Unit 2: Interactive Front End Development

Level:	Level 5
Credit Value:	16
GLH:	152
Unit Number:	A/650/3526
Unit Aim:	This unit aims to provide learners with the knowledge and skills needed to build a dynamic, interactive Front end web application. It includes understanding user input and control when interacting with a web application. It also includes understanding the principles of automated and manual testing for debugging.

This unit has 5 learning outcomes.

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
1 Design, develop and implement a dynamic Front end web application using HTML, CSS and JavaScript.	1.1 Design a web application that meets accessibility guidelines, follows the principles of UX design and presents a structured layout and navigation model, and meets its given purpose. 1.2 Design interactivity for a web application that lets the user initiate and control actions and gives feedback. 1.3 Write custom JavaScript, HTML and CSS code to create a responsive Front end web application consisting of one or more HTML pages with	M(i) Design a web application following the principles of UX design which meets accessibility guidelines, is easy to navigate and allows the user to find information and resources intuitively.	Please refer to the performance characteristics for distinction.**

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	significant interactive functionality. 1.4 Write JavaScript code to produce relevant responses to user actions. 1.5 Implement an interactive web application that incorporates images or graphics of usable resolution, legible, unobscured text, consistent styling, undistracted foregrounds.		
2 Implement Front end interactivity, using core JavaScript, JavaScript libraries and/or Application Programming Interfaces (APIs).	2.1 Write JavaScript code, that passes through a linter (e.g. JSLint) with no major issues and write validated HTML and CSS code. 2.2 Write JavaScript functions that correctly implement compound statements such as “if statements” and/or loops. 2.3 Write code that intelligently handles empty or invalid input data. 2.4 Implement appropriate working functionality for all project requirements. 2.5 Organise non-trivial JavaScript code in external file(s) linked at the bottom of the body element (or bottom of head element if needs to be loaded before the	M(iv) Write code such that users who direct to a non-existent page or resource are redirected back to the main page without having to use browser navigation buttons.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	body HTML) and CSS code in external files linked to HTML in the head element. 2.6 Write code that meets minimum standards for readability (comments, indentation, consistent and meaningful naming conventions). 2.7 Name files consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility. 2.8 Write code that does not generate internal errors on the page or in the console as a result of user actions. 2.9 Organise code and assets files in directories by file type.		
3 Test an interactive Front end web application through the development, implementation and deployment stages.	3.1 Explain the principles of automated and manual testing and when each might be deployed. 3.2 Design and implement testing procedures (automated or manual) to assess functionality, usability and responsiveness of the web application. 3.3 Insert screenshots of the finished project that align to relevant user stories.		

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 Apply test procedures during development and implementation stages and test to ensure the deployed version matches the development version. 3.5 Fully document the results of well-planned testing procedures (automated or manual) to assess the website's functionality, usability and responsiveness.		
4 Deploy an interactive Front end web application to a Cloud platform.	4.1 Deploy a final version of the interactive web application code to a cloud-based hosting platform (e.g. GitHub Pages). 4.2 Ensure that the deployed application is free of commented out code and has no broken internal links. 4.3 Use Git & GitHub for version control of an interactive web application up to deployment.	M(v) Commit often, for each individual feature/fix, ensuring that commits are small, well-defined and have clear, descriptive messages.	
5 Demonstrate and document the development process through a version control system such as GitHub.	5.1 Document the full development cycle, with clear evidence given through commit messages, the README. 5.2 Write a README.md file for the interactive web application that explains its purpose and the value that it provides to its users. 5.3 Clearly separate and identify code written for the interactive	M(vi) 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	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	web application and code from external sources (e.g. libraries or tutorials). Attribute any code from external sources to its source via comments above the code and (for larger dependencies) in the README. 5.4 Use consistent and effective markdown formatting that is well-structured, easy to follow, and has few grammatical errors, when writing a README file.	(or multiple related audiences). M(vii) Document the UX design work undertaken for this project, including any wireframes, mock-ups, diagrams created as part of the design process, and the reasoning behind it. Include diagrams created as part of the design process and demonstrate that these have been followed through to implementation. M(viii) Document testing fully to include evaluation of bugs found and their fixes and explanation of any bugs that are left unfixed. M(viii) Fully document the deployment procedure in a section 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, interactive, web application for a real-life audience, with specific content rather than placeholders. There is a range of interactive features. Data validation, API handling (when applied) and user feedback are all evident in the code and the working application. Optionally, a range of external and internal APIs (e.g. TMDb, Google Maps, etc.) have been used to produce working features. There are no logical errors in the code and the application functions as expected.

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 user is kept informed of progress and actions through feedback and, where large data sets are being loaded, progress indicators. The design of the web application 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 manual or automated testing procedures, with no obvious errors left in the code. If automated testing is implemented unit test files should be present in code commits.

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. When used, the learner shows a clear understanding of asynchronicity and the timing problems that can arise when accessing shared data values.

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 logic 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. The resulting application is original and not a copy of any walkthrough projects encountered in the unit

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
  - users who direct to a non-existent page or resource are redirected back to the main page without having to use the browser navigation buttons
  - the user is shown progress indicators where relevant
  - errors resulting from user actions are reported to the user
- 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, class name, function names and variable 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, CSS rules, code variables and function names 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, JavaScript, 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
    - all code is split into well-defined and commented sections
    - functions are well-defined and commented to indicate purpose
    - Semantic markup is used to structure HTML code
    - HTML, CSS and JavaScript are kept in separate, linked files
    - CSS files are linked to in the HTML file's head element
    - non-trivial JavaScript code files are linked at the bottom of the body element (or bottom of head element if needs loaded before the body HTML)
- Defensive design
    - all input data is validated (e.g. presence check, format check, range check)
    - internal errors are handled gracefully, and users are notified of the problem where appropriate.
- 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
- Navigating between pages via the back/forward buttons can never break the site
- All links to external pages open in a separate tab when clicked
- 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:

- navigating between pages via the back/forward buttons can never break the site, there are no broken links
- user actions do not cause internal errors on the page or in the console
- if automated testing is implemented unit tests are included in code commits

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, mock-ups, 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