Swinburne University of Technology
Faculty of Science, Engineering and Technologies

COS10011 / COS60004 / COS60007
Creating Web Applications [and Databases]

Assignment Part 3, Semester 1, 2017

Server-Side Programming

Important Dates:

Due Date ESP	**Week 12 - Normal day of you tutorial: 23-26 May,8 am.** (Late submission penalty: 10% of total available marks per day)
Demonstration	Your tutorial: Week 12 (23-26 May)

Individual Assignment. Contribution to Final Assessment: 14%

Purpose of the assignment

To gain practical skills and knowledge in coding and using PHP server-side embedded scripting to extend the functionality of the website you developed in Part 1 and 2, by creating dynamic webpage content, accessing a separate MySQL database server, and using PHP to connect to this MySQL server. In particular, you will:

- Use PHP to include common webpage code (eg. menu, header, footer code)
- Understand the ways that 'state' can be maintained between web pages
- Use PHP to validate data sent in the "payment" HTML form to a new "process_order.php" page and if any errors, provide user feedback through a new "fix_order.php" form, else if okay …
- Use PHP to create and store the order data in a server-side MySQL database table "orders", and provide feedback through a new "receipt.php" page.
- Use PHP to query and update the status of the "orders", through a new "manager.php" page.
- Understand how using a "settings.php" script, and relative links, can enable the website to be ported from a development environment to a testing environment.

As in Parts 1 and 2, there is an opportunity to enhance your website beyond the specific requirements.

Prerequisite

If you failed to meet the Essential Requirements of **Part 2**, then you must demonstrate that you have fixed these problems ***before*** *this part of the assignment is submitted and marked.*
It is advisable to get these fixes completed and signed off *well before* you hand in this part of the assignment.
How to get your fixes signed off:

1. Arrange a time with your tutor to check your work during your allocated tutorial or during a consultation time.
2. Bring a copy of the ESP assessment printout from Part 2
3. Your tutor will check that the fixes address the issues identified under the Essential Requirements on the mark sheet.
4. If the fixes are successful, your tutor will record this and you will be eligible to have this Part 3 assignment assessed. If there are issues that have not been fixed, your tutor will inform you of this and you will have *a further chance* to fix the assignment.

Note that any fixes will **not** alter the mark you received for Part 2.

Note: This assignment must be demonstrated in you Lab.
The code that is assessed in your demonstration *must be identical* to the code you submit to ESP. This will be checked before the demonstration.

Implement your website in standard HTML5, that is also well-formed XML.
And all new webpages generated by PHP should also be compliant when served to the client-side.

Specified Requirements

1. Use PHP to reuse common elements in your website

PHP provides us with techniques to modularise and reuse our web application code. Rewrite your web pages so that the common static HTML elements such a menu, header and footer are written in common text files that are then "included" back into your web pages. Name the include file(s) with a **`.inc`** extension, replace the sections of HTML in your main pages with 'include' statements, and rename your main pages with a **`.php`** extension, so the php includes will be included.

2. Create an 'orders' database table

Create an 'orders' table in your MySQL database.

This table will record the data sent from the "payment.php" form (including the data gathered from "enquire.php"). This data contains information on the customer, product and payment details as specified in the previous assignments. The format of the database fields should match appropriate validation rules defined in assignments Part 1 and 2. If no rule exists for a particular field, choose an appropriate format.

In addition to the fields that record information from the "payment" form, add the following fields with appropriate data-types to the table:

- An auto-generated primary key field called 'order_id'.
- The total cost of the order 'order_cost'
- Date / time of order (generated by the system) 'order_time'.
- A field called 'order_status'.
  An 'order status' can have one of three values: PENDING | FULFILLED | PAID | ARCHIVED.
  When an order is created its status is always set to PENDING.

Later, your "`process_order.php`" page should be able to create this table, if it does not already exist, when the first order is made.

