COMP3006
FULL-STACK DEVELOPMENT

20 CREDIT MODULE

ASSESSMENT: 100% Coursework **W1: 30% Set Exercises**
W2: 70% Report

MODULE LEADER: Dr David Walker

MODULE AIMS

- This module explores the production of dynamic web applications with a particular focus on the web environment.
- Key elements such as object oriented and event-based scripting, asynchronous client-server communication and distributed content representation are explored through practical production.
- The production of working systems use frameworks such as HTML, CSS, JavaScript/JQuery and Node.js.

ASSESSED LEARNING OUTCOMES (ALO):

1. Apply principles of object oriented and event-based scripting as well as synchronous and asynchronous client-server communication.
2. A Demonstrate an understanding of how application content is represented and communicated across the web and how this affects the user experience.
3. Design, implement and evaluate/test dynamic web-based applications.

Overview

This document contains all the necessary information pertaining to the assessment of *COMP3006 Full-Stack Development*. The module is assessed via **100% coursework**, across two elements: *30% Set Exercises* and *70% Report*.

The sections that follow will detail the assessment tasks that are to be undertaken. The submission and expected feedback dates are presented in Table 1. All assessments are to be submitted electronically via the respective DLE module pages before the stated deadlines.

	Submission Deadline	Feedback
Set Exercises (30%)	**3rd November 2020, 12pm**	1 December 2020
Report Proposal	**10th November 2020, 12pm**	17 November 2020
Report (70%)	**14th January 2021, 12pm**	11 February 2021

Table 1: Assessment Deadlines

All assessments will be introduced in class to provide further clarity over what is expected and how you can access support and formative feedback prior to submission. Whilst the assessment information is provided at the start of the module, it is not necessarily expected you will start this immediately – as you will often not have sufficient understanding of the topic. The module leader will provide guidance in this respect.

Assessment 1: Web Development Exercises

This assignment assesses Assessed Learning Outcome 3.

Task:

This assignment contributes **30%** of the overall module mark for COMP3006 and is an **individual assignment**. You are required to write HTML, CSS and JavaScript (JS) to develop a dynamic web page according to the instructions below. The coursework will be automatically marked through a set of unit tests. The goal of your code is to pass the tests. The specifications are only a guide to how this can be done. Code that satisfies the specifications without passing the tests will not be awarded any marks.

Start by downloading the ZIP archive provided on the DLE, called **"COMP3006_CW1.zip"**. This archive contains the file you should edit, **"COMP3006_CW1.html"**. The archive also contains several other files that enable the automated testing of your HTML, CSS and JS code, in particular the **"COMP3006_CW1_Tests.js"** file. You must NOT modify these files as it will defeat the purpose of the exercise. **I will be marking against fresh copies of the unit tests**.

The required files will become available on **5th October 2020**.

A function called **rgb2hex** has been provided in the unit test file, which you are welcome to use within your program to convert between the format in which the browser stores colours and the format used in these questions.

SUBMISSION

You are required to submit a ZIP archive containing all relevant files to the link provided on the module DLE page. You must submit your completed work by the specified submission dates.

REQUIRED FEATURES

There are 30 tests in total, reflecting the 30 features below. Your task is to build a web interface for a temperature monitor.

Structure/HTML (10 unit tests)

1. Set the browser title for this document to be "Temperature Monitor".
2. Add a *div* to the the page with the ID *monitor*.
3. Within the *monitor* div add a level one heading with the content "Temperature Monitor" and ID *heading*.
4. Below the heading add a text input with the ID *temperature* and the initial value "Enter temperature".
5. Below the text input add an checkbox *input* with the ID *fahrenheit*.

6. Next to the checkbox input, add a *label* with ID *fahrenheit_lbl* and use the *for* attribute to refer to the checkbox. The value of the label should be “Fahrenheit”.
7. Below the button add a *paragraph* with the ID *output*.
8. Below the *output* paragraph add another paragraph with the ID *time*.
9. Below the inputs add a *button* element with the ID *check* and the value “Check temperature”.
10. Alongside the *check* button add another element with the ID *reset* and the value “Reset”.

Style/CSS (10 unit tests)

11. Set the body background colour to be #ffff99.
12. Set the body to display text in Arial font.
13. The text in the level one heading should be 24px.
14. Set the *monitor* background colour to be #aeaeae.
15. Set the width of the *monitor* to be 200px.
16. Add a border to the *monitor* – it should be #010101 in colour, and be a 1px solid line.
17. Set the background colour of the buttons to #585858 and display the text in #ffffff.
18. The *output* paragraph should be 100px wide.
19. The colour of the *fahrenheit_lbl* label should be #0000fe.
20. Set the *check* and *reset* buttons to be disabled.

