# SENG1050 Assessment 3

**Web-Based Assignment 2 (25%)**

**Milestone 1: Checked starting from Week 8 Labs**
**Milestone 2: Checked starting from Week 10 Labs**
**Due Date: 11:59 pm on 25 Oct (Week 12)**

---



---

The objectives of this assignment are as follows:

1. To exercise your ability to create Web pages using HTML and CSS.
2. To transform the XML documents from Assessment 2 (Web-Based Assignment 1) into HTML using PHP.
3. To test your skill in deploying a website in a self-hosted server environment.
4. To evaluate your expertise with JavaScript for client side processing.

---

# 1. Outline

Following the success of the initial website launch, the *DIY Paradise* board have decided to expand the web site to faciliate selling second-hand household hardware products submitted by customers from all over the world. More precisely, your project is now to extend your website to allow people to complete a form listing their second-hand household hardware products with all the details required by *DIY Paradise*.

Your website should still be be simple, friendly and attractive. This can be achieved by satisfying the requirements and completing the tasks in the following sections:



---

# 2. General Requirements

## 2.1 Web Server Requirements

Your website must run on an *Apache* web server which uses *GetSimple CMS* to organise the site files and *PHP 8.2* to implement the server-side behaviour.

Data files must also be stored in appropriate locations of the website. For instance, content images (such as product images) should be stored in the *data/uploads* folder; whereas theme-based images (such as page backgrounds) should be stored in the *theme/<yourfirstname>/images* folder.

## 2.2 HTML Requirements

All the HTML pages that form your website should satisfy the following:

- Coded according to the HTML Living Standard.
- Contain the full HTML document structure, i.e., include <!DOCTYPE>, <html>, <head>, <meta>, <title> and <body>.
- There should be complete navigation capabilities across your website, such that:
  - Each webpage should contain hyperlinks to every other webpage.
  - The hyperlinks are implemented using *relative* URLs.
- The footer of each page should contain a copyright statement including your name, student number, email address, and degree.

### 2.3 CSS Requirements

All visual formatting of the Web site will be implemented using Cascading Style Sheets. All the following CSS requirements should apply to your website:

- A left side margin of at least 1 cm for the main content of all webpages.
- A nice background for the webpages. The text must be readable with the background.
- The text colours of a link before they are visited, after they are visited and during a user action must be different from each other. The link must appear noticeably bigger when you move the cursor over them.
- Use at least one *id* and one *class* selector.
- At least one , used to provide some interesting styling (your choice of borders, background, floating alignment, etc.)
- At least one , used to provide some formatting (e.g., a colour or font change) for a few words of text.

You must use at least one external style file (style.css) to define the style for all web pages you have created. If necessary, document-level style sheet can be used to apply styles that are unique to the particular page. Inline styles should only be used for one-off visual effects.

In general, in order to draw your target audience, make sure that your website considers factors such as navigability, content layout, element positioning on screen, text size, colour contrast, readability, consistency, and accessibility.

---

## 3. Web Server Setup and Website Import

### 3.1 Web Server Setup

This subsection involves setting up and configuring WampServer and GetSimple CMS. This could be easily achieved by following the steps outlined in Task 1 of Lab 6 and Task 1 of Lab 7.

### 3.2 Website Import

Upon completing subsection 3.1, you will import your website from Assignment 1. This can be achieved through the following steps:

- Open the admin panel of GetSimple, and create each page by copying the source HTML code (content of `<body>`) into the editor.
- Move any content-based images (such as product pictures) into the *data/uploads* folder. Change the image *src* attributes in the HTML to point to the new location.
- Create a sub-directory in the *data/pages* folder named *products*. Move your XML files (the template.xml file and the two product category XML files) into this folder.
- Create a new custom theme folder in the *theme* directory. Name it according to your first name. Copy

your external stylesheet *style.css* into this folder. Inside the folder, create a sub-directory called *images* and move any theme-based images (such as background images) into it. Update any image paths in your external stylesheet to point to the new location.

- In your custom theme folder *theme/<yourfirstname>*, create a file called *template.php*. In this file, create a template that links to the external stylesheet, and calls the appropriate methods to get the page data.
- Once again in the admin panel of GetSimple, select the *Theme* tab and change the theme from the default to the one you created.

It is recommended to double check that your webpages satisfy the requirements in subsection 2.2 and subsection 2.3. You might also find it useful to create partial PHP templates for different components of your website, such as the navigation bar, header, footer, etc.

