# Overview

Your client is happy with your work and now wants you to build the full eCommerce website. The client has heard that the best way to develop eCommerce websites is to use "PHP" and wants you to use that language.

# Submission Instructions

*If you want to break up with your previous assignment partner and/or team up with a new assignment partner, you may do so. Please talk to a staff member if you need advice or assistance.*

Students may work alone or in a pair in this assignment. We recommend pairing with someone with complementary skills (eg programmer + designer).

You may find that collaborating on cloud9 (<https://c9.io>) helpful when working in groups and justifying any work imbalances later on.

**Students should host their website on coreteaching servers in order to gain marks for this assignment and make the homepage available from this url:**

<http://titan.csit.rmit.edu.au/~s1234567/wp/a3>  **(index.php)**

The group leader should host the assignment and the partner should provide a link to the group leader's hosted assignment.

|  |
| --- |
| An official submission must be made at the start of week 12 by the group leader via **Blackboard**:  **Monday 10 October 2016, 4am**. |

Files must be **zipped into one file**. These files will be used for plagiarism testing and as a reference should problems arise.

# 

# Requirements

## 

## 1. Website Page Structure

Content common to all pages must be stored in module files and imported into your main files using either the include or require PHP functions. At a minimum, <header>, <navigation> and <footer> are to be included this way.

In addition, the debug module (or your own creation) must be included into every page. :

<?php include\_once('/home/eh1/e54061/public\_html/wp/debug.php'); ?>

Students using local hosting or who are on cloud 9 can use the "debug-lite" module. Please make sure that this is only used on private servers as it prints out all of your PHP code.

https://titan.csit.rmit.edu.au/~e54061/wp/debug-lite.phps

Putting session\_start() near the top of every page, customers can browse pages in any sequence and the site will remember any purchases they have added to the cart.

Advanced students: you may develop a **single page application** or use a **framework** if you wish, but this is not covered in this course and will not appear on the exam. Make sure you understand the simple path before embarking on a more complex path.

## 

## 2. Shop Page

In assignment 2, we asked you to create a shop page to display all 3 previous seasons of Please Like Me. In this assignment we want you to make the page dynamic using javascript.

Unit prices for each season should be displayed near to each product and as the user makes selections

* the subtotal (ie price for each season ordered) should update in real time
* the total (ie price for all seasons ordered) should update in real time

The unit prices for each season box set are:

* Season 1: $17.00
* Season 2: $22.50
* Season 3: $26.75

## 

## 

## 3. Cart Page

When customers have finished ordering products, they may go to the cart page where they are presented with all of their orders, quantity of each item, subtotals of each item, and a final order total. Quantity values must be re-checked serverside to determine that they are an integer between 0 and 5 using the filter\_var() function or an alternative solution.

The fields should line up neatly and logically, page layout should meet HCI usability guidelines.

Ideally, a user can delete items they no longer want from the cart (ie a complete boxset season order).

## 

## 4. Checkout Page

When a customer moves to the checkout page, the following information is requested. All fields are required (ie cannot be blank) and the user must be alerted to errors without using the default javascript alert box:

Fields requiring simple validation both client side and server side:

* Text boxes: first name and last name fields  
  *- contains letters, spaces, some punctuation characters*
* Textarea box: address field   
  - *a multi-line field, contains numbers, letter, spaces, some punctuation characters*
* HTML5 type boxes: email (email type) and phone (phone type)  
  - *must be a valid email or valid phone*
* Text box (or your own solution): credit card   
  - *must contain between 13 & 18 numbers, and up to 3 single spaces separating groups of digits*

Client side, use of the modern HTML5 attributes such as type, pattern and required OR javascript can be used to perform validation. Only the credit card and expiry field values need to be validated serverside.

The page should also make use of the new HTML5 localStorage object that allows data to be saved client side so that the server does not have to store the customer's personal details. When a customer selects a "please remember me" checkbox (or method of your choosing), name, address, email and phone should be stored in localStorage.

If localStorage data exists, the fields above should be pre-filled the next time that the user makes orders. The user should also be able to deselect the "please remember me" checkbox (or method of your choosing) which removes the customer's information from localStorage.

(continued on next page)

### 

More complex (or compound) fields are as follows:

* Delivery Method (drop down options or radio buttons) with regular post, regular courier, express courier options. Most add an additional cost, delivery prices as follows:
  + Regular delivery price: free
  + Regular Courier price: $7.00
  + Express Courier price: $10.50
* Credit Expiry (drop down options or your own solution): a year and a month can be entered and the date must not be expired (ie you will have to get the current date at the time of ordering)
  + A marker will test one month ahead and one month behind. Whether you believe a credit card expires at the start or end (or middle?) of a month is completely up to you.

If all the fields are valid, the order is processed and the customer is taken to the final page.

## 5. Orders Confirmation Page

When the customer finalises their order, their order should be added (ie appended) to a file so that the client can track all orders from every customer. This file can be a text file, or spreadsheet (csv file) or in JSON format. You can assume that the client is unable to use or create a database OR Advanced students only: storage in a database is allowed, but is not covered in this course and you must familiarise yourself with file writing concepts for the exam.

*You will need to create the file first and give it read and write permission (ie 606).*

This confirmation page must match the look and feel of other pages in your website and is just for the customer. It should only contain information about their most recent order, displayed in a human readable format and laid out to the same standard as a shop receipt.

If you are unsure, press the print button in your browser to see how it would look if you printed it out.

The cart must be cleared once the order has been written to the text file and the order information has been displayed to the user.

## 

**Important: Please don't store credit card details for security reasons. You won't get a deduction if you do, it's just not good practice and we don't want markers having access to your credit card details.**

# Marks Allocation

## 21 marks or 21% of your final grade

### 1. Website Structure 3 marks

All pages have access to the SESSION superglobal array 1 mark

Proper use of modules/includes throughout site 1 mark

All prices in website are displayed to 2 decimal places 1 mark

### 2. Shop Page 0 marks

Tasks are now part of week 10 Lab Projects

### 3. Cart Page 5 marks

All orders, subtotals & totals are displayed, link to checkout page present 1 mark

Quantities re-checked serverside, season prices used in calculations are serverside 1 mark

Subtotals and totals for multiple purchases calculated correctly server side 1 mark

Customer can remove an entire season order from cart 1 mark

Items are stored in the SESSION in an organised structure 1 mark

### 

### 4. Checkout Page 10 marks

All form fields correct type, lined up neatly; form uses the post method 1 mark

Delivery method fields, credit card and expiry date fields set up correctly 1 mark

Delivery prices are added to the final bill 1 mark

*Form fields: client side data validation*

Credit card number validation completed client side 1 mark

Credit card expiry date detected correctly client side 1 mark

*Form fields: server side data validation*

Credit card number completed server side 1 mark

Credit card expiry date detected correctly server side 1 mark

*HTML5 Local Storage*

A customer can store personal details into localStorage, ie 'remember me' 1 mark

A customer can remove personal details from localStorage, ie 'forget me' 1 mark

Remembered form fields are pre-filled from localStorage 1 mark

### 

### 5. Orders confirmation page 3 marks

Customer order confirmation page displays a receipt and looks professional 1 mark

All order details stored as a human readable block in orders.txt (or .json, .csv, etc) 1 mark

Cart is unset when order is complete 1 mark

**Total:** **21 marks (or 21% of your final grade)**