COMP3006
FULL-STACK DEVELOPMENT

20 CREDIT MODULE

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

MODULE LEADER: Dr Mark Dixon

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. 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 describes the assessment of *COMP3006 Full-Stack Development*. The module is assessed via **100% coursework**, across two elements: *30% Set Exercises* and *70% Project Report*.

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
Individual Set Exercises (30%)	**Tue 11 Nov 2025 16:00 GMT**	After submission
Individual Project Report (70%)	**Fri 9 Jan 2026 15:00 GMT**	Fri 6th Feb 2026

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, you are not expected to start this immediately (as you will often not have sufficient understanding of the topic). The module leader will provide guidance in this respect.

This document was last updated: Friday, 12 December 2025

Assessment 1: Full Stack Set Exercises (30%)

Task:

This coursework is an **individual** assessment. It will be administered via the DLE quiz facility (where you will need to enter short free-text responses to the exercises presented) and as such will be run entirely online. Students will be given a 3 hour period in which to submit. They will be allowed up to 3 attempts within that period and the final mark will be an average of all attempts taken. The mark should be available just after submission.

Note: This is separate from the lab exercises that are handed out each week (which are not assessed).

Acceptable level of generative AI tool use in this assessment element

Assisted Work	You are permitted to use generative AI tools in an assistive role.	☐

Be careful when using generative AI tools – they will often give suggestions that are:
- inappropriate (such as different version of the software, or assume requirements beyond the module needs)
- complex (include concepts that you are unfamiliar with, that will make further code modification difficult for you)

Assessment 2: Full-Stack Project Report (70%)

Task:

This is an **individual** assignment:

- collaboration between **students** is prohibited
- staff will provide advice during teaching sessions (each year one or two students fail because they have not spoken to staff and mis-understood this brief)
- you are advised to make use of staff time during the teaching sessions (access to staff outside these sessions is very limited)
- the **final submission** must be submitted to the DLE by the specified submission dates.

As with any software development project, the details of this brief may change slightly over time. You must check for changes and announcements (both email and in teaching sessions) regularly.

Within the guidelines given below, you must define the specific functionality that you will deliver and the process you will follow. Your project must satisfy the following requirements:

- Functional Requirements, must be selected from ***one*** of the following topics:
  - School / University / College Learning Platform
  - Project Management System
  - eCommerce System

Full-Stack Development

- Hotel Room Booking System
- Library System
- Cinema/Theatre Booking System
- GP / Hospital / Dental Appointment Booking System
- Gym/Swimming Pool Workout Record System
- Restaurant / Takeaway Ordering System
- Travel (Bus / Train / Flight) Booking System
- Estate Agent Record System
- Event Management System
- Vehicle Rental/Repair Management System
- Content Management System
- Environmental Monitoring System
- Blog System
- Social Media System
- Sports/Social Club Management System
- Court Appearance Management System
- Emergency Services (Police/Fire/Ambulance/Coast Guard) Management System

- Technical Requirements:
  - You must produce a working system, comprising:
    - must include at least 1 frontend client constructed using dynamic web technologies (***HTML***, ***CSS*** & ***JavaScript*** - client-side code must be JavaScript *(or TypeScript)*, and should use **React** or **Angular**)
    - must include at least 1 backend **Node.js** server (server-side JavaScript code)
    - can include more *optional* Node.js backend servers, and
    - can include an *optional* Python (Flask) or Java (SpringBoot) backend server
    - must include at least 1 database (either **MongoDB** or PouchDB) to store information and pass it to the client through web-service API(s)
    - other types of database are not permitted
    - no other server / language is permitted
  - ***WebSockets*** must be used to give the appearance of communication between multiple clients
  - The system must be ***interactive***, i.e., the user should be able to affect its behaviour by interacting with it using a suitable input device (such as keyboard and mouse).
  - The system must be capable of running on multiple computers (containers), i.e., it must be ***distributed*** (stand-alone applications and static web-sites are not permitted)
  - Include appropriate security considerations (such as registration and login functionality)
  - Include CRUD functionality for at least **2 entities** (but 3 is probably required for sufficient functionality).

