Glasgow Caledonian University

University for the Common Good

School of Computing, Engineering and Built Environment

Web Application Development 1 Coursework Specification

2022/2023 Trimester B

Module Code:
M2I325626-22-B
M2I326554-22-B

This module is assessed 100% by coursework.

The coursework is due by 24.04.23 at 12:00.

In this coursework, you will implement a web application using the technologies and concepts you have learned in this module.
The goal is to develop a web application that provides users with recent UK vacancies and contains a salary calculator.

Web Application

The web application should follow a well-thought-out style and consistent design. It should consist of ***three*** web pages whose content and filenames are specified as follows:

1 - Homepage (*index.html*)

The main page should serve as a landing page for users. You should include a related image[1] and general information about the application and its functions.

2 - Pay Calculator (*pay.html)*

This page provides users with a payment calculator based on their wages and hours worked per week. You should build an input form to allow users to enter their wages in GBP (with options to select year/month/week/hour) and their hours worked in addition to providing a job title.
Upon confirmation by the user, the page should then provide a styled output showing the user their yearly- monthly- weekly- and hourly- wages. Calculations need to be performed using client-side JavaScript.

Upon submission, this form should ***dynamically*** update a separate section on this site showing the following in a well-formatted and styled output that contains at least the following information:

Job {**Job Title**}
Working **{Hours}** a week for {**Wage Entered**} per {**Time (year/month/week/hour**} breaks down into:

{**Wage Computed**} per {**hour**}
{**Wage Computed**} per {**week**}
{**Wage Computed**} per {**month**}
{**Wage Computed**} per {**year**}

The area where the results will be displayed *should not* appear when the user first loads the page, only when the user fills and submits the form.
If the user, *without refreshing the page*, enters new values into the form and submits them, this result should be ***appended***, and not replaced. This allows users to compare the values burned for various samples of data entered.

[1] This image should fit the purpose of the site such as any kind of stock image. It should be integrated using HTML, *not CSS.* An example image could be a person working in a office.
An example web-app that contains copyright free images is: https://unsplash.com/

Fewer marks will be awarded if the output is only shown in a JavaScript *alert*.
More marks can be gained, by providing a clickable option, that upon clicking on the job role, ***opens a new tab*** and leads the user to the *vacancies.html* page, showing *vacancies* matching this job title.

In this output, the variables in curly brackets such as { **Wage Entered** } need to be replaced by the values the user has entered or calculated in the case of { **Wage Computed** }.

Example: User enters the following:

Job Title: Software Developer
Pay: *£2000*
Pay Timeframe: *Monthly*
Hours per Week: *35*

The values are then calculated, and the following result text should be shown in a well-designed output area:
Job: **Software Developer**
*Working **35** hours a week for **£2000** per **Month** breaks down into:*

£13.19	*per Hour*
£461.51	*per Week*
£2000	*per Month*
£24000	*per Year*

Note: All values should be rounded to two digits after the decimal dot.

3- Vacancy Search (*vacancies.html)*

This page will provide users with a list of recent vacancies on top of an option to search for vacancies across the UK. It should also provide a search box, where users can enter a job title, and show the most recent vacancies matching their query. Upon performing the search, a list of relevant vacancies will be shown.

The link to the API endpoints and documentation are as follows :

- API Documentation: http://api.lmiforall.org.uk/
- *First API Endpoint* (Vacancies): http://api.lmiforall.org.uk/api/v1/vacancies/search
- *Second API Endpoint* (Job Information): http://api.lmiforall.org.uk/api/v1/soc/search

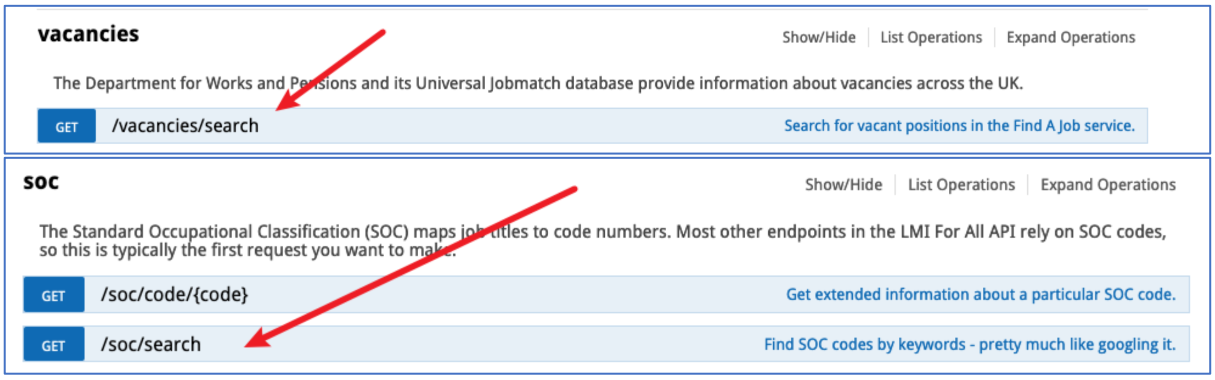
At the ***minimum***, *upon visiting the page*, users should be able to see the *top 10* most recent vacancies and their details including ***title, summary***, ***company***, **l*ocation***, and the ***link** to the job posting,* when opening the page. The data should be obtained from the *First API Endpoint* and should be displayed well-formatted and dynamically.

