Building a Personalised Learning Environment.

1. Introduction.

The idea for this coursework is to build a single page web application that allows users to build, collect and view their own personalised resources that helps them to learn all about the Client Side Web Development (CSWD) module.

> A single-page application (SPA) is a web application or web site that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server. This approach avoids interruption of the user experience between successive pages, making the application behave more like a desktop application. In an SPA, either all necessary code – HTML, JavaScript, and CSS – is retrieved with a single page load, or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions. The page does not reload at any point in the process, nor does control transfer to another page, although the location hash or the HTML5 History API can be used to provide the perception and navigability of separate logical pages in the application. Interaction with the single page application often involves dynamic communication with the web server behind the scenes.
>
> The most prominent technique currently being used is Ajax. Predominantly using the XMLHttpRequest/ActiveX Object(deprecated) object from JavaScript. Popular libraries like jQuery, normalize Ajax behaviour across browsers from different manufacturers.
>
> Requests to the server typically result in either raw data (e.g., XML or JSON), or new HTML being returned. In the case where HTML is returned, JavaScript on the client updates a partial area of the DOM (Document Object Model). When raw data is returned, often a client-side JavaScript XML / (XSL) process (and in the case of JSON a template) is used to translate the raw data into HTML, which is then used to update a partial area of the DOM.
>
> An SPA moves logic from the server to the client which results in the role of the web server evolving into a pure data API or web service.
>
> ref: https://en.wikipedia.org/wiki/Single-page_application

1.1

There is a range of content types that modern web browsers support. However, there are also a wide variety of sources that are not directly supported but can be displayed and viewed via the browser. With this in mind you should be able to pick and choose content from a variety of sources acting as a bespoke 'one-stop shop' for your specific learning requirements e.g. access eBooks(pdf), HTML5 source, JavaScript code samples, videos and audio, games, animations and simulations to aid your understanding, activities that promote learning and accommodate differing learning styles. You should be able to customise existing eLearning content to match the specific needs of your CSWD course.

Having gathered all the content into your PLE how do you know that it has been of value and has contributed to your understanding of the CSWD module? Another area of consideration could be thinking about assessing your learning by evidencing competencies for summative assessment. This could be carried out using quizzes and reflections/personal blogs, sharing material with peers and staff, and by incorporating feedback opportunities into your learning environment.

Collecting a portfolio of content for your learning and development is one of the most important reasons for building such an app. Building a collection of materials within your PLE will provide you

with evidence of the progress that you are making towards achieving your personal learning outcomes and help with your web skills development.

NB These areas of usage are not exhaustive. You can be imaginative about what you would expect to see in your CSWD module.

2. General considerations on delivering the single page web application.

You can decide on most of the specific aspects of your own PLE but the following will help you think about the sort of elements you might have in your SPA:

- A data storage repository (probably using JSON) that contains all of your digital artefacts: personal blog, announcements, achievements, lectures, labs, tutorials, notes, html, coding samples etc basically any combination of content as you see fit – with appropriate metadata to represent the different data types
- A data capture component, probably through the use of forms that allow you to collect 'content' together in some structured way from a variety of sources and for a variety of purposes
- Data display components onto which the 'content'. are "presented" by the application for you and any other viewers in an easy to use and visually interesting way
- A search capability to help you find appropriate content to be displayed perhaps using keyword searches eg 'show all lectures covering topic xyz'
- A 'to-do' facility to help you plan for future work aspect of the application
- A facility that allows you to 'test' your understanding of what has been learned
- A data administration component, again, probably using forms that allow site administrators to take the 'content' packages (create, edit, publish, archive, and delete 'content') for storage in the data storage repository

2.1 Storage repository.

I suggest that you try to identify the content that you would like to capture and then define the data structures and possible metadata that you need to store the content. Once this is done all the other elements cascade from there, so you need to define how your 'content' should be structured and from there where and how it will be stored. Remember that not all types of data need to be structured in the same way eg a photo gallery might only need tags to store the filenames of the images to be shown whereas a blog will probably require different metadata. So don't try and shoehorn all the different types of content into a single specific structure. Whilst mentioning storage it would be useful to identify the types of data that you might want to store to enhance the user experience as they use your site on a regular basis – think about local and session storage usage.

