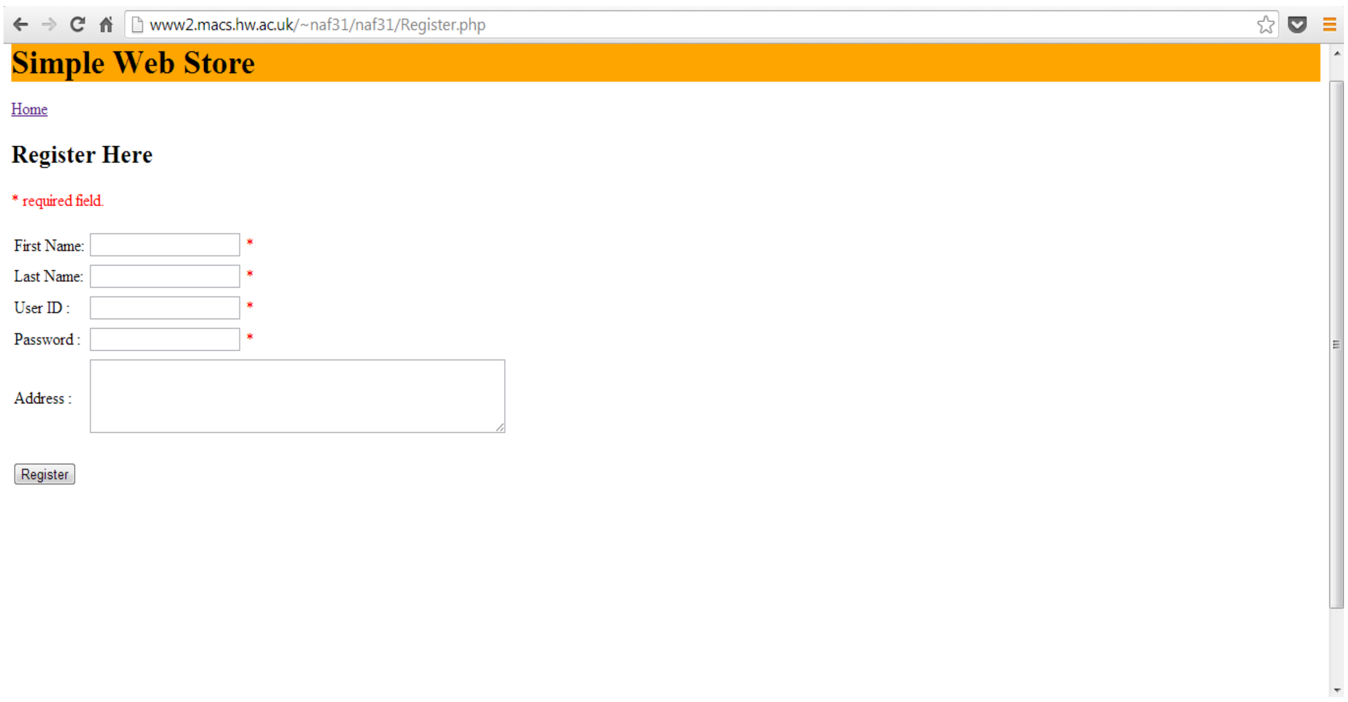
Password = f21sc_13

And

Userid =2

Password = 12ab

You can use any of the login information above to login or click on register to register a new account. You must enter all the fields, though the address field is optional.



www2.macs.hw.ac.uk/~naf31/naf31/Register.php

Simple Web Store

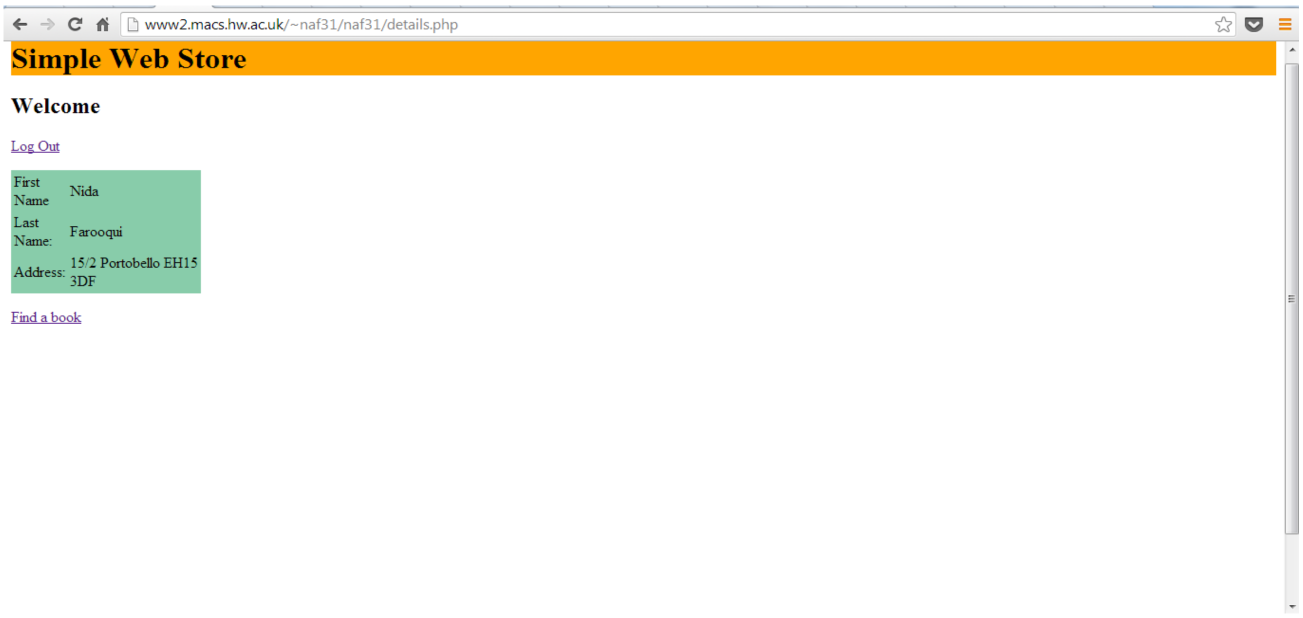
Home

Register Here

* required field.

First Name:	*
Last Name:	*
User ID :	*
Password :	*
Address :	

Once you fill in the details click on register and your account will be created and you will be directed to the details.php page which will give you show you your user details and provide a link to find a book or to log out.



