BSc Computing (All Pathways) CSY2028 Web programming			
Date of Issue:	Thursday, 13th June 2019	Date for Submission:	Friday 12th July 2019 23:59:59
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 using 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

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 result in a capped grade of F-.

CSY2028 Web Programming

PHP & MySQL Assignment 2 (resit)

Aims & Objective
The purpose of this assignment is to assess your ability to modify and improve PHP code on an existing PHP/MySQL driven website.

Brief
You are a developer for a web design agency and you have taken on a new client called *Harry's Holidays.* Harry runs a small business that sells holidays and the company's website lets customers view the holidays they have for sale. Currently online purchasing is not a requirement. The website was developed several years ago,is now a little outdated and contains bugs that need fixing.

The administration area is available in the /admin/ directory, the password is letmein.

Below is an excerpt from one of Harry's emails regarding the changes he wants you to make:

1) We're open Sundays now. Can you change the opening times to say Sundays: 10:00-16:00?

2) I want to add a new page called "Travel Insurance". Just add the page with some placeholder text that says "Insurance information coming soon" and put a link in the menu. I'll send the content over at a later date.

3) I can add a new category in the administration area, but the added categories do not appear on the list on the Holidays page, can you fix this so the customers can view holidays from all the categories we add?

4) We sell the same holidays every year but our hotels get booked up quite quickly. At the moment the only way to remove a holiday from the website is to delete it. Next year we have to re-add the holiday by providing all the information again! Can you make it so that we can hide fully booked holidays from the website and easily restore them the following year?

5) The holiday's location is just a text box at the moment. My spelling of Greek towns is terrible and I have to look it up each time. Can you add a page called "Locations" which works a bit like categories so that I can add a location with a town and a country e.g. Agios Nikolaos, Greece and can select this in a drop-down when adding a holiday, rather than typing the name manually each time.

6) I'd also like it so that customers can look for holidays in a specific location and filter by country or by town. For example, viewing all the holidays in Biscarrosse.

7) I'd like my colleagues to be able to use the website. At the moment we all use the same password. Instead, I'd like to be able to manage user accounts so I can give new staff access (And remove their access when they leave the company). Each member of staff should have their own username and password used to log in to the admin area.

8) The website lets me upload a photo of each holiday but customers would like to see a range of different photos. Can you allow me to upload at least 4 photos? Although sometimes a lot more would be useful!

9) I want to be able to add special offers to the home page. Each entry should have a date (so people know when it was posted), a title a description and sometimes a photo. This will contain information about deals and special offers we currently have. I need to be able to delete them after the offer has expired.

10) People often ask us questions about the holidays. Can you add an "Ask a question" box below the details about each holiday that lets someone enter their email address, phone number and question. When they submit, the questions should go into a page in the administration area that shows us the holiday they were asking a question about and the customer's question. We can then email or call the customer back. We'll need some way of marking the question as answered so we don't contact the same customer twice.

Harry will be sending further changes in the future. As the new maintainer of the website, you will need to take a look at the code structure and possibly change it to make future additions easier.

You have the following tasks:

1. Complete the changes asked for by the client.

2. Analyse the existing code and document anything you found about the structure of the code that made your tasks more difficult than it otherwise could have been. You should identify each problem and explain why it made your task difficult. You should also suggest problems that might cause difficulties when updating the website in the future.

   Examples of problems you should look for include repeated code or having to make the same/similar changes in multiple locations when editing the code.

3. Provide a potential solution for the problems you identified so that future changes to the website are considerably easier to make.

For a **bare pass** (D-) you must:

1. Complete at least 7 of the 10 changes requested by the client.

2. Identify some of the weaknesses in the existing code and explain why they pose a problem. You should provide snippets of the code in your report and detail why the method used causes issues for future developments.

3. Fix the major problems with the current website

For a **pass** (D – C+) you must:

1. Complete at least 8 of the 10 changes requested by the client

2. Identify several of the weaknesses in the existing code and explain why they pose a problem. You should provide snippets of the code in your report and detail why the method used causes issues for future developments.

3. Fix the major problems with the current website and some of the minor ones.

For a **good pass** (B – A+) you must:

1. Complete all 10 changes requested by the client

2. Identify most of the weaknesses in the existing code and explain why they pose a problem. You should provide snippets of the code in your report and detail why the method used causes issues for future developments.

3. Fix most of the issues on the website and re-develop the website in such a way that further amendments require smaller changes in a single location and sections of code could be reused on a different website without any modification.

Additional information:

- No functionality/features should be lost during your updates (e.g. every page currently has a different title, that should still be the case after you have finished making changes)

- You do not need to amend the HTML or CSS of the website, you may change the HTML and CSS code if you wish, but marks are awarded for PHP code, not HTML and CSS.

