CIS*2750
Assignment 3
Module 1

1 Preliminaries

You are provided with the 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 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`.
- All server-side code JavaScript code must go into `app.js` in the root directory of the stub.

As discussed in the lecture notes, 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 the Stub.

You **must** test your code on NoMachine, using **Firefox 60.5.1esr** 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 this project. 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 last minute! 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 “Calendar Viewer". 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 Calendar View Panel for display of calendar components, one component per line. You should assume that the calendar you load from a file could have at least 8 events, and the panel is appropriately scrollable.
- 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
Calendar View Panel
Additional functionality goes here (create calendar, create event)

Keep in mind that while the Status Panel should be placed on top of the page, the other two panels and all the forms/ buttons described in Sections 3 - 6 can be organized as you see fit. Do whatever you think makes a good UI. However, all specific tables **must** be organized as discussed in Section 1.

This GUI layout is a bit clunky, but it is simple, and it saves you from working on multiple windows/tabs in your very first Web app.

3 File Log Panel

This panel displays the list of **all** iCalendar 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 iCalendar file, and a summary of that file's properties:

- iCalendar version
- iCalendar product ID
- Number of events in the calendar
- Number of properties in the calendar

Here is a sample row:

File name (click to download)	Version	Product ID	Number of events	Number of properties
testCalSimpleUTC.ics	2.0	-//hacksw/handcal//NONSGML v1.0//EN	1	2

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

If a particular iCalendar file is invalid, you cannot display its summary. In this case, display its name, display "invalid file" in Product ID column, and leave all other columns blank

Place an "Upload file" button at the bottom of the File Log Panel. Upload functionality is described below, in Section 6.

4 Calendar View panel

This panel shows components in the currently open file, one line per component, in the order that `createCalendar()` returns them. This is intended to be a tabular-looking view with rows and columns (e.g. an HTML table), though how you implement it is up to you. Since there will often be many events in a single file, this view must be scrollable, both horizontally and vertically.

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. When the user selects a file, run `createCalendar()` on

the file and display its output in this panel. If `createCalendar()` returns an error, go no further (the Calendar View panel is unchanged). On good status, fill the panel from the file's components.

A component row is intended to show a few important properties at a glance. There will be 4 of them, each in a separate column. These 4 columns are mandatory, should be sized to fit on one line, and must have accurate headings.

Event No.: A sequence number counting up from 1 (the position of the Event in the event list).

Props: Total number of properties (required + optional).

Alarms: The number of alarms in the event

Summary: Value of SUMMARY property. If no SUMMARY is available, leave this empty.

Here is a sample row:

Event No	Start date	Start time	Summary	Props	Alarms
2	1954/02/03	12:30:12 (UTC)	Do taxes	4	2

Note that the date and the time have separators for readability. Also, the time must have the UTC label if it is UTC.

Actions

The user must be able to perform the following actions. How you implement the UI for them is up to you.

Show alarms: The action is to display the alarms for a specific event. You just have to create a wrapper function for it - e.g. `alarmListToJSON()` - which returns a JSON string with a readable alarm representation, and parse the JSON in the JavaScript code. You would call this function from the GUI and display the returned string in the Status Panel.

Extract optional props: The action is to display the optional properties for a specific event in the Status Panel. Again, some sort of `...toJSON()` function is your friend here.

5 Status Panel

The Status Panel is a display box at the top of the web page, 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.

6. Working with calendar data and iCalendar files

When your web client starts, it parses all `.ics` files uploaded to the server, displays their summaries in File Log Panel, and adds them to the list of files in File View Panel, and all the drop-down lists of server file names. If there are no `.ics` files on the server, the File View Panel must display appropriate messages (See Section 3).

Your Web client GUI must have the following interactive functionality:

- **Upload an iCalendar file**: Obtain a filename (see details below) and upload the .ics 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. In addition, add the file name to all the drop-down lists of server file names - i.e. in Calendar View Panel, and the Create Calendar module (see below). 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.

- **Download a file:** This is done by clicking on the link in the File Log Panel (See Section 3).

- **Create calendar**: Creates a Calendar object, which is saved to an iCalendar file. Your calendar must be created with at least one event. The file is saved on the server using `writeCalendar()`. User must provide the file name, and you can verify that the file name is unique. The new object must be validated before being saved.

  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 Calendar View 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 Calendar" button.

  Events in a new calendar

  A calendar without at least one event is invalid. As a result, you will need to make sure the user adds an event to a newly created calendar. How you do this is up to you. Whatever you do, a user **must not** be able to save a calendar without an event to a file (`validateCalendar()` can catch this for you). We never want to allow the user to create invalid files.

- **Create event**: add an event 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.

  Don't worry about adding alarms or optional properties, other than SUMMARY - just have the user enter the required properties. Give the user the option of adding the event summary, but don't require them to do so.

  Make sure the user enters all the required properties, including the two dates (which will become DTSTART and DTSTAMP properties). Again, how you organize the GUI is up to you. Whatever you do, a user **must not** be able to create an event without valid dates.

  If the request to the server to insert an event 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 event" button.

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 ".ics" or files that violate iCalendar specification- you will lose marks.