CSY2028: Web Programming			
Date of Issue	**Tuesday, 27th September 2022**	**Last Date for Submission:**	**TBC (Week 7-Week 8)**
		Module Tutor:	Thomas Butler
This assignment is weighted as 50% of the Module's assessment			
Assessment Feedback is provided on the rubric via NILE			

Submission Guidelines – Please read carefully

1. The University of Northampton's Policy on Plagiarism & Mitigating Circumstances will be strictly implemented.

2. This is not a group project, by submitting this assignment you are asserting that this submission is entirely your own individual work. You may discuss the assignment with other students but any code written should be your own. **Sharing your work with another student, or submitting code that was written by someone else may be deemed academic misconduct.**

3. If you have used any code that you did not write you must:
   i. Provide a reference, with a link where applicable as a comment at an location you have used code you did not write.

4. You must supply **all three** items of assessment and **upload them to the correct submission points**. All work must be uploaded to turnitin
   i. Source code uploaded to github (instructions will be provided separately)
   ii. Source code (zip file). The marker must be able to download and run your code.
   iii. Source code word document (uploaded to turnitin)

5. The website must be built using the docker environment we use in class. The marker **must** be able to run `docker-compose up` and access your website without any additional steps needed to get it running.

Please make sure you double check all submissions. It is your responsibility as a student to ensure these guidelines have been met.

Failure to follow the submission guidelines may result in a capped grade of F-.

If the marker has any questions you may be asked to attend an online viva to discuss your submission, failing the viva will result in a fail grade.

CSY2028 Web Programming

PHP & MySQL Assignment 1

Ensure you have read this brief carefully and thoroughly before attempting to start the assignment.

Aims & Objective
The purpose of this assignment is to assess your ability to create a database driven website using PHP and MySQL.

Brief
You are a back end developer working for web development agency. A startup auction site wants you to build a website which can be used by people to advertise products for sale at auction. The front end developer has supplied an HTML layout with the relevant CSS and Images, this has been signed off by the client and should be used for every page of the website. The contents of the <main> tag will differ on each page, but everything else in the layout should be visible on each page.

Your task is to implement functionality into the website and allow users to advertise products for auction on the website.

To keep things organised, the owner of the website wants to be able to add categories to the website that people can then assign products to. Some example categories are: Electronics, Home, Motors and Fashion, though this is not a complete list and the website should not be limited to these options.

Anyone should then be able to register, then advertise a product on the site and categorise it.

The home page should display the 10 most recently created auctions.

The public will then be able to browse products and filter by category, for example, seeing all Electronics that the shop has in stock.

Members of the public should be able to add reviews to each other. The reviews for each product should be visible below the product description and reviews should only be visible on the page of the product they were posted to. When a user posts a review, their name, email address, review text and date the review was posted should be logged and displayed on the page.

Requirements
The system must use the layout that was supplied by the designer and have two access levels. Admin users who can control the categories on the site and standard users who can post auctions.

1. Have a password protected administration area that has functionality for: *(10 functionality marks)*
   a) Add categories
   b) Edit categories
   c) Delete categories
   d) Added categories should appear on the top of the website.

2. Have a publicly visible front end that allows standard users to: *(35 functionality marks)*
   a) Register an account and log in
   b) Users should be able to post auctions on the website by supplying the product's information: Name, Description and end date/time. Auctions must be associated with the currently logged in user.
   c) When posting an auction, the user should be able to select from one of the categories managed by the administrator and see all auctions in that category (Use the HTML from the "Latest Listings / Search Results / Category listing" section of the supplied layout)
   d) The homepage should display the 10 most recently added auctions
   e) Users should be able to edit auctions they have created to fix typos or update information.
   f) Users should be able to delete auctions they have created.
   g) Click on one of the categories to view products in that category (products from other categories should not be visible)
   h) Click on an auction and see a page displaying information about that auction (Use the HTML from the "Product Page" section of the supplied layout)
   i) Add your own review of a user
   j) The reviews for a user should be displayed at the bottom of any of their auction listings.
   k) The remaining time of the auction should be calculated and displayed on the auction page using the end date/time supplied when the auction was created.

Additional requirements

In addition to the requirements outlined above, additional features can be added to improve your grade, as outlined below.

There are marks available for usability, you should use select boxes/checkboxes/radio buttons in place of text input where applicable and consider how user friendly the website is. Users should never need to type in or remember numerical IDs.

If you have used code from a third party (book, article, tutorial, etc) you **must** include a reference to to it as a comment in the source code at the point the code has been used. Failure to do so may be deemed academic misconduct.

For example

// Code for connecting to the database from https://www.sitepoint.com/re-introducing-pdo-the-right-way-to-access-databases-in-php/
```
$connection = new PDO('mysql:host=localhost;dbname=mydb;charset=utf8',
'root', 'root');
```

Your code must use the file names, field names and structure outlined below.