Behaviour/JS (10 unit tests)

21. If the value of the *temperature* input is changed to an integer value the *check* button should be enabled.
22. When the *check* button is clicked the *reset* button should be enabled.
23. Depending on the temperature in degrees **Celsius**, the monitor should alter the *output* element as follows when the *check* button is clicked:
    a. If the temperature is less than or equal to 15 degrees it should display the message “Too cold” with the background colour #6699ff.
    b. If the temperature is greater than 15 degrees and less than or equal to 19 degrees it should display the message “Ideal” with the background colour #ffcc00.
    c. If the temperature is greater than 19 degrees and less than or equal to 23 degrees it should display the message “Warm” with the background colour #ff9900.
    d. If the temperature is greater than 23 degrees it should display the message “Too hot” with the background colour #ff0000.
24. The monitor should process Fahrenheit temperatures:
    a. Write a function called *f2c* that converts Fahrenheit to centigrade using the formula:

$$c = (f - 32) * 5/9,$$

    where *f* is the temperature in Fahrenheit retrieved from the *temperature* input and *c* is the equivalent temperature in Celsius. Your function must return the Celsius value.

    b. If the *fahrenheit* box is ticked when the *check* button is clicked, the *output* element should behave correctly.

25. If the reset box is clicked the *temperature* contents should be removed, the *output* contents removed and it's background reset to #ffffff.
26. If the unchecked *fahrenheit* box is checked when the output paragraph has been populated (the *check* button has been clicked but the *reset* button has not) then the *output* paragraph should be recomputed.

Assessment Criteria:

This assignment is marked out of a total of 50 marks, and will be assessed as follows.

- **Structure/HTML:** 1 mark for each correctly passing test.
- **Style/CSS:** 1 mark for each correctly passing test.
- **Behaviour/JS:**
    - 1 mark for each correctly passing test **in the test suite distributed with the coursework.**
    - 2 marks for each correctly passing test **in an unseen test suite I will be using for marking.**

The unseen dataset provides additional tests using different test data to ensure that you have implemented sufficiently generic software (rather than hardcoding the software to the test data I have distributed). It does not test additional requirements – the test suite you have been issued with is complete.

Assessment 2: Full-Stack Project

This coursework assesses Assessed Learning Outcomes 1-3.

Task:

This assignment contributes **70%** of the overall module mark for COMP3006 and is an **individual assignment**. Both the **proposal** and the **final submission** must be submitted to the DLE by the specified submission dates.

This is a negotiated project in which you must define the specific content you will produce and the process you will follow, within the guidelines given below. Examples of previous projects can be found on the DLE. Your project **must satisfy** the following requirements:

- The server should use a database (either **MongoDB or PouchDB**) to store information and pass it to the client through an API. Other types of database are not acceptable for this module.
- WebSockets should be used to enable communication between two clients.

You must produce a working system, comprising a client constructed using dynamic web technologies (HTML, CSS and JavaScript) and a server using Node.js. The application must use JavaScript as the main development language both client-side and server-side. The system must be interactive, i.e., the user should be able to affect its behavior by interacting with it using a keyboard, mouse, or another suitable input device. The system must run on multiple computers, i.e., it must be distributed. The final product should reflect an effort of 80+ hours of dedicated work.

You must document the system, including its design and details of the implementation process you have followed, and provide a description of your DevOps pipeline. This could include your code repository, continuous integration/deployment setup, unit tests, integration tests, behavior tests, code analyses, usage metrics and usage analyses.

COURSEWORK DELIVERABLES

There are three main deliverables for this assignment. Deliverable **D1**, the project proposal, which does not affect the final module mark but is required to ensure that your project is suitable. Only deliverable **D2**, the final submission, and **D3**, the video demonstration, will contribute to the final module mark.

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 stop you spending lots of time on something that is inappropriate.

A form has been provided on the DLE which must be submitted by the deadline specified.

D2 – Final Submission

For this deliverable 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**. If you are struggling to fit into the 150MB limit you can remove you **.node_modules** directory. Please check your submitted files are correct by downloading them again and checking that they work. **You will receive a confirmation receipt by email when your work has been properly submitted** – if you do not receive this email then your work has not been submitted.

You must include the following elements in the final submission:

1. **Source Code:**
This should contain all of the code you have written yourself and should, as far as possible, contain copies of the folders and files needed to run your application.

2. **Report:**
The report must be a document of no more than 2,000 words. Please use screen shots and sketches to illustrate the functionality and UML diagrams (or appropriate alternatives) to illustrate the design. The report should explain:
    - Requirements (ca. 400 words with additional documents in appendices)
        - Who is the application aimed at?
        - What features were included and why?
    - Design (ca. 500 words with UML diagrams)
        - What is the system architecture (clients/servers/peers)?
        - How do the processes interact?
        - How are the data and code structured?
        - Why are these structures appropriate?
    - Testing (ca. 400 words)
        - What automated testing have you performed (e.g., unit testing)?
        - What usability testing have you performed?
        - Why is your chosen test strategy appropriate?
    - DevOps pipeline (ca. 400 words)
        - Describe your development environment.
        - Describe your continuous integration pipeline and how you used it.
    - Personal reflection (ca. 300 words)
        - What worked/didn't work well in terms of your work and the technologies you used?
        - What lessons would you take from this project into your next project.