www2.macs.hw.ac.uk/~naf31/naf31/details.php

Simple Web Store

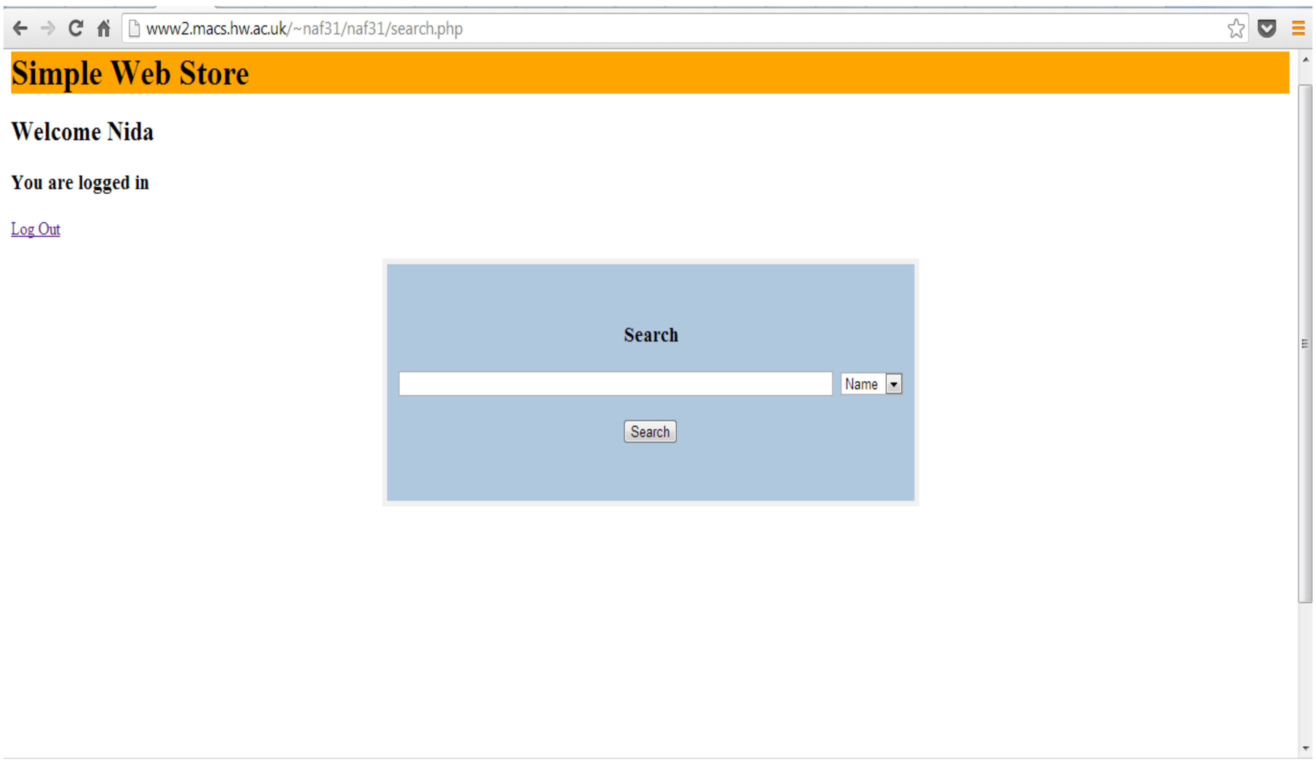
Welcome

Log Out

First Name	Nida
Last Name:	Farooqui
Address:	15/2 Portobello EH15 3DF

Find a book

If you click on Find a book, you are sent to the search.php page as shown below.


www2.macs.hw.ac.uk/~naf31/naf31/search.php

Simple Web Store

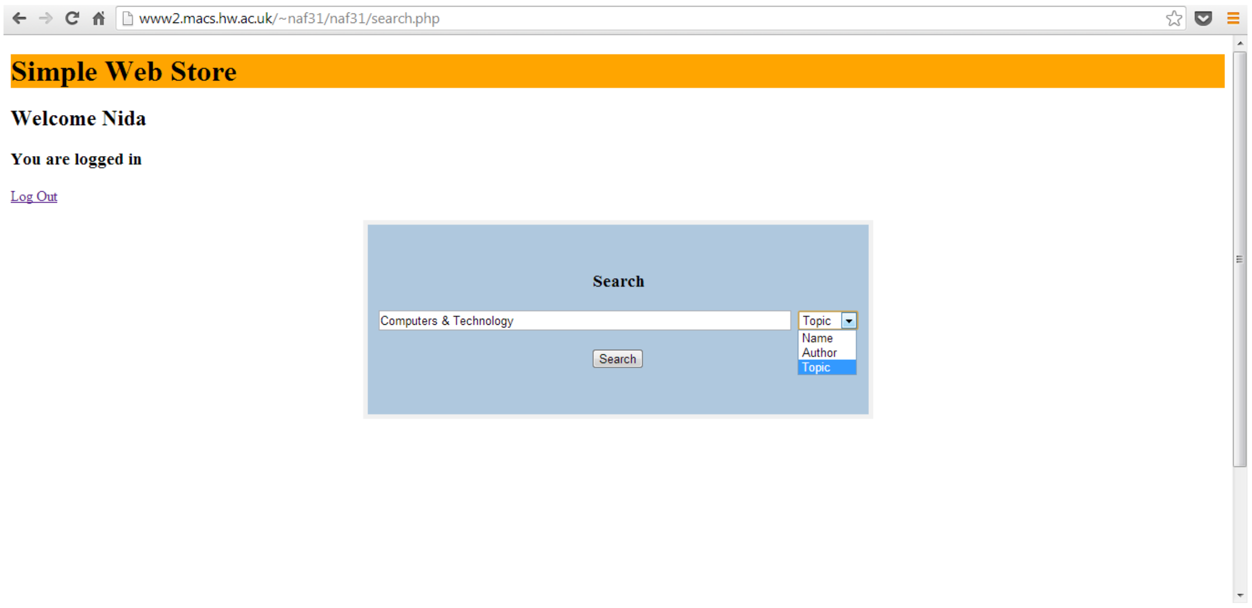
Welcome Nida

You are logged in

Log Out

Search

Here the user can find a book by three categories in the drop down menu, through name, topic or author, where name is the book's name, author is the author's name and topic is searching by subject. If a user enters the topic name and if it exists in the database system a search result page will be returned showing a link to product details, otherwise an error message will appear saying "No such book found"