3. Create a file to store your database connection variables "`settings.php`"

To enable your website to be easily ported to our "testing" environment (and in a workplace, ultimately to a production website platform), use a PHP include file, "settings.php" that contains the ***connection variables as shown below***, and use this in your PHP to connect to your MySQL database on the feenix-mariadb database server.

```php
<?php
    $host = "feenix-mariadb.swin.edu.au";
    $user = "s1234567890";  // your user name
    $pwd = "ddmmyy";        // your password (date of birth ddmmyy unless changed)
    $sql_db = "s1234567890_db"; // your database name
?>
```

4. Disable HTML5 and JavaScript form validation `part2.js, enquire.php, payment.php`

While client-side validation using HTML5 and JavaScript was used in previous assignments, in order to preserve the integrity of the server data, server-side data format checking should be implemented. In this assignment HTML5 and JavaScript form validation will be disabled.

So we can test that server-side validation works correctly:

1. Add the **novalidate="novalidate"** attribute into your forms, to disable client-side HTML5.
2. Because we will still need JavaScript to handle client-side storage, we cannot disable it entirely. You will need to temporarily disable any validate function(s) within your JavaScript.

Hint: You could disable the JS by making any *call* to the validate functions conditional.
Put them in an **`if`** statement that evaluates a global Boolean variable you create and initialize. e.g.

```
...
if (!debug) {validate()};
...
```

Set the flag variable 'debug' to *true* or *false* depending on what mode the code is run.
Have a prompt, or a check box on the page to set the variable 'debug' .

5. Processing Orders `"process_order.php"`

Change the "payment.php" form action to "process_order.php". Having disabled HTML5 and JavaScript form data validation, all form data format checking will now be implemented server-side, using PHP, after the data has been submitted by "payment.php":

1. All values received by "process_order.php" should be sanitized to remove leading and trailing spaces, backslashes and HTML control characters.
2. Before an order is written to the **`orders`** table the data format rules need to be checked. These rules are specified in Part 1 (for customer details) and Part 2 (for product quantity and credit card details), and a product with options should also be able to be selected and checked. You need to replicate this checking in your PHP code. *(See the Mark Sheet for a checklist).*

If the input data does not meet format requirements, errors should be returned to **`"fix_order.php"`** a form version of the 'payment' page, and display all form control fields filled with data passed back using a query string, and with errors marked or highlighted. Do not pass back the Credit Card details in the query string, these will need to be re-entered. The "fix_order.php" form should submit back to "process_order.php", using method post. *Hint: error msg back to fix_order could be string or an array.*

If the input data is correctly validated by "process_orders.php":

1. Calculate the total cost of the order (do not rely on the client to send this information).
2. Store the order in the orders table using a mysqli query.
3. Return an order receipt webpage **`"receipt.php"`** to the user. This page should include all the information stored in the record including the **`order_id`** and **`order_status`**.

The "process_order.php" page should not produce a html page. It should only process data and pass data to other webpages. (During development you might want to have this script echo back data.)

The "process_order.php" page should include a check that if the database table 'orders' does not exist then create it.

The "process_order.php" page, and the "receipt.php" page should not be able to be accessed directly by url through a browser. "fix_order.php" should only be able to be accessed with a query string.
Hint: check what data has been set and redirect.

6. Managers Order report and Order Update Page `"manager.php"`

For convenience add an extra menu item to access this new Manager page.
This web page allows the Manager to make queries about orders, display the result in an HTML table and update the status of an order. *In other words, this page will call itself, and have 3 sections.*

For each query clearly display: order number, order date, full details of the **product** including the **cost,** only the customer's **first** and **last** names, order status. No credit card details should be displayed.
To make the display presentable and easily readable, you might need to concatenate some fields.

The web page should give the manager the option to display:

- All orders
- Orders for a customer based on their name
- Orders for a particular product
- Orders that are pending
- Orders sorted by total cost

The Manager should be able to 'update' the status of an order from a link or button next to the order in the table, changing the status from (pending | fulfilled | paid | archived). This could be done by returning the record to a form in this webpage, to change and 'confirm' the change.

The Manager can also 'cancel' (ie. 'delete' an order) an order via this page.
Only pending orders can be cancelled. How you select the order to be deleted is up to you.

CSS

All pages should be styled appropriately using CSS as in Part 1 and 2, and should be valid CSS3.

Reflection *(optional)*

You might like to add a sub-heading to your 'Reflection' section on your `about.php` page, and include notes to yourself, about anything that might help you reflect on what you have learnt in Part 3, both now, and later when you revise your work. We will not be assessing this.

Enhancements

Please complete all the ***Specified Requirements*** before attempting any enhancements.

See theMarking Guide below.

