# Web Application Assignment

Due: Sunday 11:59pm 6th of February 2022 – **Assessment Weight: 30%**

---

Note:

- Include comments for your student ID, Name, and Practical Class Time at the top of each source file created.
- All instructions given in this document must be followed in order to be eligible for full marks for the Web Application Assignment
- This assignment is not a group assignment; collusion, plagiarism, cheating of any kind is not acceptable. Submitting your work to your TWA site signifies that the work uploaded is yours. If you cannot honestly certify that the work is your own then do not upload it to your TWA site. Breaches of the Misconduct Rule will be dealt with according to the university policy (see the learning guide for more information)
- All files must be uploaded to your TWA web site before submission due date
- Ensure all HTML written is valid. Use http://validator.w3.org to confirm before submission

---

## Assignment Overview:

For this assignment, you are going to be build a basic eCommerce site. To do this, you are given a list of files that you will need to create. These will be a mix of HTML, CSS, Javascript and PHP files. Within some of your PHP files, you will be required to connect to and execute SQL statements on a database. This database will be given to you and will only be associated with your TWA site.

Every concept or skill within this assignment has been demonstrated in lecture or has been part of a practical class exercise. You will be combining all these concepts and skills to build your basic eCommerce site.

### Design and Styling

Within lectures and your practical classes, you have been given the styling and design that needs to be met either via CSS or via a figure. For this major web assignment, you must design and style your own site. You can adopt what has been given in your Practical Class exercises or you can start from scratch. Either way, you must design and style your eCommerce site so it is professional looking. Your chosen design and styling will be assessed.

### Javascript and PHP Validation

All forms or where a user can provide input needs to be validated with both Javascript and PHP. HTML 5 validation will not be accepted as a valid form of validation.

### Web Application Assignment Database

You have been provided with your own copy of the database called **eshop** on your TWA site. To access this database, you have to use a username and password.

The following is a generic representation of the connection information to be used when connecting to your eshop database (you do this with PHP code):

Database Name: `eshop###`
Username: `twa###`
Password: `twa###XX`
Server: `localhost`

Where ### is your twa site number, and XX refers to the first two characters of your TWA site password.

For example, if your TWA site is twa999, and your password is abcd7890, then the following would be your connection information:

Database Name: `eshop999`
Username: `twa999`
Password: `twa999ab`
Server: `localhost`

Using this information, you will use similar code to that below to connect to your database:

```php
$connection = new mysqli('localhost', 'twa999', 'twa999ab', 'eshop999');
if($connection->connect_error) {
  // Echo out if it failed to connect to the database
  echo $connection->connect_error;
}
```

Once connected to your database, you will have access to the `eshop` database and all its data. Figure 1 presents the Entity Relationship Diagram and the schema of your database.

If wanting, within the assignment files the Schema has also been provided – `eshop.sql`

