**Web Application Development: Coursework**

**Allocation**

This coursework is worth 100% of the marks for module CHC5054.

You will also need skills from the following modules:

* **CHC4008** (Python Programming)
* **CHC4007** (Design reporting)
* **CHC5049** (Database design)
* **CHC5226** (Secure implementation)

**Specification**

Your task is to develop and test the full stack for a simple web-based Learning Management System designed to facilitate the management, delivery, and tracking of educational courses.

* instructors can upload Lectures, submit assessments, and mark student assignments.
* students can enrol, view course content, and submit assignments, progress tracking, and discussion forums.
* Administrator creates and manage accounts for instructors, students, courses and other admins

The following is the **basic** functionality to implement:

* **Home Page:** On visiting the first page of the site, user should be redirected to login page if not logged in, otherwise he should be able to see the ***course list*** for the student, ***upload list*** for instructor*.*- Admins should have access to user, courses, lectures management (creation, editing, deletion).  
  - Instructors should see their courses, student lists, and lecture management options.  
  - Students should see their enrolled courses, grades, and upcoming assignments.
* It is up to admin to create/edit/delete and assign a course for only one instructor.
* An “Add lecture” button when an instructor wants to create a lecture (name, description) and another “Add file” button to upload add one resource (word/pdf file).
* Students can look over the offered courses and request to be enrolled in those they would like to take enrol students in courses
* Instructors and students can communicate with each other through messaging
* **Notifications:** Receive alerts for new announcements, deadlines.
* **Search Functionality:** Search for courses, lecture, and other users within the system.

Admin:

* **Course Enrollment:** Assign a course to an instructor (or instructors), enrol students in courses

Instructor:

* **Course management:** create course content, including lectures, and assignments, an instructor can only edit his own courses.
* **Assignment Management:** Create and manage assignments (name, files, deadline) all students enrolled in the same course will be assigned automatically
* **Grading and Feedback:** access students’ submission, grade and provide feedback

Student:

* **Course Enrollment:** Look over the offered courses and request to be enrolled in those they would like to take (only administrators can decide whether to accept or enroll a student).
* **Content Access:** Access enrolled courses, lectures, assignments.
* **Submission:** Submit assignments, within specified deadlines.
* **Feedback and Grades:** Receive feedback and grades on assignments and assignments.

The marks available for each of these areas of functionality are divided based on the sections of the coursework, as described below. Simply writing code to meet the specification by any means will not earn full marks.

**Restrictions**

Your web site must run in a Python virtual environment built by running the following commands from a Command Prompt in a suitable working directory:

pip install virtualenv

virtualenv webtest

cd webtest

scripts\activate

pip install flask mysql-connector-python

It is recommended that you also use this virtual environment for development. Regardless, you must test your site within the virtual environment before submitting it. This is the virtual environment in which your code will be tested by the assessor. If it does not work, marks will not be awarded.

In addition to the libraries in the above virtual environment, you may use the jQuery library for JavaScript, but **may not use any other frameworks or libraries**. This means that you may not use libraries that are “Flask extensions” if they are not installed *with* Flask.

(Please do not send e-mails asking if you can use other extension libraries; the answer is no. The purpose of the coursework is to understand how JavaScript and server interaction work at the lowest level possible, not in terms of abstractions created by higher level code.)

Your website must be built using HTML 5 and ECMAscript 6 on the client, and Flask, MySQL and Python 3 on the server.

**Plagiarism**

This is individual work. You must not copy or share code with other students. Do not copy code from online sources, answers, tutorials or existing open-source software. Copied code will result in the plagiarism process being invoked and you may be asked to attend an online meeting to verify that your understanding of your code is consistent with you having written it.

**You must be very careful with the use of online tutorials on this module.** Your primary learning source should be the module notes and the lecturer and tutorial staff. There are a very large number of online tutorials on the topic of web development, but many are incorrect, out-of-date, or badly written.

**Sites which present code and then explain it are often cheating sites, not tutorials.** Retroactively explaining large amounts of code is not an effective learning method, but is often used as a trick to “justify” presenting code actually intended to be cut and pasted. Often the explanations are extremely poor or even incorrect and will confuse your understanding further.

**Submission**

Submission is in three stages:

* a preliminary report
* the actual website
* a final report

**Due Date**: Preliminary report: **(Week 5 - Thursday March 28, 2024)**

Website: **(Week 11 - to be announced)**

Final report: **(Week 12 - to be announced)**

The preliminary report is worth 20% of each component of the marks from the Specification section. The implementation is worth 60%, and the final report is also worth 20%.

Code implemented in the final submission but not reported on in the actual report can still score full marks on the other two components, but you will lose out on potential feedback on your report.

The final report may refer to sections of code that were not implemented, but it will likely not be possible to write the report well without having implemented those sections.

**Preliminary Report**

The preliminary report should cover the design and structure of implementation you intend to use to produce the selected functionality. This should include:

* ER diagram for database used in the website
* the division of functionality between client and server (in this context the “client” means the web browser and JavaScript code, not the human user);
* wireframes representing the user interface;
* an overview of the intended implementations at client and server.

**Website**

Students are expected to start developing the website from the first week of the semester, the website should be uploaded as a .zip file to the student website by (Date to be announced). To prepare your .ZIP file, do the following steps:

1. Prepare a fresh virtual environment via the commands described in the “restrictions” section.
2. Copy your website files into the virtual environment, activate it, and ensure that your website works inside it.
3. Use the mysqldump utility to output your database specification into a file in the virtual environment. (The mysqldump.exe file is in the bin directory of the MySQL Server install path.)
4. Delete the directories Include, Lib, Scripts, and tcl.
5. ZIP the virtual environment directory.