2.2 Building the Content Creation Tool

Clearly there are lots of data sources just waiting to be gathered together to provide the content for your application. This part of the coursework focuses on the bringing together of all these different types of data. This could well be a series of form filling screens to do this part –perhaps different forms for different types of content, with each type of content being packaged according to the required structure. A more ambitious way would be to try and use a 'WYSIWYG' creation scheme! Thoughts – do all types of the same content need to be presented in the same standard way or can content have its own 'formatting' instructions?

2.3 Building the Admin Tool

The admin tool for the application is likely to be a set of 'pages' that will allow administrators to 'log in' and create, edit, publish, archive and delete content as and when required.

2.4 Building the Presentation Side

Having collected all the content you'll need to build an appropriate 'presentation' style for displaying the content so that you and any users of the application can view and search for data. Do try and allow your users to interact with as much of the content as possible. Consider whether all types of content get displayed in the same way? Is it possible for a user to combine different types of content? Can users provide feedback on what they have seen and done? Do all viewers get access to the same/all content e.g. can stuff be marked as Private/Public?

2.5 Conclusion

That's it! – there is potentially a lot here to do and plenty of opportunity to show how well you have understood the module. Fundamentally it's a vehicle that allows you to demonstrate your JavaScript programming skills.

3. Specific Coursework Details.

You have to build a single page web application which allows users to manage the 'content' in their own personal 'learning space'. The *content* should showcase a wide range of materials integrating sound, video, text and pictures in an imaginative way. For our purposes we will assume that the spa is a software application running on the 'web' which allows users to manage all aspects of their own 'content' in their personalized learning environment for the CSWD module. The learning environment is the medium through which the content is deposited, presented and managed. This should be stored on the localhost webserver in a subfolder of the htdocs folder or equivalent. You can use jQuery and Bootstrap if you think this will help you deliver your information in an 'interesting' way but this is predominantly an exercise in showcasing YOUR javascript skills. Hence, you should not use any pre written frameworks and remember you must try and avoid 'polluting the global workspace'.

The application is expected to cover 3 main areas:

1. building the backend data repository for the different types of 'content' (10%)
2. building a presentation system for delivering the 'content' (40%) incorporating at least the following javascript modules:
    a. a '***slide' manager module*** to display 'slides' from a collection of 'slides' with appropriate controls for first, last, next, previous, goto slide number; where a slide can be a graphic image or a pdf or just a page of text from a collection of images/pdfs/text pages that make up a lecture, lab etc
    b. a ***content loader module*** that loads the appropriate content and passes control over to the appropriate display routine eg the slide manager; the page updater etc
    c. a ***page updating module*** that puts the actual content on to the page
        - watch out for things like html and javascript code that could cause potential issues
    d. a ***menu manager module*** that allows the user to navigate around your spa
    e. a ***search manager module*** that can find 'content' and then passes it to the page updater to display
    f. any other appropriate code to 'glue' all the parts together

3. building an admin tool to collect, edit and publish the 'content' (10%)

20% of the marks are for the actual 'content' that you provide in your application which should be clearly aimed at supporting your own learning, teaching and assessment. This gives you a free reign over what you want to use as evidence for this part.

20% of the total marks will be allocated for documentation, expected in 3 parts as follows:

1. a description detailing the folder, data structures and metadata that you used to store the content and an explanation of why you chose them, 5%
2. a description of the testing that you used in showcasing how to use/navigate/search/maintain your application, 5%
3. and a reflective part detailing what you personally 'learned' from both building and using your application paying particular attention to usability and software design patterns that you used, 10% - expected 1500 words.

In each case, there's plenty of room for constant improvement and refinement. The key is to get something working and then to enhance it, Keep It Simple, Stupid (KISS)!

Also remember **minimum work achieves a minimum mark**.

Submission Details

- You are required to hand in an individual piece of work, containing a copy of your completed application and documentation via GCU Learn TurnItIn. Your report, a printout of your code and a disc/usb containing your completed web application and any instructions required for me to mark your app is to be submitted by **1 p.m.** on **Wednesday 20th Dec 2017.**

Remember: You can use any appropriate tools such as the library and or the internet. However, please note that "plagiarism" is not accepted within the School and as such all material handed in should be annotated with appropriate references. Anyone found to be in breach of University regulations with regards to this subject will be dealt with appropriately. As usual any additional time required to fulfil this coursework can only be granted by the module leader.

That's all folks and all the best!