Assignment Brief – Coursework

Module:	SOFT352 Client-side Web Scripting
Assignment Title:	Web Application
Individual/group assignment:	Individual
Submission Deadline:	1. Proposal submission 3pm, Thursday 2nd November 2017
	Return date: Thursday 16st November 2017
	2. Demonstrations 8th – 11th January 2018
	3. Final submission 3pm, Friday 12th January 2018
	Return date: Friday 9th February 2018
Submitted at:	DLE module site (proposal and final submission)
Contribution to module grade:	90%

Assessed learning outcomes

ALO1. Apply principles of object oriented and event-based scripting as well as synchronous and asynchronous client-server communication.

ALO2. Demonstrate an understanding of how application content is represented and communicated across the web and how this affects the user experience.

ALO3. Design, implement and evaluate/test dynamic web-based application.

Specification

This is a negotiated, open-ended project where you must define the specific content you will produce and process you will follow, within the guidelines given below. Previous projects have included games, A-life simulations and interactive learning tools.

You must produce a working ***prototype*** using dynamic web technologies, i.e., HTML, CSS and JavaScript (JS). You may also use other web technologies you find interesting or useful, e.g., WebRTC. The application must JavaScript as the main development language both client-side and server-side. The final product should reflect an effort of 80+ hours of dedicated work.

You must also provide a description of your ***DevOps pipeline***. This may include, your code repository, continuous integration server, unit tests, behaviour tests, code analyses, usage metrics and usage analyses.

Marking Scheme

The marking of the project takes three elements into account. First, the ***system requirements*** must all be met or the project will be awarded an automatic fail. These requirements are described in detail below. Second, the ***base criteria*** will decide your base grade reflecting the quality of your software and your development process. These criteria are presented in detail in Table 1. Third, your base grade will be modified by two ***multipliers*** reflecting overall aspects of your work. These multipliers are explained in Table 2. Baselines are provided for each of the grading criteria explaining the expected achievement for a lower second (50%-60%) level grade.

Base Criteria	Weight	Explanation	Baseline
Functional Requirements, Analysis and Testing	20%	Your handling of requirements will be evaluated based on the quality and appropriateness of artefacts such as personas, use-cases and BDD. Your testing will be evaluated on the relevant documentation as well as any test-related code.	The documentation includes personas, use-cases and documentation of systematic testing. For higher grades usability testing, TDD and BDD are required.
Design and Documentation	10%	The design will be evaluated based on the quality and appropriateness of relevant documentation such as sketches for the GUIs, UML for the system architecture. It also includes the identification of relevant software design patterns.	GUI sketches and UML diagrams for high-level structure and deployment. For higher grades, sequence diagrams and other appropriate UML diagrams are also expected.
Software Quality and Structure	40%	The quality of the software will be evaluated based on the general software quality metrics, in particular the 'high cohesion', 'low coupling', 'protected variation' and 'don't repeat yourself' (DRY) principles.	The code uses appropriate functions, objects and classes and follows framework patterns, e.g., MVC for Angular. For higher grades code implements additional patterns such as the Factory or Observer patterns when appropriate.
Development and Operations Process	30%	Your use of development and operations support tools, including version control, continuous integration and DevOps.	The process includes version control and there is documentation of real-time performance analysis. For higher grades you should include CI, feature flagging and usage metrics/analysis.

Table 1: Base criteria for grading

Multiplier	Range	Explanation	Baseline
System Complexity	0.8-1.2	This multiplier reflects the overall complexity of your system, reflecting the complexity of your code and the number of features and technologies it contains.	Hundreds of lines of code written by the student. For higher grades, the inclusion of third party libraries is also expected.
Understanding	0.5-1.0	This multiplier reflects your understanding of how your system works, including your code as well as applied software technologies and third-party libraries.	Good understanding of all the code written by the student. Minor misunderstandings and omissions is acceptable.