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 should include:

- component architecture
- code repository,
- testing
    - unit tests,
    - integration tests,
    - system tests,
    - user acceptance testing (UAT),
    - static code analysis metrics
- continuous integration/deployment pipeline,

Acceptable level of generative AI tool use in this assessment element

Assisted Work	You are permitted to use generative AI tools in an assistive role.	☐

Be careful when using generative AI tools – they will often give suggestions that are:
- inappropriate (such as different version of the software, or assume requirements beyond the module needs)
- complex (include concepts that you are unfamiliar with, that will make further code modification difficult for you)

COURSEWORK DELIVERABLES

There are three deliverables for this assignment (which must be submitted to the DLE by the deadline, after which no alterations or additions are permitted).

D1 – Report

A single **report file (in PDF format)** containing a written report describing your project. The report must include (on the first page):

- A link to your code on GitHub
  (no other source will be marked, you must ensure it can be accessed).
- A link to your video on YouTube
  (no other source will be marked, you must ensure it can be accessed).

The report must be a document of no more than 2,000 words. Use screen shots and sketches to illustrate the functionality and UML diagrams (or appropriate alternatives) to illustrate the software architecture / component design.
The report should explain:

- Requirements (ca. 400 words with additional documents in appendices)
    - Who are the application's target users (such as staff and/or customers)?
    - What benefits will the system give to these users (compared with existing systems)?

    - What functionality was planned (for each user type) and prioritised?
- Design (ca. 500 words with UML diagrams)
    - What is the system architecture (is the architecture monolithic or micro-service based – describe client, server & database components and how do they interact, supported by a component diagram)?
    - What data and code structures are used (such as class diagrams)?
    - What design practices (such as SOLID, DRY or MVC) were used (describing actual examples used)?
    - Why are these structures appropriate?
- Testing (ca. 400 words)
    - What testing (System, Integration, and Unit) was applied appropriately (and documented)?
        - Which of these used manual testing (a test plan would be beneficial)?
        - Which of these used automated testing?
          A selection of actual test code should be presented, described in detail and related to the concepts (theory) covered in lectures. Choose a small number of tests that show the depth and breadth of your testing (do not include every test).
    - What usability testing was performed?
        - how many users participated?
        - what protocol (process) was followed?
        - what were the results?
        - was the system modified?
        - did the work follow University ethical requirements?
- DevOps pipeline (ca. 400 words)
    - Describe the continuous integration pipeline, how it was used, and the development environment (including your actual pipeline code, described in detail and related to the concepts covered in lectures).
- Evaluation (ca. 300 words)
    - What worked/didn't work well?
        - What functionality was delivered (compared with what was planned)?
        - How useful were the techniques used (such as unit testing)?
        - How useful were the technologies used (such as Node.js)?
    - What lessons should be taken from this project into future projects.

Marks are awarded for clear detailed evidence (in the report and video) of what was actually undertaken. This is not an essay or a pure coding exercise. Submissions that have reports and videos that do not show clear detailed evidence (even if the code works well) are likely to receive a fail grade. It is important to describe in detail what was actually applied in the project and how this relates to the theory. For example, a section in the report that gives a general overview of the theory (for example unit testing) without explicitly showing details of what was implemented (a few sample unit test code fragments with explanations) will score poorly.

Full-Stack Development

Check your submitted files are correct by downloading them again and checking that they work. **You should 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.

D2 – Source Code

The code you have written (both the software and the tests). This must use the GitHub Classroom repository created for you (which will be described in class) and a link included in your report (D1).

This should contain all of the code you have written yourself and should (including copies of the folders and files needed to run your application).
Do not include cache folders (such as ***node_modules***) as these are very large and will be generated when the stack is deployed/executed.
Use your real names for your GitHub account, not nick names.
Failure to include a working link in your report is likely to lead to a substantially reduced (possibly failing) mark.

