

| CSY2028: Web Programming | | | |
|---|--------------------------------------|--------------------------------------|---|
| Date of Issue | Friday, 30th October 2020 | Last Date for Submission: | Sunday, 17th January 2021 23:59 |
| | | 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. Correctly reference the code in the report (use Harvard referencing)
 - ii. In your report, clearly document which lines of code you used, where you used them and what they are used for.
4. You must supply **all four** items of assessment and **upload them to the correct submission points**. All work must be uploaded to turnitin
 - i. Report word document (uploaded to turnitin)
 - ii. Source code (zip file). The marker must be able to download and run your code. *Do not include your video in your zip file*
 - iii. Source code word document (uploaded to turnitin)
 - iv. Video demonstration
5. The website must be built using the virtual machine we use in class. The marker **must** be able to run `vagrant 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.

CSY2028 Web Programming

PHP & MySQL Assignment 1

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 for a web development agency. A local newspaper wants you to build a website which they can post news articles to. 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 the look of the website.

Your task is to implement functionality into the website and allow users at the newspaper to post news articles. Each news article will have a title, a date, an author and the article's text. The company should be able to log in to a secure administration area and add news stories, entering all the information for the article, making it visible to the public.

Each article will fall into a category such as Local News, Sport, Technology or Business. Categories should not be limited to these choices and new categories should be able to be added via the administration area.

The public will then be able to browse news articles and filter by category, for example, seeing all news in the *Sports* category.

Users should be able to add comments to each news article. The comments for each news article should be visible below the article text and comments should only be visible on the news article they were posted to. When a user posts a comment, their name, email address, comment text and date the comment was posted should be logged and displayed on the page.

Along with the working website, you must provide technical documentation so that other developers your company can easily work on the website.

Basic Requirements (Grades D- to C-)

The system must use the layout that was supplied by the designer.

1. Have a password protected administration area that has functionality for: *(20 functionality marks)*
 - a) Adding news articles
 - b) Adding categories
 - c) Assigning articles to categories
 - d) Editing an existing article e.g. changing the title or text
 - e) Deleting articles from the website
 - f) Editing category names
 - g) Deleting categories
2. Have a publicly visible front end that allows users to: *(20 functionality marks)*

- a) Browse all the news articles displaying newest first
- b) Views a list of news categories in the drop-down menu in the supplied HTML layout
- c) Click on one of the categories to view news articles in that category (news from other categories should not be visible)
- d) Add a comment to a news article
- e) See comments added to that article by other users. Comments should be visible on the news article page, and only comments for the selected article should be visible.

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.

It is up to you how you structure your application and you may extend it with additional functionality that you think would be useful. Possible enhancements include:

- Moderation of comments. When a comment is added, it's placed in a holding area in the administration area for administrator approval before appearing on the website (*10 functionality marks*)
- Allow uploading an image with the news article and have it appear on the page with the title and news article text (*5 functionality marks*)
- Social media buttons allowing users to easily share news articles (*5 functionality marks*)
- Allow searching news stories by typing in a search term (*5 functionality marks*)
- Allow administrators to manage administrator accounts. Admins should be able to create, update, and delete other admin users who can then log in and post stories. Stories posted should be associated with user who posted them. The news article's author should be visible on the news article's page (*10 functionality marks*)
- Securely store passwords with an appropriate method (*5 functionality marks*)
- Prevent users from adding comments unless they create an account. Users should be able to register an account and log in, once logged in they should be able to comment on an article by entering only their comment text. (*10 functionality marks*)
- Allow comments to be *nested* so that users can comment on other user's comments (*10 functionality marks*)
- Allow users to sign up to be notified via email when new articles are posted (*5 functionality marks*) *n.b. you will not be able to test this but you can include the code*
- The ability to click on a user and see any comment they have made (*5 functionality marks*)
- The ability to see all news articles posted by a specific author (*5 functionality marks*)

There are marks available for code quality, you should try to avoid repeated code and use Object-Oriented Programming as well as valid HTML and CSS.

Technical Report

In addition to the website you are required to provide technical documentation for other developers who will be working on the website.

1. A checklist of implemented features that follows this format:

You must include a checklist of functionality you have included.

| Functionality | Completed? (Yes/No/Partially) | Relevant files containing code for this feature | Any comments (e.g. usernames/passwords, |
|---------------|----------------------------------|--|--|
|---------------|----------------------------------|--|--|

| | | | known bugs/issues) or URL if different to filename |
|--------------------------|------------|------------------|--|
| Administrator log in | (e.g.) Yes | (e.g.) admin.php | (e.g.) Username: admin, password: admin |
| Add categories | | | |
| Edit categories | | | |
| Delete categories | | | |
|[extend as needed] | | | |

Please extend the checklist with any additional functionality you have implemented.

