Testing-Strategy

Testing Hooks:

Testing hooks will be implemented by having additional testing interfaces in conjunction with normally implemented interfaces. These additional interfaces will only be available to the testing team, and will aid in them writing deep probing test cases.

We will use this functionality to:

- Inject additional errors
- Break typing contracts
- Mock broken functions

Logging:

Logging will be implemented at a few different levels of granularity to help track down bugs. During development we will run tests with full logging. Final builds will only have logging enabled at a shallow level. This will be done with a JS logging library such as log4javascript.

User Interface Testing:

Due to the nature of single-page web applications, UI components may need to be re-rendered multiple times to reflect changes that the user has made or interactions that they have with the system. In the case of this project, the user will be able to manipulate how the data is visualized as well as interact with various tools. We will be using Jest's snapshot testing functionality to ensure that these components re-render correctly after they are triggered by specific DOM events. Some snapshot tests may also be a part of the smoke test.

We will also be using Enzyme in tandem with Jest to provide additional testing functionality, one of which is to simulate specific events and user interactions. Enzyme provides many other methods that simulate traversing/interacting the UI much easier for testing.

Use of Assertions:

Unfortunately, javascript currently does not have an assert function. To work around this we will be making use of the expect() function, which is a part of the Jest Testing Framework. The expect() function is used in combination with various "matchers" (eg. toEqual(value), toHaveReturnedWith(value), etc.) to validate different things.

Assertions will be written by programmers at the time of implementation. Any assumptions which a developer believes requires a comment probably requires an assertion. This will require if statements to check if we are on a development branch or a release branch.

Mocks:

Mocks will be implemented for any stub which is deemed of sufficient difficulty to implement, or for functions which cut across many concerns. This will allow for us to test the implementation of dependant functions more easily.

Addressing The Separation of Concerns:

This should be addressed by an MVC architecture.