

Web Programming - Assignment 3

Instructions	2
Overview	2
Requirements	3
1. Website and Page Structure	3
2. Products (Services) Page Upgrades	4
2.1 Spreadsheet Read	4
2.2 Combined Product / Products Page	4
2.2a Combined Product / Products Page Update	5
2.3 Javascript Functionality	6
2.4 PHP Functionality	6
3. Cart Page / Facility	7
4. Checkout Page / Facility	8
5. Receipt Page and Orders Spreadsheet	9
Marks Allocation	10
25 marks or 25% of your final grade	10
Website and Page Structure (5 marks)	10
Products (Services) Page Upgrades (6 marks)	10
Cart Page / Facility (5 marks)	10
Checkout Page / Facility (6 marks)	10
Receipt Page and Orders Spreadsheet (3 marks)	10
Helpful functions to put in tools.php	11

Instructions

Students can work **individually** or in **groups of 2** for this assignment. All are encouraged to seek help from classmates and teaching staff.

Students must make sure that their assignment 3 is available from this url:

<https://titan.csit.rmit.edu.au/~s1234567/wp/a3/index.php>

An official submission must be made by 11:59pm Friday 12th October via **Canvas**.

Please zip up your **wp** folder and rename it with your student id(s), ie

[wp-s1234567.zip](#) (individual)

[wp-s1234567-s1234568.zip](#) (leader / partner)

This submission will be used for plagiarism testing and as a reference should problems arise.

If all these things are not done, you may receive zero marks.

Keep the [wireframe.css](#) stylesheet and button intact, this will help you during the development process and also your assignment marker to visually assess your website structure.

Please Note: If there are issues with your assignment that you need to fix before marking can take place, a temporary mark of 0.25 will be assigned until your assignment is ready for marking.

Overview

Assignment 2 asked you to build the front end of the assignment, now the client wants you to build the full eCommerce website to the point that orders can be made and stored in a spreadsheet.

Requirements

1. Website and Page Structure

Content on all pages / facilities above must be neatly laid out, attention to alignment and proximity must be observed.

All prices must be numeric and displayed with a "\$" in front and to 2 decimal places at all times.

All your CSS and JS must be in external files (ie do not use inline and embedded versions unless there is a very good reason for doing so).

All pages must be PHP and must have access to the session object, ie use `session_start()` near the top of every page. Framework code, ie code common to all pages, must be moved out to functions or some other method explicitly approved ahead of time by the instructor.

A debug module must also be included. Instructions are provided in the tute lab sheets and will work for offline and online hosting options.

Implementation of all the above tasks must be complete in order to earn marks.

The client would like to have three more pages (or facilities):

1. **Cart Page / Facility**

When a customer has finished making selections from the product page / facilities, they can progress to the cart page or facility and see all of their purchases, item subtotals and order total.

2. **Checkout Page / Facility**

When a customer has finished reviewing their purchases in the cart page / facility, they can progress to the checkout page or facility, enter their details and complete their purchase.

3. **Receipt Page**

When a customer has finished entering their details in the checkout page / facility, they can progress to the receipt page. This page must display their complete order in a printable format. What this means is that the page should be styled with a different stylesheet and must look like an A4 letterhead page with business / organisation branding when viewing in "print preview" mode.

Your client no longer needs a single **product** or **service** page. This functionality will be built into the existing **products** or **services** page.

You should also create a file called **tools.php** that contains functions for each page to use. Some starting functions (blue text) have been provided on the last page of this assignment document with examples of calling code (red text) to use in your php files.

2. Products (Services) Page Upgrades

2.1 Spreadsheet Read

The client would like to supply you with a spreadsheet of product / service information. The spreadsheet must be read and the product information in the site must come from a **tab delimited** spreadsheet called [products.txt](#) or [services.txt](#). Make sure that your supplied spreadsheet has a row of headings and multiple rows of data in the following order:

ID	OID	Title	Description	Option	Price
T123	SML	Mens T-Shirt	This is a small men's Tshirt ...	small	25.5
T123	LRG	Mens T-Shirt	This is a large men's Tshirt ...	large	25.5

A link to your spreadsheet must be placed in the footer.

2.2 Combined Product / Products Page

All features present in the single and combined product / service pages must be combined into a single products or services page.

- When no valid product or service id is present in the [GET](#) header, all products or services are displayed with no purchasing controls.
eg [products.php](#) and [products.php?id=FAKE_ID](#)
- When a valid product or service id is present in the [GET](#) header, a single product or service is displayed with a [form](#), purchasing controls (quantity field; -, + and add to cart buttons; options dropdown or radio buttons) and a price subtotal in a [span](#) element for that item.
eg [products.php?id=LEGIT_ID](#)

```
if (isset($_GET['id']) && this_id_actually_exists($_GET['id'])) {  
    // show a single product or service matching id with a purchasing form  
} else {  
    // show all products or services without purchasing form  
}
```

Web Programming - Assignment 3

2.2a Combined Product / Products Page Update

UPDATE 20/09: This was discussed and agreed upon in the lecture in week 9.

