10 Appendices

10.1 Appendix 1 – Unit Details

Unit 1: User Centric Front End Development

Level:	Level 5
Credit Value:	12
GLH:	102
Unit Number:	Y/650/3525
Unit Aim:	This unit aims to provide learners with the knowledge and skills needed to build a Front end web application. It includes understanding the principles of responsive design, documentation and effective layout of content.

This unit has 5 learning outcomes.

LEARNING OUTCOMES	ASSESSMENT CRITERIA - PASS	MERIT CRITERIA*	DISTINCTION
The learner will:	**The learner can:**	**In addition to the pass criteria, the learner can:**	
1 Design a Front end web application based on the principles of user experience design, accessibility and responsivity.	1.1 Design a website that incorporates a main navigation menu and a structured layout. 1.2 Design a website that meets accessibility guidelines (e.g. contrast between background and foreground colours, non-text elements have planned alternative text equivalents to cater for the visually impaired). 1.3 Design the organisation of information on the page following	M(i) Design a website with a flow of information layout, and interaction feedback which are clear and unambiguous.	Please refer to the performance characteristics for distinction.**

WireFrame

WebPage

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	the principles of user experience design (headers are used to convey structure, information is easy to find due to being presented and categorised in terms of priority). 1.4 Ensure that foreground information is never distracted by backgrounds. 1.5 Include graphics that are consistent in style and colour. 1.6 Design the site to allow the user to initiate and control actions such as pop-ups and playing of audio/video.		
2 Develop and implement a static Front end web application using HTML and CSS.	2.1 Create a website of at least 3 pages, or (if using a single scrolling page) at least 3 separate page areas, to match the design and to meet its stated purpose. 2.2 Write custom CSS code that passes through the official (Jigsaw) validator with no issues. 2.3 Write custom HTML code that passes through the official W3C validator with no issues. 2.4 Incorporate images that are of sufficient resolution to not appear pixelated or stretched.	M(ii) Implement a website whose purpose is immediately evident to a new user without having to look at supporting documentation. M(iii) Implement a website that provides a solution to the user story demands and expectations.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	2.5 Code all external links to open in a separate tab when clicked. 2.6 Use CSS media queries or CSS Grid/Bootstrap across the application to ensure the layout changes appropriately and maintains the page's structural integrity across device screen sizes. 2.7 Use Semantic markup to structure HTML code. 2.8 Present the finished website with clearly understandable site-specific content, rather than Lorem Ipsum placeholder text. 2.9 Implement clear navigation to allow users to find resources on the site intuitively.		
3 Maximise future maintainability through documentation, code structure and organisation.	3.1 Write a README.md file for the web application that explains its purpose, the value that it provides to its users, and the deployment procedure. 3.2 Insert screenshots of the finished project that align to relevant user stories. 3.3 Attribute all code from external sources to its original source via comments above the code and (for larger dependencies) in the README.		

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	3.4 Clearly separate and identify code written for the website and code from external sources (e.g. libraries or tutorials). 3.5 Organise HTML and CSS code into well-defined and commented sections. 3.6 Place CSS code in external files, linked to the HTML page in the HEAD element. 3.7 Write code that meets at least minimum standards for readability (consistent indentation, blank lines only appear individually or, at most, in pairs). 3.8 Name files consistently and descriptively, without spaces or capitalisation, to allow for cross-platform compatibility. 3.9 Group files in directories by file type (e.g. an assets directory will contain all static files and may be organized into sub-directories such as CSS, images, etc.)		
4 Use version control software to maintain, upload and share code with other developers.	4.1 Use a cloud-based, git-based, version control system (e.g. Git & GitHub) throughout the development and implementation process.	M(iv) Commit often, for each individual feature/fix, ensuring that commits are small and well-defined, with clear, descriptive messages.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	4.2 Document the development process through descriptive commit messages. 4.3Use consistent and effective markdown formatting to produce a README file that is well-structured, easy to follow, and has few grammatical errors.		
5 Test and deploy a Front end web application to a Cloud platform.	5.1 Design and implement manual testing procedures to assess functionality, usability and responsiveness. 5.2 Document the testing in the README or in a separate file. 5.3 Deploy a final version of the code to a cloud-based hosting platform (e.g. GitHub Pages) and test to ensure it matches the development version. 5.4 Remove commented-out code before pushing final files to version control and deploying. 5.5 Ensure that there are no broken internal links.	M(v Present a clear rationale for the development of the project, in the README, demonstrating that it has a clear, well-defined purpose addressing the needs of, and user stories for a particular target audience (or multiple related audiences). M(vi) Document testing fully to include evaluation of bugs found and their fixes and explanation of any bugs that are left unfixed. M(vii) Fully document the development life cycle procedures in the README file.	

*Additional guidance for merit

To achieve merit learners, need to meet the assessment criteria outlined above for a pass and a merit.

The following additional guidance describes characteristics of performance at MERIT.

The learner has a clear rationale for the development of this project and has produced a fully functioning, well-documented, website for a real-life audience.

The finished project has a clear, well-defined purpose addressing the needs of a particular target audience (or multiple related audiences). Its purpose would be immediately evident to a new user without having to look at supporting documentation. The design of the web site follows the principles of UX design and accessibility guidelines, and the site is fully responsive.

Code is well-organised and easy to follow, and the application has been fully tested, following a planned, manual testing procedure, with no obvious errors left in the code.