For ***more marks*** in this category (with increasing difficulty) you should include the following:

- Users are ***only*** shown the title, and upon clicking, the remaining information expands dynamically.

- There should be a "search bar" where users can enter certain keywords and are only shown the *top 10* relevant results. (Based on passing the search parameters to the "vacancies/search" API)
- Information from the second API Endpoint should be integrated. The title from the vacancy should be extracted and used as a keyword to query the second endpoint. The information from there, including ***description*** and ***tasks*** should be shown below the vacancy, as "General Information" about this type of job.

You can obtain information about the required API parameters by clicking on the relevant endpoints in the documentation:



vacancies

Show/Hide | List Operations | Expand Operations

The Department for Works and Pensions and its Universal Jobmatch database provide information about vacancies across the UK.

GET /vacancies/search — Search for vacant positions in the Find A Job service.

soc

Show/Hide | List Operations | Expand Operations

The Standard Occupational Classification (SOC) maps job titles to code numbers. Most other endpoints in the LMI For All API rely on SOC codes, so this is typically the first request you want to make.

GET /soc/code/{code} — Get extended information about a particular SOC code.

GET /soc/search — Find SOC codes by keywords - pretty much like googling it.

The first endpoint needs to use the *keywords* parameter when the user enters a search and the *title* from the vacancy needs to be used in conjunction with the *q* parameter in the second API endpoint.

NOTE: Using the LMI For All Widget (https://www.lmiforall.org.uk/widget/) will **NOT** give you any marks, you are expected to build the API access and dynamic display part on your own.

Example:
User goes onto the page and sees the *top 10 most recent vacancies*.
They then enter "Tester" in the input field and submit it. The *first API* queries the URL:
http://api.lmiforall.org.uk/api/v1/vacancies/search?keywords=Tester
The top 10 results are shown, and the user selects the first result with the title: "*Software Tester / Test Engineer*"[2]. The user is then shown the *title, summary, company, location,* and *link* of that vacancy.
You should then get hold of the **title** ("Software Tester / Test Engineer" in that example) and use it to query the *second API:*
http://api.lmiforall.org.uk/api/v1/soc/search?q=Software%20Tester%20%2F%20Test%20Engineer
This returns a list of jobs. The first result of that list should be selected, and the user should be shown this job's *title* and *description* as supplementary information to the vacancy.

[2] Due to the API relying on live vacancy data, the results can change on a daily basis.

Other Web Application Criteria

In addition to the aforementioned functions of the web application other criteria have to be met:

- The web application should be responsive and should be designed to display across a range of different platforms (at least mobile and desktop). Content that is side-by-side on a desktop-sized display should stack vertically on mobile devices.
- Sufficient navigation options should be provided such as links between the different pages and internal links (if used). On mobile devices, the navigation bar should stack and collapse.
- All pages should have descriptions and titles.
- For all input fields, sufficient warnings should be added if the user does not enter the correct data. This means that when the user enters text into a field that is designated for numeric values or the user enters something which is not an email into an email field, an appropriately styled warning should appear
- You should use a user-friendly colour scheme. Some inspirations for this can be drawn from pages like: https://coolors.co/, http://colormind.io/, https://designs.ai/colors.
- You should **NOT** use sample text; your web application should serve as a finished product.
- The use of **Content Management Systems** such as WordPress, Drupal, Joomla or prewritten frameworks is **NOT allowed.**
- Code libraries such as jQuery and Bootstrap are allowed. However, if you are using CSS libraries, you still need to write *sufficient custom CSS* to show that you are able to use it effectively.
- You should use semantic-markup as appropriate (<main>, <header>, <footer>, <section>, <figure>, <article>…..)
- External Stylesheets and JavaScript files *should* be used
- Styles should use different kinds of selectors, in particular
    - Element, Class, ID and Pseudo-Selectors
- All code files **must** include sufficient comments. It is not necessary to comment on every single line of code but larger- or more complicated code should include them (This also applies to CSS)
- The web application must work correctly in *Chrome* or *Firefox*.
- All HTML and CSS files must be validated. The results of the validation for **each** of the four pages as well as the validation report for CSS used need to be included in the submission. Use the validators at:
    - https://validator.w3.org/
    - https://jigsaw.w3.org/css-validator/

  Take a screenshot of the validation reports, label it accordingly such as "*validation index.jpg"* and save it within your submission.

Plagiarism

You should pay attention to the university's codes and practices[3] as well as their plagiarism regulations[4].

Any kind of content (images, text, ***code***) that was copied from any source and used in your coursework **without** acknowledging the source is bad academic practice and could fall under plagiarism.

