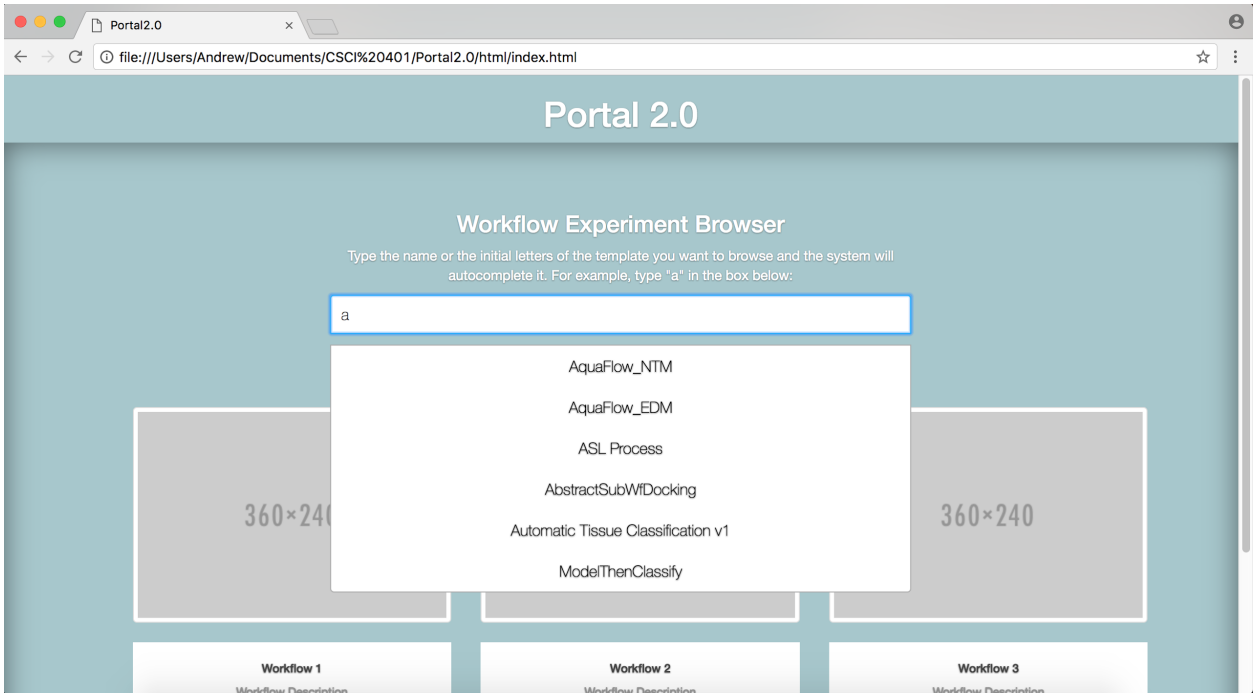
Huy Ngo
Andrew Nguyen
Tiffany Truong

Deliverable 3

Home Page



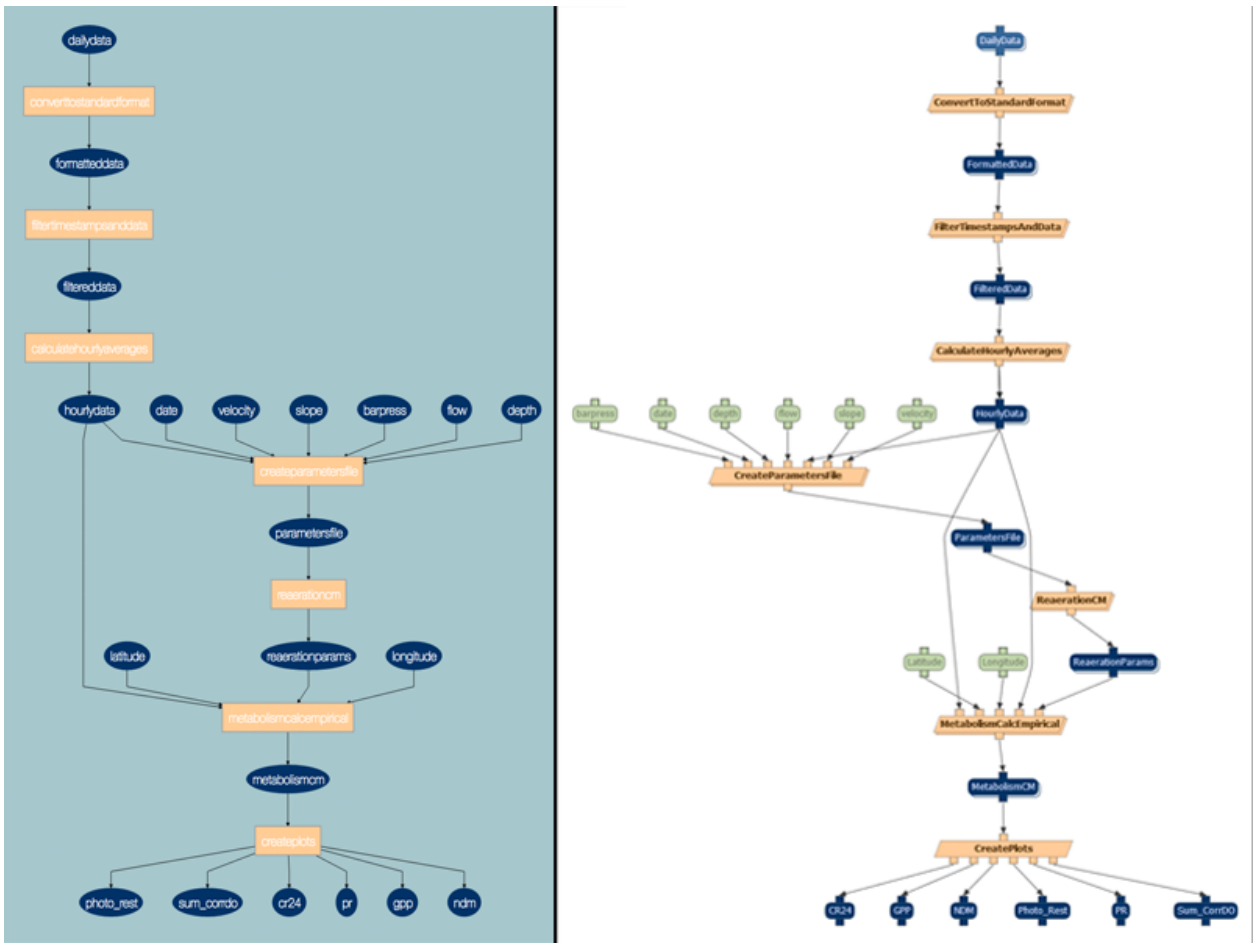
Current State:

Since the past deliverable, we have added an autocomplete react component in place of our search bar. In our html file, we simply added a div with an id of “search-bar” with the react component lying in js/landing-page.jsx. The autocomplete computes suggestions by searching our list of workflows for any match with the user’s input. The list of workflows we have is currently hardcoded so the results are minimal. However, when the number of suggestions becomes large enough to stretch beyond the bottom of the page, we will have a scrollbar that allows users to explore all the different options.

Goals for next deliverable:

For the next deliverable, we want to implement REST calls so that we dynamically populate the list of workflows from the database. When we get the results from the database, we will have to parse the URIs and retrieve their workflow names. We will still need to store the URIs with their workflow names so that when a user selects their desired workflow, we can pass its URI to the workflow-main page so it can generate the visualization. Also, our stakeholder wanted the queries to the db to be easily modified so we will need to avoid hard-coding the various queries and endpoints.

