Lab 4 – node.js

In this lab we are going to create a node server to read from yet another streaming API (remember: it's about the real-timeness of the data); do not pick the same API you used for labs 2 or 3, but you may pick another API in the same broad area as them.

Your program should read and display data in a visually interesting way from your selected API. It is **node** that should ultimately be handling the API requests.

Your node server must also accept connections on port 3000. When a user connects to http://localhost:3000 you must serve a web page. Using Angular and jQuery, that web page must be able to make a request to node to get API data. Node should write out the data to a JSON file called *RCSID-APIname*.json. This file should be what the frontend then reads to display data. You can delete and replace the old data file, if it exists, if that is easier than appending to an existing file.

In short, the ideal travel path might look something like this:
Press button on frontend → communicate to node "I want to get API data now" → node sends the API request → node receives answer → node writes out answer to a local file → frontend reads the local file and updates view. You *will* want to poke around with Express.js for some of this, but you don't have to go super in depth—remember that we will spend some time in lectures on Express soon so don't feel like you need to be a pro right now.

Alternatively, you can update the frontend directly via node (socket.io), but you still need to write out the local file.

There are many methods to accomplish these features in the lab. Any method, so long as it uses all the outlined technologies, is acceptable.

In case it's not clear: this is lab1 except now you're responsible for getting the API data instead of me giving it to you.

Objective 1 – For node.js portion of the lab:

Install Express via npm, and feel free to install any other modules you feel might be useful to you.

Information about express can be found here: http://expressjs.com/

Please make sure to read the API documentation for whatever API you choose! Additionally, it would be quite wise to select an API that has a module installable via npm that will do all the heavy lifting for you. *Do not reinvent the wheel!*

You must include a package.json, which will correctly install your app's dependencies by running the command "npm install" – please do not upload a 100 MB zip file at the end of the week—I am going to tell the TAs not to grade such submissions.

Objective 2 – For the interface portion of your lab:

Please use Angular(.js) and optionally jQuery as well to create a frontend for your application. The interface can be very simple but it must have at least the following:

- A field in order to enter search criteria (if your API has search functionality)
  - If your API does not have search functionality, a field to enter some sort of user-specified parameters that the API does understand (e.g., weather API might allow the user to specify location)
- A button to get the data
- An input field to tell the application how many data entries to display on screen (particularly useful if you get back *a lot* of data!)
- A place to output the data on the screen.

Info about angular can be found at https://angularjs.org/ and https://angular.io

You may wish to stream the data from your server to the web page. Checkout https://www.npmjs.com/package/socket.io

Objective 3 – For the output portion of your lab:

You are building an input file that in theory could be used for next year's lab1, therefore, your output file needs to be in the proper format so that we may test your output file as the input for lab1. We probably won't though :)

However, if it's not in JSON format you might have a more difficult time getting the rest of the lab to work.

As you will be writing data to a file;

You will need the fs module (http://nodejs.org/api/fs.html) in order to write to your file.

Objective 4 – Extra credit – Writing your first node module

This is totally optional, but if you'd like some extra credit, you can write a node module. TO GET THE EXTRA CREDIT: You must identify a piece of your node code that is solving a generic problem, separate that code out into its own npm module, and install and use it back in your app. Don't forget to include your module in your submission on LMS! You can't just take all your node code and call it a module; that's not going to work. Whatever you separate out must solve a generic problem. It is OK if that generic problem is one that an existing module already solves (that will almost certainly be the case) but you **may not plagiarize code.** See the syllabus/first day slide deck for the stern warnings about plagiarism.

You will be graded on the following;

Objective 1 – node server and API	:	10
Objective 2 – Interface and output	:	10
Objective 3 – output portion	:	10
Objective 4 – extra credit; node module	:	(+5)
Creativity/Coding style	:	10
Documentation/Read.me	:	10

Max 55 points