CIS*2750
Assignment 3, Module 1

1. Preliminaries

You will be provided with the Assignment 3 stub, which includes both the client and the server stubs. See A3 Stub documentation for details. You **must** follow the code organization described in the A3 Stub documentation.

Your Module 1 functionality must be placed into three stub files:

- All HTML code must go into `public/index.html`
- All Module 1 (client-side) code must go into `public/index.js`. Do not embed any JavaScript code directly into `index.html`.
- A template CSS file has been provided for you in `public/style.css`. Do not embed any CSS code directly into `index.html`.

However, you do need the A3 Stub to gets started on Module 1. Simply use the provided `index.html`. It is identical to `index.html` that will be provided in the A3 Stub. Once the A3 Stub is released, replace the `index.html` file in `public/` directory of the A3 Stub with the one you've been working on.

As discussed in class, you will be using the de-facto standard of modern Web graphical user interfaces - HTML + JavaScript with the jQuery library. Some jQuery examples are provided for you (see Week 7 examples). There are also multiple JavaScript/HTML/jQuery examples in *"Learning PHP, MySQL, JavaScript, CSS & HTML5*".

You have some freedom in how you want to lay out the GUI and determine its exact appearance, but the items below items will be required. Items specified as drop-down lists or buttons must stay that way; you cannot switch them to other UI elements.

The `index.html` file provided for you already includes a number of JavaScript libraries, including jQuery and Bootstrap.

You may wish to use Cascading Style Sheets (CSS) for prettifying your GUI, but you are not required to do so. You will not lose any marks if you use the default CSS file included in A3 Stub.

You **must** test your code on NoMachine, using **Firefox 52.6.0** installed on the SoCS servers. If your GUI does not work correctly on this browser, you **will** lose marks - up to and including getting a **zero (0)** in the assignment. Testing it on your home browser is **not** sufficient.

Make sure you test your code on the Linux server **as you develop it** - **do not** save the testing until the night the assignment is due! Use the NoMachine Graphical Linux Environment: **https://wiki.socs.uoguelph.ca/techsupport/guides/nomachine**

2. Overview of Web GUI

The UI web page has a title showing "Genealogy App". It has a number of required forms and buttons. There will be three major display "panels" (how you implement them in HTML is up to you):

- The GEDCOM View Panel is for display of individuals in a GEDCOM file, one individual per line. This panel must be scrollable, both horizontally and vertically. If the GEDCOM View Panel contains more than 4 individuals, vertical scrolling must be enabled.
- The File Log Panel for displaying file summaries and download options. It is also scrollable, both horizontally and vertically. If the File Log Panel contains more than 2 files, vertical scrolling must be enabled.

- The Status Panel is displayed at the top of the window, and displays the status of various commands (success or various errors). It is not a particularly elegant solution, but it is easy to implement and it simplifies debugging. If the Status Panel contains more than 4 lines, vertical scrolling must be enabled.

Sample app layout:

Status Panel
File Log Panel
GEDCOM View Panel
Section 3 functionality goes here

Each panel is described in its own section below.

Keep in mind that while the Status Panel must stay on top of the page, the other two panels and all the forms/ buttons described in Section 3 can be organized as you see fit. Do whatever you think makes a good UI. However, all code **must** be organized as discussed in Section 1.

3. Working with genealogy data and GEDCOM files

When your web client starts, it parses all `.ged` files uploaded to the server, displays their summaries in File Log Panel, and adds them to the list of files in GEDCOM View Panel, and all the drop-down lists of server file names. If there are no .ged files on the server, File Log Panel and GEDCOM View Panel must display appropriate messages(See Sections 4 and 5).

For the Module 1 stub, when the app loads, provide one row of dummy data in File Log Panel, and one row of dummy data in GEDCOM View Panel.

Your client GUI must have the following interactive functionality:

- **Upload a GEDCOM file**: Obtain a filename (see details below) and upload the .ged file to the server. If the server request (upload) is unsuccessful - i.e. server returns some sort of error, go no further - the File Log Panel is unchanged. Display the error in the Status Panel. If the server request is successful - i.e. server returns some sort of OK message - add a row to the File Log Panel using the file's components obtained from the server. If the file you are trying to upload already exists on the server, display an appropriate message in the Status Panel.

  Obtaining a filename*:* you must open a file browser for the user to select a filename. If an input file does not exist, display the error in the Status Panel.

  For the Module 1 stub, obtain the filename using a file browser and display it in the Status Panel with an appropriate message (e.g. “Uploaded shakespeare.ged”). In addition, display a dummy row of fake file data, including the appropriate file name, in the File Log Panel.

- **Download a file:** This is done by clicking on the link in the File Log Panel (See Section 5).

  For the Module 1 stub, simply display an appropriate message in the Status Panel (e.g. “Downloaded shakespeare.ged”) when the user clicks on a link in File Log Panel. Make sure there is at least one row of dummy information in the File Log Panel, so the user has something to click on.

- **Create Simple GEDCOM**: Creates a GEDCOM file with a header and a submitter, but no individuals. The file is saved on the server. User must provide the file name, and you can verify that the file name is unique. Source, encoding, and GEDCOM version may be filled in by the UI automatically (it’s up to you). User must provide submitter name, and may also provide submitter address.

Depending on what happens on the server, the file may or may not be saved. If the server response indicates an error, file was not saved. Display the error message in the Status Panel. If the server response indicates success, file was saved. If the file was saved, update the contents to the File Log Panel to include the summary of the new file, and add the file name to the drop-down list in GEDCOM File Panel, and all the other drop-down lists of file names.

The user input must be entered using forms, and there must be a "Create GEDCOM" button.