D3 – Video

A video in which **you** present a demonstration of:

- the main functionality working
- the supporting processes (tests running and CI/CD pipeline running)

The video must be no longer than **4 minutes** (only the first 4 minutes will be marked).
The video must show the screen and be narrated by your voice (computer generated speech must not be used).
Only 1 video is permitted (if multiple videos are included only the first will be marked).
A link must be included in your report (D1) and the video must be accessible for 2 months after submission (if your video is deleted by YouTube for violating YouTube policies this will seriously affect the mark given).
The video must be uploaded as an **unlisted YouTube video**.
Failure to include a working link in your report is likely to lead to a substantially reduced (possibly failing) mark.

Frequently Asked Questions (FAQ)

Q) Can we use Angular?
Yes
Q) Can we use React?
Yes
Q) Can we use Next.js?
Yes, but there is limited support from staff available, and if it causes issues consider removing it.
Q) Can we use CSS frameworks (such as Bootstrap, Tailwind CSS, and Material UI)?
Yes, but there are no marks for aesthetics (so be careful to prioritise carefully)
Q) Can we use TypeScript?
Yes
Q) Can we use Jest?
Yes
Q) Can we use JSON web tokens?
Yes
Q) Do we have to use TypeScript?
No (you can choose to use either classic JavaScript or TypeScript)
Q) Can we use Firebase?
No.
Q) Can we use .Net?
No
Q) Can we use C#?
No
Q) Can we use PhP?
No
Q) How much functionality should I include?
Your choice of functionality and rationale is part of the assessment. I would say try not to select functionality that just duplicates (i.e. it makes no sense to implement 20 entities that essentially have the same structure, where the first demonstrates some concepts / technology and the others add nothing conceptually or technically), but it may be beneficial to have 2 of something over 1. For example if I wanted to demonstration inheritance, I would have an Animal class and then at least 2 inherited classes (such as Dog and Snake). If I only had 1 inherited class it would be difficult to see the power / benefit of inheritance. However, I would not have 20 inherited classes (2 or 3 would be sufficient).

Assessment Criteria:

Submissions will be assessed according to the rubric found in Table 2. The mark for this piece of coursework will be based on an aggregation of the marks for each category. Marks will be awarded based mainly on the evidence in the **video** and **report** (the code will be checked to ensure it matches what is described). No marks are awarded solely for effort.

Category	Fail (< 40%)	>= 40%	>= 50%	>= 60%	>= 70%
Analysis (10%)	Insufficient evidence of analysis. Functional requirements are not described in sufficient detail. Unclear who the target users are.	Evidence of some vague functional requirements based on a basic problem outline.	Reasonably detailed functional requirements but based on weak analysis.	Functional requirements are well specified with some evidence of strong analysis.	Functional requirements are extensive and based on clearly evidenced strong analysis.
Design (20%)	Little or no evidence of design work being undertaken or understood. The architecture and/or major components are unclear.	Some evidence of architecture and design work but insufficient detail. Little or no evidence to show the structure and operation of the system.	Some of the system's design is evidenced in detail but other areas are unclear. Some diagrams are included.	Design is mostly complete, further detail would enhance some areas. Most functionality and design specified with diagrams.	Strong evidence of system design and comprehensively described with the use of appropriate diagrams.
Testing (20%)	Little or no evidence of testing of any kind being conducted or understood.	Evidence that some testing (such as unit or integration or system or usability) has been completed and understood, but lacks detail and breadth.	Evidence that an adequate level of testing (such as unit or integration or system or usability) has been undertaken and understood with adequate depth and breadth.	Evidence that reasonable testing (unit or integration or system or usability) has been undertaken and understood. Clear that most of the system has been covered.	Evidence of comprehensive testing (unit, integration, system, and usability) with both depth and breadth (such as participant numbers, results and resultant modifications).
CI/CD (10%)	No or insufficient evidence of version control or an attempt at an automated CI/CD pipeline.	Evidence of some use of version control, but insufficient evidence of CI/CD pipeline.	Evidence of good use of version control but weak of absent CI/CD pipeline.	Clear evidence of effective version control use, with indication of attempted TDD or CI/CD pipeline.	Evidence that version control was used well and working CI/CD pipeline.
Evaluation (10%)	Little or no evidence of evaluation.	Some evaluation, but lacking both breadth and detail.	Some detailed evaluation of limited range of aspects of project, but other areas weak.	Good evaluation of functionality, techniques & technology (with broad coverage and detail).	Comprehensive evaluation of functionality, techniques & technology (detailed & broad).
Software (30%)	No or inadequate evidence that system has been constructed, functions sufficiently and has been understood. No evidence of security considerations (such as login). No use of WebSockets.	Evidence is lacking or shows that large aspects of the functionality do not operate (such as major runtime errors or large numbers of minor bugs or WebSockets and/or a database have been attempted but incomplete).	Evidence of some requirements being implemented but major omissions. WebSockets and database used but one of them weakly. Limited functionality.	Evidence that sufficient functionality has been implemented, functions as expected and has been understood. Some runtime errors. WebSockets and a database are used, both work correctly. There are some minor bugs that do not impede the main functionality.	Evidence clearly shows implementation of extensive working functionality. Very few apparent bugs. The approach taken to implementing the database and WebSockets is nearing commercial standards.

