|  |
| --- |
| **COURSEWORK ASSESSMENT SPECIFICATION** |

|  |  |
| --- | --- |
| **Module Title:** | *Web Programming* |
| **Module Number:** | *KF5002* |
| **Module Tutor Name(s):** | *Garry Elvin* |
| **Academic Year:** | *2020-2021* |
| **% Weighting (to overall module):** | *100%* |
| **Coursework Title:** | *Assignment Outline and Requirements* |

**Dates and Mechanisms for Assessment Submission and Feedback**

|  |
| --- |
| **Date of Handout to Students:**  Week commencing the 26th October 2020. |
| **Mechanism for Handout to Students:**  The assignment is available via Blackboard. |
| **Date and Time of Submission by Student:**  Tuesday the 5th of January 2021 before 11.59 pm. |
| **Mechanism for Submission of Work by Student:**  **Submission instructions**  This assignment **must** be submitted electronically in two ways as follows.   * 1. 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 **Tuesday the 5th of January 2021 before 11.59 pm** (your files should not be altered after that time).   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.***   * 1. 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 **Tuesday the 5th of January 2021 before 11.59 pm**.   **Instructions for submitting your assignment solution via blackboard**   1. In blackboard go to Assessment and Submission > Assessment Submission > Assignment Submission 2. In the **Assignment Submission** section go to View assignment > Add Content and do two things (you MUST do both)    1. 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***    2. Click on the ***+***  button, click on the down arrow, choose **Insert Local Files** and navigate to the required zip file, click **Save** and then click **Submit** (note that it must be Submit and not Save and Close)   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* |
| **Date by which Work, Feedback and Marks will be returned to Students:**  During the week commencing the 1st February 2021. |
| **Mechanism for return of assignment work, feedback and marks to students:**  Students will receive written feedback on their work from their tutor via the electronic learning portal (Blackboard). |

**Further Information**

**Learning Outcomes tested in this assessment (from the Module Specification):**

1. 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
2. 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
3. Analyse a web development problem and develop a solution
4. 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
5. Ability to work individually to develop a web-based system, managing time and resources

**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 (*make sure that you read carefully*) and the specific requirements.

# Assignment Case Study Background

Northumbia Books Limited (NBL) (a fictional company) sells a range of books. They have asked you to design and build the first stages of a new online presence for their company. You are to start the creation of the web-based system.

**Assignment Development Tasks**

In this assignment you are required to start the development of the “NBL” web based system by completing the following tasks:

*Task 1 pre-task*

An SQL script, NBL.sql, has been provided on Blackboard in the Assessment and Submission section that contains the SQL statements required to create four database tables called ‘NE\_category’, ‘NBL\_books’, ‘NBL\_publisher’ and ‘NBL\_users’ and to add a number of records to them in MySQL. You will need these tables and records for various tasks.

1. Download the file NBL.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 book currently held in the database. Components:

* *Choose a book to edit:* the administrator should be able to choose a book to edit from a list (for best marks created dynamically from the database records using PHP, PDO and SQL) of all the current books held in the NBL\_books database table by clicking on its title
  + Full details of all books must be included, **except** the publisher. You will need to format the book details (their layout etc.) in an appropriate way. For full marks divs or a table should be used and styled in an appropriate way, books should be ordered on title and you will need to display the category description
* *Edit details of book:* after choosing an book (clicking on its title) to edit, the user should be able to edit **all** of the details of that book, **except** the ISBN, using a form and the book’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 5 compliant and that it conforms to the module’s ‘house standard’ on coding style (outlined in appendix I)*
* *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 publisherID. 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 included 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. Again, ideally when a user logs in or logs out they should return (be redirected back) to their originating page.