**Note: All files in the folders (*theme*, *data/pages*, and *data/uploads*) must be your own work. You are not permitted to use any third-party plugins or themes for GetSimple CMS, including any that came pre-installed. While you may look at these files for reference, you cannot copy the code, this will be considered plagiarism.**

---

### Milestone 1

Upon completing section 3, approach your lab instructor to check your first milestone during the labs between Week 8 and Week 12. The first milestone aims to check that you have completed your setup of GetSimple CMS and that you have started to import your website from Assignment 1.

The following will be checked:

- The directory structure of your virtual host - the root directory must contain the GetSimple CMS installation in the correct files and folders.
- The default document for your Apache web server while running - this should be your homepage from Assignment 1 which should be stored in GetSimple CMS.
- All webpages in your website should be loading all linked resources (hyperlinks, images, etc.) correctly.
- Your external CSS should be correctly linked from a theme in GetSimple CMS.

Fulfilling the criteria listed above will grant you 1% of your Assessment 3 score.

---

# 4. Category Pages

You will be writing a separate PHP template to dynamically generate HTML pages (via query elements) from your two XML files (for product categories, not template) in Assignment 1. The HTML pages that are generated should fit in with the style of the rest of your website (i.e., similar formatting, colors, etc).

Note that all your XML files have the same tags and DTD. Thus, you only need to make 1 PHP template which allows the creation of a HTML page containing:

- A level 1 heading which mentions the product category you chose in Assignment 1 (e.g., kitchen, plumbing, building, gardening, etc.)
- List of all the products in the category, sorted alphabetically by name and including information in your XML files such as:
  - Product description
  - Pricing
  - Search tags
  - Reviews
  - Any picture(s) of the product
  - URL of a webpage related to the product.

- All extra data that you defined

Since the HTML pages are generated via query elements, your PHP template should be able to read the name of the XML file (i.e., product category) from the GET parameters of the URL. As an example, to access the kitchen product category HTML page, which is generated from *kitchen.xml*, the URL may be http://diyparadise/products?category=kitchen. In this case, the PHP template should be able to retrieve the query string 'kitchen' from the URL. *[Hint: refer to Lab 7 Task 3.]*

You are free to give your PHP template a preferred name, but it must be placed in your custom theme folder (which also contains *template.php*, *style.css* and other theme files).

Upon completing the PHP template, create an empty page in the GetSimple admin panel called *products*. Under Page Options, change the 'Page Template' to your newly created PHP template.

You will also need to include and update the navigation links on your website to load the correct page for each product category. Carefully note that all product category pages link to the same *products* page, with subtle differences governed be the query string *?category=* at the end of the URL.

Finally, in the category pages, you should add JavaScript which allows you to store the name of the category page viewed, along with the date and time it was viewed, in a cookie on the user's browser. Also, in the hompepage, add JavaScript which reminds a returning user of the product category they were last viewing, along with the date and time they viewed it, by reading the stored cookie.

---

# 5. Data Collection Page

As mentioned before, the site staff want a form for *Data Collection* where users can list and provide details about their second hand household hardware products. This will be done as a form, which sends the information as a POST request to a script *submit.php* located in the document root of your website when they click the "Submit" button. The following things need to be included on the page.

1. An appropriate level 1 heading.
2. Information about the person submitting:
    - Their first name.
    - Their last name.
    - Their email address.
    - A binary choice question on whether they have submitted a product before. Apply a default response here.
3. Information about the product - Include *all* information in your XML template, with the following additional requirements:
    - Required tags and attributes (elements with no modifiers or the '+' modifier, and attributes with #REQUIRED) should be required fields on your form before it can be submitted. These fields should be visually marked to the user with an asterisk '*'.
    - A dropdown box for the user to indicate the category of the product. The dropdown box should list all existing categories, and an option for the user to create a new category (you will need to use PHP to get the existing categories if doing the bonus task below).
    - A checkbox to indicate whether the user has read the terms and conditions. This checkbox should be required before the form can be submitted.
    - There should be buttons to add search tags for each product as per the XML. Each individual item should also have a button to delete it (there is a minimum of one, so the delete button of the last remaining item should be removed). Use JavaScript to implement the behaviour of these buttons.
    - If any reviews are entered, each review must contain the phrase "x out of 5 stars" (where x must be a number from 0 to 5 allowing a single decimal place, i.e, 0, 1.1, 3, and 4.5 are all valid). Use JavaScript to check this when the submit button is clicked. The form should not submit if this phrase is not present in **each** review, and an error message should be displayed to the user.

