Southampton Solent University
Assessment Brief

Assessment Details

Unit Title:	Web Technologies
Unit Code:	COM710
Unit Leader:	Darren Cunningham
Level:	7
Assessment Title:	Website and Documentation
Assessment Number:	AE1
Assessment Type:	Website and Supporting Documentation
Restrictions on Time/Word Count:	No more than 2000 words
Consequence of not meeting time limit:	It is essential that assignments keep within the time/word count limit stated above. Any work beyond the maximum time/word length permitted will be disregarded and not accounted for in the final grade.
Individual/Group:	Individual
Assessment Weighting:	100%
Issue Date:	10/02/2023
Hand In Date:	8/03/2023
Planned Feedback Date:	20 working days after submission
Mode of Submission:	On-line via 'Solent Online Learning'
Anonymous Marking	Yes

Scenario

In this assessment you will be developing a website that provides information on the "Solent Hackathon" which is a fictitious hackathon event in which teams of coders will compete to make the best application. The theme for the hackathon is "the environment" – so teams will be looking to develop apps to help with environmental issues. This website will have both a public facing element that provides information to the public and an API that allows the website to provide and receive information to/from other sources.

This website will be deployed into a node environment and therefore must be written in and operate using Node.js

Alongside this document, you have also been provided with an index.js file. This should be the index.js file for your node project. You should follow the comment instructions in the index.js file – do not change any of the code I have already provided.

Attention!

To speed up the marking process you should:

- Please use the index.js file provided!
- Use your student ID as the name for your submitted files. For example, if your student ID is 123456, then you should submit a .zip file called 123456.zip and either a word document or pdf called 123456.pdf or 123456.docx respectively.
- You should use the routes listed in the next section; you should also ensure they are spelt correctly in your code
- You should only install the node packages as listed in the technologies section of this document. You should not install any others.
- Please use port number 5000
- When the marker downloads your submission, they will do the following to make it run – you should ensure that this will work
    - Unzip the zip file
    - Open the now unzipped project folder in VSCode
    - In the terminal run the command: npm install
    - In the terminal run: node index.js
    - The marker should now be able to view the website by going to localhost:5000 and get api results back by going to localhost:5000/api/teams

By ensuring the above, the marking process is likely to be quicker. Delays can occur if enough submissions do not follow the above points.

Requirements

Database

The database that the system interacts with must be **SQLite.** Said database will need to store the following about teams who will be taking part at the event (**No further information should be provided**):

- ID – As with most database tables, all records should have a unique ID
- The name of the team
- Project name – name of the project they will be entering
- About – A brief description of their project

Do not get real data/information about real people. For the sake of this assessment, just make up names etc and use filler text where needed.

The database should be submitted with the assessment – do not code scripts that create your database when the website first runs.

API

As you are creating the system in Node.js, you will be setting up routes that will allow the website to be accessed. There will be two different sets of routes, the first will allow access to the API and the second will all access to the webpages which make up the website.

I have already provided these routes in the accompanying file called **index.js.** Please use this file!

The API will facilitate the adding, removing, updating and retrieval of team information.
Responses should be given in JSON.
The following routes **MUST** be provided:

- Retrieve all team in JSON format:

  GET /api/team

- Retrieve the information about a specific team – the response should provide a single result in JSON format

  GET /api/team/:id

- Create a new team – A JSON response indicating success or error should be returned

  POST /api/team

- Update the details of a team – A JSON response containing the new details of the team, or an error should be returned

PUT /api/team/:id

- Remove a team by their ID – A JSON response indicating success or error should be returned.

  DELETE /api/team/:id

The Website

The website itself will be made up of the following pages and should be suitably displayed using HTML and CSS. You should also include JavaScript functionality as well.
As with the API, the following routes should be set up in your **index.js** file all these routes should render HTML from an EJS template.