*Hint: look up the php header function for redirection and to find out about the originating page search for 'referrer'*

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 ‘*Choose a book to edit’* (the list of books to choose from) *or to* ‘*Edit details of a book’* (the edit form) functionality to edit details of a book currently held in the database (see task 1). Without logging on successfully via the form, the user **should not** be able to gain access to **either**. Components:

* 1. *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# |

* 1. *Logon script:* this should check that the username entered by the user in the logon form exists in the NBL\_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 NE\_users for the given username and the methods covered in the module
  2. *Sessions*: sessions should be used to restrict access to the administrator functionality
  3. *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:
   1. 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.)
   2. 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
2. Use appropriate *server-side code* (i.e. PHP) to make data retrieved from database queries that originally came from the user, safe before displaying it in the browser to, for example, reduce the likelihood of successful cross-site scripting (XSS) attack. This will include handling any html and Javascript entered by the user.

*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 orderBooksForm script*

A PHP server-side script, orderBooksForm.php, which includes the PHP code to dynamically generates the html for a form with the html required to display one checkbox for each of the books currently held in the NBL\_books database table that has been provided for you in the assessment section for the module on blackboard. The user can select one or more books 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 orderBooksForm.php from the server in your browser and use the browser to view source.

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

**Important notes**

* you **MUST NOT** in any way change the php code
* You **MAY** 
  + Add Javascript for task 4
  + Modify the pageto add your own navigation menu and link to your own stylesheet. *However, please note that styling will not be marked for this assignment*

Task 4 - Select book(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 which will add the following functionality to orderBooksForm.php
2. 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. 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
4. 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 surname or company, and at least one book has not been selected

(remember that users don’t always fill out forms in top to bottom order)

1. The dynamically created form contains details of different books. Next to each is a checkbox that has been given a data-price attribute corresponding to the price of a particular book. The user is also given the choice of collection method for the books 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 book checkboxes that are selected and the choice of delivery method. The total cost should be displayed appropriately to allow the user to consider the cost before committing themselves to buying a book. If the user un-checks all of the books that they had previously checked, the total cost should always go back to zero

1. A select box has been provided to do with the type of customer. It (“Customer Type”) allows the user to select whether they are a standard customer or a corporate customer (a business).

If the user selects “Customer” for non-corporate customer then the div containing the “forename” and “surname” input areas should be visible, while the “Company name” input area should not be. On the other hand if the user selects “Corporate” for a corporate customer the opposite should be true.

*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 NBL\_SpecialOffers.sql from Blackboard. You **MUST NOT** alter this file in any way. It is an SQL script that contains the SQL statements required to create a new sample database table (called ‘NBL\_special\_offers’) and to add a number of records (each containing details of one special offer) to the table in MySQL. You will need this table and the records for the next task
2. Import it using phpMyAdmin to create the ‘NBL\_special\_offers’ MySQL database table
3. 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 NBL\_special\_offers and NBL\_category tables and retrieve one of the offer records at random. By **default** the function getHTMLOffer is called and the offer is placed inside an HTML paragraph and then sent to the requesting web browser. In this way the script acts as a web service that can be used by different web pages. Alternatively the programmer can specify (in the AJAX code) that the function getJSONOffer should be called to send details of the special offer in a JSON format by calling getOffers.php?useJSON

*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 named either index.html or index.php that
   1. provides links to the other pages in the site
   2. contains two aside elements, one with the id ‘**offers**’ and one with the id ‘**JSONoffers**’

### AJAX in the home page

* 1. Use appropriate AJAX code to request the getOffers.php script when the home page of the site is loaded, then display the **default** response (a special offer placed inside an HTML paragraph) 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)

1. Every 5 seconds a new special offer should be requested and then displayed inside the ‘offers’ aside in the **home page**
2. In a manner similar to a), use appropriate AJAX code to request getOffers.php?useJSON when the home page of the site is loaded, and to specify that the response should be in JSON format. Then use AJAX to display the response in an appropriate manner in an aside tag with the id ‘**JSONoffers**’ inside the **home page** (again it **MUST** be in the home page otherwise ZERO marks will be awarded for it)

