CIS*2750 Assignment 3 grading scheme

All the grades have the form **N/M**

- **N** is the max grade you get for a fully functional UI connected to a fully functional back-end
- **M** is the max grade you get for a fully functional UI without a connected back-end, but with all stubs that "fake" a functional backend.

To get grades for the UI elements, make sure that they are connected to callback functions. Each callback should provide a visible output, as described below.

Your assignment is graded based on the functionality it displays. Having a broken server (back-end) route for some functionality is the same as not implementing that functionality. In both cases the grade is zero for that part of the assignment. In fact, a fully functional UI with stubs and without the back-end could be worth more marks than a UI that connects to the back-end incorrectly (or results in a server crash), and has no stubs.

For example, the file Log Panel functionality is worth 25 marks if fully implemented (UI and JavaScript/C back-end), and 12.5 marks if the UI is not connected to the back-end, but has all the stubs - i.e. when the web page loads, a table with dummy SVG filer summaries is displayed, using the format specified in Module 1.

Another example:

- For the SVG View Panel, if the grader selects a file from the pull-down menu, and nothing happens - e.g. because you have messed up an Ajax call to the server or the C library on the server has crashed - you will get a very low grade for this functionality - much less than the grade for the UI.
- On the other hand, if the grader selects a file from the pull-down menu and the UI displays a “dummy” table of SVG file details without connecting to the server, you will get a higher grade - up to the lower of the two marks below (i.e. up to 10 marks), assuming the rest of this functionality is up to spec.

As a result, do not submit an assignment with a completely broken back-end and no UI stubs, and expect a passing grade. You should create stubs for all UI functionality, and only replace stubs with a back-end connection once you have verified that the back-end for this functionality actually works.

Grade breakdown

Note: all functionality must display errors using alerts, as specified in Module 1. Failure to do so will result in deductions for every UI component that does not handle error correctly.

- **File Log Panel (25 marks / 12.5 marks)**
  - correctly updated when a file is uploaded or created
  - displays all the required details in table format, as specified in Module 1
  - includes a downloadable link for every valid file on the server
  - does not display invalid files

- **Upload a file (5 marks / 2.5 marks)**
  - Contains the file selection window and a button.

- **SVG View Panel (20 marks / 10 marks)** - showing / editing attributes and changing title / description is graded separately
  - Drop-down list contains all valid files on the server
  - Displays all the required details in table format, as specified in Module 1

- **Show attributes (7 marks / 3.5 marks)**

  - This is done for a specific element
  - Displays all the required details, as specified in Module 1
  - The text must be formatted for human readability - do not just display a raw JSON string

- **Edit Attribute (7 marks / 3.5 marks)**
  - This is done for a specific shape inside an SVG image, or the SVG image itself (i.e. the <svg> element)
  - User enters all details using forms
  - Validates user input and does not allow a user to create an invalid SVG file
  - When a .svg file is updated, the File Log Panel must be updated and reflect the change. The change must also be visible in the image itself, as shown in the File and SVG panels.

- **Edit title/description (6 marks / 3 marks)**
  - This is done for a specific SVG image
  - Validates user input and does not allow a user to create an invalid SVG struct due to length of strong exceeding max title/description length in `SVGParser.h`.
  - When a .svg file is updated, the summary for that image in SVG View Panel must reflect the change.

- **Create SVG (10 marks / 5 marks)**
  - Create a new SVG struct, pass it to the parser, save it to the .svg file
  - User enters all details using forms
  - Validates user input and does not allow a user to create an invalid SVG image
  - When a new .svg file is created, the File Log Panel and all the file lists must be updated to include it.

- **Add Shape (10 marks / 5 marks)**
  - Adds a circle or rectangle to a specific file, as described in Module 1
  - Drop-down list contains all valid files on the server
  - User enters all details using forms
  - Validates user input and do not allow a user to create an invalid shape or image
  - The new shape must be visible in the image, and the File view panel entry for that file must be updated accordingly.
  - If the user requests the details for the updated .svg file, the new route must show up in the SVG View Panel.

- **Scale Shapes (10 marks / 5 marks)**
  - Scales all circles or rectangles in a specific file, as described in Module 1
  - User must be able to select what shape to scale
  - Drop-down list contains all valid files on the server
  - User enters all details using forms
  - Validates user input and do not allow a user to create an invalid shape or image
  - The changes to the shape must be visible in the image, the File view panel entry for that file must be updated accordingly, and the shape must appear in the summary in SVG View panel
  - If the user requests the details for the updated .svg file, the new route must show up in the SVG View Panel

- **Total: 100 marks (UI+back-end) / 50 marks (UI+stubs)**

Bonus:

- **Scale Image (3 marks)**

- Must correctly scale the image by a factor provided by the user
- No part marks will be given for the bonus
- Bonus functionality will only be graded if the main A3 functionality is fully connected to the backend, received at least 90 marks, and Scale Shapes received full 10 marks.

Evaluation procedure - installation and execution

Your code must compile, run, and implement all functionality listed in Modules 1 and 2. Your code will be compiled using your Makefile, which must create a shared library (see Module 2 for details) that will be used by `app.js`.

Your backend will be compiled and executed on `cis2750.socs.uoguelph.ca` using a port number of our choice. **Do not hard-code your port number!**

Your Web front ent will be graded using from Firefox on `linux.socs.uoguelph.ca`. It will be accessed as `cis2750.socs.uoguelph.ca:portNum`

Before submitting the assignment, make sure it is compatible with the grading requirements:
- delete your `node_modules` and compiled shared library (`.so` file)
- run `npm install`, recompile your shared library using your Makefile
- run your server, and make sure everything works!

If your server fails to execute for any reason when we run `npm run dev portNumber` (with a port number of our choice), only your `index.html` will be graded - see below.

Make sure you compile and run the assignment on the SoCS Linux server before submitting it - this is where it will be graded. You must test your code on NoMachine, using Firefox installed on the SoCS servers. If your GUI does not work correctly on this browser, you will lose a lot of marks - up to and including getting a **zero (0)** in the assignment.

If you do not provide a Makefile, you will lose all the marks for the back-end, and your maximum assignment grade will be **50 marks**.

If you provide a Makefile, but it fails to compile, you will lose all the marks for the back-end, and your maximum assignment grade will be **50 marks**.

As stated in Module 2, all your Node.js modules must be automatically downloaded when your code is downloaded on the SoCS server and the grader types "`npm install`". If your A3 back-end does not run due to missing dependencies when we grade it, you will lose all the marks for the back-end, your maximum assignment grade will be no more than **50 marks** - but it can be much less if the missing dependencies prevent your server from executing.

NOTE: If your UI lacks the stubs and the backend cannot be run for any reason, we will grade your `index.html`. However, your maximum assignment grade will be **20 marks**.

Deductions

You will lose marks for run-time errors and incorrect functionality.

Additional deductions include:

• Any compiler warnings:	**-15 marks**
• Any additional failures to follow assignment requirements:	**up to -25 marks**

- This includes creating an archive in an incorrect format, modifying the assignment directory structure, creating additional `.js` and `.html` files, etc.
- Unprofessional language is also included in this category, and will be penalized.