CIS*2750
Assignment 3
Deadline: Friday, March 18, 9am
Weight: 25%

1. Description

In this assignment, you will create a web-based graphical user interface (GUI) for a simple image database that would display SVG images, and give you some ability to manipulate them. The front end (GUI) will be a web client written in HTML/JavaScript. The backend will use Node.js platform as the server and will call your C-based SVG parser created in Assignments 1 and 2 - compiled into a shared library - using the FFI-NAPI Node.js module.

This assignment is individual work and is subject to the University Academic Misconduct Policy. See course outline for details.

This assignment will be released as a series of modules, instead of all at once, to facilitate incremental development. I strongly encourage you to start working on Module 1 as soon as it is released.

Module releases will be announced on Moodle. Each module will be described in a separate document.

You will also be provided with an A3 Stub, which will contain stubs for both server and client code. You will need it to develop and run your assignment, but you can get started on the HTML portion of Module 1 without it.

The A3 Stub will include a script to install all the Node.js libraries you will need for the server code. A3 Stub documentation will describe how to use the stub. In addition, the A3 Stub documentation will describe your Assignment 3 submission structure and Makefile details.

If you want to work from home, you must install Node.js **16.14.0** on your home machine. It has already been installed for you on `cis2750.socs.uoguelph.ca`.

For this assignment, we are using a slightly different server setup:
- All your code, including the Node.js server, will be tested and graded on a dedicated server: `cis2750.socs.uoguelph.ca`. It has Node.js installed on it, and is set aside specifically for this course. This server will have you home directory automatically available on it, so you will have access to all your files.
- We will **not** be using `linux.socs.uoguelph.ca` for running your code. It does not have Node.js installed, and you will not be able to use it for working on Assignment 3.

The one minor snag as that `cis2750.socs.uoguelph.ca` does not have NoMachine access to it. To develop your code, you can use one of two options:
- Install Node 16.14.0 at home, develop A3 at home, and upload the code to the cis2750 server for testing. You can then connect to the cis2750 server from your browser at home to run your Web app (see details below).

- Use NoMachine to connect to `linux.socs.uoguelph.ca`, write your code on `linux.socs.uoguelph.ca`, but keep a separate terminal window with an SSH connection to `cis2750.socs.uoguelph.ca`. Compile and run your code on `cis2750.socs.uoguelph.ca`, and access the Web server **from a browser** running on `linux.socs.uoguelph.ca`. This is how we will evaluate your assignment.

2. Modules

Module 1 is the client-side (browser) component, and is the graphical user interface GUI of your app. Module 1 document will describe the user interface functionality for the client code running in the browser.

Module 2 is the server-side (backend) component. Module 2 document will describe the server-side functionality that will interact with your parser library written in C, and communicate with the web client.

3. Evaluation

Your code must compile, run, and have all of the specified functionality implemented. Any compiler errors will result in the automatic grade of **zero** (0) for the assignment. An evaluation scheme will be posted in Assignment 3 dropbox.

You can develop A3 at home. However, as always, make sure you compile and run the assignment on the SoCS servers before submitting it - this is where it will be graded. You will need to use the following setup

1. Deploy the Node.js server on `cis2750.socs.uoguelph.ca`
2. Connect to the regular Linux server using NoMachine. Connect to your Node.js server using Firefox installed on Linux: the URL will be `cis2750.socs.uoguelph.ca:your_port_number`

   Individual port numbers are discussed below

If Module 1 (GUI) is implemented, but not connected to Module 2, you will get some part marks - approximately 50% of the overall assignment grade. However, if you do not implement Module 2 and want to get part marks for Module 1, you must implement stubs for Module 1 functionality. These stubs will be described in the A3 Grading Scheme. In a nutshell, the stubs need to indicate that the UI elements are fully function, and simulate the correct output.

Module 2 (backend) functionality must be connected to a functional Web app GUI described in Module 1. Module 2 functionality that is not connected to a Module 1 GUI will be worth zero (0) marks.