*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/*](http://validator.w3.org/)*). In addition, ensure that it conforms to the module’s ‘house standard’ on coding style (outlined in appendix I)*

### *Test that any CSS code that you use is valid using the W3C CSS Validation Service (this is available at* [*http://jigsaw.w3.org/css-validator/*](http://jigsaw.w3.org/css-validator/)*)*

* *The use of HTML generation tools such as Adobe Dreamweaver (other than in Code view), or templates or files for either html or css not authored by you, is* ***NOT PERMITTED****. Anyone who is found to have done so will receive 0 marks for this part of the assignment. You have been warned. We expect you to generate your HTML by hand, e.g. using Notepad++*
* *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 Criteria/Mark Scheme**

This part of the assignment is worth up to 100% of the total marks available for the module.

**Marking Scheme**

Marks will be awarded in the following categories:

|  |  |
| --- | --- |
| **Category** | **Marks** |
| **Administrator functionality** - edit/update details of a book | **24** |
| * A dynamically generated list (from db content) to allow the user to choose a book to edit (books should be formatted appropriately and ordered on title) * Edit *form* fields are populated with existing data (all fields should be editable, except the ISBN) * Books are actually updated | **10**  **9**  **5** |
| **Logon functionality** | **10** |
| * Users can successfully logon with a correct username and password and can logout on every page * Use of sessions to restrict access to the edit functionality | **6**  **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 * Make data safe before it is added to the database * Make data retrieved from database queries safe before displaying it in the browser | **4**  **3**  **1** |
| **Select book(s)** | **20** |
| * Client‐side validation that at least one book is selected (checkboxes) * Client-side validation for text entry form fields, e.g. for surname, to check if completed * Terms and conditions   + must be checked before form can be submitted   + 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 * Total price calculation and display * Visibility of the customer detail divs | **2**  **2**  **2**  **2**  **10**  **2** |
| **AJAX functionality for the home page** | **18** |
| * On page load, get and display a special offer in an aside with the id ‘**offers**’ in the home page * Every 5 seconds get and display a new special offer in the aside with the id ‘**offers**’ * On page load, get a special offer, send the response in a JSON format and display it in an aside with the id ‘**JSONoffers**’ in the home page | **8**  **2**  **8** |
| **Build quality** | **20** |
| * Use of functions (e.g., use of external .php and .js files, the range of functions, use of parameters, returning values) * Valid HTML and conformance to the module’s ‘house standard’ on coding style * General quality of code – e.g., use of exceptions, correct scoping of Javascript functions and variables by appropriate wrapping inside window load handlers, well written (e.g. clear and indented), degree to which code is commented (file, function, parameter and line comments) | **10**  **5**  **5** |
| **Total** | **100** |

**Viva-voce**

As part of the normal marking of your work you may be asked to attend a viva-voce to talk about your work and explain the code you’ve written. The viva will normally take place in the assessment weeks and you may be required to attend, do make sure you’re available.

**ASSESSMENT REGULATIONS**

You are advised to read the guidance for students regarding assessment policies. They are available online [here](https://www.northumbria.ac.uk/about-us/university-services/academic-registry/quality-and-teaching-excellence/assessment/guidance-for-students/).

**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.

**Appendix I**

## html5 House Standard

1. Web pages must have the following elements (and must be in the following order): the HTML 5 doctype (<!doctype>) - and must start with one; <html>; <head>; <meta …>; <title>; <body>
2. Tags and attribute names must be in lower case
3. Tags must be nested correctly
4. Container tags must be closed
5. Inline elements can't contain block elements (the W3 spec also describes inline and block as 'text-level' and 'grouping' in the [html5 specifications](http://www.w3.org/TR/html5/Overview.html#contents))
6. Attributes must have values, except where that value is the same as the attribute name
7. Attributes with values must be quoted
8. Web pages should only have **one heading 1** <h1> per page and headings should be nested appropriately