ASSESSMENT BRIEF	
Module Title:	Web Programming
Module Code:	KF5002
Academic Year / Semester:	2022-23 / Semester 1
Module Tutor / Email (all queries):	Garry Elvin garry.elvin@northumbria.ac.uk
% Weighting (to overall module):	100%
Assessment Title:	Assignment Outline and Requirements
Date of Handout to Students:	Week commencing the 31st October 2022
Mechanism for Handout:	Module Blackboard Site
Deadline for Attempt Submission by Students:	Wednesday the 11th of January 2023 before 4.00 pm
Mechanism for Submission:	Upload to Module Blackboard Site and to the newnumyspace web server (see below)
Submission Format	zip file and web files (see below)
Date by which Work, Feedback and Marks will be returned:	During the week commencing the 6th February 2023.
Mechanism for return of Feedback and Marks:	Mark and individual feedback will be uploaded to the Module Site on Blackboard. For further queries please email module tutor.

LEARNING OUTCOMES

The learning outcomes (LOs) for this module are:-

LO1 A critical understanding necessary for developing dynamic, interactive web-based applications using both server-side and client-side technologies and of issues related to their use

LO2 A critical application of appropriate server and client side processing techniques and technologies to develop a dynamic, interactive web based application, considering relevant security issues

LO3 Define and apply an appropriate range of tools, methods and technologies to solve a given web development problem and implement a standards compliant web application

LO4 Ability to work individually to develop a web-based system, managing time and resources

This assessment addresses learning outcomes LO1, LO2, LO3, and LO4.

Mechanism for Submission of Work

This assignment **must** be submitted electronically in two ways as follows.

a. FTP ALL of the files (e.g. HTML, graphics, CSS, php, js etc.) that you use to implement this assignment to your newnumyspace account by **Wednesday the 11th of January 2023 before 4.00 pm** (your files should not be altered after that time). *It is recommended that you create a sub-folder of a name of your choosing inside public_html and put your assignment files in there.*

   Your web solution MUST be designed to work on your newnumyspace web space. You should test your web solution to ensure that it is working prior to the specified hand-in date/time.

   If you fail to FTP your files to newnumyspace correctly or do not make your work accessible via a web address on newnumyspace by the specified deadline it will not be possible to mark your work.

b. Submit **one** zip file (this MUST be a zip file and not an alternative like a rar file) containing a **zipped** copy of all of the files (e.g. html, graphics, CSS, php, js etc.) that you have used to complete the assignment using the link provided under **Assignment Submission** in the Assessment and Submission area on blackboard for this module by **Wednesday the 11th of January 2023 before 4.00 pm**.

   Instructions for submitting your assignment solution via blackboard

   1. In Blackboard go to Assessment
   2. In **Assignment Submission** section go to View assignment > Add Content and do two things (you MUST do both)

      a. enter the **EXACT** web address of the home page for your solution to the Assignment. ***You MUST do this. If you don't it will not be possible to mark your work***
      b. insert you zip file and click **Submit** (note that you must do this)

      You must only submit only **ONE** zip file that contains all of the files you used to complete the assignment. Do NOT submit lots of files individually. *The zipped copy of the files that you submit using via blackboard must be **identical** to the files you uploaded to your newnumyspace account.*

Hints:

- *Make sure all links, images etc. use relative web addresses to avoid problems*
- *It is best to have all file names in lower case, and not to have gaps in file names*

Assignment Outline and Tasks Required

This is an individual assessment. It is worth 100% of the marks available for the module and will involve starting the development of a web based system. This document gives the scenario for the assignment and the specific requirements (*make sure that you read carefully*).

Assignment Case Study Background

Northumbria Music Company (NMC) (a fictional company) sells a range of vinyl records. They have asked you to build the first stages of a new online presence for their company.

