Summative Assessment
Programming Assignment 2: Server-Side Programming

Outline

- Assessor: *Amitabh Trehan, amitabh.trehan@durham.ac.uk*
- Handout Date: 22 February 2021
- Hand-in (Submission of code and video): by 14:00 23 April 2021
- Return by 21 May 2021
- Expected workload: 20 days, 3-4 hrs/day = 60 hrs

- Total Marks: 100
- Components marked: Code, Documentation, Video
- Contributes 60% of module marks

Submission

Source code (all zipped in a directory with correct file structure)

- README.txt with execution instructions
- HTML and CSS and any media
- Client and server-side JavaScript
- package.json including test and pretest scripts
- .eslintrc
- jest test cases e.g., app.test.js
- documentation of API
- demonstration video

You should not include `node_modules` in submission

Subject-specific Knowledge

- A knowledge and understanding of good programming practice (for example, reuse, documentation and style)

- Building collections of data within a program and using JavaScript Object Notation (JSON)
- Making programs robust through the use of exceptions and exception handling
- A knowledge and understanding of good programming practice (for example, reuse, documentation and style)

Subject-Specific Skills

- An ability to realise solutions to problems as working JavaScript programs
- An ability to apply reuse by exploiting predefined components
- An ability to use software tools related to programming (programming environments, code management, documentation tools, etc.)

Key Skills

- An ability to communicate technical information
- An ability to recognise and apply the principles of abstraction and modelling

Task summary

1. Construct a **dynamic web site** for an application of your choosing meeting constraints as written below in the Dynamic Web Site section.
2. Use static *HTML* pages loading dynamic *JSON* content from server via *AJAX*
3. Server written in *nodejs* to provide *JSON* through *REST API*

Dynamic web site

- Design a Business/Advertisement/Community/Organisation website which has Objects/entities of at least two types e.g., say, an object of commercial etc interest (books, electronics, pets, services etc…), people/users/buyers/sellers, pictures, places, events, comments …

  The website must have a clear purpose which is evident from/highlighted on the main/single page.

- If you are stuck for ideas, feel free to consult me

Static HTML loading JSON via AJAX

- 'Single page app': page content loaded as JSON via AJAX
- Can have more than one page e.g., for user and admin
- Should provide clean and simple User Experience (UX)
- Should be responsive i.e., work well on desktop and mobile
- Recommend using framework such as Bootstrap, semantic-ui

Message sequence chart

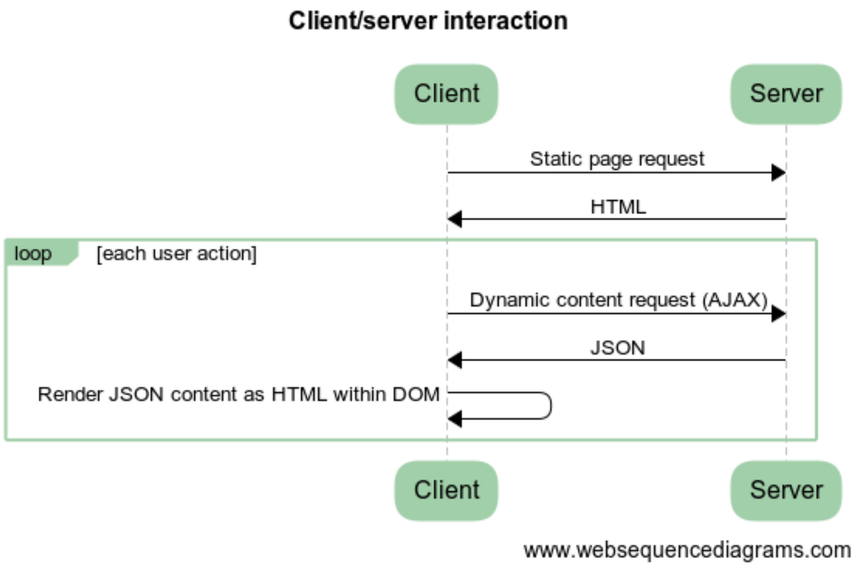
Client/server interaction
Client
Server
Static page request
HTML
loop
[each user action]
Dynamic content request (AJAX)
JSON
Render JSON content as HTML within DOM
Client
Server
www.websequencediagrams.com