- You cannot pass the assignment solely by implementing the requested changes. You must identify issues with the code and fix the major maintainability issues.

- Marks will be lost if implemented features are buggy or break functionality which currently exists.

- You should keep security and maintainability in mind when implementing the changes

Deliverables

1. Technical report (Guideline length 2000 – 2500 words)
   a) Table of contents
   b) A checklist of the requested changes you have implemented clearly showing which changes you have/haven't completed (e.g. ticks/crosses or red/green backgrounds on each row)
   The checklist should have the following structure. The second row is an example:

Feature	Completed?	Relevant files	Notes
Admin accounts	Partially	admin/login.php , admin/adduser.php	There is currently no way to delete an account

c) Identification of weaknesses including screenshots or formatted code snippets and an explanation of why the identified code causes maintainability issues .
d) Proposed fixes that would solve the problems caused by the problems identified in part (c).
e) Evidence of testing:
   i. Test logs providing information of all the tests carried out (including any failed tests for functionality not implemented).
   ii. List of any bugs/weaknesses you have discovered along with what you did to fix them.
   iii. Output of PHPUnit along with PHPUnit code coverage reports
f) 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.

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 *website* directory and *Vagrantfile*. Your website folder must contain the up to date `database.sql`
      ii. A word document contain all your PHP files (Copy and paste each file into a single document) for uploading to turnitin
   b) Please note the following when uploading your work:
      i. You must include the database.sql with your code
      ii. If the marker cannot run your work your grade may be capped to F
      iii. Do not include your video in the source code zip
      iv. You must only upload code you wish to be marked. Do not include, for example folders called “Assignment2a”, “assignment 2 version 3”, “as2” and “assignment backup”. It should be clear to the marker which folder you intend to have marked.
3. In Addition to the report you must provide a video demo of your assignment. The demo should be 5-10 minutes (No longer than 15 minutes) and demonstrate all of the new features of the website as if you were showing the client the completed changes that were asked for.

Marking Criteria

This is a programming course, not a design course. As such, marks will be awarded for code quality instead of visual design. You do not need to change the look of the website. However, thought should be placed in how easy to use any changes you made are for non-technical users. Users should never need to write PHP code or change the files to make changes to the website. You should also consider usability, users should never be expected to remember numerical ID numbers, manually amend the address bar or type in information which is already in the system.

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
Implementation of requested changes (20%) Note: These numbers are *indicative* and marks may be lost if a feature is incomplete, implemented with bugs or in a way that is difficult to use or makes future modifications more difficult.	All 10 requested changes have been implemented and work without bugs and the changes are easy to use.	8 of the 10 changes have been implemented and work without bugs and the changes are easy to use.	7 of the 10 changes have been implemented and work without bugs and the changes are easy to use.	7 of the 10 changes have been implemented but contain bugs, do not work correctly or are done in an unintuitive way.	Fewer than 7 of the 10 changes have been implemented.	No submission or no submission of merit
Identification of weaknesses (25%)	Almost all weaknesses have been identified and strong reasoning has been provided for those that have	Most weaknesses have been identified and good reasoning has been provided for those that have	Several weaknesses have been identified and good reasoning has been provided for those that have	Few weaknesses have been identified but good reasoning has been provided for those that have	Very few weaknesses have been identified or identified problems are poorly explained or do not contain reasons for being identified as a weakness.	No submission or no submission of merit
Testing (15%)	Everything for B and/or use of PHPUnit to run automated tests for form submissions and any functions/classes that have been written.	Tests have been carried out on all completed functionality. There is enough information provided for someone else to replicate each test exactly	Tests have been carried out on all completed functionality	Tests have been carried out on most completed functionality.	Little to no testing, no clear testing strategy	No submission or no submission of merit
Code quality and efficiency (35%) Note: These are indicative features. You can get marks for implementing parts in (A) even if you have not completed everything from (C)	Everything for B plus: Use of controllers with one function per “page” Code is written so that sections can be moved to a different website without modifying a single character in the file Use of entity classes	Everything for C plus: Use of objects and classes. Database abstraction (no hardcoded queries) Classes are loaded via an autoloader Everything goes through a single entry point Classes are in namespaces	Everything for D plus: No repeated bootstrap code (e.g. requires statements) at the top of each file Separation of concerns: Use of template files with no queries or database logic in templates No separate add/ edit pages	Repeated code is in its own files. Making a change should not require making the same change in multiple locations.	F+: No attempt to improve the structure of the code. F - F-: Changes implemented have introduced more repeated code than there was at the start.	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. Any known bugs are highlighted.	Covers the functionality from the basic requirements in detail.	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