You should have:

- GET /

  Home Page: A welcome page for the users that will just look nice and promote/provide general information about the hackathon

- GET /teams

  Teams Page: This page should list all the teams from the database. This should be formatted appropriately. This page only needs to provide the name of each team and their project name.
  You should also be able to select an individual team and be taken to a page about them (below)
  Additionally, this page should give the user options to add and delete teams

  You should not provide pictures

- GET /team/:id

  Team page: Display a page that provides all details about the team with the specified ID. The user should also be given the option to update the team's information.

  You should not provide images

(Please note, further requirements are added in the grading criteria)

Technologies

You must only use the following to develop this website:

- HTML
- CSS
- JavaScript
- Node JS
    - Express
    - SQLite
    - EJS

Any technologies that were not covered in this unit will not be counted and any features developed using them will be overlooked.
(Bootstrap and JQuery, whilst mentioned in the unit, are not to be used)

Please note, more requirements can be found in the grading criteria (Later in this document)

Supporting Documentation
In addition to the website, you are also expected to provide a word document or PDF.

This will be a brief and concise document that will be written after you have completed your website.
It should detail the following.

- Proof that your website conforms to W3C standards (use their online validator)
- Legal and ethical considerations
    - What makes the website accessible? (i.e. alt text on images)
    - What legal considerations have been made? (sourcing images from websites like pexels for example)
- (Where applicable) – What security considerations have been made.
- You should also document any use of version control (i.e., a screenshot of any commits made to git)
    - If you have been pushing your code up to GitHub, you **must** declare it here with screenshots clearly showing your GitHub username

This should be to the point! I do not expect definitions of any concepts or lengthy explanations.

What you should submit

You will need to upload **two** files to Solent Online learning:

- A **.zip** file containing your web project.
- A word document or PDF containing the supporting documentation.
- The .zip file and the word/pdf document should be uploaded as separate files.

Reminder!

When the marker downloads your submission, they will do the following to make it run – you should ensure that this will work

- Unzip the zip file
- Open the now unzipped project folder in VSCode
- In the terminal run the command: npm install
- In the terminal run: node index.js
- The marker should now be able to view the website by going to localhost:5000 and get api results back by going to localhost:5000/api/team

By ensuring the above, the marking process is likely to be quicker. Delays can occur if enough submissions do not follow the above points.

Grading Criteria

To Achieve an F Grade

Any of the following will result in an F (or S) grade being awarded:

- The submission does not run in a node environment, or the features are not accessible using the routes outlined above
- The website uses unapproved technologies that are not in line with those listed on the previous page.
- The website has little to no layout
- Very poor practice has been demonstrated.
    - i.e. using deprecated HTML tags that style the page rather than CSS
- The criteria for a D grade (below) have not been met
- The submitted website is very similar to a class task (regarding HTML and CSS) – has perhaps been slightly restyled or just given different content.
- No client-side JavaScript has been added to the website at all
- No supporting documentation has been submitted
- A .zip file containing the web application has not been submitted.
- A package.json file has not been submitted
- The supporting documentation is off topic or does not follow the points requested in the 'supporting documentation' section of this brief.

- The supporting documentation must be submitted outside of the zip file and must generate a Turnitin report.

- The use of bad, rude or offensive language/imagery. This includes its usage as 'filler text.'

- Evidence of plagiarism or collusion

To Achieve a D Grade

To achieve a D grade, the website should:

- Have a basic API with database driven functionality – allowing the adding, removing, and viewing of teams

- Have the webpages as listed in the "website" section above. These should be accessible via the specified routes. Information about teams should be retrieved from the database. The website should also allow the adding or removing of teams

- The website should be fully navigable and should make use of node.js routes

- Have a basic layout

- Include some basic client-side JavaScript functionality

- Mostly conform to W3C standards

- Run in a Node.js environment using the specified routes (above).

- An up-to-date package.json file has been provided.

To Achieve a C Grade

To achieve a C grade, (in addition to the D criteria) the website should:

- Provide update functionality for teams in both the API and the website.