Fig: Server providing JSON through a REST API

Entities

Each entity type (e.g., picture) has

- GET method to list/search (returns a list of ids and names)
- GET method for individual details (includes details of related entities)
- POST method to add new entity
- Document your API in the style of the Twitter API
- Response provided as JSON
- Content-type needs to be correct
- HTTP codes should be correct: use 200, 400 or 403 (if using authentication)

Server written in nodejs

- Use npm for management
- Make sure you use --save or --save-dev option with packages you add
- Write jest test cases: run with `npm test`
- Use eslint: run with `npm run pretest`
- Recommend using express

Assessment Criteria

- **Client-side functionality (15%)**
- **Client-side quality (15%)**
- **Server-side functionality (30%)**
- **Server-side quality (30%)**
- **Video presentation (10%)**

	Marks /100
Functionality (Client and Server)	**45**
-Functionality (both Client: 15 marks, and Server: 30 marks) assessed by what is shown/demonstrated in the video (as per criteria below).	
Quality (Client and Server)	**45**
- Assessed by Assessor as per criteria below.	
Video Presentation (2 minutes with penalty for going over time). Please read carefully below.	**10**

Testing Environment

- Mac OSX. Firefox (default).
- Visual Studio Code
- Standard packages covered in class (npm, express etc) (remember you need to include package.json)

Client-side functionality

- User Experience (UX): clean layout and minimal clicks/entry required
- App complexity: entities can be listed and edited
- 'Single page' style: asynchronous updates

Client-side quality criteria

- Standards compliant (HTML5)
- Responsive to different viewport sizes
- Gracefully handles server disconnection
    - useful error messages
    - recommences on server restart

Server-side functionality criteria

- More than one entity type, with relationships
- REST API provides each entity with appropriate GET/POST methods
- Installs with `npm install`
- Starts with `npm start`

Server-side quality criteria

- Appropriate development tools and frameworks (Fetch, Express etc) used
- Successful eslint (run with `npm run pretest`)
- Successful jest tests with good coverage (run with `npm test`)
- Testing includes content-type and HTTP code
- Completeness of API documentation

Video Presentation

- Submit a 2-minute (max) video demonstrating your software
- Include demonstration of how to start the program
- All functionality will be assessed by what is demonstrated in the video
- If it is not demonstrated in the video, you will not get a mark for it
- Quality of video presentation will be marked separately from functionality:
    - Structure: Visual Presentation; Audio explanation
- **You will lose 10% of marks for video presentation (10 marks), for every block of 10 seconds over 2 minutes. That is, if your video is 2 mins 1 second long, you lose 10%, if it's 2 mins 11 seconds long, you lose 20%, and so on.**

How to do the assignment

- Design HTML
- Design web service
- Join with Fetch

- Read the FAQ (Ver 1 below, updated versions (if required) will be posted separately, and class will be informed during lectures).

Frequently Asked Questions (FAQs)

Client-side functionality

Q. Do we need authentication, like username and password?

Not recommended: if you are going to do authentication you should do it properly, and it's very unlikely that you could write something yourself that would be robust. My usual advice would be to use an external authentication service (e.g., through firebase), but that requires have tokens etc embedded into your code. Sharing that with others would most likely be outside the terms and conditions of its use. So if you want to do authentication, just include a function to authenticate, which maybe pops up an alert saying 'this page is private' or something like that.

Client-side quality

Q. Can I use react to build the website?

Not recommended: It may be difficult to assess the quality of your solution.

Q. Can I use jQuery to build the website?

Not recommended: 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 handle 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.

Q. Will I get marked on documentation of the client-side code?

No.

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

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.

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.

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 entity 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.

Server-side quality

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.

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.

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.

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.

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

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.

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.