Remember, there could be zero reviews.

- When the user enters or changes any pricing field, separate read-only fields should display the price of the product as well as the corresponding price category of the product. If the price is $0, the price category should be 'Free'. If the price is less than $25, the price category should be 'Cheap'. If the price is above $100, the price category is 'Expensive'. Any price between $25 and $100 falls under the price category of 'Affordable'. The price cannot be allowed to be negative when the form is submitted. Use JavaScript to calculate and update these fields.

4. Buttons to submit and clear the form should be present.

Each field should use an appropriate input type that is intuitive to the user and minimizes the chance of an invalid input being entered.

---

### Milestone 2

As you are progressing through section 4 and section 5, approach your lab instructor to check your second milestone during the labs between Week 10 and Week 12. The second milestone aims to check that you have started working on pulling data from your XML files, and you have a reasonable grasp of the basics of JavaScript.

The following will be checked:

- The PHP template file in your custom theme folder that pulls the XML data. This should correctly open the appropriate file and retrieve the data.
- Both of your product category pages. These should be HTML pages that use a PHP template to pull data from the XML files. These pages are not required to be neat at this point, but you should be able to display all of the data from each file, including links and images, each in their appropriate elements (e.g., and ).
- A working version of one of the yellow-highlighted JavaScript tasks in section 4 and 5. You only need to show one, but it must be fully functional.

Fulfilling the criteria listed above will grant you 1% of your Assessment 3 score.

---

# 6. Responsive Web Design

The site staff want the website to be well-designed, and easily accessible for both humans and computers (especially search engines). To that end, the website must look visually appealing across a wide range of devices, and use a consistent style independent of which web browser is used.

To achieve this, all webpages must link a CSS reset file *reset.css* as the first set of CSS rules to be processed. Download a copy of Eric Meyer's CSS reset here, and place it in your theme folder along with your own external CSS file. Update the template.php file to link this before your own *style.css*. You will need to update your *style.css* to specify styles that were removed by the CSS reset.

All webpages should use a responsive layout, that grows and shrinks when the screen size changes. The website should also use a different layout for small screens (and specifically mobile devices) and you must achieve this by using media queries in your *style.css*. You should set the *viewport* meta tag in your template file to indicate that your webpages are mobile-friendly.

In addtion, you must optimise your website. This involves:

- Optimising images such that overly large files are not unnecessarily loaded for small display sizes. All images should be in a compressed, efficient format such as webp or avif. You can use an image compression service to optimise your images. All images must include a descriptive *alt* attribute that describes the image for screen readers.
- Minimizing content layout shift by setting explicit widths and heights for images.

- Minimizing page load times by reducing the amount of content the browser needs to download in order to initially display the page. As your *style.css* file grows, you might find it more efficient to use multiple external CSS files, to split the styles that need to be loaded immediately, vs the styles that won't be needed until the user interacts with the page in some way. You might also consider using media queries in the link tags to further split the external CSS for desktop vs mobile layouts.
- Adding appropriate information for search engines, such as optimising page titles, level 1 headers, and setting a *description* meta tag (which can be done in the GetSimple CMS admin panel under Page Options).

Finally, check whether your website meets the requirements by using the Lighthouse tool in either Google Chrome or Microsoft Edge to inspect your website. Check both desktop and mobile devices under the categories: performance, accessibility, best practices, and SEO (you can deselect progressive web app tests). Aim to score 100% in all categories (except for the HTTPS check under Best Practices which can be ignored). If the website is loading slower than you expect, try disabling extensions (you can use a command line flag to do this e.g. "chrome.exe --disable-extensions". This will be done when marking.

---

## 7. [Bonus Task] Form Submission Script

You will write a PHP script *submit.php* to receive and process the form for new products.

1. The client can always bypass client-side JavaScript to submit invalid data (remember, zero-trust for client operations). For this reason, you must validate that the submitted form data complies with all of the above requirements.
2. If the submitted data is invalid, the script should return the user to the form page and display an error message indicating what the problem was.
3. If the data is valid, the script should check if the user is creating a new product category or using an existing one. For a new category, the script should create a copy of your *template.xml* file, and name the copy after the new category *(e.g., gardening.xml)*.
4. Once the script has confirmed that the required category file exists (either by creating it, or referencing an existing one), the XML file should be opened, and the new product data should be added. The resulting XML file MUST be valid (that is, it must validate).
5. At the end of a valid submission, the user should be redirected to the appropriate category page showing their new product listed (the page should jump directly to the user's entry - use an ID '#' at the end of the link for this). Note that if a new category is created, the navigation links of the website should be updated to include it. A form submission success message should be displayed on this page to the user.