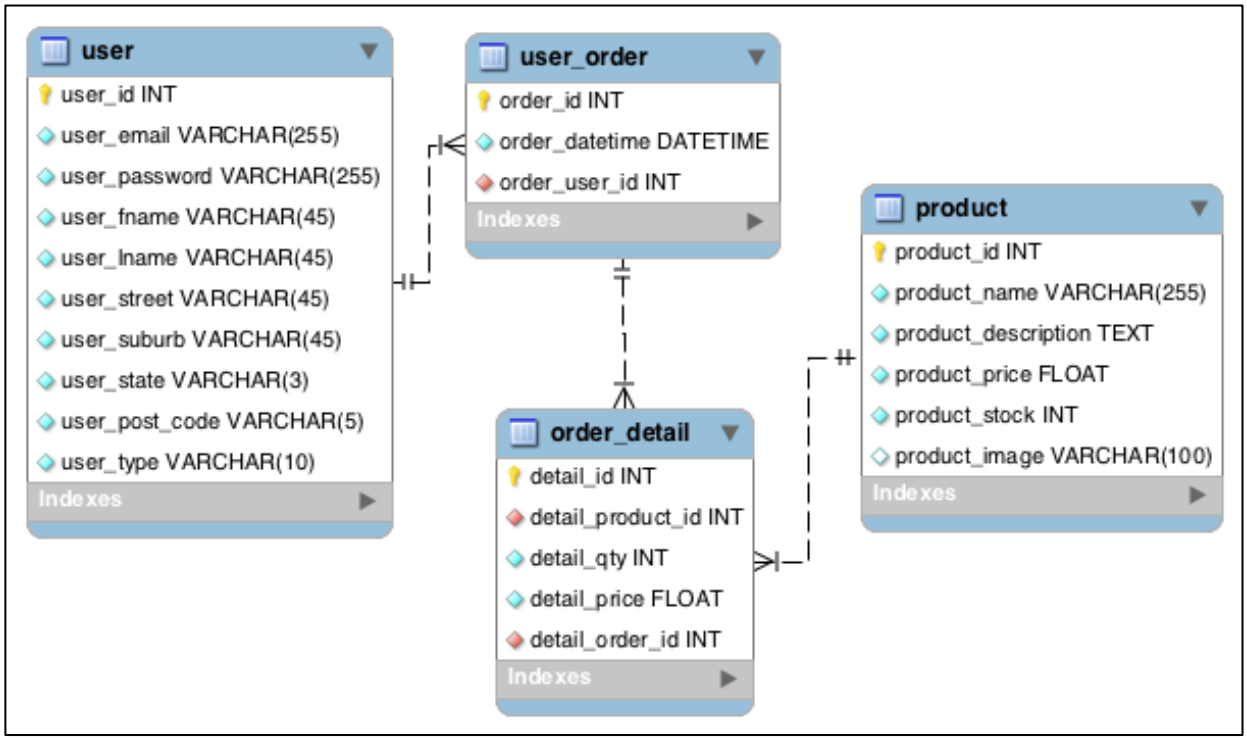


Figure 1

*The* **`eshop`** *Database Data Dictionary:*

| User Table | | | |
|---|---|---|---|
| **Column** | **Description** | **Type** | **Required** |
| user_id | This is an auto incrementing number to uniquely identify a table row. You do not insert this number into the database it is determined automatically | Integer | Yes |
| user_email | This is the email address of the user. It will be used as their login | VARCHAR(255) | Yes |
| user_password | This is an encrypted password. The encryption used is sha256. If you store an unencrypted password the admin cannot login. | VARCHAR(255) | Yes |
| user_fname | This is the first name of the user | VARCHAR(45) | Yes |
| user_lname | This is the last name of the user | VARCHAR(45) | Yes |
| user_street | This is the home street of the user, e.g. 4 Silly Street | VARCHAR(45) | Yes |
| user_suburb | This is the home suburb of the user, e.g. Parramatta | VARCHAR(45) | Yes |
| user_state | This is the home state of the user, e.g. NSW | VARCHAR(3) | Yes |
| user_post_code | This is the home post code of the user, e.g. 2000 | VARCHAR(5) | Yes |
| user_type | This is the type of user they are. There is 'general' and 'admin' | VARCHAR(10) | Yes |

| User Order Table | | | |
|---|---|---|---|
| **Column** | **Description** | **Type** | **Required** |
| order_id | This is an auto incrementing number to uniquely identify a table row. You do not insert this number into the database it is determined automatically | Integer | Yes |
| order_datetime | The date and time the order was placed | DATETIME | Yes |
| order_user_id | The id of the user who placed the order. It is a foreign key to the user table's user_id column. | Integer | Yes |

| Order Detail Table | | | |
|---|---|---|---|
| **Column** | **Description** | **Type** | **Required** |
| detail_id | This is an auto incrementing number to uniquely identify a table row. You do not insert this number into the database it is determined automatically | Integer | Yes |
| detail_product_id | The id of the product that was part of an order. It is a foreign key to the user_order table | Integer | Yes |
| detail_qty | This is the number of items bought with the product id that appears in the detail_product_id column a particular row | Integer | Yes |

| | | | |
|---|---|---|---|
| detail_price | This is the price the product was bought for at the time the order was placed. It is not a multiple of the price per product. It is the price of one product. | Float | Yes |
| detail_order_id | The id of the order this detail belongs to. It is a foreign key to the user_order table's order_id | Integer | Yes |

| Product Table | | | |
|---|---|---|---|
| **Column** | **Description** | **Type** | **Required** |
| product_id | This is an auto incrementing number to uniquely identify a table row. You do not insert this number into the database it is determined automatically | Integer | Yes |
| product_name | The name of the product | VARCHAR(255) | Yes |
| product_description | The description of the product. | TEXT | Yes |
| product_price | This is the price of an individual product. | Float | Yes |
| product_stock | This is the quantity in stock. When a purchase is made the quantity of the product is updated | Integer | Yes |
| product_image | This is the image file name of the product. It does not include the path to the file, just the filename | VARCHAR(100) | No |