www2.macs.hw.ac.uk/~naf31/naf31/search.php

Simple Web Store

Welcome Nida

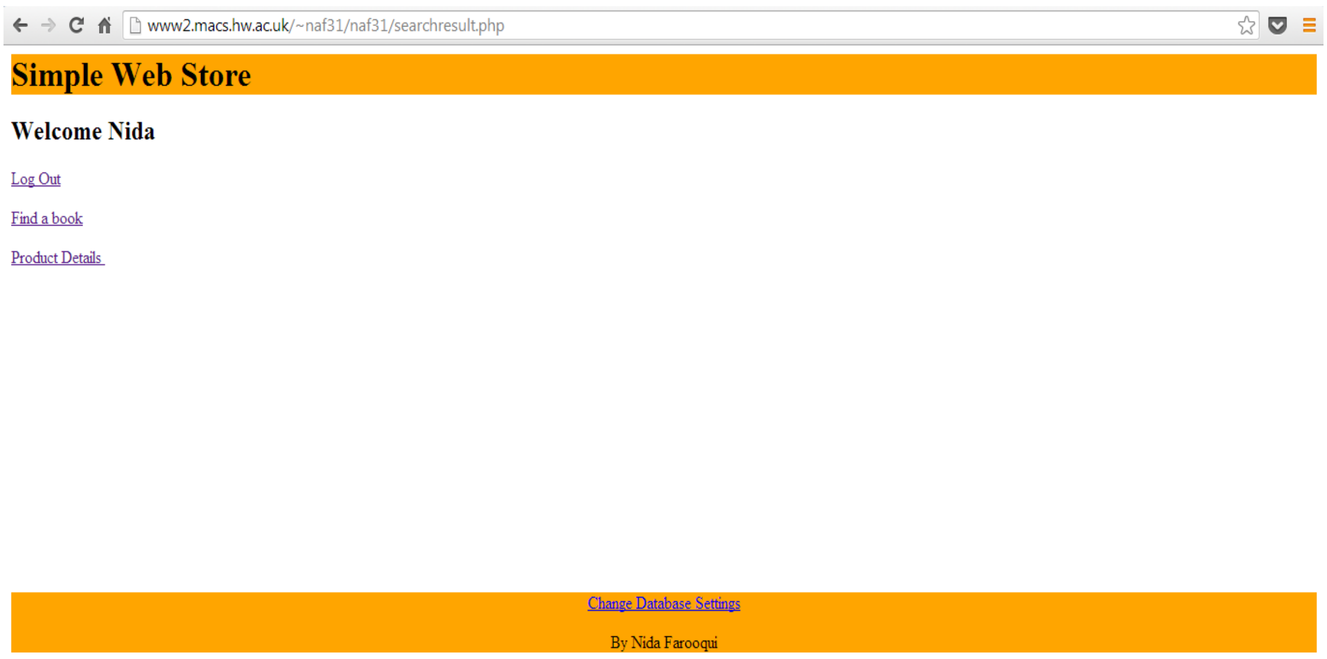
You are logged in

Log Out

Search

Computers & Technology

Topic



www2.macs.hw.ac.uk/~naf31/naf31/searchresult.php

Simple Web Store

Welcome Nida

Log Out

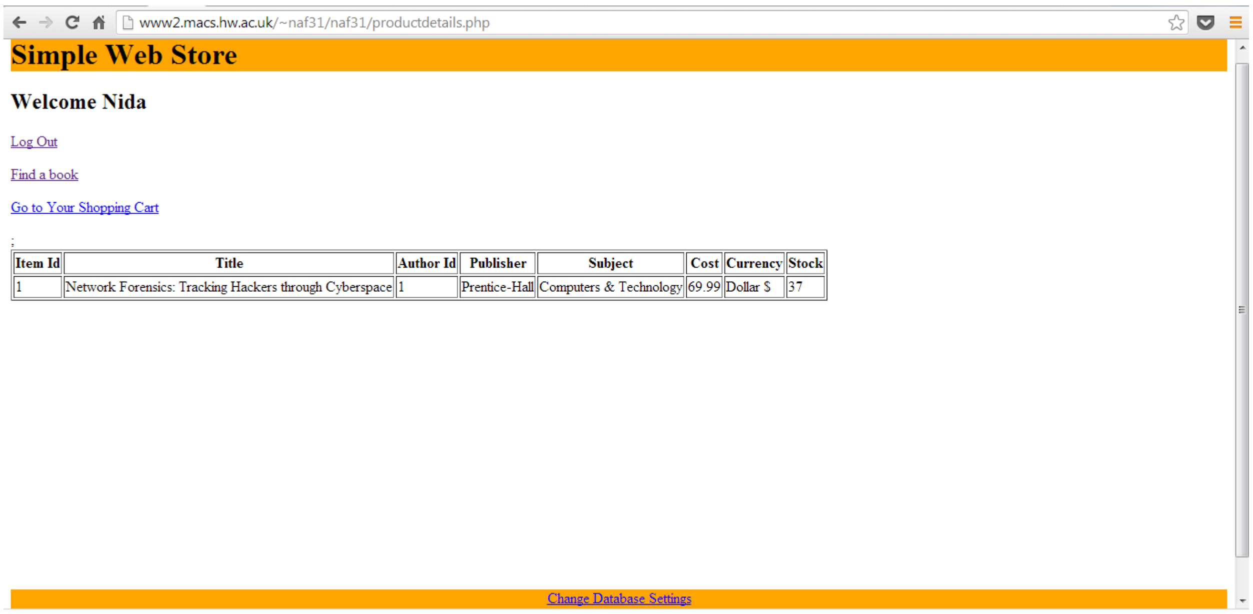
Find a book

Product Details

Change Database Settings

By Nida Farooqui

If the user clicks on Product Details he/she is directed to another page in which the details of the product are shown.



www2.macs.hw.ac.uk/~naf31/naf31/productdetails.php

Simple Web Store

Welcome Nida

Log Out

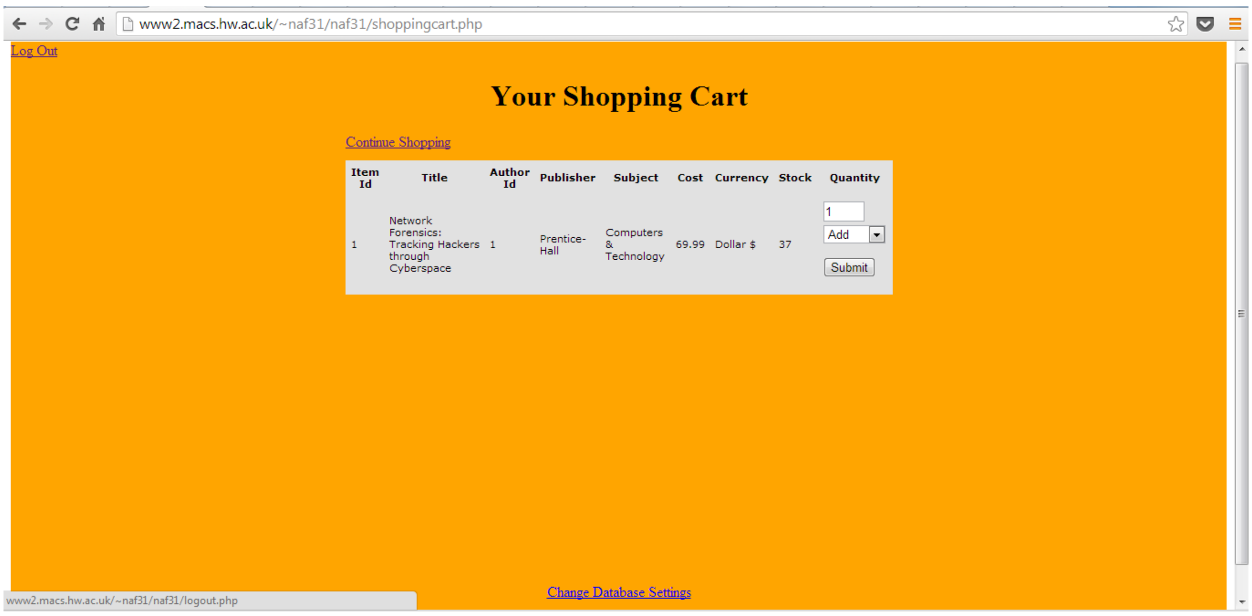
Find a book

Go to Your Shopping Cart

Item Id	Title	Author Id	Publisher	Subject	Cost	Currency	Stock
1	Network Forensics: Tracking Hackers through Cyberspace	1	Prentice-Hall	Computers & Technology	69.99	Dollar $	37

Change Database Settings

The user can now click on shopping cart to be directed to a page where he/she has the option of selecting the number of items she wants to buy.


