User Journey Tool - Tagging Design

Context

As the user journey tool approached a MVP state, we coordinated with various stakeholders to discuss additional features or potential use cases that could be valuable to integrate into the UJT. A few ideas included:

1. Support for different types of dependencies. A dependency is currently a generalized connection between graph nodes, which doesn't carry semantic information regarding it's type. We could add support to indicate RPCs, async tasks, queues, database actions, etc.
2. Support for viewing non-functional dependencies, such as those to loggin services. These could be hidden by default but displayed upon request by the user.
3. Differentiating between request priority/criticality levels.
4. Introducing the ideas of flows, a set of dependencies in the graph that could be (a) conditionally taken within a single user journey (microvideo upload) or (b) shared between multiple user journeys (user authentication)

Solution

To flexibly address many of the situations above, we propose adding the notion of tagging for Nodes, Clients, and Dependencies. This feature allows us to include support for a variety of semantics in the dependency topology of a system, at the cost of a potentially less-intuitive UX compared to first-class support of each individual case.

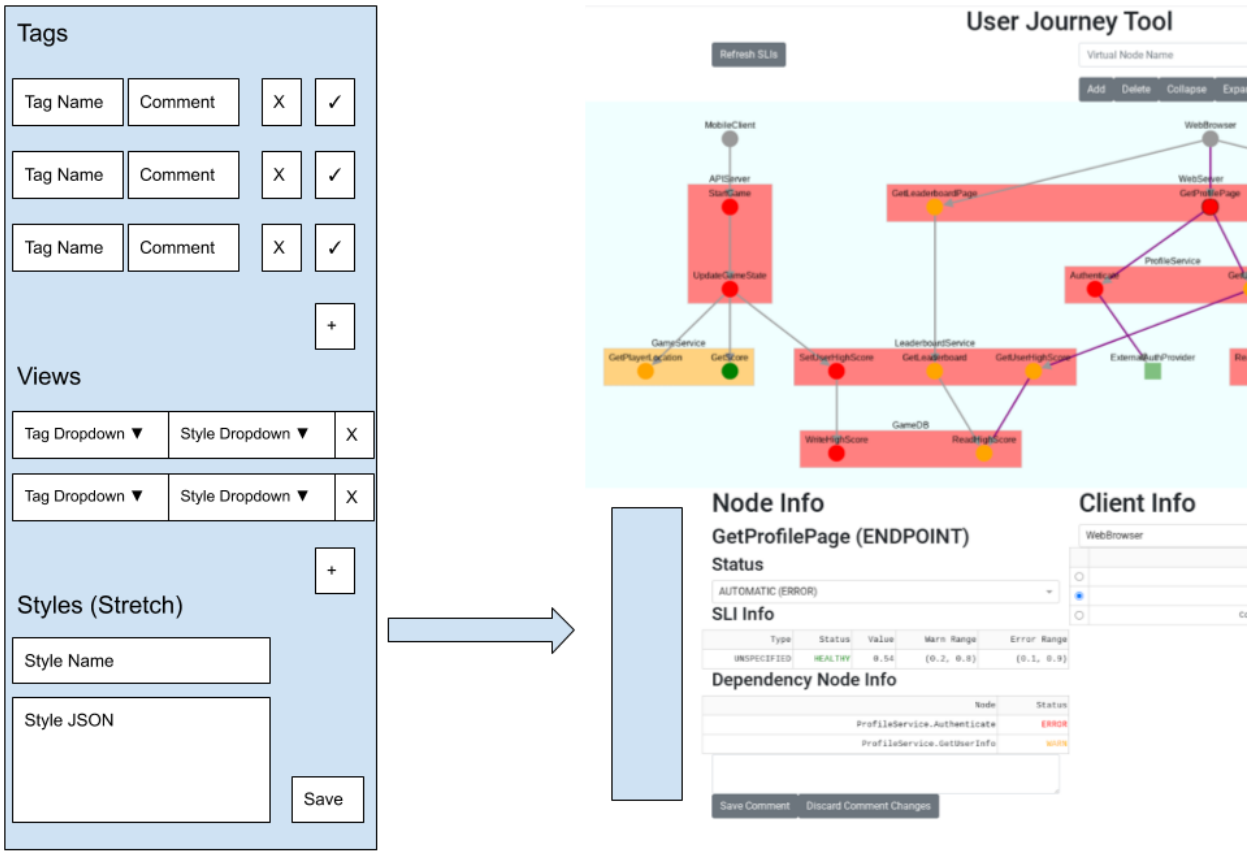
Tags will be persisted on the reporting server. To avoid confusion, tags are also only applied to an individual node and not propagated down to its children. Moreover, tags will be merely a visual update to the graph and will not support status computation operations (e.g. viewing aggregate status for a tag that represents a flow).

As such, we expect to use tags to address cases 1-3 above. It's still to be determined if we will add first-class support for flows, pending additional design decisions regarding changes to the user journey specification/representation.