### Use of jQuery

IMPORTANT: This should only be used by students with understanding of these libraries. Lecture 6 contained resources for jQuery

If you feel you want to use jQuery to assist with Javascript, you may. Only the jQuery library is allowed. You are **not** allowed to use any CSS framework.

## Required Functionality and Files

Hint: Write your HTML, CSS first before you do any Javascript or PHP programming. This way create your designed design and then add the programming logic. This approach will make development considerably easier. Plus, if you fail to implement the Javascript or PHP you can still get marks for the HTML and CSS used to construct the page.

### Navigation

The site must have a main navigation bar. Where you place this within your page is up to you. Every page should have this navigation bar. Pages which fail to have a navigation bar will lose marks. The navigation bar should display the following links:

- Home – links to `index.php`
- Products – links to `products.php`
- Cart – links to `cart.php`
- My Profile – links to `profile.php`
    - *This should only be present if the user is logged in*

- Order History – links to `history.php`
  - *This should only be present if the user is logged in*
- Register – links to `register.php`
  - If the user is logged in this link should not be present
- Login – links to `login.php`
  - *If the user is logged in this link should change to 'Logout' (`logout.php`)*

### Style and Design – `styles.css`

The styles.css file is your master CSS stylesheet for your site. Although you are allowed embedded and inline styles, a focus on external styles should be maintained. For maximum marks, your style and design should be responsive. This means it can be viewed both on a mobile screen and a desktop. Use the device preview available in Google Chrome under Developer Tools.

### Javascript – `actions.js`

This file is to hold all your javascript required for your pages; all your form validations or actions in javascript.

### Home page – `index.php`

This file is the home page to your site. This page only requires:

- A navigation bar
- A welcome message

### List Products Page – `products.php`

This page lists the products found within the database table `product`. The listing of the products can either be in a rows like *Figure 2* or grid like *Figure 3*.

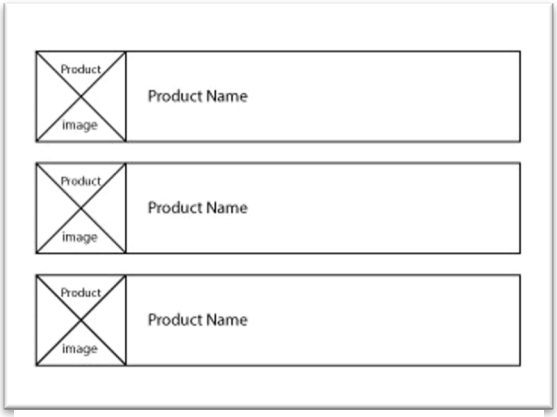


*Figure 2*

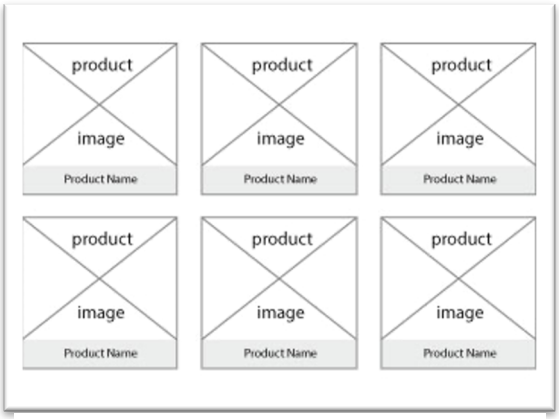


*Figure 3*

The page also allows a user to search for products via the product name via a call back. Remember a callback is where the data is within the URL, i.e. `list-products.php?query=ball` using a `GET` request sent to the page that called it. If a search query does not return any results the user needs to be notified that no results were found. If it does, the results need to be shown. If the user just navigates to the page, by default the page should list all products within the products table.