www2.macs.hw.ac.uk/~naf31/naf31/shoppingcart.php

Log Out

Your Shopping Cart

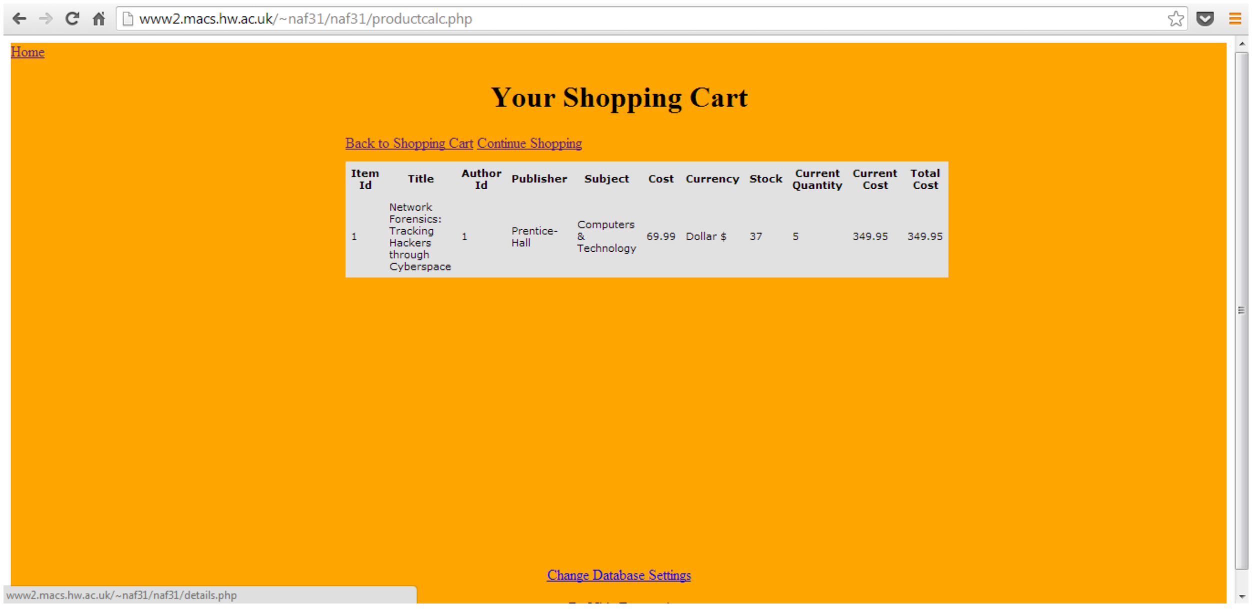
Continue Shopping

Item Id	Title	Author Id	Publisher	Subject	Cost	Currency	Stock	Quantity
1	Network Forensics: Tracking Hackers through Cyberspace	1	Prentice-Hall	Computers & Technology	69.99	Dollar $	37	1 Add Submit

Change Database Settings

www2.macs.hw.ac.uk/~naf31/naf31/logout.php

The quantity of items can be selected and there is also an option which states whether the user is adding another item or updating the item to the quantity he/she is giving above. So if a user wants to buy 5 books and chooses add and then clicks on submit the following page appears.


www2.macs.hw.ac.uk/~naf31/naf31/productcalc.php

Home

Your Shopping Cart

Back to Shopping Cart Continue Shopping

Item Id	Title	Author Id	Publisher	Subject	Cost	Currency	Stock	Current Quantity	Current Cost	Total Cost
1	Network Forensics: Tracking Hackers through Cyberspace	1	Prentice-Hall	Computers & Technology	69.99	Dollar $	37	5	349.95	349.95

Change Database Settings

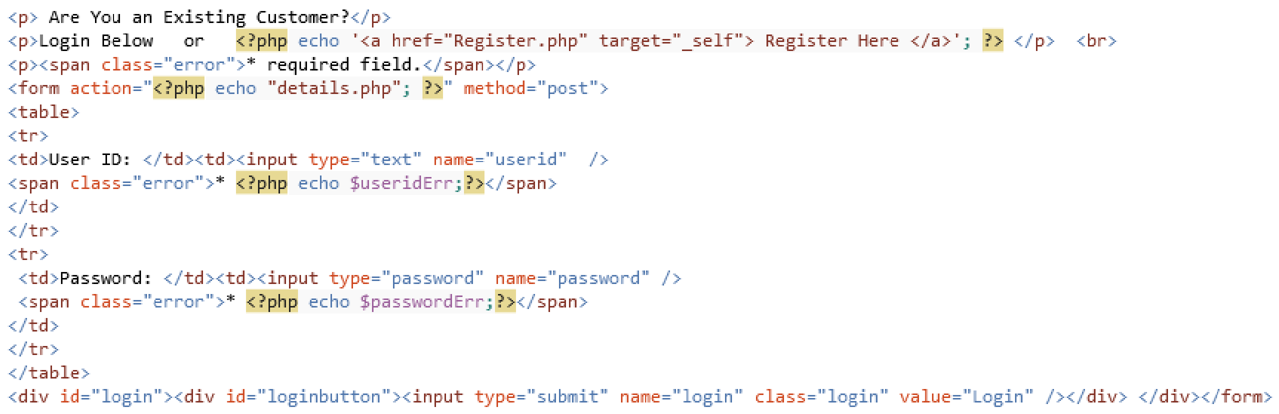
www2.macs.hw.ac.uk/~naf31/naf31/details.php

The user can now click on back to shopping cart to add more items or click on Continue Shopping to be directed back to the search page to continue searching for a book.

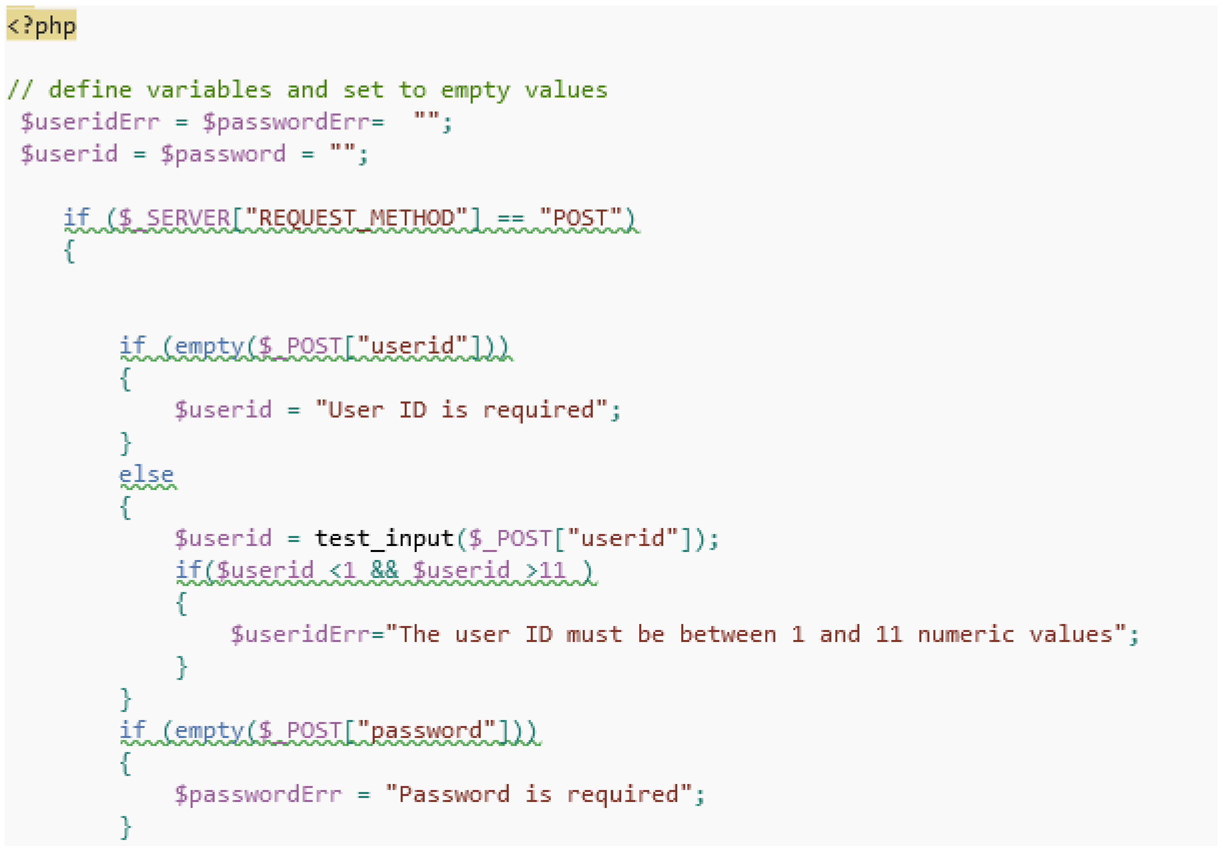
The current cost shows the cost of the book/books currently selected by the user and the total cost represents the cost of all the items the user has added to the shopping cart.