As with Part 1 and 2 you have an opportunity to extend your learning by adding extensions/ enhancements to the main pages of your Web site, using techniques not covered in the tutorials.

Briefly **list** and **describe** each enhancement implemented on a page called **`enhancements3.php`**:

- What it does and how it goes beyond the specified requirements.
- What does a programmer have to do to implement the feature.
  (A reminder to your future self of what you have done.)
- Reference any third party sources used to create the extension/enhancement

Any enhancements that are not listed on the PHP enhancements page will not be assessed.
It is a good idea to discuss any proposed enhancements with your tutor before you implement them.

In this assignment we will consider PHP and MySQL enhancements.
You are encouraged to be creative in thinking up possible enhancements.

Examples of PHP / MySQL enhancements you might make that will contribute a higher mark include:

- Create Manager security, with a "Manager registration" page with server side validation requiring a unique username and a password rule, and store this information in a table. Create a "Manager Log-in" page to use the stored data, and control access to the manager web pages. Ensure the manager web page cannot be entered directly using a URL. Create a "Manager Log-out" page. Provide a 'log-out' link on the manager page if 'logged in'.
- Provide a number of more advanced Manager reports based on compound queries. For example
  - the most popular product ordered
  - fulfilled orders purchased between two dates the vendor enters
  - the average number of orders per day
- On the table on the Manager page, provide the ability to select a column heading, and re-sort the table in the order of that field. If selected again, reverse the order.
- Use php sessions to pass data back to the "fix_order.php" page from "process_order.php", in which case you could pass back the credit card details.
- Use sessions to process the forms in two stages. Store and pass data from 'enquire' to a "process_enquire" page to check this data, then redirect back if errors, else if okay, pass the data to 'payment' using sessions, and display this 'session' data in payment.
- Store customers' details in a separate 'customers' data table and create a primary-foreign key link between the 'customers' and 'orders' tables.
- Store the product details and options in a separate 'products' data table, and dynamically fill the product page with that data. (If possible, write code to create and populate the table, so the data can be tested, in a "testing environment".)

Up to 8 marks will be allocated to each enhancement. A maximum of 24 enhancements will be assessed.

Web Site Folder Structure and Deployment Requirements

Your website folder structure should follow a similar structure as in Part 1 and 2.
All files should be under a folder `/assign3`.

```
assign3/            You must have this folder – case sensitive!
  index.php
  product.php
  enquire.php
  about.php
  enhancements.php
  enhancements2.php
  enhancements3.php
  header.inc        } You could put these in an
  menu.inc          } 'includes' folder
  footer.inc        }
  settings.php
  ...other php function or include pages
  process_order.php
  fix_order.php
  receipt.php
  manager.php
  ...other php pages
  scripts/          Folder for JavaScripts
     part2.js
     enhancements.js
  images/           Folder for images for your page content
  styles/
     style.css      other css files
  styles/images/    Folder for images referred to by your css files e.g. background
```

Notes:

- PHP/HTML files should only be in the base "assign3/" folder – not anywhere else.
- All links to your files (includes, JavaScript, CSS or images) should be ***relative***.
  Do not use absolute links, as these links will be broken when files are transferred for marking.
  No marks will be allocated if links are broken.

Assignment Submission

An electronic copy of your assignment should be submitted through ESP at https://esp.ict.swin.edu.au on or before the deadline.

- Make sure all your files are in the correct folders and compress your root folder with all your sub-folders with PHP, HTML, CSS, JavaScript, and image files into a zip file named "**assign3.zip**". Submit this to ESP. When the zip file is decompressed, the entire Web site should be able to be run from index.php without needing to move any files.
  You can submit more than once through ESP.
- Note that all deliverables must be submitted electronically. There is no need to submit an assignment cover sheet as ESP generates a receipt upon successful submission.

Make sure you complete your ESP submission process.
You should get a PDF receipt if you have submitted successfully.

MAKE SURE YOU ENTER THE CORRECT UNIT CODE WHEN YOU SUBMIT YOUR ASSIGNMENT TO ESP
Submitting to the wrong unit will not be accepted as grounds for granting an extension.

Demonstration Procedure:

1. Make sure you attend your allocated lab.

   Demonstrate your assignment to the tutor in your allocated Tutorial in Week 12.

   You must attend this session to receive a mark for this assignment.