The 60% of the section mark for the website is divided as follows:

* The basic functionalities: 30%.
* Quality of user interface design: 10%.
* Quality of back-end architecture: 20%.

Remember to break down processes in detail to allow these to be described. “The user can click to login” is not sufficient detail - the process of logging in is a multi-stage one that will likely involve several communications between the client and server, and use of the database.

It is not necessary to include source code in the report. The report will be your main opportunity to gather feedback before the submission (although you may also ask for help in practical sessions) so make sure to provide detail on any decisions you are not sure about, so that the assessor can provide useful feedback.

The preliminary report should be submitted on the student website by (**Date to be announced**). You may have already started developing your implementation before this submission.

**Final Report**

The final report should be at standard font size, and should address the following issues:

**Describe the accessibility measures added to your site** and who they are intended for. The measures must be documented in WCAG as correct ways to increase accessibility. You could also describe any additional accessibility features that might be added with more development time. Note that the absence of inaccessible features that could never be relevant to the site (such as video) cannot be listed here and is not worth any marks. (2.5%)

**Describe how you would adapt your site for mobile phone users.** You do not have to write actual changes to the CSS, HTML, or JavaScript, but you should describe the changes you would make and the means by which they would adapt the site. It is not acceptable to write that you would replace your site with an app, (5%)

**Describe how you have tested your site.** This should include a formal test plan and results as covered on previous programming and software engineering, and any fixes you made. This should include testing both the functionality and accessibility of your site. You should also test that your site works when multiple sessions interact with it at once, by using multiple browsers at once on your computer. (10%) Note: if you present your test plan and results in a table, lines and extra space taken by that table do not count against the limit of 6 sides of A4.

**Choose one of the major JavaScript frameworks** such as ReactJS, Bootstrap or Angular, and research it only. Describe how you would modify your site to work within this framework and what difference it would make to implementation of your site. Note that your description must be specific to **your site** - a general description of the framework will score no marks. (2.5%)

The final report should be uploaded to Student Website by (**Date to be announced**).

**Overall Mark Scheme**

|  | **Basic functionality** |
| --- | --- |
| Initial Report | 20% |
| General Implementation | 30% |
| User Interface | 10% |
| Back-End Implementation | 20% |
| Final Report | 20% |
| Component total | 100% |

**Tips**

**Using the lecture material**

The lecture material and practical’s do not directly walk you through the coursework, but it can be completed by applying the skills they teach.

* **Session 7** will help you think about designing your database. The specification includes **administrators**, **instructors**, **students..**. Think about what needs to be stored about each one and how they relate to each other, and what data needs to be stored about the relationships.
* Think about database **transactions** - what is the list of actions that the user can take that alter the database? How can you write each one as a query? Think back to **CHC4007** and consideration of use cases.
* What input does each transaction need, and how can you get that from the user over the web? **Sessions 1-4** will help here.
* How will you send the input to the server? First, in what form? How can you make it easy for your program to work with but also secure? Again, see **sessions 1-4**. Second, how will you actually send it over the web? See **sessions 5 and 8** for your options.
* Can you now write the Python program for the transaction and assign it to be a Flask route? See **sessions 5 and 6** as well as general Python programming from **CHC4007**.
* How will you return the response to the user? Will you use a web template (**session 6**) or maybe a JavaScript program that reads from the server using AJAX (**session 9**)?
* Is the transaction “stateful” - meaning that it changes the treatment of other transactions after it, such as logging in? If so, what do you need to store in the Flask session variable (**session 6**) to allow this?

**Previous Common Errors**

Here are some previous mistakes frequently made by students on Web Development coursework. Please be careful not to repeat them!

**Preliminary report**

* Repeating what is stated in the specification as if it was your decision rather than what you have been told to do.
* Telling the story of how a function will be implemented rather than how the implementation will work.
* Failing to break down tasks into client and server components or attributing all of a multi-stage task to one of these; for example “the student logs into the site” which actually requires multiple steps and cannot be purely done on the client.
* Describing what is **stored** on the client or server, but not what functionality they have.
* Over-emphasising the choice of technologies, rather than how they are used.
* Writing many-to-one relationships with the foreign key on the wrong side.
* Trying to store foreign keys to two different tables in the same field without distinguishing them.
* Not showing data table descriptions, only sample data that does not formally indicate structure.
* Omitting field types from specifications.
* Stating that the “user will be logged in” without indicating what method is used to signal this.
* Large amount of dead space on UI designs.
* Showing UI prototypes instead of wireframes.
* Where search is implemented, not providing a wireframe for the search results page.
* Where AJAX is used, not providing a structure for the JSON messages that will be used.

**Web site**

* Using unauthorized libraries.
* Failing to implement all basic functionality.
* Functions on the server that require the user to be logged in don’t actually check that the user is logged in, for example “add resources” can not add lecture resources unless the instructor is logged in.
* Storing passwords in cleartext in the database.
* Making the database or on-screen fields for a comment text too short for messages to be entered.
* Duplicated data in the database, violating normal form.
* Missing or redundant foreign keys in tables.
* Search loads entire table and searches it in Python rather than using SQL database search.
* Can only search exact, not partial, matches.

**Final Report**

* Listing the general principles of mobile friendliness, framework features, etc. but not how they would be used on this particular site.
* Testing complete functionality paths instead of individual interactions. While testing paths is a good idea, remember that users may not follow them; web requests can be sent in any order as a result of shared links, the back button, etc.
* Not testing edge, exceptional or error cases.