Developer Guide

The index page starts the session, displays a form in html and checks for form validation in php while posting the form once submitted to the details.php page.



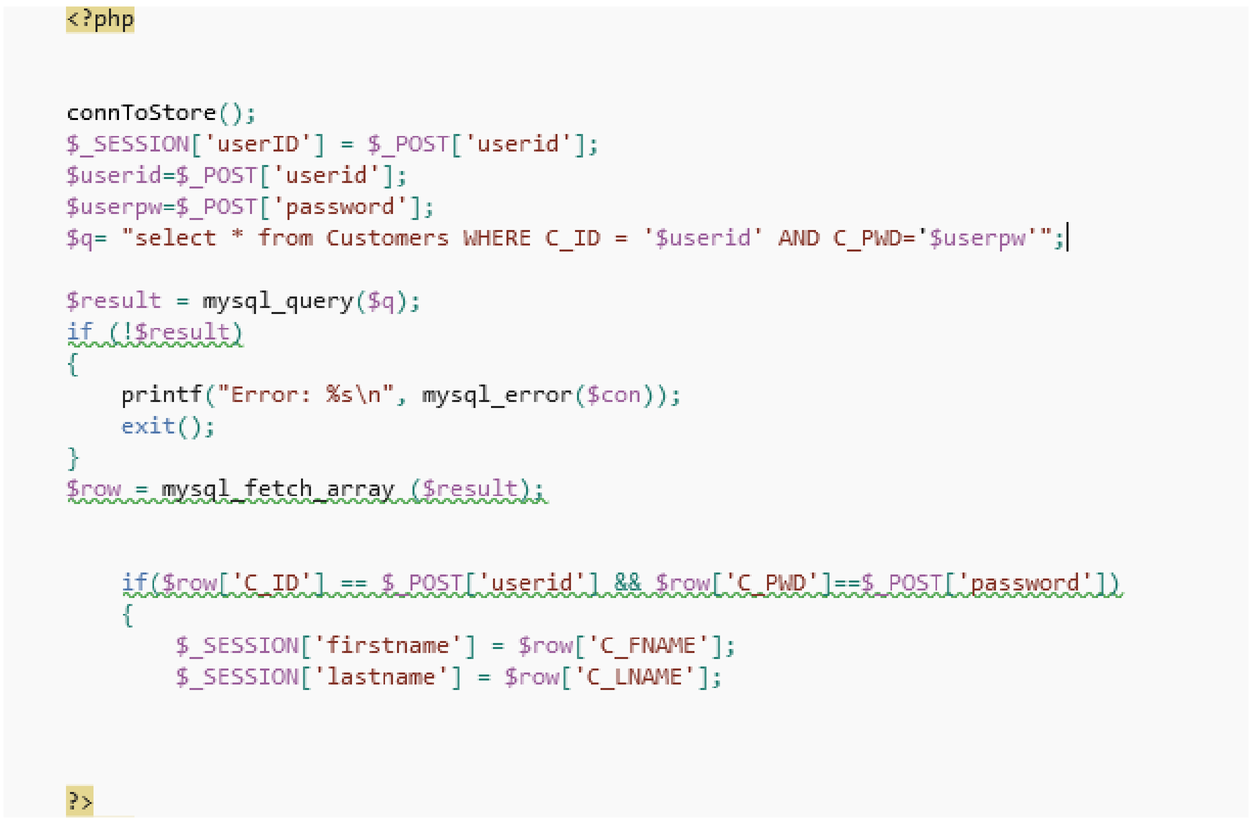
```
<p> Are You an Existing Customer?</p>
<p>Login Below   or   <?php echo '<a href="Register.php" target="_self"> Register Here </a>'; ?> </p>  <br>
<p><span class="error">* required field.</span></p>
<form action="<?php echo "details.php"; ?>" method="post">
<table>
<tr>
<td>User ID: </td><td><input type="text" name="userid"  />
<span class="error">* <?php echo $useridErr;?></span>
</td>
</tr>
<tr>
 <td>Password: </td><td><input type="password" name="password" />
 <span class="error">* <?php echo $passwordErr;?></span>
</td>
</tr>
</table>
<div id="login"><div id="loginbutton"><input type="submit" name="login" class="login" value="Login" /></div> </div></form>
```



```
<?php

// define variables and set to empty values
 $useridErr = $passwordErr=  "";
 $userid = $password = "";

    if ($_SERVER["REQUEST_METHOD"] == "POST")
    {


        if (empty($_POST["userid"]))
        {
            $userid = "User ID is required";
        }
        else
        {
            $userid = test_input($_POST["userid"]);
            if($userid <1 && $userid >11 )
            {
                $useridErr="The user ID must be between 1 and 11 numeric values";
            }
        }
        if (empty($_POST["password"]))
        {
            $passwordErr = "Password is required";
        }
```

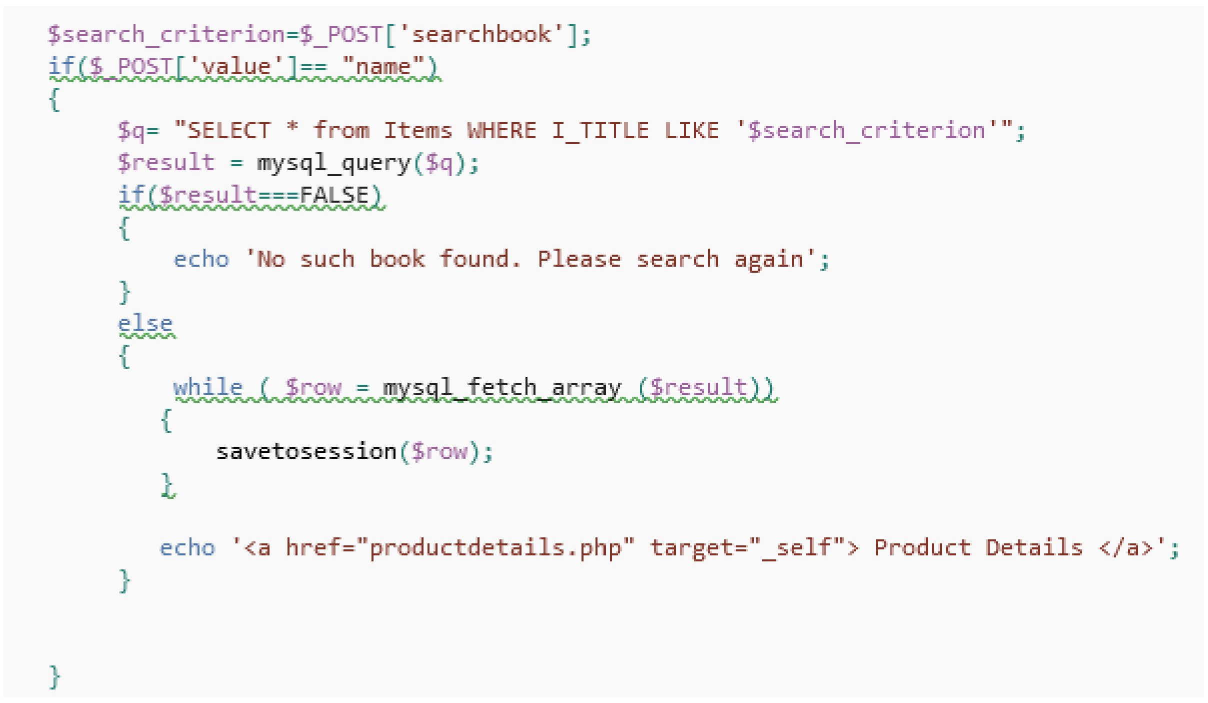
Shown above is just a snapshot of a part of the code and not the entire code itself.

