Swinburne University of Technology

COS10011
Creating Web Applications

Assignment Part 3

Server-Side Programming

Important Information

Due Date	**10am on Monday in Week 12** (Late submission penalty: 10% of total available marks per day)
Submission Method	**Canvas + Mercury**
Demonstration Date	**Your tutorial, Week 12**

Individual Assignment. Contribution to Final Assessment: 40%

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.
- 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.

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 as 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 an **`.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" (**or simply **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 code by putting them in an `if` statement that evaluates a global Boolean variable you create and initialize. e.g.

```
...
if (!debug) {some JS code;}
...
```

Set the flag variable 'debug' to *true* or *false* depending on what mode the code is run.

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 entered in enquire page and payment page, and with errors marked or highlighted. Do not fill the Credit Card details, 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", "fix_order.php" and "receipt.php" pages should not be able to be accessed directly by url through a browser.

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.

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).

The Manager can also 'cancel' (ie. 'delete') an order via this page. Only pending orders can be cancelled.

CSS

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

Enhancements

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

See the Marking 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.

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:

- 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.
- 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.

Note: If you are going to implement functionalities for the manager, please use "admin" as the username and "password" as the password. Your tutor will need these credentials to login and validate these functionalities.

Up to two enhancements will be assessed, i.e., up to 20 marks will be awarded for enhancements.

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 (Mercury + Canvas)

Your assignment should be uploaded to Mercury on or before your deadline.
An electronic copy of your assignment should be submitted through Canvas on or before your deadline.

- Make sure all your files are in the correct folders and compress your root folder with all your

sub-folders with HTML, CSS, JavaScript, PHP, inc, images and etc into a zip file named "assign3.zip". Submit this to Canvas.

- You can submit more than once through Canvas. Your last submission will be marked.
- Note that all deliverables must be submitted electronically. There is no need to submit an assignment cover sheet.
- Turnitin has been integrated into Canvas. Turnitin allows the marker to identify any similarities between your work and that of other students and from the Internet in general.

Make sure you complete your Canvas submission process.

Assignment Demonstration

1. Make sure you attend your allocated lab. You will 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
    a. Fill in and sign the Declaration on the Marking Sheet.
    b. Make sure your website is running on Mercury. (Your tutor will check the URL). All demonstrations will be done on Firefox.
    c. Load Web Developer in Firefox. Validate all your **new/altered** web pages for both HTML5 and XML and the results display in separate browser windows.

    Remember: this is HTML generated by your PHP - not the PHP code itself.

    d. Load MySQL Monitor command line client or the phpMyAdmin web interface so you can demonstrate the changes to your table as your demonstration progresses.

3. As you demonstrate your website your tutor will ask you to explain how you have implemented various aspects of it. In the week following the demonstration tutorial your tutor will mark your source code and documentation.

Mark Sheet ***Marker: ……..***

Declaration:
I hereby confirm that the assignment on the Mercury server is identical to my Canvas submission.

Student number ……. Student name ……..

Signature ……. Date ……

Product ……..

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

A3

Specified Requirements Place ☑ or ☒ in box	**Mark**
PHP common static includes: menu, footer, header, static code blocks included ☐ ☐ ☐	/3
settings.php created, included and used ☐ [1]	/1
Orders Table (view in phpMyAdmin) - scheme can store all the necessary data ☐ [1] - primary key order_id auto generated ☐ [1]	/2
enquire.php payment.php part2.js - client-side HTML5 form validation disabled ☐ ☐ [1] - client-side JS form validation disabled ☐ [1]	/2
process_order.php - unable to access directly through URL ☐ [1] - all data read from payment.php ☐ [2] - text data inputs sanitised ☐ [1] **Data errors validated**, displayed back in **fix_order.php** - fnames ☐ [1] lnames ☐ [1] email ☐ [1] str_addr ☐ [1] state ☐ [1] pcode ☐ [1] phone ☐ [1] - preferred contact ☐ [1] qty ☐ [1] product ☐ [1] options ☐ [1] - total cost calculated ☐ [1] - CC type ☐ [1] CC name (alpha) CC number (15-16 digits) ☐ [1] CC expiry date ☐ [1] - CVV ☐ CC formats checked ☐ [1] post code checked against state ☐ [1]	/4 /18
fix_order.php - all data sent from **process_order.php** to fix_order.php [4] - all data displayed in form ☐ *(CC details not sent shown blank)* [4] - errors displayed in page ☐ [2] - form submits all data back to **process_order.php** ☐ [1]	/11
receipt.php - all data sent from **process_order.php** to receipt ☐ [2] - all order data displayed in receipt page ☐ *(Secure CC details shown ****)* [2] - record added to orders table by **process_order.php** *(view in phpMyAdmin)* ☐ [10]	/14
manage.php - linked from menu ☐ [1] - form to make queries, with required options ☐ [1] - results in HTML table order fields (#,date,product,cost,name,status) ☐ [6] Manager can query: - all orders ☐ [1] - orders for a customer based on name ☐ [2] - orders for a particular product ☐ [2] - orders that are pending ☐ [2] - orders sorted by total cost ☐ [2] - can 'update' status of an order (pending | fulfilled | paid | archived) ☐ [2] - can 'cancel' a pending order ☐ [2]	/8 /7

Create database table automatically (in process_order.php) - creates 'orders' table if it doesn't exist on first order ☐ [10] *(can be tested by dropping table in phpMyAdmin)*	/10

Subtotal	**/80**

Enhancements listed and linked from ***enhancements3.html***

Maximum of 2 Enhancements will be assessed (put your best ones at the top of the list).
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	
Subtotal				**/20**

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

Requirement	Deduction
HTML - Deprecated elements/attributes have been used [-5] - Inappropriate use of HTML semantics [-5] (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) [-5] - HTML Image height, width attributes missing or incorrect [-5] - HTML has embedded Style markup. CSS is not fully separated from HTML [-5] - Code comments inadequate to inform later code understanding/maintenance [-5] - HTML webpages not fully valid [-5 each webpage]	
PHP *(deduct up to 5 marks each)* - Inappropriate header comments - do not match in-house standard. [-5] - Code comments inadequate to inform later code understanding/maintenance [-5] - Uses only mysqli commands [-5]	
Web site - Directory and file structure not as specified [-10] - Third party content inadequately acknowledged [-10] - 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**. [-100]**	
Other Deductions - other deductions not listed above [-100]	
Total Deductions	