Table 2: Feedback Template (marks based mainly on video and report evidence)

Acceptable levels of AI use:

The table below provides the acceptable use categories for GenAI. Each assessment element may allow different uses. Please check the brief for each element carefully to see what uses are allowed.

Solo Work	**S1 - Generative AI tools have not been used for this assessment.**
Assisted Work	**A1 – Idea Generation and Problem Exploration** Used to generate project ideas, explore different approaches to solving a problem, or suggest features for software or systems. Students must critically assess AI-generated suggestions and ensure their own intellectual contributions are central.
	A2 - Planning & Structuring Projects AI may help outline the structure of reports, documentation and projects. The final structure and implementation must be the student's own work.
	A3 – Code Architecture AI tools maybe used to help outline code architecture (e.g. suggesting class hierarchies or module breakdowns). The final code structure must be the student's own work.
	A4 – Research Assistance Used to locate and summarise relevant articles, academic papers, technical documentation, or online resources (e.g. Stack Overflow, GitHub discussions. The interpretation and integration of research into the assignment remain the student's responsibility.
	A5 - Language Refinement Used to check grammar, refine language, improve sentence structure in documentation not code. AI should be used only to provide suggestions for improvement. Students must ensure that the documentation accurately reflects the code and is technically correct.
	A6 – Code Review AI tools can be used to check comments within the code and to suggest improvements to code readability, structure or syntax. AI should be used only to provide suggestions for improvement. Students must ensure that the code accurately reflects their knowledge and is technically correct.
	A7 - Code Generation for Learning Purposes Used to generate example code snippets to understand syntax, explore alternative implementations, or learn new programming paradigms. Students must not submit AI-generated code as their own and must be able to explain how it works.
	A8 - Technical Guidance & Debugging Support AI tools can be used to explain algorithms, programming concepts, or debugging strategies. Students may also help interpret error messages or suggest possible fixes. However, students must write, test, and debug their own code independently and understand all solutions submitted.
	A9 - Testing and Validation Support AI may assist in generating test cases, validating outputs, or suggesting edge cases for software testing. Students are responsible for designing comprehensive test plans and interpreting test results.
	A10 - Data Analysis and Visualization Guidance AI tools can help suggest ways to analyse datasets or visualize results (e.g. recommending chart types or statistical methods). Students must perform the analysis themselves and understand the implications of the results.
	A11 - Other uses not listed above Please specify:
Partnered Work	**P1 - Generative AI tool usage has been used integrally for this assessment** Students can adopt approaches that are compliant with instructions in the assessment brief. Please Specify:

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/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/extenuating-circumstances

For this module the way the CW1/W1 Set exercises are managed means you will have ample opportunity to overcome any minor setbacks during the time of the module and still be able to deliver your best work.

Plagiarism

All of your work must be of your own. 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/essentialinformation/regulations/plagiarism

Examination Offences: https://www.plymouth.ac.uk/student-life/your-studies/essentialinformation/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://help.turnitin.com/feedback-studio/blackboard/basic/student/student-category.htm