   If you cannot attend your allocated tutorial due to illness you must provide a copy of the medical certificate to the convenor.

2. Before your demonstration starts

   - Fill in and sign the Declaration on the Marking Sheet.
   - Make sure you website is running on Mercury.
     (Your tutor will check the URL).
     All demonstrations should be done on Firefox.
   - Load Web Developer tools in Firefox. Validate all your ***new/altered*** web pages for both HTML5 and XML and show the results display in separate browser tabs.

     Remember: validate the served HTML generated by your PHP - not the PHP code itself.
   - Display a copy of the submission receipt from ESP (a soft copy for viewing on your computer).
   - Load the phpMyAdmin web interface, (or the MySQL Monitor command line client), so you can demonstrate the changes to your 'order' table as your demonstration progresses.
   - ***Have some orders in your 'order' table***, so you can demonstrate your 'manage' queries.
   - Near the end of the demonstration, if you have implemented the create table in process_order.php, you will be asked to drop your 'order' table, and demonstrate this.,

3. As you demonstrate your website your tutor will ask you to explain how you have implemented various aspects of it.

Later, after the demonstration, your tutor will mark your source code and documentation, and enter the marks into ESP. You will be sent an email with the final marks.

Mark Sheet – Assessed by demonstration in your tutorial ***Marker: ……***

Declaration: **Fill this in before you start**

I hereby confirm that the assignment to be demonstrated is identical to my ESP submission.

Student number ……. Student name ……..

Signature …… Date ……

Product ……..

Tutorial Day …… Tutorial Time …….. Tutor Name ……..

A3

Marker checks: *Prerequisite – Essential Requirement in Assignment Part 2 - OK?* ☐

ESP with receipt sighted ☐ *File date:* ☐

During the demonstration, the marker will ask you about various aspects of your Web site, and ask you to explain the code you have used in the assignment.

Specified Requirements Place ☑ or ☒ in box	**Comments**	**Mark**
PHP common static includes: menu, footer, header, static code blocks included ☐ ☐ ☐		/6
settings.php created, included and used ☐ [2]		/2
Orders Table (view in phpMyAdmin) - scheme can store all the necessary data ☐ [2] - primary key order_id auto generated ☐ [2]		/4
enquire.php payment.php part2.js - client-side HTML5 form validation disabled ☐ ☐ [2] - client-side JS form validation disabled ☐ [2]		/4
process_order.php - unable to access directly through URL ☐ [2] - all data read from payment.php ☐ [4] - text data inputs sanitised ☐ [2]		/8
Data errors validated, displayed back in **fix_order.php**: (2 marks each) - fnames ☐ lnames ☐ email ☐ str_addr ☐ state ☐ pcode ☐ phone ☐ - pref.contact ☐qty *(not <1)* ☐ product ☐ options ☐ - total cost calculated ☐ - CC type ☐ CC name *(alpha)* CC number *(15-16 digits)* ☐ CC exp 2+2 *(>today)* ☐ - CVV ☐ CC formats checked ☐ pcode checked against state ☐		/36
fix_order.php - sent page validates to HTML5 ☐ and XML ☐ *(deduct 4 if not)* - all data sent from **process_order.php** to fix_order.php in query string [8] - all data displayed in form ☐ *(CC details not sent shown blank)* [8] - errors displayed in page ☐ [4] - form submits all data back to **process_order.php** ☐ [2]		/22
receipt.php - sent page validates to HTML5 ☐ and XML ☐ *(deduct 4 if not)* - all data sent from **process_order.php** to receipt in query string ☐ [4] - all order data displayed in receipt page ☐ *(Secure CC details shown ****)* [4] - record added to orders table by **process_order.php** *(view in phpMyAdmin)* ☐ [20]		/28
manage.php - sent page validates to HTML5 ☐ and XML ☐ *(deduct 4 if not)* - linked from menu ☐ [2] - form to make queries, with required options ☐ [2] - results in HTML table order fields (#,date,product,cost,name,status) ☐ [12]		/16
manage.php reports - all orders ☐ [4] - orders for a customer based on name ☐ [4] - orders for a particular product ☐ [4] - orders that are pending ☐ [4] - orders sorted by total cost ☐ [4]		/20

process_order.php - *(drop table in phpMyAdmin)* - creates 'order' if it doesn't exist on first order ☐ [10]		/10

Subtotal		**/156**

Enhancements to Specified Requirements *listed and linked* from ***enhancements3.html***

Maximum of 3 Enhancements will be assessed (put your best ones at the top of the list).
Up to 18 marks are available overall. Poorly implemented or trivial enhancements may receive zero marks.

