Summative Assessment Programming Assignment 2
Server-Side Programming

Outline

- Assessor: Amitabh Trehan, amitabh.trehan@durham.ac.uk
- Handout Date: 16 February 2023
- Hand-in (Submission of code and video): by 14:00 16 March 2023
- Return by 04 May 2023
- Expected workload: 20 days, 2.5 hrs/day = 50 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):

1. README.txt with execution instructions
2. HTML and CSS and any media
3. Client and server-side JavaScript
4. package.json including test and pretest scripts
5. the file *.eslintrc*
6. jest test cases e.g., app.test.js
7. documentation of API
8. demonstration video

Note: You should not include the directory *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 page(s) loading dynamic **JSON** content from server via **AJAX**
3. Server must be written in **nodejs** and **Express** to provide JSON through REST API

Dynamic Web Site

- Design a website related to a **Business, a Charity, or/and an Education related venture** (*feel free to use your imagination but justify it in the submission!*)
  - The site must have at least two **entities** e.g. say, people (artists, musicians, scientists, customers, beneficiaries, study participants), business/charity/education related objects, places, events, comments etc etc... At least two entities must have **One-to-Many** or **Many-to-Many** relationship e.g. a Company may sell many products, a product maybe sold by exactly one company (This is a One-to-Many relationship) or maybe sold by many (sometimes maybe just one) company (This is a many-to-many relationship).
  - 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*

The picture in Figure 1 shows the interaction required as a Message Flow chart

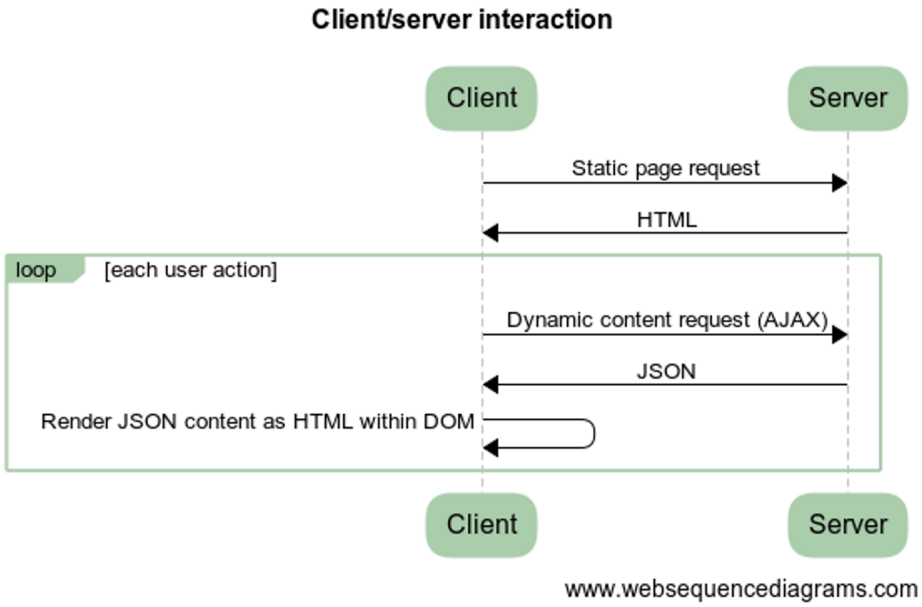

Figure 1: Server providing JSON through a REST API

Entities

Each entity (e.g., picture) must have at least:

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

Server (to be written in nodejs)

Please follow these:

- 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*
- Use **express**. Use any other packages along with as you see fit.

Assessment

Assessment Criteria

The assessment is divided along the following criteria:

1.	Client-side functionality	15%
2.	Client-side quality	15%
3.	Server-side functionality	30%
4.	Server-side quality	30%
5.	Video Presentation	10%

In more detail, the critreria and requirements are as follows:

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

Testing Environment

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

Client-side functionality

Client-side functionality listed as follows:

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

Client-side quality criteria

1. Standards compliant (HTML5)
2. Responsive to different viewport sizes
3. Gracefully handles server disconnection
4. Useful error messages
5. Recommences on server restart

Server-side functionality criteria

1. At least two entity types, with relationships
2. REST API provides each entity with appropriate GET (at least 2)/POST (at least 1) methods
3. Installs with *npm install*
4. Starts with *npm start*

Server-side quality criteria

1. Development tools and frameworks - NodeJs, Fetch, Express are mandatory. Additional tools/frameworks may be used.
2. Successful *eslint* (runs with *npm run pretest*)
3. Successful *jest* tests with good coverage (runs with *npm test*)
4. Testing includes content-type and HTTP code
5. Completeness of API documentation

Video Presentation

1. Submit a 2-minute (max) video demonstrating your software
2. Include demonstration of how to start the program
3. All functionality will be assessed by what is demonstrated in the video
4. If it is not demonstrated in the video, you will not get a mark for it
5. Quality of video presentation will be marked separately from functionality:
   (a) Structure, (b) Visual Presentation, (c) 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.

You may view your marks along the following scale:

1. **Fail (**< 40**)** – Most of the basic requirements were not met; there are major errors in the website's functionality.
2. **Approaching expectations (**$40 - 59$**)** -Many requirements were met, with some issues in the design and development methods, and some minor functionality errors are present.
3. **Meeting expectations (**$60 - 79$**)** – The basic/mandatory quality/functionality requirements were mostly or all met; the website is fully or almost fully functional.
4. **Exceeding expectations (**≥ 80**)** – the website is fully functional, proficiently addressing all the criteria asked and possibly beyond, high quality code development.

How to do the assignment

1. Design HTML
2. Design web service
3. Join with Fetch
4. Read the FAQ (Ver 1 as follows, updated versions (if required) will be posted separately, and class will be informed during lectures).