Successfully completing this task will earn you a 10 mark bonus.

---

## 8. Coding Style

Your HTML code, XML code, CSS code, JavaScript, and PHP code must be neat, clear, indented, and commented.

Other than the specifications mentioned, the way you design your page is up to you. However it should use semantically-correct tags. All of your work (HTML, XML, CSS, XSLT, and Javascript) should be validated and valid (meaning no errors **or warnings**). Any HTML pages generated by PHP must validate.

All code files must include comments for your code, including comments at the top of the file, stating the name of the file, the name of the author and the date the file was created.

---

## 9. Submission

In the GetSimple CMS admin panel, navigate to Backups -> Website Archives, and select Create New Archive Now. You will find the created zip file in *backups/zip* This zip should contain all the files that contribute to the Web site, including HTML, CSS, XML, PHP and image files. This consists, at a minimum of the entire content of the *data* and *theme* directories. You must also include a `README.txt` file, containing your student number, list of the files that are part of your submission, and the original sources of any data, images, etc taken from other places.

Finally, your submission should include a completed Assignment Cover Sheet. You should download a cover sheet template, then fill in the details and include it with your submission (*Note that an Assignment Cover Sheet submitted electronically is deemed to have been signed*) Your assignment **cannot** be marked until a completed cover sheet is received. Both the `README.txt` and assignment cover sheet should be added to the *zip* archive, which should be re-named A2_c1234567.zip (replace 1234567 with your student number).

Submission will be via the Modules page on Canvas. To submit your files, place them all into a single **`.zip`** archive containing every file and submit it using the Web form in the Assignment 1 area. The `.zip` archive should be named `A2_cXXXXXXX.zip` archive where `XXXXXXX` is replaced by **your** student id.

Note that failure to follow submission instructions will incur a penalty. Assignments submitted after the deadline will also be penalized by 10% per day late (including weekend). For example, an assignment worth 20% marked at 78% may be graded as:

- On time: final mark = 0.78 * 1.00 * 20 = 15.6
- 1 day late: final mark = 0.78 * .9 * 20 = 14.04
- 3 days late: final mark = 0.78 * .7 * 20 = 10.92
- 5 days late: final mark = 0.78 * .5 * 20 = 7.8

Extensions will only be granted with a valid and approved Adverse Circumstances application for the assignment. Adverse Circumstances application should also be submitted at least a day before the due.

## Warning - HTML, XML, CSS Generating Tools

*You are **NOT** allowed to use tools which generate HTML, XML, XSLT or CSS code, either from a visual representation or otherwise. This includes DreamWeaver, Microsoft Frontpage, ChatGPT, saving to HTML from MS Word, etc. You must write all code yourself in a **plain-text** editor such as Notepad++. Any assignments that use code generated from a tool will be given **ZERO**.*

## Warning - Plagiarism

*Plagiarism is defined as presenting the work of others' as if it was your own. This includes the copying of content from other people's Web pages (and other sources) without "appropriate referencing" (indicating the parts of your assignment which have been copied and where they were copied from).*

*If you "quote" or use other people's work (or images) in your assignments, then you **must** reference them appropriately (use a README.txt file to do this). Failure to do so will cause you to lose marks for the assignment. An assignment that consists largely of work copied from elsewhere will receive very low (zero) marks, and may be reported to the School for suspicion of plagiarism.*

*The use of generative AI (such as ChatGPT) is forbidden for the writing of any code or for the research task. However, generative AI may be used for purely creative aspects of the website (such as coming up with descriptions, welcome text, and images) as long as such content is clearly marked. AI detection tools will check each file, and any content with a sufficiently high detection score will be reported to the School for suspicion of plagiarism.*

*You can find more information on the University's Policy on Student Academic Integrity in the Course Outline and at Policy Library.*

## 10. Marking Guide

The following serves as a rough guide to how many marks are allocated to each section of the assignment. Please refer to the rubrics on Canvas for a complete breakdown.

Total Marks: 92

- General Requirements: 8
- Web Server Setup and Website Import: 12.5
- Category Pages: 27
- Data Collection Page: 21.5
- Responsive Web Design: 13
- Coding Style: 10
- [Bonus Task] Form Submission Script: [10]
- You may be penalised for poor aesthetic appearence, untidiness, spelling and grammar problems, and not correctly following instructions