CEng 445 Fall 2024 Projects Phase 3 Description

In phase 3, you are going to convert your phase 2 servis into a web based service.

The web application will be a static one. You will write a Django application to generate the HTML content on the browser. In the backend, it is going to connect to the phase 2 server. Since Django does not keep session open by default. It is going to run a,

- Connect TCP server
- Attach an object (if operation is on an object)
- send command
- Get result
- Convert to HTML
- detach/disconnect

loop.

In this phase there will be no notifications. You can suppress notifications on the server side or read and ignore them. Model updates by other connections will be accessible via *refresh* links. You can set auto refresh with 3-5 second intervals in simple Javascript environment.

Primary purpose of this phase is to convert your model into HTML, convert all your user commands into HTML forms. You can use simple browser scripts to draw the model, and filling a form from a mouse location, but Javascript should not be involved in the business logic yet. Use SVG, tabular data and images instead of text lists.

All commands in the phase 2 should be converted into web forms. Add CRUD like operations should be POST operations on forms. If you feel the form will not be useful in the last phase, make it as simple as possible.

The topic specific descriptions of Phase 4 are given below.

Race Map

It is time to associate views of components with images. Define image ids as the views of the components. Than make web application subsitute them with `<img>` tags to actual `.png` or similar format images. You can use `transform:rotate(angle)` CSS function to make browser rotate the components. The grid can be simply drawn as a table with 0 margins and padding, or you can use SVG.

There will be no race mode in this phase, showing the edit facilities are sufficient.

Web Dash

The `Dash`, `Tab` and components will be rendered as HTML in the browser window as expected. You can use simple Javascript/JQuery UI to switch between tabs. You can use tables or `flex/flexbox` properties to position components.

You need to render parameters as forms and form posts will be send as the triger events. There will be no notification. You need to press a refresh button to reload the attached `Dash`.

Graph Data

There are different options to render a graph in a web page. There are libraries like **Joint.js** for intereactive graphs. Your graph does not have to be interactive. You can do one of: 1. layout nodes in a `SVG` element with `x y` coordinates. `<foreignObject>` elements can include text based html elements. 1. Create a `relative` positioned large `<div>` element and make your node `<div>` elements as `absolute`. By using CSS `top` and `left`, you can place your nodes on in any position.

or a more clever solution you can come up with.

You do not have to draw the lines between the components. You can use same colored ports to indicate the connection. In SVG, it is possible to add `line` objects by using relative position of ports in the nodes by using Javascript, but it is optional.

In this phase, you will use HTML tables in `Viewer` components. You need to show the `params` dictionary as an HTML form (inside of nodes view) and let user submit them to update the parameters. Link modifications can be done in classical dropdown HTML components.