Note: This will be marked partially via automation. As such, you must use all file names, database table names, and form field names **EXACTLY** as they appear below where indicated, including the correct case. **All database tables must be in schema assignment1 and all files must be placed in the websites/as1 directory, accessed via https://as1.v.je** . Failure to do so will result in potentially losing all marks for the file as the automated marking will assume you haven't completed the work. Time will not be spent by the marker working out what you got wrong and awarding marks for it. **It is up to you to ensure your code uses the file names, field names and database columns EXACTLY (Case sensitive) as specified below or you will not get marks for them**

Your project must include the files below to perform the tasks as described:

Each page must use the consistent header/footer supplied. In addition to the existing links, links to each category should be displayed in the menu

You must produce the following files:

index.php

Purpose: Home page. It must display the 10 auctions which are due to finish soonest.

Requires log in to view: No

Form: None

[Category pages]

Purpose: To display a list of links to auctions in a specific chosen category

Requires log in to view: No

The navigation bar on each page must provide links which take the user to a page that displays the auctions in a specific category. For example, if the user clicks "Electronics" in the menu, it should show only auctions which have been added to the "Electronics" category.

Links to the categories in the navigation must have the HTML class `categoryLink` this exists in the supplied HTML layout and will be used by the automated marking tool to identify your navigation links to categories. **Ensure that class="categoryLink" remains on these links.**

When clicking the links, it should show all the auctions in the selected category using the Category listing block from the supplied layout.

Each auction has its own page linked from the *More* button. This button must have the CSS class **more auctionLink** e.g. *<a class="more auctionLink" href="[auctions-page-name]">Title of the Auction</a>*, the contents of the href attribute can be chosen by you (as per [Auction Pages] blelow) but the *class="more auctionLink"* ***must remain on the link.***

Form: None

[Auction pages]

Requires log in to view: No

Purpose: To display a single auction on its own page. Use the **Product Page** section of the supplied layout. Display the auctions' name, description and category.

In addition to the auction title, description and end date, it should display a list of reviews which have been added to the **auction's author (NOT auction product)**. Reviews added should be associated with a single user and the page should display the name of the person who posted the review.

Form: Yes (x2)

Review form

A form should be displayed to enable users to add a review of the **user** who posted the auction. The add review form should only be visible if a user is logged in and associated with the logged in user when the review is added.

Form fields: reviewText (textarea), submit (submit button)

Form method: POST

Database: table name: **review**, columns: you may use any columns you like for this table

Bid form

A secondary form is required for placing a bid. When a bid is placed, it should log the amount bid, the auction being bid on and the user who placed the bid. The highest bid (not necessarily the most recent), should be displayed on the page if any bids have been made.

Form fields: bid (text), submit (submit button)

Additional optional requirement for better grade: Keep track of the bid history and not just the latest bid user/amount.

---

register.php

Requires log in to view: No

Purpose: Registration form for users to register with their email address, password and a display name

Form: Yes

Field names: email, password, name

Additional optional requirement for better grade: Passwords should be securely hashed in the database

---

login.php

Requires log in to view: No

Purpose: When the form is submitted the user is logged in

Form: Yes

Fields names: email, password, submit

(Additional requirement for better grade): Passwords should be securely hashed in the database

---

logout.php

Requires log in to view: No

Purpose: If a user is logged in, logs the user out when the page is visited

Form: no

---

addAuction.php

Purpose: Adds an auction to the website by writing it to the database. The user will enter the auction, title, description and the auction will be associated with the logged in user.

Requires logging in to view: Yes

Form: Yes

Form method: POST

Form field names: title, description, category (select box), auction end date (date selector or text box)

Requires login: Yes

Database: Table name: auction, columns: title, description, categoryId, endDate

Additional optional requirement for better grade: Allow uploading an image. This must be an image upload which, when chosen, is stored in the public/images/auctions directory. It is up to you how to name the image, but it must be in this folder and displayed on the page for the auction it is associated with.

---

editAuction.php

Purpose: Allow editing an auction. All input boxes must be pre-filled with the title and content of the auction being edited. Auctions must only be able to be edited by the user who posted them.

Form: Yes

Same fields as the addAuction page

Additional optional requirement for better grade: Add a delete button which deletes the auction from the website.

---

adminCategories.php

Purpose: A list of all categories which allows the admin to manage the categories in the database. The page will present a list of a all the categories in the database and each category will have a link to edit or delete it. There must also be a link to add a new category (addCategory.php)

Requires logging in to view: Yes, admin users only.

Form: No

---

addCategory.php

Purpose: Add a category to the website. When a category is added here, it should appear in the Categories submenu at the top of every page.