Assignment Development Tasks

Pre-task

An SQL script, `NMC.sql`, has been provided on Blackboard in the Assessment and Submission section that contains the SQL statements required to create five database tables called 'nmc_category', 'nmc_records', 'nmc_special_offers', 'nmc_publisher' and 'nmc_users' and to add a number of records to them in MySQL. You will need these tables and database records for various tasks. You **MUST NOT** alter this file in any way.

1. Download the file `NMC.sql` from Blackboard
2. Import it using phpMyAdmin to create the MySQL database tables referred to above.

Task 1 - Administrator functionality

1. Create a facility that allows the administrator of the site to edit details of a record currently held in the database. Components:

    - *Choose a record to edit:* the administrator should be able to choose a record to edit from a list (for best marks created dynamically from the database records using PHP, PDO and SQL) of all the current records held in the nmc_records database table by clicking on its recordTitle

      Full details of all records must be included in the list, **except** the publisher. You will need to format the record details (their layout etc.) in an appropriate way. Records should be ordered on recordTitle and you will need to display the category description (not catID)

    - *Edit details of record:* after choosing an record (clicking on its recordTitle) to edit, the user should be able to edit **all** of the record's details (including publisher), **except** the recordID, using a form and the record's details should be amended in the MySQL database.

      Ideally the form should initially display the current category description and publisher name as the default values in pre-defined lists that are dynamically generated from database content with one option for each of the categories and publishers, respectively, in the database. You should program this functionality without generating duplicate entries in the list.

    - *Server-side data validation and making data safe*: the validity of the data entered should be checked and the values made safe for each field used (using PHP) – see task 3 for more details on the requirements for this

Notes

- *To achieve this functionality you should use PHP, including PDO (and not any other database library like mysqli), and SQL and the methods covered in the module*
- *It is expected that any output generated by PHP should be HTML compliant*
- *You should not display system type attributes to the user, they're of no interest to the user, so make sure you don't display any of the ids used in the SQL tables, e.g. catID and pubID. You will need to use them, of course, in list boxes or as hidden fields etc. in order to link back to other tables but you shouldn't display them to the user. Check you don't.*

Task 2 – Logon

1. Create a facility to allow the user to logon via a form. Ideally for the non-logged in user, the logon form should be accessible on every page within the site in a consistent location (such as in the top-right area of the page). For the logged in user, a logout feature should instead be included in the same location. Logging in will cause the login form to show a logout mechanism and logging out will display the login form.

   Only users who have successfully logged in should be able to gain access to the '*Edit details of a record*' (the edit form) functionality to edit details of a record currently held in the database (see task 1). Without logging on successfully via the login form, the user **should not** be able to gain access to the edit form. Components:

   a. *Form elements*: these should allow the user to enter a username and a password (from those provided in the table below)

Username	**Password**
admin1234	45$Qr87$482d
admin1235	29t$tGf2583#

   b. *Logon script:* this should check that the username entered by the user in the logon form exists in the nmc_users table. If it does exist, the script should then use the PHP **password_verify** function to verify whether the password entered by the user in the logon form was valid (correct) for the given username, using the password hash retrieved from the database table nmc_users for the given username and the methods covered in the module
   c. *Sessions*: sessions should be used to restrict access to the administrator functionality
   d. *Server-side data validation and making data safe*: see task 3

   Notes

   - *To achieve this functionality you should use PHP, including PDO (and not any other database library like mysqli), and SQL and the methods covered in the module*
   - ***If you can't get the logon functionality to work, but have attempted the administrator functionality, make sure that the marker can still access the administrator functionality in order to mark it***

Task 3 – Server-side data validation and making data safe

1. For all areas of functionality that involve the user entering /selecting data (such as the admin edit pages or clicking a link which sends data), use appropriate *server-side code* (i.e. PHP) to:
   a) Check that the data entered / selected is valid before it is added to the database. This will include, as appropriate, validating that values have been entered (i.e. no empty fields), validating that the values entered for each field used are appropriate (i.e. validate data type/length etc.)
   b) Make data safe before it is added to the database to, for example, reduce the likelihood of successful SQL injection attacks. This will include using prepared statements

   Notes

   - *To achieve this functionality you should use PHP, including PDO (and not any other database library like mysqli), and SQL and the methods covered in the module*

Task 4 pre-task - download the orderRecordsForm script

A PHP server-side script, `orderRecordsForm.php`, has been provided for you in the assessment section for the module on blackboard. This contains code to build a form, including PHP code to dynamically generate the html to display one checkbox for each of the records currently held in the nmc_records database table within the form. The user can select one or more records that they are interested in ordering by clicking the checkboxes.