Visualization Graph



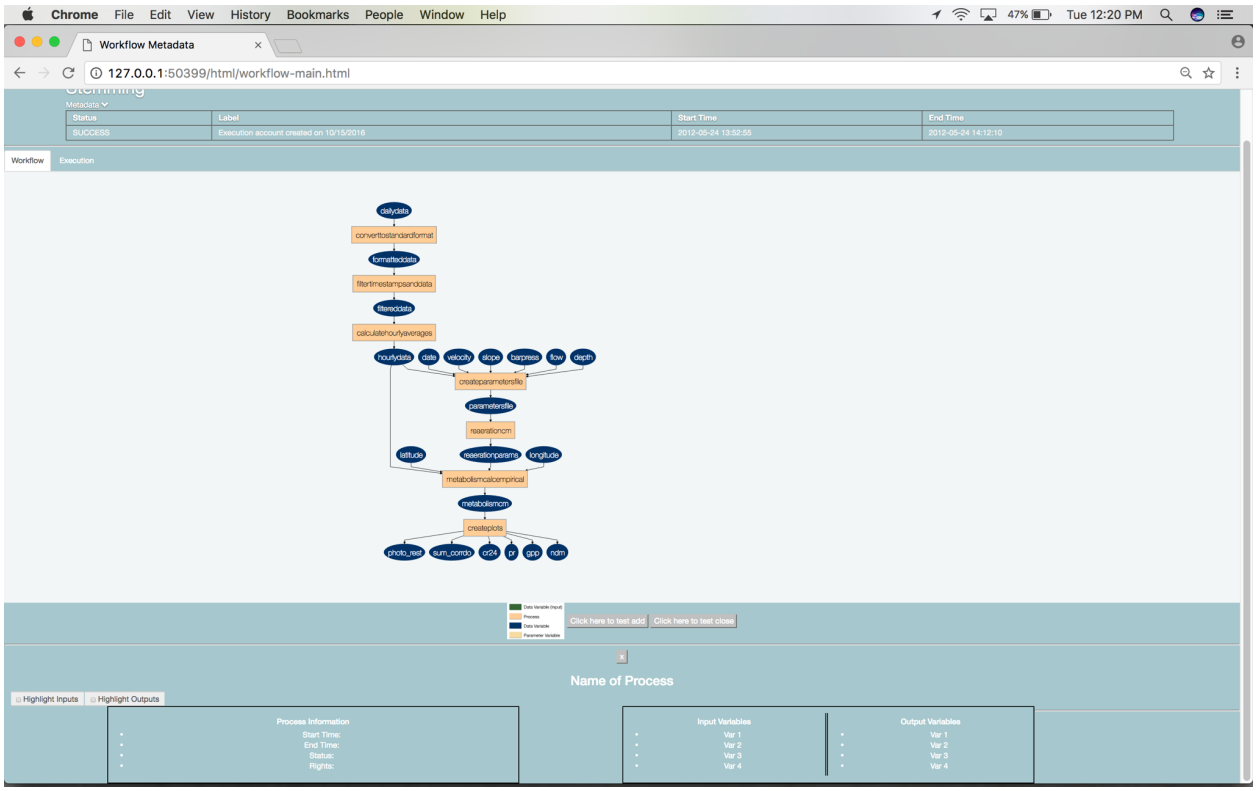
Current State:

The left section of the above image represents the Aquaflow_CM workflow diagram generated by our d3.js visualization logic. The graphic is generated by using a locally pre-loaded .json file with a JSON blob generated by a SPARQL query. We have written logic to take the .json file's data and process it within a file called 'visualization.js' and populate the graph by injecting the image into a DOM element called <svg>. The graph is lightly styled as far as shape and color in order to distinguish between processes and data, but no further styling has been done yet. On the right, is the original graph from our stakeholder's previous solution. The structure of both visualizations match, marking a success in our logic to generate the image that is similar to the original.

Goals for next deliverable:

The main goal is to start loading the visualization dynamically, as a result of an AJAX call, rather than loading in a pre-populated .json file. This step is dependent on receiving information from the search bar from the landing page. Furthermore, we wish to add on click listeners to each node within the visualization so that information about that node can be shown beneath the image. Additionally, we would like to add some additional styling to the visualization to aid end users in further distinguishing between nodes.

Workflow Main Page



Current State:

Javascript for the process information section at the bottom of the page is in the process of being added. Currently, clicking on any node in the graph (or the test button at the bottom) will insert a new section containing the process information. The second test button is connected to a Javascript function that will remove a section from the page. A legend image from the graph has also been added (this is a static image that will remain the same for every workflow page).

Goals for next deliverable:

In order to have the process information section have relevant data, the Javascript function adding the new section will also make use of SPARQL queries to populate the information sections. Test buttons will be removed once the correct graph clicking functionality is connected. The metadata section at the top of the page is to be moved to the executions tab once it is implemented. Javascript for the legend should be added to allow the user to hide and show the legend to avoid taking up extra space on the page. Finally, the overall UI styling will continue to be touched up.