In your listing, each product must have a Qty option for user input, and an 'Add to Cart' button. When the button is clicked the product id and Qty should be sent to **cart.php** to be added to the items in the cart.

**The Qty and Add to Cart should only be shown if there is stock available**. Otherwise, 'Out of Stock' should be shown. In addition to this:

- The user is only allowed to enter a quantity up or equal to the stock available. Meaning, they cannot add more than what is available in stock to their cart.

- For top marks, if the user already has the product in their cart they cannot add more than what is in stock and their cart. To achieve this the cart checks the available stock and what's in the cart before adding to cart.

The images for each product are found within the image directory within the assignment zip.

Hint: To handle search you just need to capture the search query and modify the SQL for the listing of products

### Shopping Car Page – `cart.php`

This page will show the contents of a shopping cart. The contents of the shopping cart will show regardless if the user is logged in or not. Also, if a user has items in their shopping cart before they login, after they log in, the contents must still be present. In addition to this ability:

- A user can add products to the shopping cart if they are logged in or not
- To display a table similar to *figure 4*. The table must have the product image, product name, price, qty in cart, total for qty in cart for each product. A total for all products in cart must also be present
- The user can only checkout if they are logged in. If the user has not logged in there must be a link or button to allow the user to navigate to `login.php`

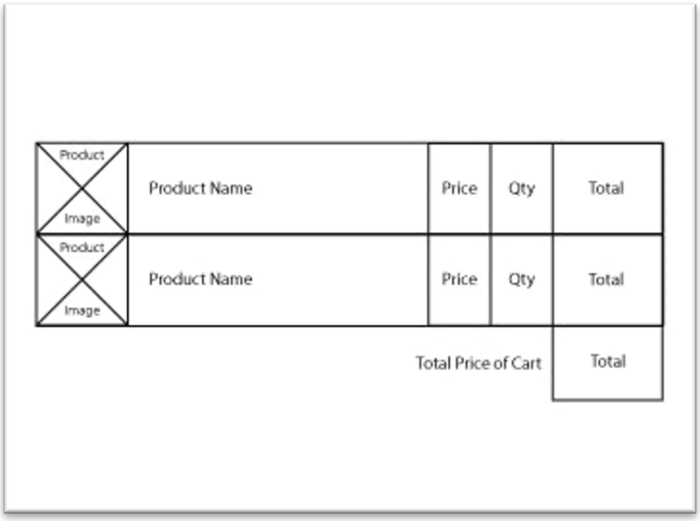


*Figure 4*

Checkout – `checkout.php`

A user must be logged in to see this page, if not they are redirected to `login.php`

This page will present the user a form for payment details. Once the payment details have been validated the order will be processed. Processing of the order means the currently logged in user will have a new order listed with all details of that order (products ordered) within the database. Once processed a thank you message is shown (`process-order.php`) needs to be presented. For the payment details the following is required:

- Credit Card Name
  - Must be alphabetical characters and single inverted commas only
- Credit Card Number
  - Must be 16 digits long
- Expiry Date
  - Must be greater than current date
- CSV Number
  - Must be 3 digits long

These details are not to be stored, it is a mock payment form. However, the mock form must be validated before the order can be processed.

### Process Order – `process-order.php`

A user must be logged in to see this page, if not they are redirected to `login.php`

After a user validates their payment method the user is sent to this page to process their order. To process an order, you must create a new order, and then add the details of that order to the database from the shopping cart. The two tables relevant to this process are `user_order` and `order_detail`.

First, insert a new order into the `user_order` table, and then use the last inserted id to add the products order to the `order_detail` table. Once you have created a new order and inserted all product details into the database for the order, present the user with a thank you message and echo the order id for their reference.

### User Profile – `profile.php`

A user must be logged in to see this page, if not they are redirected to `login.php`

This page presents the currently logged in users' details within a form (except their user id, user type and password). If the details have changed the user can updated details and click submit to perform a **postback** and save the modified details to the database. This page requires the following validations before the **postback** can occur:

- Basic form validation so all fields are required
- The email address is a valid email address form
- Post code must be 4 digits
- State must be 3 characters or less (Hint: use a drop down)

### User Registration – `register.php`

This is the user registration page. When users are registered through this page they become a 'general' user. For this reason, your registration form must not have the `user_type` field displayed to the user. Also, the `user_id` field is not to be entered in by the user. The database automatically populates this field.