In order to understand the structure of the form html, its attributes, IDs and classes, you should load orderRecordsForm.php from the server in your browser and use the browser to view the source.

1. Download a copy of `orderRecordsForm.php` from the assessment area on blackboard

Important notes

- you **MUST NOT** in any way change the php code provided OR the html within the body tags, **except** to
  - Add Javascript for task 4
  - Modify the page to add your own navigation menu and link to your own stylesheet using additional html or php. *However, please note that styling will not be marked for this assignment*

Task 4 - Select record(s) page

*Note that any JavaScript code that you include for this task **MUST** be written by hand and the use of external libraries is **NOT** permitted. In addition, the JavaScript used **MUST** follow the techniques covered in this module and not any other alternative ones.*

1. Create Javascript code to add the following functionality to `orderRecordsForm.php`

a. Make the text saying "I have read and agree to the terms and conditions" change to the colour black and remove the bold if the user checks the terms and conditions checkbox, and return to the original formatting if they uncheck the checkbox

b. The form's submit button should be disabled until the user has checked the checkbox to say that they have read and agree to the terms and conditions

c. It should not be possible to submit the form if the user does not enter a value into the text entry fields, e.g. for forename or surname, and at least one record has not been selected (remember that users don't always fill out forms in top to bottom order)

d. The dynamically created form contains details of different records. Next to each is a checkbox that has been given a data-price attribute corresponding to the price of a particular record. The user is also given the choice of delivery method for the records that is selected via a radio button, each with a corresponding price.

   Use Javascript to calculate the total cost of the order based on the record checkboxes that are selected and the choice of collection method. The total cost should be displayed appropriately to allow the user to consider the cost before committing themselves to buying a record. If the user un-checks all of the records that they had previously checked, the total cost should always go back to zero.

Note that there is no need to actually process the order and store it anywhere. A full system would require this kind of functionality, but this is beyond the scope of the assignment.

Task 5 pre-task– download and set-up files

1. Download the file `getOffers.php` from Blackboard and place a copy into your public_html folder in your space on the newnumyspace web server. You **MUST NOT** alter this file in any way.

   Take a look at the code. You will see that this script when requested by a web browser is designed to query the nmc_special_offers and nmc_category tables and retrieve one special offer record at random. The function getJSONOffer is called and sends details of the special offer in a JSON format to the requesting web browser. In this way the script acts as a web service that can be used by different web pages.

   Note that the third line of the code is **`require_once('functions.php');`** *You may need to modify this so that it refers to the name and location of any file in which you have your database connection code that you have created.*

Task 5 – Home page

1. Create a basic home page that
   a) provides links to the other pages in the site
   b) contains an aside element with the id '**offers**' and a suitable html sub-heading with the text 'Special offers'

2. AJAX in the home page
   a) Use appropriate AJAX code to request the getOffers.php script when the home page of the site is loaded. Using the JSON data in the response to the AJAX request, display the record title, category description and price in an aside tag with the id '**offers**' inside the **home page** (it **MUST** be in the home page otherwise ZERO marks will be awarded for it). Do not display any ids. The record title should be wrapped in appropriate html heading tags (e.g., h2, h3…). Similarly, wrap the category description and price in appropriate html tags

   b) Every 5 seconds a new special offer should be requested and then displayed inside the 'offers' aside in the **home page**

   Notes
   - *The* AJAX / *JavaScript code that you include **MUST** be written by hand*
   - *The use of external libraries is **NOT** permitted*
   - *The code used **MUST** follow the techniques covered in this module and not any other alternative ones.*

Task 6 - Credits page

1. Create a web page that identifies you (include your *name* and *student ID*) and credits all the sources of material either: used specifically as a component within your site; or, which contributed to the overall development of your work (e.g. books, web sites).

