Hello!

Ok, so before we hit the scope of work for the next (last) phase of work on the Chart Widget, I need to make sure that we get the import thing sorted. After polling a sampling of folks expected to be participating in the course that will use the Chart Widget, my client has finally decided that there isn't a "template" that he can provide for how the data will look when it comes in. So that "template" I gave you is good enough to go on.

But given that template, as I mentioned in an earlier message, the import function isn't working. Whenever I or my client try to do an import (I'm on a mac, he's on Windows 7), after selecting the file and clicking "import," we just get an error message.

So the very first thing is to make sure that we can import that sample document we've been working with. Once that bit is sorted, I'll pay the other half of the little interim contract, and everything below will be a new contract. If it's working for you and you need me to help troubleshoot, let me know so we can hop on skype and go over what I'm seeing.

OK.

Now then. The next (last?) phase of work for next week (Feb 8 - 14). Here are the things that need to be done, in order of importance. Please look through these, and let me know your estimate on time to completion.

1. **Have the widget create a new user if an id does not exist in the database.**

You may remember that the user will not use a login page to enter the chart widget, but will basically be pressing a button from another page that will take them to the chart widget with their user ID as part of the URL. We need to incorporate the functionality where if a user tries to access the chart widget with an unknown user ID, we create that user's record.

For example, if I go to healmydisease.com/MartyYoungProject/public/?user=1000000, there is data for that user and I will just be able to navigate to my data/charts. What we need to add is: if I go to healmydisease.com/MartyYoungProject/public/?user=1111111, the chart widget will *create* the user record for 1111111 and then proceed to the existing “What would you like to do?” page.

1. **Support “sets” of data.**

We need to add a table with an ID and Name field that will capture the names of “sets” of data, and then add a column to the data table that can capture the Set ID for each row of data, so that we can have varying sets of data.

The user experience will be like this: When a user imports data, after they select the file to be imported and click continue, a screen will ask them if they want to add the data to an existing set, or create a new set for the data they are importing. They should see a drop-down list with the names of all their existing sets (from which they will choose one) with a button that says “import my data in to this dataset.” Beneath that should be a text field that they can enter text in to, with a button that says “create this new list for my dataset.”

Depending on the user's choice, the data they import should be tagged with the ID of the set they have chosen (or create the new set and tag the data with that id). They should see an error message if they don't choose a set but try to import to existing, or if they try to create a new set with a non-unique name.

Oh and just so it has been said: the sets will *not* be compared *to each other* – they are completely standalone. We just need the user to be able to have multiple sets rather than one.

1. **On import, allow the user to define the columns of data.**

As I mentioned above, my client has determined that we won't be able to have a “template” describing the data as we will get it. So here is the compromise I was able to come up with: Like the “dummy data” spreadsheet we've been working with, every spreadsheet the user will import will have a DATE in the first column (formatted as YYYY-MM-DD) and a TIME in the second column (formatted as HH:MM on a 24 hour scale). Each row will begin with the time and date (and if either is blank, that row can be ignored by the importer).

What we won't know are what the remaining columns will contain. So. In order to determine that, we will present the user with a page labelled “Column 1” “Column 2” “Column 3” etcetera for as many columns as we see in their spreadsheet. Next to each will be a dropdown list containing all of the fields we support in the database. Column 1 and Column 2 will already be selected as “Date” and “Time” and the user won't be able to change them, but from column 3 onward, they will have to pick from the dropdown list to indicate what data is contained in that column. Once they click “continue,” we will import the data into the columns as selected.

Not selecting an answer should be a valid option (for instance, column 6 of the data may be completely irrelevant to our Chart Widget), but the user cannot choose two columns to belong to the same data type (columns 4 and 8 cannot both be “Chilled-Water Return Temperature,” for instance).

I know that setting up a single page with all the columns and corresponding dropdowns is a pretty hefty feat of programming, so an alternate idea might be to go one column at a time (here's column 1, we know it will contain the date, click next – here's column 2, we know it will contain time, click next – here's column 3, please tell me what it is and click next – and so forth). Or if there's another way to do it, I'll defer to your design. I'm just trying to describe above that we need to give the user the ability to tell us where their data is.