The development process is clearly evident through commit messages. The project's documentation provides a clear rationale for the development of this project and covers all stages of the development life cycle.

**Characteristics of Performance at Distinction

To achieve a distinction, a learner will have achieved all pass and merit criteria, as described above, and will demonstrate characteristics of high level performance as described below:

Characteristics of performance at DISTINCTION:

The learner has documented a clear, **justified**, rationale for a real-world application and a comprehensive explanation of how it will be developed. The development of the project has resulted in a fully-functioning, interactive, web application, developed using at least one more advanced technique (e.g. CSS media queries)

The finished project is judged to be publishable in its current form with a clearly evidenced professional grade user interface and interaction adhering to current practice. There are no obvious errors in the code. Where there is a clear breach of accepted design/UX principles, or of accepted good practice in code organisation, these are fully justified, appropriate and acceptable to the target user. It clearly matches the design and demonstrates the characteristics of **craftsmanship** in the code.

Code is clearly signposted and there is clear, documented evidence of testing at all stages of development. There is full evidence of end testing, and there is an evaluation of bugs that may remain in the code.

The development and testing process is clearly evident, **and justified**, through commit messages. The project's documentation provides a **clear and detailed** rationale for the development of this project, covering **all stages** of the development life cycle.

Amplification (craftsmanship) This amplification is only applicable to performance at distinction.

Design

The design of the web application demonstrates the main principles of good UX design:

- Information Hierarchy
  - headers are used to convey structure - each section has a header that is easy to see and clear to understand
  - all information displayed on the site is presented in an organised fashion with each piece of information being easy to find
  - all resources on the site are easy to find, allowing users to navigate the layout of the site intuitively
  - information is presented and categorised in terms of its priority
- User Control
  - all interaction with the site would be likely to produce a positive emotional response within the user. This is down to the flow of information layout, the clear and unambiguous navigation structures and all interaction feedback
  - when displaying media files, the site avoids aggressive automatic pop-ups and autoplay of audio; instead letting the user initiate and control such actions
- Consistency
  - evident across all pages/sections and covers interactivity as well as design
- Confirmation
  - user actions are confirmed where appropriate, feedback is given at all times
- Accessibility
  - there is clear conformity to accessibility guidelines across all pages/sections and in all interactivity

Any design decisions that contravene accepted user interaction, user experience design principles are **identified and described** (comments in code and/or a section in the README)

Development and Implementation

Code demonstrates accepted standards for readability (including characteristics of 'clean code'):

- Consistent and appropriate naming conventions within code and in file naming, e.g.
  - file names and class names, are descriptive and consistent
  - for cross-platform compatibility, file and directory names will not have spaces in them and will be lower-case only
  - all HTML attributes and CSS rules, are consistent in format, follow standards for the language and are appropriate and meaningful
- File structure e.g.
  - whenever relevant, files are grouped in directories by file type (e.g. an assets' directory will contain all static files and code may be organised into sub-directories such as CSS, etc)
  - there is a clear separation between custom code and any external files (for example, library files are all inside a directory named 'libraries')

-
    - files are named consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility.
- Readability
    - code is indented in a consistent manner to ease readability and there are no unnecessary repeated blank lines (and never more than 2)
    - function/class/variable names clearly indicate their purpose
    - CSS code is split into well-defined and commented sections
    - Semantic mark-up is used to structure HTML code
    - HTML and CSS are kept in separate, linked files
    - CSS files are linked in the HTML file's head element
- Defensive design
    - any potential user errors are identified and dealt with
- Comments
    - all custom code files include clear and relevant comments explaining the purpose of code segments
- Compliant code
    - HTML code passes through the official W3C validator with no issues
    - CSS code passes through the official (Jigsaw) validator with no issues
    - JavaScript code passes through a linter (e.g. jslint.com) with no major issues
- Robust code
    - no logic errors are found when running code
    - errors caused by user actions are handled
    - where used, API calls that fail to execute or return data will be handled gracefully, with the site users notified in an obvious way
    - inputs are always validated.

The full design is implemented providing **a good solution to the users' demands and expectations**.

Real world application

- Clearly understandable site-specific content is used rather than Lorem Ipsum placeholder text
- The final application is aligned to the user stories presented at the start of the project

Testing procedures are comprehensive, with a good level of coverage, and have clearly been followed. All noticeable errors have been corrected or documented.

Version control systems are used effectively:

- all code is managed in git with well-described commit messages
- there is a separate, well-defined commit for each individual feature/fix
- there are no very large commits which make it harder to understand the development process and could lead the assessor to suspect plagiarism

The full application development process is documented:

- the purpose of the application is clearly described in the README
- the project's documentation describes the UX design work undertaken for this project and the reasoning behind it
    - wireframes, mockups, diagrams, etc., created as part of the design process are included in the project

- there is a clear separation between code written by the learner and code from external sources (e.g. libraries or tutorials). All code from external sources is attributed to its source via comments above the code and (for larger dependencies) in the README
- the testing procedure is well documented either in the README or a separate file
- the deployment procedure is fully documented in a section in the README file
- clear, well-described, explanatory commit messages describe the development process
- the README is well-structured and easy to follow
- the README file is written in markdown and uses markdown formatting consistently and effectively
- project documentation and the application's user interface have few errors in spelling and grammar.