Please note that, as mentioned earlier in the course, Assignments 3 and 4 are not graded using test harnesses. As a result, there will not be any preliminary test harness releases.

Starting from this assignment, all evaluation will be done through your GUI. Unlike the previous assignments, the lower-level modules will not be individually tested. In general, you can obtain marks for correct front-end GUI (HTML/JavaScript) functionality and correct back-end functionality (JavaScript/C code), or for only the GUI in case the server back-end doesn't work.

Front-end functionality means that the GUI elements are present and can be manipulated. For example, if the back-end of getting details of a SVG image does not work yet, you should at least display a message that proves that the GUI element for SVG View Panel is connected to a callback function for the corresponding drop-down list. See Module 1 for details. Passive/static GUI elements with no callbacks will not get full front-end credit.

When A3 Stub is released, each of you will be assigned a unique port number (plus a backup number). This port number will be used only by your assignment. Failure to use port numbers as specified in A3 Stub documentation will result in penalties, up to and including receiving a **zero (0)** for the assignment. Connecting to someone else's code (client or server), either on purpose or accidentally, will constitute Academic Misconduct and will be treated as such.

4. Bonus marks

You can get a bonus mark for additional GUI functionality (3%). The additional GUI functionality will be discussed in Module 1.

5. Incremental development

A suggested road map would go like this:

1. Design your GUI on paper.
2. Familiarize yourself with the Web UI development: HTML, JavaScript, and jQuery. Read Week 6-8 lecture notes, review the posted code examples, and use *"Learning PHP, MySQL, JavaScript, CSS & HTML5*" as a reference. Chapters 13 - 15 are good starting point, since they cover JavaScript and HTML.

3. Convert your GUI design to static HTML code that you can run in a browser.

   You can do steps 1-3 without the use of the A3 Stub.

4. Get your program working so it successfully displays your GUI. Populate it with enough fake component data to prove that the scroll bars are working.
5. Create and connect all your JavaScript callbacks, so if a user clicks on any of the buttons, appropriate output is displayed using JavaScript alerts.
6. **Test** this setup code - run the server on cis2750.socs.uoguelph.ca and connect to it from linux.socs.uoguelph.ca using Firefox.

Do all this before you start working on Module 2. Remember, Module 2 is not worth any marks without a working GUI.

The overall functionality of Module 2 should be fairly obvious from the description in Module 1. However, Module 2 description will include the necessary implementation details and guidelines.

Once Module 1 is complete:

7. Shift your attention to the server side: design and implement any additional C functions that you might need (in addition to the functions in A2 Module 3). Implement them first as stubs that print their arguments when called, and pass back some dummy data to server-side JavaScript code.
8. Update your JavaScript server code so it actually calls these functions and displays the dummy data from them. Test this until it works reliably. You have now established two-way communication between JavaScript and C.
9. Create server endpoints for these dummy functions. Have your Web GUI make some simple Ajax requests, and have your server code pass the dummy data it got from C functions to the web GUI as a response to a get request from the GUI. Display the dummy data in the browser. You have now completed the circuit and established two-way communication between the Web GUI and the JavaScript / C backend.
10. Modify your C wrapper functions so they do their intended job. Now you can read real SVG images into your server code. Keep on mind that most of the wrappers have already been implemented in A2, i.e. converting between C struct sand JSON strings.
11. Complete the callbacks for client Ajax requests, so you can pass the JSON information between the server and the GUI.
12. **Test** this as you did before: run the Node.js server on cis2750.socs.uoguelph.ca, and connect to it from NoMachine using Node.js and Firefox installed there (see Module 1 and 2 descriptions for details).

6. Submission

You are working on the same GitLab code base as before. You must commit and push your code - which will include C and JavaScript code, along with the necessary Makefile - into the A3 branch of your CIS*2750 project on gitlab.socs.uoguelph.ca by the specified date. The details of the submission structure will be discussed in A3 Module 2.

Late submissions: see course outline for late submission policies.

This assignment is individual work and is subject to the University Academic Misconduct Policy. See course outline for details)