The details page accesses the details of the user from the database and saves it into a session array and then echoes it in html code.



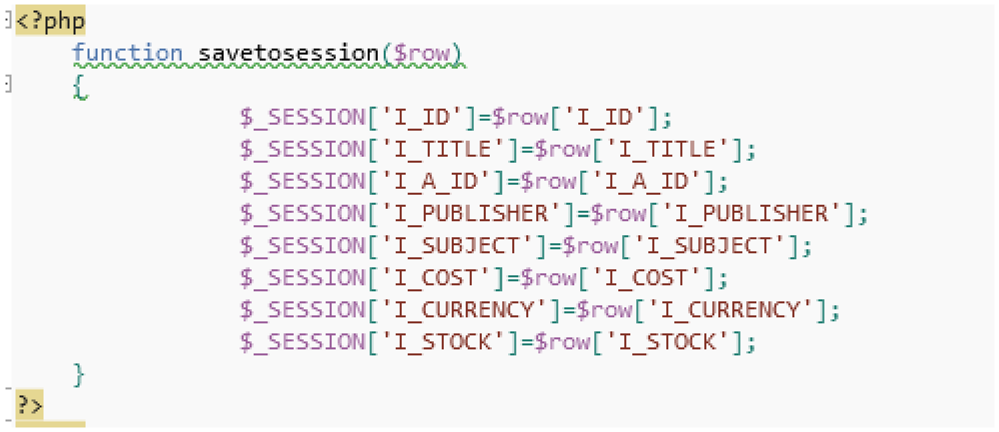
```
<?php


connToStore();
$_SESSION['userID'] = $_POST['userid'];
$userid=$_POST['userid'];
$userpw=$_POST['password'];
$q= "select * from Customers WHERE C_ID = '$userid' AND C_PWD='$userpw'";

$result = mysql_query($q);
if (!$result)
{
    printf("Error: %s\n", mysql_error($con));
    exit();
}
$row = mysql_fetch_array ($result);


    if($row['C_ID'] == $_POST['userid'] && $row['C_PWD']==$_POST['password'])
    {
        $_SESSION['firstname'] = $row['C_FNAME'];
        $_SESSION['lastname'] = $row['C_LNAME'];



?>
```

The search.php is written mainly in html and the form submitted from there goes to the searchresult.php page. This page checks the search criterion entered to see whether it is a name, topic or author and then searches the database for such a criterion. If none found an error message is displayed to the screen.



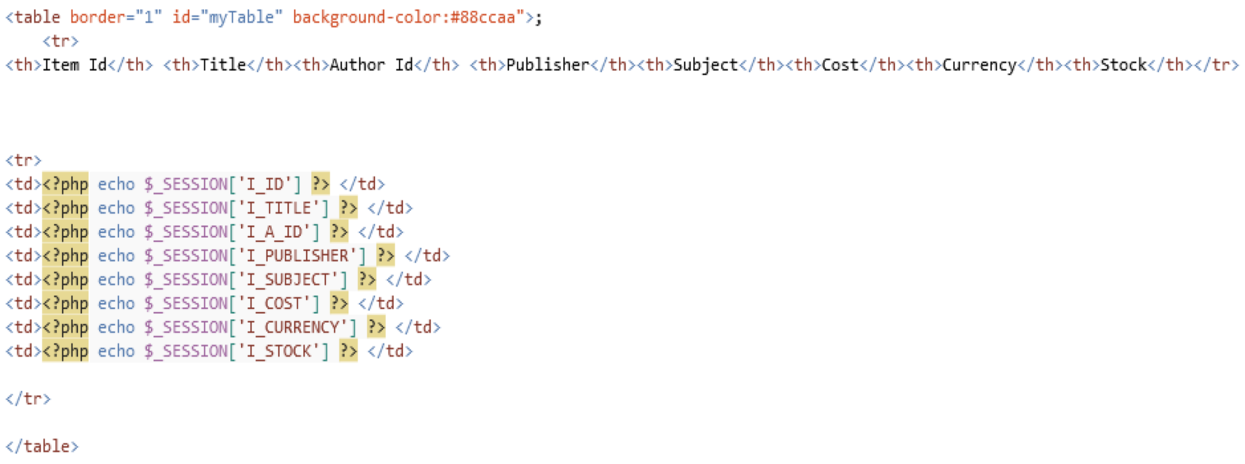
```
$search_criterion=$_POST['searchbook'];
if($_POST['value']== "name")
{
     $q= "SELECT * from Items WHERE I_TITLE LIKE '$search_criterion'";
     $result = mysql_query($q);
     if($result===FALSE)
     {
         echo 'No such book found. Please search again';
     }
     else
     {
         while ( $row = mysql_fetch_array ($result))
        {
            savetosession($row);
        }

        echo '<a href="productdetails.php" target="_self"> Product Details </a>';
     }


}
```

If the book details are found, they are saved to session variables to be accessed later.



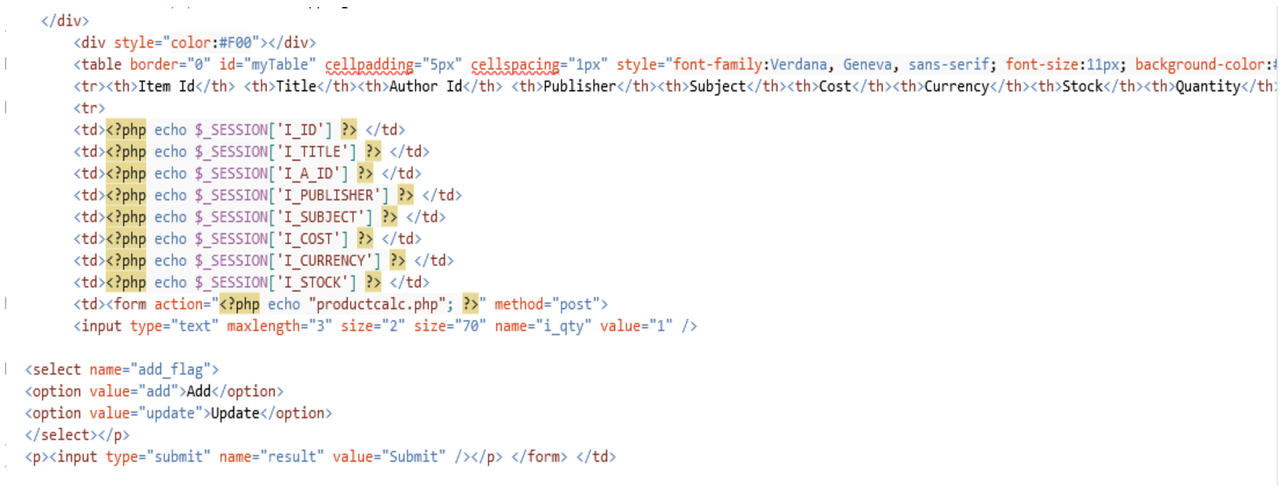
```
<?php
    function savetosession($row)
    {
                $_SESSION['I_ID']=$row['I_ID'];
                $_SESSION['I_TITLE']=$row['I_TITLE'];
                $_SESSION['I_A_ID']=$row['I_A_ID'];
                $_SESSION['I_PUBLISHER']=$row['I_PUBLISHER'];
                $_SESSION['I_SUBJECT']=$row['I_SUBJECT'];
                $_SESSION['I_COST']=$row['I_COST'];
                $_SESSION['I_CURRENCY']=$row['I_CURRENCY'];
                $_SESSION['I_STOCK']=$row['I_STOCK'];
    }
?>
```

The productdetails.php is also mostly written in html and echoes the session variables to the user in a table form.