The discussion of coursework between students is encouraged but the work must be undertaken individually. ***Collusion*** (copying work between students) may result in a zero mark being recorded for everyone involved and further action being taken.

Sources

To avoid plagiarism or poor academic practice you need to ensure that you specify where you obtained any material you use and how you have modified it.

This **MUST** include specific web addresses (not just google.co.uk).

A template for this is included on GCU Learn and an example is shown in Table 1.

Table 1 Sources Template

Type	Sources
CSS	Table Styling https://support.awesome-table.com/hc/en-us/articles/115001399529-Use-CSS-to-change-the-style-of-each-row-depending-on-the-content
JavaScript (modified)	Splitting String in split_string JavaScript function: https://stackoverflow.com/questions/96428/how-do-i-split-a-string-breaking-at-a-particular-character
Image	GCU Logo https://www.gcu.ac.uk/

Coursework Submission

All files for your web application should be submitted in a compressed *.zip* file. The file should follow the following naming scheme: LastName_FirstName_StudentID.zip.

If your name is *Nicola Sturgeon* with the StudentID *S12345*, name the file:

sturgeon_nicola_S12345.zip

Coursework files are submitted using GCU Learn.

[3] https://www.gcu.ac.uk/academicquality/regulationsandpolicies/universityassessmentregulationsandpolicies/

[4] https://www.gcu.ac.uk/library/smile/plagiarismandreferencing/

You should double-check your files before submission to ensure that you did not miss any files.

Marking Scheme

The marking scheme which will be used to assess the coursework is appended below.

Practical-Based Web Programming Assignment Total: 100 Marks	100 % of overall coursework mark				
Topic	Fail <40%	3rd 40%-49%	2.2 50%-59%	2.1 60%-69%	1st >70%
Visual Design & CSS 25 Marks	No attention to the visual design of the pages of this application and the placement of information. No colours, or inadequate colour schemes were used. Content and layout are not properly separated, and very little styling has been applied	Little attempt at providing a coherent and consistent interface style. There is little evidence of attention to the layout and consistency of the visual design. Very simple rules, some selectors and common properties used to create a very basic layout	Appropriate layout and design of information within the pages of this application. External stylesheets used. Different rules for different elements have been applied to create a straightforward layout. Multiple types of selectors have correctly been used to apply a straightforward layout	Good layout and design of the information within the pages. The visual design appears consistent and there is a clear differentiation between different kinds of screen sizes. External stylesheets have correctly been applied. A variety of selectors, properties and values such as relative units have been used.	Excellent visual design of the web application. The design appears appropriate, clear and consistent and fits to the specification. There is a clear differentiation between different kinds of screen sizes. Well-structured, external stylesheets have correctly been applied to create a responsive, original and well-designed web
HTML Markup, Site Structure, and content 25 Marks	Inadequate navigation or structure of the web application. Major omissions or placeholder content included. No real attempt at using HTML markup. Errors are present in the markup and deprecated features have been used.	Poor structure of the application and its content. The navigation does not appear to be thought out and may include occasional omissions and placeholder text as content. Some attempt at using the appropriate markup for the web application.	Reasonably structured web application. Necessary navigational tools provided. All necessary areas included and no placeholder content. Structured markup used to create an adequate site. Shows use of relevant semantic elements.	Good, well thought out structure. Clear and consistent navigation between the different web pages of this application. Semantic elements have been used throughout the application and it is built with regards to current standards.	Excellent, clear and concise navigation and structure. The pages are very user friendly and adhere to modern conventions allowing users to find content and use them with ease. Complies with current standards and is built with regard to future development. Consistent use of the relevant HTML elements using well-structured code.
Client-Side Functionality, Interactivity & API Access 30 Marks	No real attempt at providing interactivity and the required functionality using JavaScript	Occasional Flaws in a straightforward attempt to provide some level of required functionality. Little attempt at using JavaScript or HTML5 to receive data from external resources.	The client-side functionality demonstrates reasonable understanding and competence in the technologies taught. Minor flaws or omissions may be included.	Good attempt at providing the required functionality. JavaScript or HTML5 has successfully been used to fetch data from remote APIs and display them appropriately	Excellent client-side functionality covering all the required specifications with well-structured and well-designed code. Student demonstrates a very good understanding of the technologies chosen.

		Or Data is fetched from the API but not integrated into the webpage			All relevant data from the remote APIs has been fetched and is displayed in a structured, well designed, engaging interface
Project Scope & Validation 20 Marks	Inadequate attempt at matching the coursework specification. Validation errors present, files missing, incorrect URLs used.	Little attempt to provide the appropriate levels of interactivity and functionality of this application. Some validation errors present.	Reasonable attempt at providing the appropriate levels of interactivity and functionality with some features missing or erroneous. Little to no validation errors	Good attempt and fulfilling the project specification with little flaws. Good file organisation, suitable file formats used and mostly no validation errors.	Excellent well-developed application showing high level of file organisation and no validation errors of HTML and CSS files. This clearly fulfils the project specification