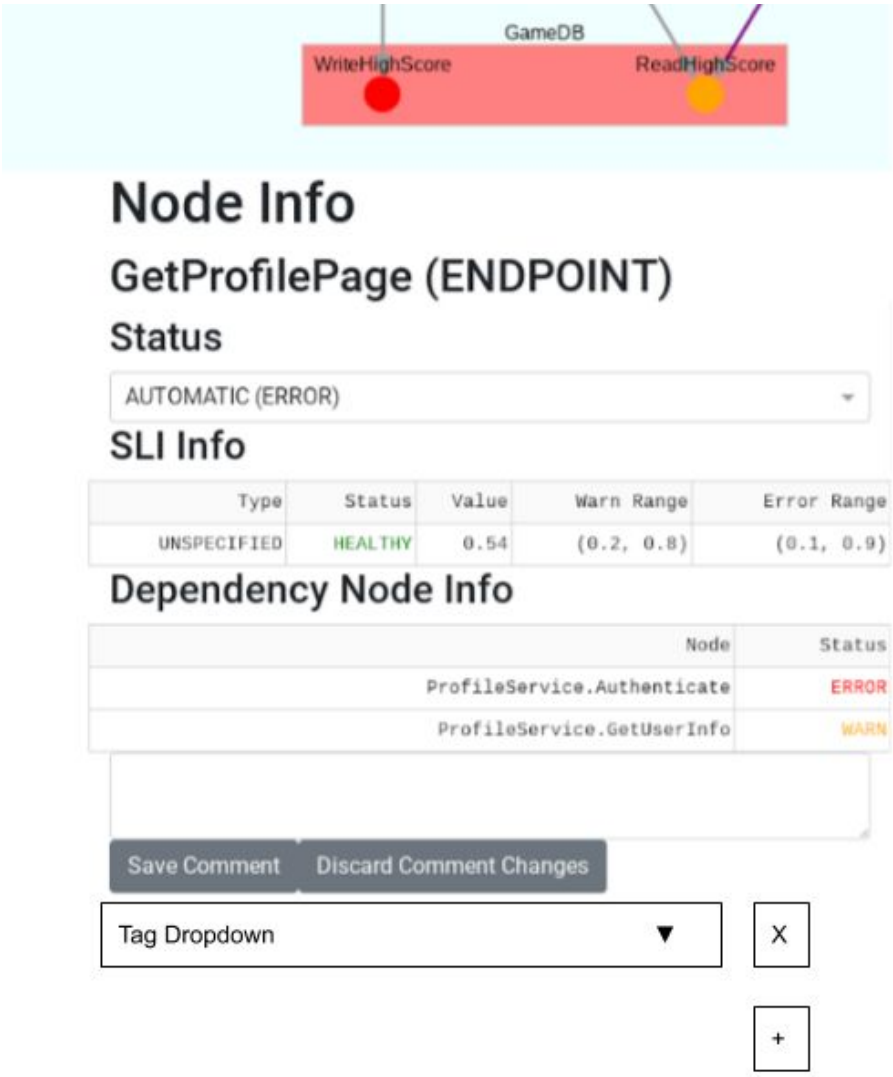
We might also want to consider refactoring the Status and NodeType fields of Nodes into tags. Not sure if this will increase or reduce complexity.

Tag Operations

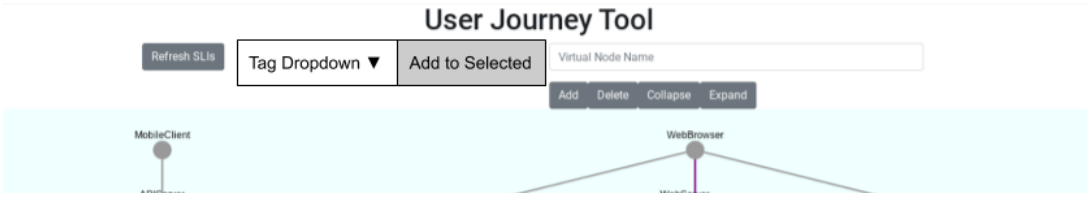
I plan on adding another panel to create and delete tags.



In order to add and remove tags from graph elements, we add dropdown components in their respective info panels. I plan on combining the node, client, and dependency information panels into a single panel. I'll convert the right panel to a UserJourney explorer panel that allows selection of user journey by Client, Node, and tag.



We also add a button to select multiple graph elements and apply a tag in batch.



We use dropdowns wherever we apply tags to ensure that typos aren't an issue. We can make these dropdowns searchable to make it easier to find specific tags. However, this comes as the cost of being able to dynamically create new tags by adding them to an element.

Tag Styling

Built-in tag styles will include:

- Hidden - all graph elements with the tag are hidden.
    - This style requires first class support, since we will need to support the case where the source or target of an edge are hidden.
    - During implementation, we found that cytoscape supports a hidden style for the elements. This automatically hides associated edges. However, when the hidden style is removed, the hidden edges don't reappear. This seems to be a cytoscape bug.
- Highlighted - all graph elements with the tag are highlighted
    - Still TBD exact highlighting mechanism, candidates include changing: outline thickness, outline color, background color shade, label color, etc.
    - Alternatively, instead of a single highlighted style, we can split these up into builtin color and boldness styles.

Styles will be persisted on the reporting server, but specific views will not.

Custom Styling Functionality (Stretch goal - Implemented!)

Cytoscape supports a variety of styling choices. It's probably infeasible to add 1:1 support for all the options. For advanced users, we can consider allowing them to input a stringified (JSON) dictionary of properties to add to the cytoscape stylesheet via the side panel. I believe loading JSON isn't a security vulnerability.

Multi-Tag Selection Functionality (Stretch goal?)

Another feature we might want to consider is the ability to combine tags with logical operators for a more fine grained view filter. For instance, we could support filters such as "tagA && tagB" or "tagA || ~tagB". If this proves to be useful in many cases, we could add this. We would probably modify the tag dropdown on the side panel to a freeform text input field, and parse the input string.

Persisting View Presets (Stretch goal?)

We may want to introduce the notion of a view preset, which could be represented as a set of views that are persisted on the reporting server. Users would be able to select a view preset to modify their view of the graph, and modify/save/remove presets.