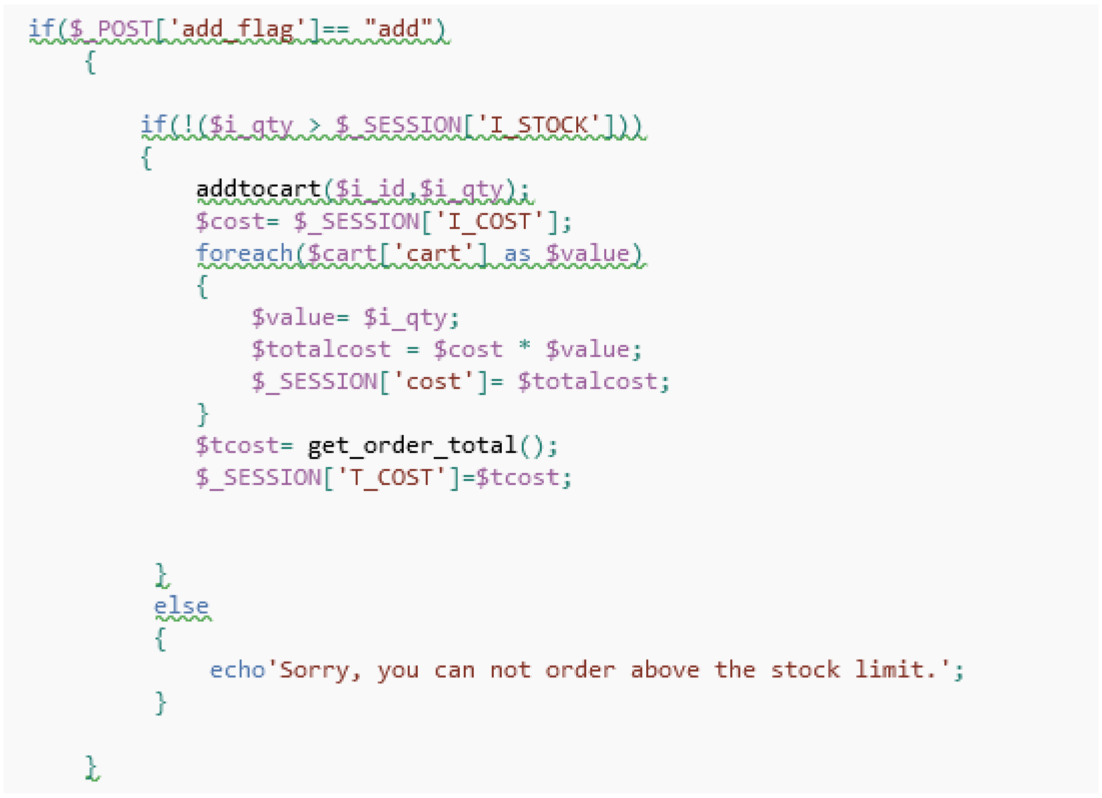
```
<table border="1" id="myTable" background-color:#88ccaa">;
    <tr>
<th>Item Id</th> <th>Title</th><th>Author Id</th> <th>Publisher</th><th>Subject</th><th>Cost</th><th>Currency</th><th>Stock</th></tr>



<tr>
<td><?php echo $_SESSION['I_ID'] ?> </td>
<td><?php echo $_SESSION['I_TITLE'] ?> </td>
<td><?php echo $_SESSION['I_A_ID'] ?> </td>
<td><?php echo $_SESSION['I_PUBLISHER'] ?> </td>
<td><?php echo $_SESSION['I_SUBJECT'] ?> </td>
<td><?php echo $_SESSION['I_COST'] ?> </td>
<td><?php echo $_SESSION['I_CURRENCY'] ?> </td>
<td><?php echo $_SESSION['I_STOCK'] ?> </td>

</tr>

</table>
```

The shoppingcart.php displays the book details again and shows a small text field to add the amountof books to be ordered.



```
    </div>
        <div style="color:#F00"></div>
        <table border="0" id="myTable" cellpadding="5px" cellspacing="1px" style="font-family:Verdana, Geneva, sans-serif; font-size:11px; background-color:
        <tr><th>Item Id</th> <th>Title</th><th>Author Id</th> <th>Publisher</th><th>Subject</th><th>Cost</th><th>Currency</th><th>Stock</th><th>Quantity</th
        <tr>
        <td><?php echo $_SESSION['I_ID'] ?> </td>
        <td><?php echo $_SESSION['I_TITLE'] ?> </td>
        <td><?php echo $_SESSION['I_A_ID'] ?> </td>
        <td><?php echo $_SESSION['I_PUBLISHER'] ?> </td>
        <td><?php echo $_SESSION['I_SUBJECT'] ?> </td>
        <td><?php echo $_SESSION['I_COST'] ?> </td>
        <td><?php echo $_SESSION['I_CURRENCY'] ?> </td>
        <td><?php echo $_SESSION['I_STOCK'] ?> </td>
        <td><form action="<?php echo "productcalc.php"; ?>" method="post">
        <input type="text" maxlength="3" size="2" size="70" name="i_qty" value="1" />

  <select name="add_flag">
  <option value="add">Add</option>
  <option value="update">Update</option>
  </select></p>
  <p><input type="submit" name="result" value="Submit" /></p> </form> </td>
```

Once directed to the product calculations page the two arrays are created. One is the session array that contains the items id and quantity and the other is a multidimensional array that contains the user id and the item id and quantity as key value pairs.

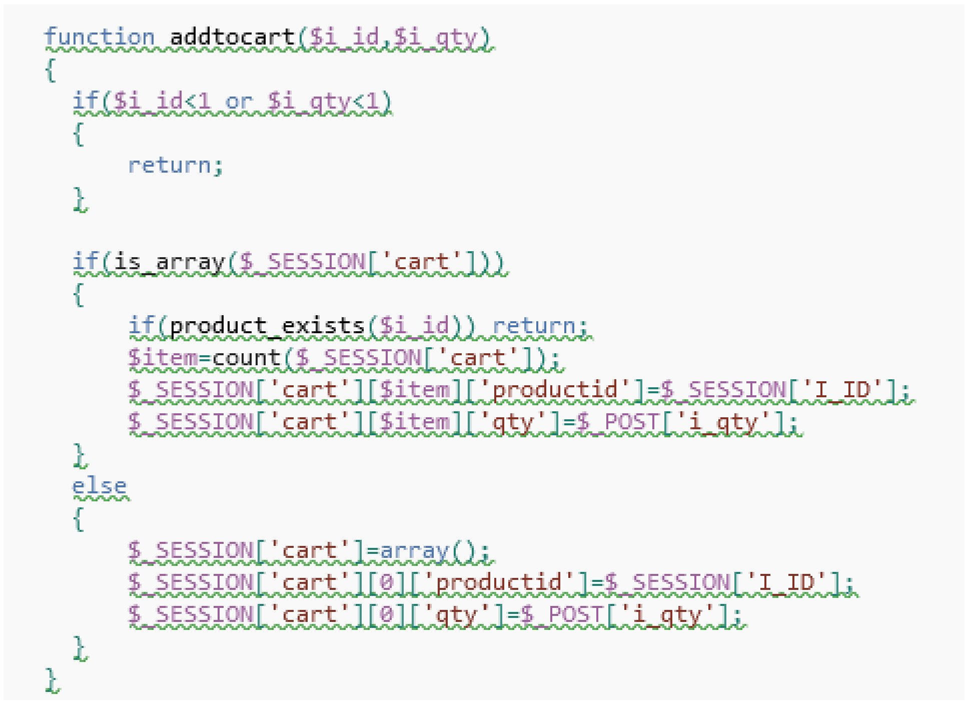


```
if($_POST['add_flag']== "add")
    {

        if(!($i_qty > $_SESSION['I_STOCK']))
        {
            addtocart($i_id,$i_qty);
            $cost= $_SESSION['I_COST'];
            foreach($cart['cart'] as $value)
            {
                $value= $i_qty;
                $totalcost = $cost * $value;
                $_SESSION['cost']= $totalcost;
            }
            $tcost= get_order_total();
            $_SESSION['T_COST']=$tcost;


         }
         else
         {
             echo'Sorry, you can not order above the stock limit.';
         }

    }
```

