**Faculty of Engineering, Environment and Computing**

# 5001CEM Software Engineering

**CW Assignment Brief 2020/21 (RAB)**

| Module Title<br>**Software Engineering** | Ind/Group<br>**Individual** | Cohort<br>**SEM1** | Module Code<br>**5001CEM** |
|---|---|---|---|
| Coursework Title<br>**CW Software Engineering Project** | | | Hand Out Date<br>**14/09/20** |
| Lecturer<br>**Mark J Tyers** | | | Due Date:<br>**11/12/20** |
| Estimated Time<br>**50 hours** | Coursework Type<br>**Codio, GitHub, Screencast** | | Module Credits<br>**10** |
| Submission Arrangement Online via CUMoodle: **Upload to TurnItIn**<br>File Types: **Link to Repository** and **Screencast**<br>Mark and Feedback Date: **08/01/21**<br>Mark and Feedback Method: **Rubric comments and marks.** | | | |

## Module Learning Outcomes Assessed

ILO 4 Select, evaluate and use tools and techniques to successfully manage a large scale software project, including configuration management and version control.

ILO 5 Use a range of appropriate tools to contribute to the development of a solution to a real-world problem.

ILO 6 Select, evaluate and apply standards, tools and techniques for assuring software quality.

## Task

You have been employed by a firm of software engineers to develop a full-stack solution for one of their clients. The work was started by your predecessor who left to get a more senior job with another company but before they left they had developed a completed registration and login system backed by a full suite of tests. You should use their code as the basis for the work you develop. You **must** develop your website using the **Assignment** base code supplied in the **Codio online IDE**.

The topic you have been assigned is based on the last two digits of your 7/8 digit university ID number, see the **Topic** section for more information.

Detailed descriptions of the topics can be found at *https://github.coventry.ac.uk/agile/projects*, you are expected to focus on the **stage 1 functionality** only, although for the highest grade you must also complete the *stage 2* tasks. Read the tasks carefully!

## Getting Started

Do the following **immediately** (right now) and get in touch with the module leader if you are having any difficulties:

1. Install the Aula app on your smartphone:
    - Ask any questions in the assessment channel, do not message the module leader directly unless the question is of a personal nature.
    - Before posting a question, read the **Glossary of Terms** section below as well as the previous messages in the assessment channel.
2. Use this link (https://codio.co.uk/p/join-course?token=monica-vienna) to create an account and access the technical lab exercises and coursework template. You may need to register using your University email address and this *must not include* the .uni part, for example doej@coventry.ac.uk, **not** ~~doej@uni.coventry.ac.uk~~.
3. You should now be able to access the **SEM 1** template in the **Assessment** Section on Codio. This is where you will be writing and testing your code.
4. Create a private GitHub repository in the 5001CEM organisation:
    - https://github.coventry.ac.uk/orgs/5001CEM-2021OCTJAN
    - If you can't create a repository contact the module leader through Aula.
    - The *repository name* should be your university username (eg doej).
    - You should add a suffix to reflect the semester in which you are taking the assessment (eg. doej-sem1)
    - The description should be the *assignment topic*.
    - The repository must be **private**.
5. Add this remote repository to you project in Codio:
    - Use the Clone or download button and copy the URL to the clipboard.
    - Open the terminal in Codio and use the git remote add origin **xxx** command to add the remote (where **xxx** is the URL).

## Technical Skills

You will need a specific set of technical skills and knowledge in order to be able to complete this assignment. To ensure this, you should complete the set of self-paced labs that can be found on Codio in the **Teaching Materials** section. These cover:

1. The Hypertext Markup Language
2. Cascading Stylesheets
3. The JavaScript language.
4. Server-Side JavaScript
5. Client-Side JavaScript.
6. Version Control.

You should aim to complete these, including all the exercises, before starting your assignment.

## Submission

Before submitting your assignment you need to create a number of **testing accounts** to allow the assessor to test the functionality of your software. If you read the topic descriptor carefully you will see that these are listed under the **Testing** heading. Before submitting please ensure that there are accounts registered on your system for each of these. Set the password in all cases to **p455w0rd**.

You are required to submit a short 3 min screencast using the Google Chrome Screencastify plugin. After recording the screencast, export as an MP4 file and upload this to Moodle using the Add media button.

The screencast must cover the following points *in this order*:

1. A demonstration of the stage 1 functionality you have achieved, clearly showing which of the tasks are working.
2. A demonstration of the stage 2 functionality if this was attempted.

You are also required to host your private repository in the 5001CEM-2021 organisation.

The assessor will attempt to run the linter and unit tests and this will be used to grade you on these categories so make sure these commands are working before submitting your work.