These tasks are proving tricky and may damage a student's products.php / services.php page.

This functionality can be incorporated into a separate product.php / service.php page.

Students now have the option of making their:

- products.php / services.php page dynamic - OR -
- product.php or service.php page dynamic - OR -
- individual products / services appear in modal boxes in products.php / services.php

To minimise word confusion, if your website has services, substitute the word "product" with "service" in the language and filenames below.

Where language differs significantly from the original text, it is displayed in bold text.

Original (harder) option	New (easier) option	Modal box (advanced) option
All products (plural) summaries are displayed in products.php when no valid id is present in the GET header. No form controls are displayed, only links to individual products , eg <code>products.php?id=LEGIT_ID</code> , where <code>LEGIT_ID</code> is a valid product id in your assignment.	Single product page redirects user to products page where all products (plural) summaries are displayed in products.php when no valid id is present in the GET header. No form controls are displayed, only links to individual products , eg <code>product.php?id=LEGIT_ID</code> , where <code>LEGIT_ID</code> is a valid product id in your assignment.	No modal windows are displayed , all products (plural) summaries are displayed in products.php or services.php when no valid id is present in GET header. No form controls are displayed other than those necessary to open individual product modal boxes.
Single product in products.php is displayed when a valid id is present in the GET header.	Single product in product.php is displayed when a valid id is present in the GET header.	Single product is displayed in a modal box when a valid id is present in the GET header.
Purchasing controls are only shown on single product page, eg <code>products.php?id=LEGIT_ID</code> , where <code>LEGIT_ID</code> is a valid product id in your assignment.	Purchasing controls are only shown on single product page, eg <code>product.php?id=LEGIT_ID</code> , where <code>LEGIT_ID</code> is a valid product id in your assignment.	Purchasing controls are only shown on a single product modal box window eg where <code>id=LEGIT_ID</code> is a valid product id in your assignment.

2.3 Javascript Functionality

Use javascript to calculate and display a subtotal price for the item as the user makes selections (ie as the user clicks the buttons or enters a quantity directly). *Advanced users: you may wish to factor in different prices for different options.*

2.4 PHP Functionality

When the customer clicks the add to cart button, the product or service ID, option id and quantity must be submitted to the server via the POST method and added to a shopping cart in the SESSION. The user must then arrive on the cart page, for example:

```
if (isset($_POST['add'], $_POST['id'], $_POST['qty'], $_POST['oid'])) {  
    // server side code is required here to validate and check if  
    // - qty is a positive integer (ie 1 or more)  
    // - product/service and option ids are valid  
    $_SESSION['cart'][$id]['oid'] = $oid;  
    $_SESSION['cart'][$id]['qty'] = $qty;  
    // redirection BEFORE any HTML is outputted  
    header("Location: cart.php");  
}
```

This can either be in a separate processing script or near the top of [cart.php](#), in which case the header redirect is not needed.

3. Cart Page / Facility

The cart page or facility must display all products, options and quantities ordered in an organised fashion.

A button or link should be present to advance the user to the checkout page / facility.

Another button or link should be present to cancel the entire order, after which the user should be redirected to the products page, for example:

```
if (isset($_POST['cancel'])) {  
    unset($_SESSION['cart']);  
    header("Location: products.php");  
}
```

This can either be in a separate processing script or near the top of `products.php`, in which case the header redirect is not needed.


Web Programming - Assignment 3

4. Checkout Page / Facility

The checkout page or facility must collect the following information which must be validated server side and conform to these criteria:

Name	Type	Description	Validation	Required
Name	text	All names of customer ie first and last names.	Alphabetic characters, plus some punctuation chars: space, full stop, comma, single quote and hyphen.	Yes
Email	email	email address	A <code>filter_var()</code> function check the using <code>FILTER_VAR_EMAIL</code> flag is fine	Yes
Address	text area	Multiple lines of text.	Numbers and alphabetic characters, plus some punctuation chars: space, full stop, comma, single quote, hyphen, forward slash and line feed char.	Yes
Mobile Phone	text	Australian mobile.	Must start with +614, (04) or 04; then any grouping of 8 more digits and single spaces is permitted.	Yes
Credit Card	text	Credit card number.	Any grouping of 12 - 19 numbers and single spaces are permitted.	Yes
Expiry Date	date or selects	The expiry date of the credit card.	Cannot expire within one month of purchase, for example, if making purchase in December 2017, expiry date must be February 2018 or later. Update: alternative option: 28 days in the future.	Yes

As an added feature, when a valid visa number is entered, a small visa logo should appear next to the credit card field and disappear / not appear if not a VISA number. This should be done using a javascript event handler function and be triggered by the credit card's oninput event.

	-blank-
4123 4567	-blank-
1234 5678 9012 3456	-blank-
4123 4567 8901 2345	
4123 4567 8901 2345 6789	-blank-

Visa cards numbers begin with the number 4 and have 13 - 16 digits inside (and optional spaces).