To process this form, you are to use a **postback** to submit the data back to `register.php`. Once the form is valid, a new user is saved to the database, and a new user session is created where you must redirect the user to the home page (`index.php`).

When processing the password field, it must be hashed (encrypted). The encryption algorithm to be used is `sha256`. The following code outlines how to use the `hash` function to perform this encryption algorithm:

```php
$password = "234jkldd";
$hashedPassword = hash('sha256', $password);
// $hashedPassword now contains:
// a5d95f8ebc8d8d592cfe772d33d8833909f526233d9ae12c32ada2bbea6a0bba
```

This form needs to be validated before submitting. The required validations are:

- All fields are required
- First name and last name must start with a capital letter
- The email address must be a valid email address
- Post code must be 4 digits only
- State must be 3 alphabet characters or less
- Password must be 8 characters long and must include at least 1 number [0-9]

### User Login – `login.php`

This page will hold a login form containing a username (the user's email) and a password field. Both 'general' and 'admin' users can login through this login form. Remember, when checking if a user's password is valid you will have to encrypt it before you can check if the password is correct. This encryption is the same process found in `register.php`.

A **postback** must be used to login the user. If all credentials are valid, a user session is created and then the user is redirected to the home page (`index.php`). If the credential are invalid, the user is presented with an error informing them the username and password entered is incorrect. Remember to store the `user_type` field in your session, as this will be used throughout the site in the navigation bar to determine if an 'admin' is shown the Users link and if they can get access to the Users list page (`users.php`).

There are two users already present in the database:

- User 1:
  - Username: admin@example.com
  - Password: `admin`
- User 2:
  - Username: user@example.com
  - Password: `user_password`

### Order History – `history.php`

A user must be logged in to see this page, if not they are redirected to `login.php`

This page lists all the orders placed by the currently logged in user. The page only needs the show following details for each order:

- The order number
- The order datetime
- The total cost of the order

### Logoff – `logoff.php`

This page is not a presentation page. When a user navigates to this page their user session is destroyed. Everything within the user's shopping cart should also be destroyed. Once destroyed, the user is then redirected to the home page (`index.php`)

# Submission Instructions

All files are required to be on your TWA site under the `/project` directory before the submission due date. Submitting your work to your TWA site signifies that the work uploaded is yours. If you cannot honestly certify that the work is your own then do not upload it to your TWA site. Breaches of the Misconduct Rule will be dealt with according to the university policy (see the learning guide for more information).

## IMPORTANT: Submission Script Details

1. Prepare a `README.txt` file that lists some customers' usernames and passwords for marks to test your website. If you have some special notes for the markers, you can also include them in this file.
2. Upload all your practical files in the `/project` folder in your TWA website.
3. Run the submission script located at: https://twasum.cdms.westernsydney.edu.au/submit/submit.asp
4. As part of the submission, you will be prompted for your TWA website username and password. You will then be asked to read the WSU policy on plagiarism and certify that work submitted by you is your own work. This action will be logged in a database for future reference and is deemed to be evidence that you claim that your work is original. Next, you will need to select from a drop down list the 'project', and click the "Submit Assessment" button. The web page will then display a listing of the files you have submitted along with a receipt number. You can take a photo/screenshot of or print this page for proof of submission.

Notes:
The submission script may be run more than once if needed. A record of all submissions will be automatically created by the submission script in a set of web log files and in a database for future verification purposes. This submission script copies the files from your web site to a duplicate version on a marking web server. You will not be able to access the marking web server.

Submission Checklist is on the following page:

## Submission Checklist

For a checklist, the list of files which need to be on your TWA site before the submission due date are:

- ☐ `styles.css`
- ☐ `actions.js`
- ☐ `index.php`
- ☐ `products.php`
- ☐ `cart.php`
- ☐ `checkout.php`
- ☐ `process-order.php`
- ☐ `profile.php`
- ☐ `register.php`
- ☐ `login.php`
- ☐ `history.php`
- ☐ `logoff.php`
- ☐ `README.txt`