Summative Assessment Programming Assignment 2
Server-Side Programming
Frequently Asked Questions

Client-side functionality

1. Q. Do we need authentication, like username and password?
   Not recommended: Doing authentication properly may be quite hard. If you want to do authentication, maybe just include a function to authenticate, which maybe pops up an alert saying 'this page is private' or something like that.

Client-side quality

1. Q. Can I use *react* to build the web-site?
   No: It maybe difficult to assess the quality of your solution.

2. Q. Can I use *jQuery* to build the web-site?
   No: As discussed in class, JQuery is on the way out. It used to be essential to make things like event handling and AJAX work across browsers. However, now modern browsers offer the Fetch API which handles this. The small issue is with Internet Explorer support. However, IE has a minor share now and *polyfills* are available to make it look like it has Fetch. Many systems still do use jQuery (i.e. bootstrap) but are looking to remove is as a requirement (e.g. Bootstrap 5) so I wouldn't recommend it for a new project. It would also be difficult to assess.

3. Q. Will I get marked on documentation of the client-side code?
   No

4. Q. What do you mean by 'gracefully handles server disconnection'?
   This means that your client-side code should do something sensible if the connection to the server goes down (as it might do if it were connected via the Internet). It should display an informative message to the user, and maybe try again later. You can test this by stopping the server and trying to interact with it through the client. Once the server is started up again the client should be able to carry on as before.

Server-side functionality

1. Q. Are we allowed to include additional modules via NPM that provides additional functionality?
   Yes. This is a good idea. This makes the code easier to read, more robust and more maintainable. Some frameworks (e.g. React) installed via npm essentially use a different language, so are difficult to read for non-experts. These are best avoided.

2. Q. How are we advised to save the data, i.e using database?
   In a real system, a database would be the best way to store data. However, databases are not part of this course. The recommendation would be to write functions for saving the state, which just write to file a JSON string representing the state. This will not work with multiple users but would be enough for simple testing.

3. Q. What do you mean by an 'entity type'?
   Refer to the lecture on REST. An entity or entity type is what you would have studied in your Databases module. In Objected Oriented programming, the counterpart would be what you would refer as a class. This is a distinct 'type' or 'kind' of thing. As an example (courtesy Steven Bradley):
   If your app is about poets and their poems then you would have two entitity types: 'poet' and 'poem'. For each poet you might want to store their iID name(s), date of birth, url of image. For each poem, you might want to store an ID, the title, date of writing, and the text. The relationships between the entities would be of the form poet1 authored poem2. Exactly how you store the information about the relationship is up to you: you could store the author id with the poem, or store a list of poems ids with the author, or have a separate store relating the two. In either case, when you get the details of an entity you should include everything, including the relationships. It may be a good idea to devote some initial effort on your entities and relationships e.g. using E-R diagrams.

4. Q. What kind of relationship should the entities have with each other?
   You should have (at least two) entities with **One-to-Many** or **Many-to-Many** relationships. You may have additional entities which may have One-to-One relationships but these are not likely to be as useful.

Server-side quality

1. Q. Is the testing solely on the server.js file or on the other .js files the website uses also?
   You only need to test your server side javascript.

2. Q. Are we allowed to use an API documentation generator like Postman or must we create our own documentation from scratch?
   Yes, if you have a good tool for API generation such as https://learning.postman.com/docs/postman/api-documentation/documenting-your-api/ that would be fine, and a good idea.

3. Q. Will adding comments to the API be sufficient for API documentation, or would it be preferable to use something like postman or 'https://www.npmjs.com/package/node-api-doc-generator' (an npm package)?
   One should not need to have to read the code to see the API documentation. You could write it in HTML by hand, or you could use a tool to do that. The postman tool maybe a good idea, the npm tool looks less maintained.

4. Q. What do we need in the API documentation?
   For the API documentation the ideal is something like the documentation of the Twitter API which lists the methods in the API and then provides the details for each method including details of parameters and response.

5. Q. How should we configure the ESLint file?
   Unless you have a good reason to do otherwise you should use this module.exports = "extends": "standard", "rules": "semi": [2, "always"], "indent": "off" ; But if you are using something else reasonable that is fine, but you must include the relevant .eslintrc. It shoud not require a whole load of work to set up.

Video Presentation

1. Q. What information are we supposed to include in the video?
   Treat it as a sales pitch for the criteria listed under client-side functionality and server-side functionality. You don't need to show every single thing that your site does, just show how it meets the requirements. Things like HTML validation, automated testing and the API

documentation do not need to be covered.

2. Q. What software should we use to record the presentation, and where could we find it?
   The university provides Panopto (which is used for sharing lecture recordings) for recording and simple editing. Here are some instructions for use. Freely available desktop tools include OBS Studio and daVinci Resolve for recording and editing, which are both powerful but have more of a learning curve. MacOS provides *QuickTime player* and *iMovie* for recording and editing. TechRadar have a list of screen recorder software for other alternatives.