*Please note:* this seems like it might be the biggest hassle. If it seems like this piece of functionality might be too big by itself, we could separate it into its own project. I know I began by saying “here's what needs to be done in order” – but this one piece can be pulled out on its own if it's too big.

1. **Allow the user to change the chart type.**

I'm hoping this one is easy. The way we display the charts now should be the default view, but we should give the user the ability to change the chart so it displays *all* the data in that chart as: (1) a line graph, (2) a bar graph, or (3) a point graph. (Just a little toggle that will switch between those three options and (4) default.)

1. **Port the chart widget to run locally on a user's machine.**

I've noticed that Laraval supports mySql and Sqlite right out of the box. Can we make a version that flips that switch to Sqlite and runs on the user's machine locally (using the browser as the interface, still)? I can help with the “installation” part of the equation (I'm assuming we'll need to walk the user through installing Sqlite and getting the files for the chart widget on to their machine in a specific location). The only functional change that would be needed is that when running locally, we will assume that there will only ever be one user of that data – we will give them a user ID and that's the only user ID that will exist, ever (they will still need to be able to have multiple sets of data, just only one user will ever use the standalone version).

Please note that the two versions are meant to be completely separate from each other. The standalone version is the same functionality, it just uses a local database and doesn't need to support any more than one user. There is no data sharing between the two versions.

1. **Support the management of data sets.**

This goes with number 2 above, but it's not as important. There should be a place where a user can rename a dataset, or delete it altogether.

1. **Support printing a chart.**

Put a “print this chart” button on each chart page that will print just the chart (none of the menu or navigation, just the chart, its title, and the legend). Or more specifically, that will open just the chart in a new browser window (the user can print with the browser’s print function).

1. **Split the display of the data to be a little nicer on a screen.**

This one can definitely be dumped if it is more than we can accomplish in the week, but you mentioned this before, and I thought it was a good idea. We support something like 34 columns of data, and if you view it all, it runs off the right side of the screen. If we split the display of the data into smaller groups of columns (in alphabetical order), it will be easier to manage. (The date/time column will appear on each page, since that is how we demarcate each row of data.)

Group 1

* Date
* Time
* Chilled-Water Loop Differential Pressure (ChWLDP)
* Chilled-Water Loop Differential Pressure Set Point (ChWLDPSP)
* Chilled-Water Return Temp (ChWRT)
* Chilled-Water Supply Temp (ChWST)
* Chilled-Water Supply Temp Set Point (ChWSTSP)
* Cooling-Coil Valve Signal (CCV)
* Consumption (ConskWH)
* Damper (name)
* Discharge-Air Temp (DAT)
* Discharge-Air Temp Set Point (DATSP)
* Duct Static Pressure (DSP)
* Duct Static Pressure Set Point (DSPSP)

Group 2

* Date
* Time
* Heating-Coil Valve Signal (HCVS)
* Hot-Water Loop Differential Pressure (HWLDP)
* Hot-Water Loop Differential Pressure Set Point (HWLDPSP)
* Hot-Water Return Temp (HWRT)
* Hot-Water Supply Temp (HWST)
* Hot-Water Supply Temp Set Point (HWSTSP)
* Mixed-Air Temp (MAT)
* Occupancy Mode (OM)
* Outdoor-Air Damper Position Signal (OADPS)
* Outdoor-Air Fraction (OAF)
* Outdoor-Air Temp (OAT)

Group 3

* Date
* Time
* Return-Air Temp (RAT)
* Supply-Fan Speed (SFSpd)
* Supply Fan Status (SFS)
* VAV Damper Position Set Point (VAVDPSP)
* Zone (name)
* Zone Damper Position Signal (ZDPS)
* Zone Occupancy Mode (ZOM)
* Zone Reheat Valve Signal (ZRVS)
* Zone Temperature (ZT)