5. Receipt Page and Orders Spreadsheet

Once the client's details are validated, orders must be appended to your existing ~~comma~~ **TAB** delimited spreadsheet called `orders.txt` according to the following format:

Purchase Date	Name	Address	Mobile	Email	ID	OID	Quantity	Unit Price	Subtotal

Servers normally allow PHP scripts to create files, however due to permissions issues on Coreteaching machines, put the heading row in manually and give the spreadsheet `chmod 606` permission.

A link to your `orders.txt` spreadsheet must be placed in the footer.

Unit prices and subtotals should be calculated server side based on quantities provided to the server. Do not trust the client to tell you what the unit price and the subtotal price is.

Also, make sure that no orders with invalid product / service and option ids are stored in the spreadsheet.

For security reasons, the credit card number and expiry date should not be stored as part of this assignment.

Marks Allocation

25 marks or 25% of your final grade

Website and Page Structure (5 marks)

1. All web pages are present, have the .php extension and access to the session object. 1 mark
2. Framework code common to all pages is separated out into module functions. 1 mark
3. A complete debug module has been created, is working in most pages. 1 mark
4. All calculated prices are present, numeric, displayed to 2 decimal places with '\$' in front. 1 mark
5. Attention to proximity, alignment, margin and spacing is thorough throughout website. 1 mark

Products (Services) Page Upgrades (6 marks)

Server side programming tasks

6. Products descriptions and options are added by reading a spreadsheet with fgetcsv. 1 mark
7. All products (plural) summaries are displayed when no valid id is present in GET header. 1 mark
8. Single product is displayed when a valid id is present in the GET header. 1 mark
9. Purchasing controls are only shown on single product page. 1 mark

Client side programming tasks

10. Subtotal price for each item changes in real time. 1 mark
11. Subtotal price calculation is correct. 1 mark

Cart Page / Facility (5 marks)

12. Purchase information is added to a server side shopping cart. 1 mark
13. Only orders with valid ids, oids, positive quantities and options are added to cart. 1 mark
14. Prices are re-calculated server side. 1 mark
15. Cart data is structured in an organised "object-like" structure. 1 mark
16. User can cancel entire order (ie cart is unset), is redirected to products page. 1 mark

Checkout Page / Facility (6 marks)

Client side validation programming tasks

17. When a valid **visa card** number is entered, a small visa logo is displayed next to field. 1 mark

Server side validation programming tasks

18. Name, address, mobile phone data validated to match specifications. 1 mark
19. Credit card validated to match specifications (ie not just Visa). 1 mark
20. Credit card cannot expire within 1 whole month (or 28 days) of purchase date. 1 mark
21. If validation/sanitisation issues exist, user is directed to or kept back on the checkout step. 1 mark
22. Error messages are placed next to invalid fields server side with appropriate styling. 1 mark

Receipt Page and Orders Spreadsheet (3 marks)

23. Receipt page is styled differently to print on A4 letterhead, contains all details. 1 mark
24. Receipt page has company branding, style and layout quality is high. 1 mark
25. Orders are appended to a spreadsheet file using fputcsv in the specified format. 1 mark

Total: 25 marks (or 25% of your final grade)

Helpful functions to put in tools.php

Instructions when using code examples in a debug module and/or when debugging in general:

- The **blue code** is "function declaring code" to put in tools.php.
- The **red code** is "function calling code" to put in your web pages or tools.php where you want the return or output of the function.

1) "preShow()" function prints data and shape/structure of data:

```
function preShow( $arr, $returnAsString=false ) {  
    $ret = '<pre>' . print_r($arr, true) . '</pre>';  
    if ($returnAsString)  
        return $ret;  
    else  
        echo $ret;  
}
```

```
preShow($_POST);  
preShow($_SESSION);
```

```
$aaarg = preShow($my_bad_array, true);  
echo "Why is \n $aaarg \n not working?";
```

2) Output your current file's source code:

```
function printMyCode() {  
    $lines = file($_SERVER['SCRIPT_FILENAME']);  
    echo "<pre class='mycode'>\n";  
    foreach ($lines as $lineNo => $lineOfCode)  
        printf("%3u: %1s \n", $lineNo, rtrim(htmlentities($lineOfCode)));  
    echo "</pre>";  
}
```

```
printMyCode();
```

Web Programming - Assignment 3

3) A "php associative array to javascript object" function

UPDATE: Now handles multidimensional arrays:

```
function php2js( $arr, $arrName ) {
    $lineEnd="";
    echo "<script>\n";
    echo "    var $arrName = ".json_encode($arr, JSON_PRETTY_PRINT);
    echo "</script>\n\n";
}

$pricesArrayPHP = array ( ... );
php2js($pricesArrayPHP, 'pricesArrayJS');
```

4) Output dynamically CSS to style the current nav link:

```
function styleCurrentNavLink( $css ) {
    $here = $_SERVER['SCRIPT_NAME'];
    $bits = explode('/', $here);
    $filename = $bits[count($bits)-1];
    echo "<style>nav a[href$='$filename'] { $css }</style>";
}

styleCurrentNavLink('background-color: rgba(255,255,255,0.6);');
```