Once the add_flag is checked to see whether it is add or update. The order is added to the cart and the total price and the current price is calculated.

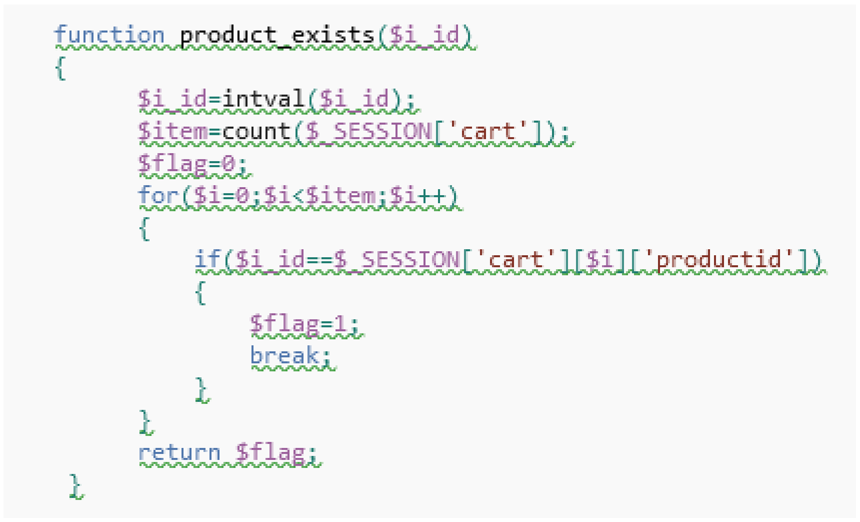
$_SESSION['cost'] is the current cost whereas $_SESSION['T_COST'] is the total cost.

The function addtocart is shown below.



```
function addtocart($i_id,$i_qty)
{
  if($i_id<1 or $i_qty<1)
  {
      return;
  }

  if(is_array($_SESSION['cart']))
  {
      if(product_exists($i_id)) return;
      $item=count($_SESSION['cart']);
      $_SESSION['cart'][$item]['productid']=$_SESSION['I_ID'];
      $_SESSION['cart'][$item]['qty']=$_POST['i_qty'];
  }
  else
  {
      $_SESSION['cart']=array();
      $_SESSION['cart'][0]['productid']=$_SESSION['I_ID'];
      $_SESSION['cart'][0]['qty']=$_POST['i_qty'];
  }
}
```

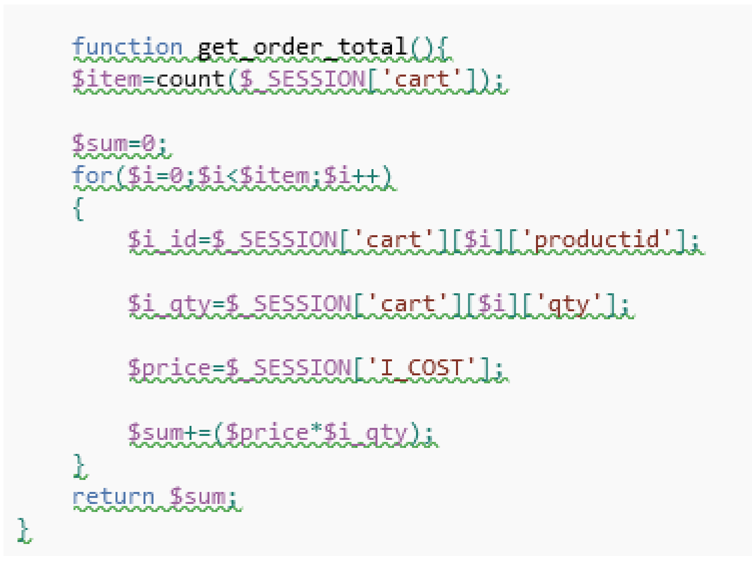
It checks whether the id exists and whether the quantity also exists. Then after checking whether the cart isn't empty it checks if the product exists and then counts the number of items in the array of product items and chooses that as the index of the array to be added. These are then taken from the session variable and the post variable and stored into the array. If the session cart array is empty the id and quantity are stored in to the first index of the array. The product_exists function is shown below.



```
function product_exists($i_id)
{
      $i_id=intval($i_id);
      $item=count($_SESSION['cart']);
      $flag=0;
      for($i=0;$i<$item;$i++)
      {
          if($i_id==$_SESSION['cart'][$i]['productid'])
          {
              $flag=1;
              break;
          }
      }
      return $flag;
 }
```

This returns 1 if product exists and 0 if the product doesn't exist.

The total order of all the items in the cart has been attempted in the following manner as shown below.



```
    function get_order_total(){
    $item=count($_SESSION['cart']);

    $sum=0;
    for($i=0;$i<$item;$i++)
    {
        $i_id=$_SESSION['cart'][$i]['productid'];

        $i_qty=$_SESSION['cart'][$i]['qty'];

        $price=$_SESSION['I_COST'];

        $sum+=($price*$i_qty);
    }
    return $sum;
}
```

The array size is calculated, which is the total number of items and is the looped through the entire array till the last index is reached and added to sum each time.

Testing

TEST CASE	RESULT
Check database interface fields by erasing all values and submitting an empty form	Database is not created and an error message is displayed
If * marked fields are ignored in the database settings change interface	Error message is displayed
If the person successfully registers	A message is displayed
If the person input invalid details in the login page	The user does not login and an error message is displayed
If the user clicks on log out	The database connection closes successfully
If the user tries sql injection	Not entirely prevented but in some cases it fails due to checks
If the user tries to search for a book not available in the database	A message is displayed telling the user no book is found
If the user tries to add items in the shopping cart that exceed the stock limit	An error message is displayed
If the user tries to search for a book by not entering the complete search criteria i.e. David instead of David Walliams for author	The book is not found
If the user clicks on home after done shopping	The details of the user are no longer displayed.

Conclusions

In conclusion I would like to thanks my instructor for teaching this course as it has been very insightful and has clearly shown the difference between programming in C# and server side scripting and also its advantages and disadvantages. I am most proud of my search page and the shopping cart page and would like to improve the design of the entire website and add more CSS styling to it. If the option to have done this differently I would have liked to add JavaScript coding to it to make it more user friendly. I would also like to implement this whole website in C# for fun to see the difference between them. Working on C# had been challenging but I found it a little more challenging to work in PHP due to my limited knowledge of HTML/CSS but I learned a lot along the way.