- Add a table to the database that stores information on the individual members that make up a team. It should, at the very least store their name and job. This table should have a relationship with the teams table.
  The 'GET team/:id' page should then list the members for the selected team, making use of the relationship in the database

- Fully conform to W3C Standards (Passes W3C validation)

- Be fully responsive to different devices.

- Any HTML forms should have some simple validation – performed by a JavaScript

- General good practice should be observed (i.e. an organised folder structure with images, styles and scripts separated appropriately)

The above should be documented in the supporting evidence, along with all points mentioned on page 4 of this document (supporting documentation section).

To Achieve a B Grade

To achieve a B grade, (in addition to the C criteria) the website should:

- An additional page should be created for an activity that you should come up with. This 'activity' will be interactive and involve JavaScript.
    - It should be related to coding and aim to get younger audiences interested in a future career in software development
    - This should be original and not taken from the internet.
    - Can be any kind of activity but I would expect more than a simple quiz.

To Achieve an A Grade

To achieve an A grade, the website should:

- Meet the D, C and B criteria (listed above) with no exceptions or leniency

- Look professional – would be publishable for its intended purpose.

- AJAX should be used (appropriately) across the website to connect to the API and perform add, remove and update functionality. The website should then be updated appropriately

- In addition, a search feature should be added to the "GET /" page. This will use AJAX to connect to the API, retrieve the relevant teams (as JSON) and then display them on the home page (as HTML).
  Relevant results should appear as the user types their search in.
  This should be done without reloading the page.
  If the user clicks one of the search results, they should be taken to that specific teams page.

Learning Outcomes

This assessment will enable students to demonstrate in full or in part the learning outcomes identified in the unit descriptors.

Late Submissions

Students are reminded that:

i. If this assessment is submitted late i.e. within 5 working days of the submission deadline, the mark will be capped at 40% if a pass mark is achieved;
ii. If this assessment is submitted later than 5 working days after the submission deadline, the work will be regarded as a non-submission and will be awarded a zero;
iii. If this assessment is being submitted as a referred piece of work then it must be submitted by the deadline date; any Refer assessment submitted late will be regarded as a non-submission and will be awarded a zero.

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-assessment-principles-and-regulations.pdf?t=1534423842941

Extenuating Circumstances

The University's Extenuating Circumstances procedure is in place if there are genuine circumstances that may prevent a student submitting an assessment. If students are not 'fit to study', they can either request an extension to the submission deadline of 5 working days or they can request to submit the assessment at the next opportunity (Defer). In both instances students must submit an EC application with relevant evidence. If accepted by the EC Panel there will be no academic penalty for late submission or non-submission dependent on what is requested. Students are reminded that EC covers only short term issues (20 working days) and that if they experience longer term matters that impact on learning then they must contact the Student Hub for advice.

A summary of guidance notes for students is given below:

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2p-extenuating-circumstances.pdf?t=1534423896787

Academic Misconduct

Any submission must be students' own work and, where facts or ideas have been used from other sources, these sources must be appropriately referenced. The University's Academic Handbook includes the definitions of all practices that will be deemed to constitute academic misconduct. Students should check this link before submitting their work.

Procedures relating to student academic misconduct are given below:

http://portal.solent.ac.uk/support/official-documents/information-for-students/complaints-conduct/student-academic-misconduct.aspx

Ethics Policy

The work being carried out by students must be in compliance with the Ethics Policy. Where there is an ethical issue, as specified within the Ethics Policy, then students will need an ethics release or an ethical approval prior to the start of the project.

The Ethics Policy is contained within Section 2S of the Academic Handbook:

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2s-university-ethics-policy.pdf

Grade marking

The University uses a letter grade scale for the marking of assessments. Unless students have been specifically informed otherwise their marked assignment will be awarded a letter grade. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-annex-2-assessment-regulations-grade-marking-scale.pdf?t=1534424273208

Guidance for online submission through Solent Online Learning (SOL)

http://learn.solent.ac.uk/onlinesubmission