For the Module 1 stub, when the user presses the "Create GEDCOM" button, display an appropriate message in the Status Panel (e.g. "saved a new file newFamily.ged"), add a new dummy row of data in the File Log Panel for that file, and update all the drop-down lists.

- **Add Individual**: Add an individual to one of the files currently uploaded to the server. File name must be selectable with a **drop-down list** - do not force the user to type in the file name. If the request to the server to insert an individual into a file succeeds, update the contents to the File Log Panel entry for that file. If the request to the server fails, display an appropriate error message in the Status Panel.

  The user input must be entered using forms, and there must be an "Add individual" button.

  For the Module 1 stub, provide a dummy file name for the drop-down list. Display an appropriate message in the Status Panel (e.g. "added a new individual Jane Doe to file newFamily.ged") when the user presses "Add individual" button.

- **Get Descendants**: display the descendants of the individual. The user must be able to select the file that they wish to query with a **drop-down list**. The user will need to enter the individual's first and last name, as well as the max number of generations, using forms. You may also allow the user to select an individual using a drop-down list, or through other means.

  The request would be sent using the "Get Descendants" button.

  If the request to the server succeeds, display a table of descendants (1 generation per row) below the "Get Descendants" button. Make the table scrollable both horizontally and vertically.

  If the request to the server fails, display the message "No Descendants" below the "Get Descendants" button.

  For the Module 1 stub, provide a dummy file name for the drop-down list. When the user selects the file and presses the "Get Descendants" button, display two generations of dummy descendants below the "Get Descendants" button.

- **Get Ancestors**: display the ancestors of the individual. The user must be able to select the file that they wish to query with a **drop-down list**. The user will need to enter the individual's first and last name, as well as the max number of generations, using forms. You may also allow the user to select an individual using a drop-down list, or through other means.

  The request would be sent using the "Get Ancestors" button.

  If the request to the server succeeds, display a table of ancestors (1 generation per row) below the "Get Ancestors" button. Make the table scrollable both horizontally and vertically.

  If the request to the server fails, display the message "No Ancestors" below the "Get Ancestors" button.

  For the Module 1 stub, provide a dummy file name for the drop-down list. When the user selects the file and presses the "Get Ancestors" button, display two generations of dummy ancestors below the "Get Ancestors" button.

4. GEDCOM View Panel

This panel shows individuals in the currently selected file, one line per individual. This is intended to be a tabular-looking view with rows and columns (e.g. HTML table), though how you implement it is up to you. Since there will often be many individuals in a single file, this view must be scrollable, both horizontally and vertically.

The individual row is intended to provide a summary of the individual. There will be 4 properties, each in a separate column. Two of these columns must be filled it, the other two may be filled in for bonus grades. Columns must have accurate headings.

Given name (Mandatory): The individual's given name. Must be the given name of that individual in the GEDCOM file. Leave empty if the GEDCOM file does not list the given name of that individual.

Surname (Mandatory): The individual's surname. Must be the surname of that individual in the GEDCOM file. Leave empty if the GEDCOM file does not list the surname of that individual.

Sex (Bonus): The individual's sex. Leave blank if you decide not to implement it. If you do decide to implement it, it must display the individual's sex as specified in the GEDCOM file.

Family size (Bonus): The size of the individual's family, including the individual itself. Individual with no spouse and children has family size 1, individual with a spouse and no children - 2, etc. Leave blank if you decide not to implement it. If you do decide to implement it, it must display the correct family size for that individual, so this value would always be 1 or more.

Above the panel, there is a **drop-down list of file names**. It must include all the files currently stored on the server. This list is empty if there are no files on the server.

Here is a sample row:

Given name	Surname	Sex	Family size
William	Shakespeare	M	5

If the selected GEDCOM file contains no individuals, display the message "No individuals" in the GEDCOM View Panel.

For Module 1 stubs, put dummy data here, as described in Section 3. Remember to remove the dummy data once you connect Module 2 to the GUI.

5. File Log Panel

This panel displays the list of **all** GEDCOM files on the server, including all the files uploaded from client and the all files created from scratch by the client. The panel is scrollable horizontally and vertically. The panel contains a link to the downloadable GEDCOM file, and a summary of that file's properties:

- Source
- GEDCOM version
- Encoding
- Submitter name
- Submitter address
- Number of individuals
- Number of families

If any of the values are not specified in the source GEDCOM - e.g. submitter name is not provided - leave that field empty. Number of individuals and number of families cannot be empty, and must be 0 or greater.

Here is a sample row:

File name (click to download)	Source	GEDC version	encoding	Submitter name	Submitter Address	Number of individuals	Number of families
simpleGEDCOM.ged	PAF	5.5	ANSEL	Submitter		3	1

If there are no files on the server, display the message “No files” in the File Log Panel.

For Module 1 stubs, put dummy data here, as described in Section 3. Remember to remove the dummy data once you connect Module 2 to the GUI.

6. The Status Panel

The Status Panel is a display box at the top of the window, which is be used to display various status messages, as described in Section 3. This panel is scrollable both horizontally and vertically. Messages generated by the UI must clearly identify which file and/or error is implicated. For example, messages that only say "File not opened" (without giving the file name) or "Syntax error" (without giving a humanly readable description of the error) are not acceptable.

The Status Panel keeps scrolling up so that the latest messages are in view at the bottom, but the user may operate the scroll bar to reposition the view. In addition, it must have a **Clear** button that clears the contents of the panel.

For Module 1 stubs, use this space to display the messages from the callbacks, as described in Section 3. Remember to remove these messages once you connect Module 2 to the GUI.

7. Defensive programming

Validate user input, and explicitly report errors to the user (use Status Panel or other means). If your GUI silently accepts invalid user input, you will lose marks. If you allow invalid input and create invalid files - e.g. files with an extension other than “.ged” or files that violate GEDCOM specification - you will lose marks.