Table 2: Grade multipliers

System Requirements

The final application must have the following properties:

P1. The application should be ***interactive***, i.e., the user should be able to affect its behaviour by interacting with it using a mouse, a keyboard or another suitable input device.

P2. The application should be ***distributed or parallel***, i.e., it should interact with more than one computer or processor through any suitable technology, e.g., AJAX, WebSockets, RTCPeerConnections, WebWorkers or WebGL.

Deliverables

There are three main requirements for this assignment. Deliverable D1, the project proposal, is a pass/fail deliverable, i.e., it does not affect the final module mark beyond deciding whether the project achieves a pass or fail grade. Only deliverable D2, the demonstration, and deliverable D3, the final submission, will contribute to the final module mark beyond pass/fail.

D1. Project Proposal

This is a brief summary of your planned application. It should explain what the application will do and which technologies you plan to use. The proposal should also explain how the application will satisfy each of the system requirements and how you plan to evaluate its performance.

The purpose of the proposal is to give you early feedback on the suitability and feasibility of your planned work in order to avoid you spending a lot of time on something that is not appropriate.

D2. Demonstration

The demo will take place at a location and time that will be announced nearer to the time. During the demonstration, you must both demonstrate and explain how you implemented your application including both the DevOps process and the application functionality. This will include detailed questions about how your code works.

You are expected to write your own code. Discussing ideas with other people is acceptable, but getting help to write code is not acceptable. Failure to demonstrate a good understanding of your own code during the demonstration may lead to significantly

reduced grade on this assignment or, in the worst case, actions against you under the University's regulations on examination and assessment offences[1].

D3. Final Submission

For this requirement, you must submit a single zip archive containing two main elements, your source code and a written report describing your project.

There is a **150MB limit on the final submission**. Please consider submitting your files from a computer on the University campus as submitting files of this size over an unreliable network connection can be problematic. Please check that your submitted files are correct by downloading them again and checking that they still work.

You must include the following elements in the final submission:

a. Source Code

This should include all the code you have written yourself and should also, as far as possible, contain copies of the folders and files needed to run your application. Remember to include any supporting class files you have developed and any server-side code you have produced. It is often not possible to submit a runnable project. If you rely on a major framework such as the Flash Media server, it is clearly not feasible to include everything.

If you submit code files or libraries you have not written yourself you need to make this clear in your report. Submitting code you have not written yourself without making this clear in the report may lead to actions against you under the University's regulations on examination and assessment offences.

b. Report

The report must be an MS Word document of no more than 2,000 words. Please use screen shots and sketches to illustrate the functionality and UML diagrams to illustrate the design. The report should explain:

- **Functionality** (ca. 500 words with screen shots)
  - What does the application do?
  - How do users interact with it?
  - What technologies where included?
- **Requirements** (ca 200 words with additional documents in appendices)
  - Who were the app aimed at?
  - What features were included and why?
  - Functional, e.g., unit, testing
- **Design** (ca. 200 words with UML diagrams)
  - What processes run where (clients/servers/peers)?
  - How do the processes interact?
  - How are the data and code structured?
  - Why are these structures appropriate?

[1] Further detail on the University's assessment regulations can be found at https://www1.plymouth.ac.uk/essentialinfo/regulations/Pages/Plagiarism.aspx.

- **Performance** (ca. 500 words, may include quantitative and qualitative data, result statistics, screenshots or diagrams)
    - How effective were the user interaction mechanisms (feedback from user testing)?
    - How responsive is the application?
    - How quickly does it load?
    - How heavy/light is the network load generated?
    - How robust is your application to network failures and unreliable users?
    - How are users interacting with it?
- **Development Process** (ca 400 words)
    - Describe your development process
    - Describe tools and technologies supporting your development process
- **Personal Reflection** (ca. 200 words)
    - What worked/didn't work well in terms of how you worked and the technologies you used?
    - What lessons would you bring from this project into your next project?