Feature Name	Described	Linked (to place in website)	Sourced (if applicable)	Mark
	Y/N	Y/N	Y/N/na	
	Y/N	Y/N	Y/N/na	
	Y/N	Y/N	Y/N/na	
Total Additions				**/24**

Other Deductions *based on demonstration, documentation, code and file inspection*

Requirement	Max Deduction if requirements not met	Deduct
HTML *(deduct up to 3 marks each)* - Invalid XML - (-2 / page) **up to -6** - Deprecated elements/attributes have been used - Inappropriate use of HTML semantics (e.g. use of when <section> <article> should be used) - HTML usability does not follow standards (e.g. alt on images, label in forms, tables) - HTML Image height, width attributes missing or incorrect - HTML has embedded Style markup. CSS is not fully separated from HTML - Code comments inadequate to inform later code understanding/maintenance	-24	
PHP *(deduct up to 4 marks each)* - Inappropriate header comments - do not match in-house standard. - Code comments inadequate to inform later code understanding/maintenance - Uses only mysqli commands	-12	
Web site - Directory and file structure not as specified - Third party content inadequately acknowledged *- Failure to acknowledge third party code or content at all is plagiarism and may result in zero marks for this assessment, or other penalties in accord with Swinburne policy.*	-4 -4 - All marks	
Total Deductions		

A final assignment mark is *not* provided during the demonstration.
All code is inspected after the demonstration by your tutor, before a final mark is allocated.

Overall Comments:

……

……

Suggested task process:

Just suggestions, ☺ ... there might be things I have left off this quick list ☹

1. Create a new assign3 directory on the server.
2. Copy your assign2 to this folder.
3. Change all the file names from .html to .php, and change all the links in all the files as well.
4. Change all the pages by converting the common content (header, nav, footer) to php server-side includes, eg: `header.inc, menu.inc, footer.inc`.
5. Check that the pages all work and that the delivered pages are still valid HTML5 and well-formed XML and that the linked CSS is still valid.
6. Create your `settings.php` file – as required.
7. Add a link to the `manage.php` in the `menu.inc`
8. Create the database table 'order' in your database on the mysql server, with fields that meet your data requirements for your 'payment' page, plus the additional fields 'order_id', 'order_cost', 'order_time', 'order_status'.
9. Look at your table through phpMyAdmin, understand your field names, load some records, try some SQL queries, and look at the results.
10. Change `enquire.php` and `payment.php` and `part2.js`
    - disable HTML5 form validation,
    - disable JavaScript form validation
11. Create `fix_order.php`
    - a simple modified form version of payment page
    - processes data sent from process_order in a query string (including error messages)
    - form action is to process_order.php,
12. Create `process_order.php`
    - has no html or css
    - process the data from payment page,
    - initially just return all data to fix_order.php in a query string using a simple redirect
    - once the processes are working, add one error check then if okay write the record to the 'order' table
13. Create `receipt.php`
    - a simple echo version of the payment page information, plus order details
    - processes data sent from process_order as post, after oreder has been written to the table.
14. Continue developing `process_order.php` and `fix_order.php`
    - add more data checking in process_order send back data and errors to fix_order
15. Create `manage.php`
    - 3 sections: form to query, table to display results, form to change status
    - create the query form with options, action is post back to manage.php
    - process the self-sent data, query the database table, then display the results in your table.
    - add a 'form' in the table to update status, pass the order_id in a hidden input, and with a hidden 'update' flag, and process the self-sent new post, and fill the 'change status interaction.

All the pages should now be working with new data. – you are now well on the way to a good pass.

16. Probably time to write the create table part of `process_order.php` so you can clean up your test table and start final testing.
17. Test again and cheer.
18. Check off your assignment against all the requirements
19. Have a look at the extensions exercise
20. Test again and cheer
21. Test your assignment again for robustness, before you
22. Zip your assignment and submit it to ESP early

    … ready for the tutor's "final testing" ☺