Requires logging in to view: Yes, admin users only.

Form: Yes

Form fields: name

Database: Table: category, columns: name

---

editCategory.php

Purpose: Edit a category on the website. A specific category is chosen and can be renamed. The original category name should appear in the text box for editing.

Requires logging in to view: Yes, admin users only.

Form: Yes

Form fields: name

Database: Table: category, columns: name

---

deleteCategory.php

Purpose: When visiting this page by clicking the "Delete" button/link on adminCategories.php, the requested category should be deleted

Requires logging in to view: Yes, admin users only.

Form: No

Below are optional pages which can be added to improve your grade

userReviews.php

Purpose: Display all reviews posted by a specific user. In the review list for any given auction, a user should be able to click on the name of the user who posted a review and see a list of all reviews posted by that user.

Requires logging in to view: No

Form: No

---

addAdmin.php

Purpose: Allow admins to create new administrator accounts. An account should be able to be added and the user should then be able to log in with the newly added account.

Requires logging in to view: Yes, admin users only.

Database: Up to you, but a user added must be able to log in via login.php

---

manageAdmins.php

Purpose: The page should display a list of admin users with an option to add and delete them

Requires logging in to view: Yes, admin users only.

Form: No

editAdmin.php

Purpose: Allow admins to edit existing administrator accounts. An account should be able to be edited and the user should then be able to log in with the newly added account.

Requires logging in to view: Yes, admin users only.

Database: Up to you, but a user edited must be able to log in via login.php with updated details

---

search.php

Purpose: Allow using the search bar at the top of each page to search for products matching keywords entered. Both descriptions and titles should be searched for occurrences of the keyword. Results should be displayed in the same way as the products listed on the home page/category page and ordered by relevance.

Form: Yes

Form fields: search (text box) – used from the top of any page

---

Submission Requirements

1. Your supplied website must be built using the Docker Environment used in class and your assignment must be placed in **websites/as1/public and be visible on https://as1.v.je**. You should zip the *websites* directory and make sure that the file *database.sql* is included. **You must run docker-compose down before submission to trigger a database export.**

2. The company's web server uses PHP 8. The website you build must work on this version and not use functions that have been removed or deprecated (e.g. old mysql functions. Using removed functions such as mysql_connect() will result in an F- grade as the website will not work on the company's web server). *If the website works in the docker environment, it will work on the company's web server.*

3. Each page on the website must use the layout supplied by the designer. You may make changes to it and extend the CSS but the overall layout should remain the same.

4. Excluding the supplied layout and code from CSY2028 lecture notes, any code you submit which you did not produce yourself **must be referenced and you must make it clear in your code which sections of the code you didn't write and where you found it.**

5. This is an **individual assignment** and any code you submit should be written by you unless properly referenced. This is not a group project and any work submitted must be your own. The

University of Northampton Policy on Plagiarism & Mitigating Circumstances will be strictly implemented. By submitting this assignment you are asserting that this submission is entirely your own/individual work.

Failure to follow the submission requirements may result in a capped grade of F+.

Deliverables

1. Source code zip file, zip up the directory and submit it to NILE

2. Source code as above, but as a **private github link.** Instructions will be provided for setting this up closer to the submission deadline.

3. Source code word document. Copy and paste all php files into a single word document and upload it to turnitin.

   The source code must be well documented with relevant comments. Consistent and clear indentation of the code is also important. The github repository and zip file must contain the websites directory and docker configuration files. The marker must be able to extract the files and type “docker-compse up” and access your website

   Only upload work you wish to be marked. Your zip should not include multiple copies of the work e.g. *assignment1*, *as1, as1backup, as1version2, etc*. There must only be one version of each page. For example, addAuction.php must only exist once in the source code upload.

If the marker cannot run the docker-compose up command and view your website, your work will be capped at F-.

Marking Criteria

The grade for this assignment will form 50% of the overall assignment grade for the module. The rubric below gives an indication of how the marks are split. In general the following criteria will act as a guide to what you should expect:

	A	B	C	D	F	G
Functionality (85%)	All basic requirements and all additional requirements completed	All requirements and some additional requirements completed	All basic requirements completed and working	Over 75% of basic requirements completed	Less than 75% of basic requirements completed or Code does not use correct file names, database table names, etc outlined in the brief or Code does not run on the docker environment we are using	No submission or no submission of merit
Code quality (15%)	Everything for B plus code is broken up into reusable functions	Everything for C plus Use of arrays and loops where applicable. Correct use of $_GET and $_POST	Attempts have been made to reduce repetition, code is consistently formatted.	Code works but not attempt has been made to reduce repetition. Sections of code are repeated multiple times throughout the project.	Code submitted does not work as intended, contains many bugs or errors.	No submission or no submission of merit