## Important Links

**Moodle** (assignment submission)
https://cumoodle.coventry.ac.uk/course/view.php?id=78422
**Codio** (technical labs and assignment code)
https://codio.co.uk/p/join-course?token=monica-vienna
**Aula** (communication)
https://coventry.aula.education/?#/dashboard/nDyGIQmECE
**Zoom** (synchronous labs)
https://coventry-ac-uk.zoom.us
**GitHub** (code versioning)
https://github.coventry.ac.uk/5001CEM-2021OCTJAN

## Restrictions

It is very important that you read and fully understand the restrictions listed below. If you ignore any of these your submission will be considered invalid:

1. Your work must be your own, do not share your work with any other students or copy work from someone else.
2. You must develop the work using the Codio IDE and not use any other software.
3. Every day you work on your project you must make *at least* one git commit of the changes made.
4. The contents of the README.md file must be replaced with details of the features you completed lifted from the assignment topic and the usernames and passwords to be used by the person marking your work to allow them to log in and check the features.
5. You have been provided with a template, you must retain this and *build on it*.
6. You may use any of the modules from the tutorials plus the following: crypto-js, image-watermark, Koa-compose, music-metadata, nodemailer, node-id3, node-lame, pdf, sharp

You should refer to the **5001CEM SEM1 CW Rubric** (at the end of this assignment brief) and ensure that your submission meets its requirements remembering that the grading rubric is cumulative, you need to meet the lower grade descriptors even when attempting the higher grades.

There is a detailed **glossary of terms** attached to this assignment brief. You should read this carefully and ensure you are familiar with the requirements outlined in the grading rubric.

## Running the Tests

As part of the assessment the person marking your work will be running live tests on your code inside the Codio IDE. All tests will be run from the directory /workspace and will assume all the required tools are installed and configured correctly.

1. The functionality will be assessed through the screencast and may be tested by logging in using the predefined accounts (these are defined in the topic descriptor) and checking the required features work.
2. The linter will be run using the command **npm run linter** and should detect and use the linter rules provided in the .eslintrc.json file and ignore files defined in the `eslintignore file. Neither of these files should be modified.
3. The unit tests will be run using the command **npm test** and should detect and use the ava rules defined in the package.json file. These rules must not be changed.
4. The version control score is based on the network graph on the GitHub server which is located under the Insights tab.
5. The DevOps criteria will be tested by: clicking on the URL in the README file that points to the live site; studying the git hooks and GitHub action scripts. For 100% you should also provide access to your repository on GitHub.com and provide access to the user: marktyers.

## Topics

You are assigned a topic based on the last two digits of your 7/8 digit student ID number. For example if your ID number were 12345**67**. You use the number **67** to determine your topic which, in this example, would be *Stock Inventory*. https://github.coventry.ac.uk/agile/projects

| | |
|---|---|
| 01-04 | Auction |
| 05-08 | Domestic Repair |
| 09-12 | Bookshop |
| 13-16 | E-Learning |
| 17-20 | File Sharing |
| 21-24 | Forum |
| 25-28 | Frequently-Asked Questions |
| 29-32 | Gallery |
| 33-36 | Game Reviews |
| 37-40 | Gift List |
| 41-44 | Local Community |
| 45-48 | Local News |
| 49-52 | My Music |
| 53-56 | Pledge |
| 57-60 | Restaurant Ordering |
| 61-64 | Restaurant Reviews |
| 65-68 | Stock Inventory |
| 69-72 | Student CVs |
| 73-76 | Survey |
| 77-80 | Theatre |
| 81-84 | Collection Tracker |
| 85-88 | Expenses Tracker |
| 89-92 | Library Loans |
| 93-96 | Mortgage Applications |
| 97-00 | Sandwich Ordering Service |

## Glossary of Terms 1/2

### Version Control

**Regular daily commits** – There is at least one commit for every day you have been working on the project and these commits have been pushed to the GitHub server.

**Simple branching model** – Each feature has been written in a separate branch with each branching directly off the master branch and merging back to the master branch.

**SWW format** – all commits should be multi-line with the first line being a summary, followed by one or more lines explaining why the changes were made and finally one or more lines explaining what was changed.

**Branch names linked to issue tracker** – the names used for the feature branch match the names used in the issue tracker with one branch per issue.

### DevOps

**Git Hooks** – Bash scripts that are automatically run as a result of specified events triggered by Git commands.

**Client-Side Hooks** – Hook scripts that are run on the development workstation.

**Development Workstation** – The computer environment used by the developer. In most cases this will be the online Codio IDE.

**Code Deployment** – The code is automatically deployed to the Heroku server so the user sees the latest version of the system.

**GitHub Server** – The University GitHub server (github.coventry.ac.uk).

**GitHub Actions** – The Continuous Integration service provided by github.coventry.ac.uk

**Deployment Pipeline** – A sequence of automated steps where if all the previous steps pass, the code is automatically pushed to the Heroku server and made available to the clients.

**Branch Health Status** – A visual indicator on the GitHub server of whether all the tests and checks have passed.

**Pre-Receive** – automated tests run on the GitHub server whenever commits are pushed.

**Auto Deployment** – If all the tests pass the software should be automatically deployed to a Heroku server.

### Functionality

**Instructions** – the readme file includes the tasks completed (copied from the assignment topic) as well as the different usernames and passwords to be used when demonstrating the functionality.

**Testing Accounts** – You are required to create accounts to allow the system to be thoroughly tested. The usernames of the accounts are shown in the assignment topic in italics. You are to assign a password of p455w0rd for all these accounts.