   Please credit ALL sources using the Harvard referencing method that you use for anything, i.e. photos, graphics, logos, widgets, text, books, web sites etc. If you have created graphics or taken photos yourself, please credit yourself. Please note that we are aware that there are sites on the Internet that provide code. Do **NOT** submit code from other people or web sources as your own, **this is academic misconduct**. The module team are aware of such sources. We realise that the Internet coding community encourages sharing and re-use of code. The purpose of this assignment is to show us what YOU can do not that you can copy somebody else's code.

Important notes

- *Test your html code (including any generated using PHP) to ensure that is HTML5 compliant and validates using the W3C Markup Validation Service (this is available at http://validator.w3.org/)*
- *You should test your web page using various web browsers (e.g. Chrome and Firefox) and at different screen resolutions to make sure that it displays as you would expect it to in all.*

Assessment Marking Scheme

Category	Marks
Administrator functionality - edit/update details of a record	**22**
• A dynamically generated list (from db content) to allow the user to choose a record to edit (records should be ordered on recordTitle)	**8**
• Edit *form* fields are populated with existing data (all fields should be editable, except the recordID)	**9**
• Records are actually updated	**5**
Logon functionality	**10**
• Users can successfully logon with a correct username and password and can logout on every page	**6**
• Use of sessions to restrict access to the edit functionality	**4**
Server-side data validation and making data safe for all relevant areas of functionality	**8**
• Check that the data entered is valid before it is added to the database	**4**
• Make data safe before it is added to the database	**4**
Select record(s)	**25**
• Client-side validation that at least one record is selected (checkboxes)	**3**
• Client-side validation for text entry form fields, e.g. for surname, to check if completed	**3**
• Terms and conditions	
○ must be checked before form can be submitted	**3**
○ make the text saying "I have read and agree to the terms and conditions" change to the colour black and remove the bold if the user checks the terms and conditions checkbox, and return to the original formatting if they uncheck the checkbox	**3**
• Total price calculation and display	**13**
AJAX functionality for the home page	**13**
• On page load, get a special offer, send the response in a JSON format and display it in an aside with the id '**offers**' in the home page. Wrap the record title, category description, and price in appropriate html tags	**7**
• Every 5 seconds get and display a new special offer in the aside with the id '**offers**'	**2**
• Re-usability of the code	**4**
Build quality	**22**
• Use of functions (e.g., external .php and .js files, range of functions, parameters, returning values)	**12**
• Valid HTML	**5**
• General quality of code – e.g., use of exceptions, well written (e.g. clear and indented), degree to which code is commented (file, function, parameter and line comments)	**5**
Total	**100**

ASSESSMENT REGULATIONS

You are advised to read the guidance for students regarding assessment policies. They are available online here.

Late submission of work

Where coursework is submitted without approval, after the published hand-in deadline, the following penalties will apply.

For coursework submitted up to 1 working day (24 hours) after the published hand-in deadline without approval, **10% of the total marks available for the assessment** (i.e.100%) **shall be deducted** from the assessment mark.

Coursework submitted more than 1 working day (24 hours) after the published hand-in deadline without approval will be regarded as not having been completed. **A mark of zero will be awarded for the assessment and the module will be failed**, irrespective of the overall module mark.

These provisions apply to all assessments, including those assessed on a Pass/Fail basis.

Academic Misconduct

The Assessment Regulations for Taught Awards (ARTA) contain the ***Regulations and procedures applying to cheating, plagiarism and other forms of academic misconduct***.

You are reminded that plagiarism, collusion and other forms of academic misconduct as referred to in the Academic Misconduct procedure of the assessment regulations are taken very seriously. Assignments in which evidence of plagiarism or other forms of academic misconduct is found may receive a mark of zero.