2. Testing table

Along with the checklist, you should provide documentation about how you tested the website. Please use a testing table. A good test log provides enough information for someone else to repeat the test exactly.

3. References

The references section should include a list of references and documentation of where you used the resource in your code. For example, if you use a login system tutorial you must explain where in your code the tutorial has been used.

Submission Requirements

1. Your supplied website must be built using the *Vagrant* Virtual Machine used in class. You should zip the *websites* directory and make sure that the file *database.sql* is included. **You must halt the virtual machine before submission to trigger a database export.**
2. The company's web server uses PHP 7. 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 virtual machine, it will work on the company's web server.*
3. Your 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 report 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 or fail grade.

Deliverables

1. Checklist/Testing report (Guideline length 500 words, use tables where appropriate).
 - a) Checklist of completed features
 - b) Evidence of testing:
 - i. Test logs providing information of all the tests carried out (including any failed tests for functionality not implemented).
 - c) References (use Harvard referencing). If you have used code from a book or that you have found online you **must** indicate clearly which parts of your code they are and include references. Failure to reference code you have used may result in a capped grade or failing the assignment.
2. Source code
 - a) The source code must be well documented with relevant comments. Consistent and clear indentation of the code is also important. You must submit the source code in two forms:
 - i. A zip file containing the *websites* directory directories as well as the Vagrantfile. The marker should be able to extract the files and type “vagrant up” and access your website.
 - ii. A commented full listing as a word document named “Appendix”. Copy and paste all the PHP code into a single word document and upload it to turnitin.
3. Video Demonstration
 - a) In Addition to the report you must provide a video demo of your assignment. You must demonstrate every feature you have implemented and do not need to show the code in the video. The demo should be 5-10 minutes (No longer than 15 minutes) and uploaded to Kaltura.

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 (65%) | 75+ functionality marks | 60-74 functionality marks | 50-59 functionality marks | 40-49 functionality marks | 0 – 39 functionality marks or Basic requirements not met. Marks will be lost if the website is unnecessarily difficult to use e.g.: <ul style="list-style-type: none"> • Users having to remember numerical IDS • Users have to type in information that is already in the database • Adding/modifying data on the website requires manually amending the database or editing PHP files | No submission or no submission of merit |
| Report: Checklist, testing and references (15%) | All functionality has been tested. Tests include information on exactly what was tested: The value entered, the expected outcome and the actual outcome. Tests cover all functionality and no unidentified bugs remain. There is enough information for someone else to replicate the test exactly. Complete checklist with relevant notes All used code referenced using Harvard referencing | Most functionality has been tested. The value entered, the expected outcome and the actual outcome. Tests cover all functionality and no unidentified bugs remain. Complete checklist with relevant notes All used code referenced using Harvard referencing | Tests have been carried out on most features and documented but lack critical information on exactly what was tested e.g. the value entered and expected outcome are not present in the test logs Complete checklist with relevant notes All used code referenced | Some features have been tested but the testing strategy is not comprehensive and obvious bugs have not been found during the testing process. Incomplete checklist Missing references | Unspecific, ambiguous, needs some overhaul No checklist Incomplete references | No submission or no submission of merit |
| Code Quality & Efficiency (15%) | Everything for (B) plus excellent use of available tools: Arrays, Loops, Objects and Classes. Making changes to the website does not require making the same change in multiple locations. | Consistent and clear file/folder structure with a focus on security: Relevant files outside the public directory and good password hashing | Program works but functions/arrays/loops/objects would reduce complexity. Any repeated code (e.g. database connections) are placed in their own files and included when necessary. | Some attempt has been made to reduce repeated code. | Marks are lost if: Sections of the code are never used/irrelevant. Code is repeated frequently and no thought has been made into the structure of the code. Chunks of code are repeated on every page or very inefficient, e.g. changing a password requires editing every page. Links are hardcoded and would not work if run from a different URL Adding auctions/categories to the website requires editing PHP files or manually changing the database | No submission or no submission of merit |
| Demonstration (5%) | Covers all implemented features in sufficient detail. Any known bugs are highlighted. Validation is tested (e.g. entering invalid values) | Covers all implemented features in sufficient detail. | Covers the functionality from the basic requirements in detail. Any known bugs are highlighted. | Covers the functionality from the basic requirements but needs more detail. | Video does not demonstrate all of the basic requirements. | No submission or no submission of merit |