**Failed attempt** – you tried to get some of the listed functionality working but failed.

**Some functionality** – you have *completed* one of the three features.

**Most functionality** – of the three features listed you have *completed* two of them.

**All functionality** – you have completed all the functionality at the appropriate stage.

# Glossary of Terms 2/2

## Design

**Poor Website Design** – The website looks terrible with little attempt made to improve its look and feel.
**Inconsistent Style** – The look and feel is different on different pages of the website.
**Poor Layout** – The
**Low Accessibility** – Some pages have an accessibility score (https://www.webaccessibility.com) of under 60%.
**Poor Choice of Data** – The data displayed on the pages does not reflect what would be used if the site were real, for example using xxx@yyy.com for an email address or qwerty for a name.
**Consistently Applied** – All the pages in the website share the same look and feel including all form elements.
**High-Quality Design** – A design that helps the user navigate and interact with the website.
**Professional Design** – The website looks and feels like one developed for a high-end organisation.
**Responsive Design** – A single website design which adapts itself as the screen size changes.
**Adaptive Design** – A series of different layouts applied based on the screen-size and mode of interaction (keyboard/touchscreen).

## Code Quality

**Modules not taught** – Only use modules used in labs or listed in this assignment brief.
**Linter rule changes** – You must not change or override any of the linter rules.
**Logic in modules** – The routes files should contain only the *event handlers*. All *other code logic* must be isolated in one or more modules that are imported into the route file This must.use the ES6 (Harmony) Module syntax style.
**Event handlers** – code that listens out for HTTP requests and user interactions.
**Other Code Logic** – any code that is not directly part of listening for events, for example code that interacts with a database.
**Consistent file naming** – All code should be split into files based on shared logic and every file *name* should reflect the content of the file.
**Descriptive comments** – simple code comments to add context to any parts of the code to help anyone working on your code in future. It should focus on explaining why the code is as it is, not how the code works. Code that is clear does not need commenting.
**Auto docs** – you have added clear *JSDoc* comments to *every module* and *function* including parameters and return values. You have then run the tool to convert this to full html documentation which is correctly structured and viewable in your Codio project.

## Test-Driven Development

**Flawed unit tests** – these are tests where the implementation of the test does not match the name, eg. a test to add a new item without checking the item was added.
**A range of tests** – there are tests for every code module with at least 80% code coverage.
**Comprehensive tests** – every function, line and branch in the code is being tested.
**Useful unit tests** – these are tests that check for important functionality in the system.
**AAA tests** – all tests are split into clearly labelled sections: Arrange, Act and Assert.
**FIRST tests** – all tests comply with FIRST principles:
**Isolated tests** – each test is independent meaning tests can be run individually and in any order.
**Bad data and edge cases** – Tests that make sure the code can handle data that is not considered valid, for example if a string value is passed instead of a number.
**Edge cases** – values that fall outside the accepted range such as trying to access an array index of -1 or passing a negative number where the function required a positive one.
**Continuous integration** – GitHub actions configured to be triggered each time commits are pushed. Actions should run a suitable range of tests, flagging issues and blocking actions if errors are found.

## Notes

1. You are expected to use the APA 7th Edition referencing style. For support and advice students can contact the Centre for Academic Writing (CAW).
2. Please notify your registry course support team and module leader for disability support.
3. Any student requiring an extension or deferral should follow the university process as outlined here.
4. The University cannot take responsibility for any coursework lost or corrupted on disks, laptops or personal computers. Students should therefore regularly back-up any work and are advised to save it on the University system.
5. If there are technical or performance issues that prevent students submitting coursework through the online coursework submission system on the day of a coursework deadline, an appropriate extension to the coursework submission deadline will be agreed. This extension will normally be 24 hours or the next working day if the deadline falls on a Friday or over the weekend period. This will be communicated via email and as a CUMoodle announcement.
6. Assignments must be uploaded in the file format specified in the assignment brief. Submissions that are not in this format will receive a zero mark.
7. Assignments that are more than 10% over the word limit will result in a deduction of 10% of the mark i.e. a mark of 60% will lead to a reduction of 6% to 54%. The word limit includes quotations, but excludes the bibliography, reference list and tables.
8. You are warned to check your work and the filename before uploading to the link. You have only one chance to submit which will be checked by Turnitin on your Moodle Web.
9. Collusion between students (where sections of your work are similar to the work submitted by other students in this or previous module cohorts) is taken extremely seriously and will be reported to the Academic Conduct Office. This applies to both courseworks and exam answers.
10. A marked difference between your writing style, knowledge and skill level demonstrated in class discussion, any test conditions and that demonstrated in a coursework assignment may result in you having to undertake a Viva Voce in order to prove the coursework assignment is entirely your own work.
11. If you make use of the services of a proof reader in your work you must keep your original version and make it available as a demonstration of your written efforts.
12. You must not submit work for assessment that you have already submitted (partially or in full), either for your current course or for another qualification of this university, unless this is specifically provided for in your assignment brief or specific course or module information. Where earlier work by you is citable, ie. it has already been published/submitted, you must reference it clearly. Identical pieces of work submitted concurrently will also be considered to be self- plagiarism.