D3 – Video

You are required to produce a video of **no more than 5 minutes in length**. During this video you must show the key features of the project (show them working in the browser), outline its architecture, and talk briefly about how you have used DevOps.

You are required to **upload the video to YouTube and provide a link** to the video in your report for deliverable D2. **Failure to do this will cost you significant marks as this is your only opportunity to demonstrate the software working**.

If the Module Leader decides that it is warranted, you may be required to demonstrate your system in person. This will happen during weeks 26 and 27 (Semester 1 assessment weeks).

Assessment Criteria:

Your work will be assessed according to the rubric found in Table 2. Your mark for this piece of coursework will be based on an aggregation of the marks for each category. Marks will be awarded based on **both the demonstration and the** report.

Category	Fail (< 40%)	>= 40%	>= 50%	>= 60%	>= 70%
Analysis (10%)	Insufficient problem analysis. Functional requirements are not described in sufficient detail.	There are some vague functional requirements based on a basic outline of the problem.	Reasonably detailed functional requirements but based on weak analysis.	Functional requirements are well specified with some evidence of strong analysis.	Functional requirements are complete and are well motivated based on strong analysis.
Design (10%)	Little or no design work. It is not clear how the system will be built, or what it's major components are.	Some overview of the design but no detail. The architecture is not defined. Little or no use of diagrams to show the structure and operation of specific sections of the system.	Some of the system's design is specified in detail but other areas are weak. Some diagrams are included.	Design is mostly complete, further detail would enhance some areas. Most functionality and design specified with diagrams.	System design complete and comprehensively described with the use of appropriate diagrams.
Software (50%)	Very little software has been constructed, and what there is does not work.	Most of the functionality has not been implemented. Major errors at runtime. WebSockets and/or a database have been attempted but incomplete. Much of the system is affected by bugs.	Some requirements implemented but major omissions. Software contains major runtime errors. Either WebSockets or a database used. The software has some bugs affecting the main functionality.	Largely complete implementation of most requirements. Some runtime errors. WebSockets and database used, one of them weakly. There are some apparent bugs but they do not impede the main functionality.	Implementation of all requirements is complete. WebSockets and a database are used, both work correctly. Very few apparent bugs.
Testing (20%)	Little or no testing of any kind.	Some testing – either code (unit or integration) or usability testing has been completed, but large parts of the system are untested.	A reasonable level of testing – either code (unit or integration) or usability testing has been completed. A good amount of the system is tested in some way.	Reasonably complete code (unit or integration) and usability testing. A good amount of the system is tested with both approaches.	Comprehensive code testing with both unit and integration tests. A thorough usability study has been performed and feedback has been taken on board. Other types of testing (e.g., load) have been used.
CI/CD (10%)	No attempt at an automated CI pipeline.	No automated CI pipeline. Some use of version control.	Good use of version control but no automated CI pipeline.	Version control has been used effectively. Some indication of test-driven development. CI pipeline has been attempted.	Version control is used well. CI pipeline is well organised and has been demonstrated to work.

Table 2: Feedback Template for Assessment 2

General Guidance

Extenuating Circumstances

There may be a time during this module where you experience a serious situation which has a significant impact on your ability to complete the assessments. The definition of these can be found in the University Policy on Extenuating Circumstances here:
https://www.plymouth.ac.uk/uploads/production/document/path/15/15317/Extenuating_Circumstances_Policy_and_Procedures.pdf

Plagiarism

All of your work must be of your own words. You must use references for your sources, however you acquire them. Where you wish to use quotations, these must be a very minor part of your overall work.

To copy another person's work is viewed as plagiarism and is not allowed. Any issues of plagiarism and any form of academic dishonesty are treated very seriously. All your work must be your own and other sources must be identified as being theirs, not yours. The copying of another persons' work could result in a penalty being invoked.

Further information on plagiarism policy can be found here:

Plagiarism: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/regulations/plagiarism

Examination Offences: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/examination-offences

Turnitin (http://www.turnitinuk.com/) is an Internet-based 'originality checking tool' which allows documents to be compared with content on the Internet, in journals and in an archive of previously submitted works. It can help to detect unintentional or deliberate plagiarism.

It is a formative tool that makes it easy for students to review their citations and referencing as an aid to learning good academic practice. Turnitin produces an 'originality report' to help guide you. To learn more about Turnitin go to:
https://guides.turnitin.com/01_Manuals_and_Guides/Student/Student_User_Manual