JavaScript Testing

In this assignment, you will write unit tests for a basic React app - download link. The unit tests you will write will test two units:

- Colorize functionality
- <Weather> component

The unit tests should ensure that these units behave exactly the way their documentation contracts specify. The contracts are written in code comments in the respective modules where these units are written (`Colorize.js` and `Weather.js`). The contracts are completely unambiguous from my perspective, so if you have *any* questions whatsoever on how they are supposed to behave - immediately ask on Piazza.

Instructions

The only file in the project you are allowed to modify is `App.test.js` - your unit tests should go in there.

Your tests should be well organized, using "describe" to group tests together when it makes sense, and your test cases should have descriptive names of what they are testing. Example test names:

- Bad - "tests that colorize returns the right color"
- Good - "returns 'orange' for a warm temperature value"

Keep your test cases small and specific.

Since these are units tests, the test cases should be independently testing a unit. What this means in practice is that if I introduce a bug into colorize, this should not affect your tests for <Weather>, and vice-versa. And the same goes for TemperatureDataSource. I will break the data source to simulate an API or database going down, and this should not affect your test cases. This can be achieved by using mocking.

The usual submission instructions apply for both the homework assignment and extra credit. The only difference is that for this assignment, I will run `npm test` instead of `npm start`. Assignments that do not run will receive 0 points.

Grading

The grade will be determined based on how well your tests catch bugs that I will introduce into the app. Some examples of bugs I will introduce are:

- Forgetting to call certain functions required to fulfill the functionality of <Weather> or colorize.
- Modify logic to break the contract - like returning 'purple' for hot weather.
- Introduce bugs into places like colorize and TemperatureDataSource in hopes of breaking tests written for a different unit like <Weather>.
- Show the temperature value using different types of DOM elements that show text. (In this case, your tests should still pass as long as the temperature value text is displayed to the user.)

Helpful Reading

Mock Object

Jest - Using matchers

Jest - Mock functions

Jest - Bypassing module mocks

React-testing-library - Cheatsheet

React-testing-library - Async Testing

Extra Credit

+2 points

Write an end-to-end test that runs in an actual web browser using webdriverio. Use the "Getting Started" documentation to set this up. This should be done as a separate npm project and separate submission. It should include a README file telling me how to run the test.

The end-to-end test should not modify the homework project functionality at all. The test needs only one test case: that the temperature value "73" eventually displays in the CSS color 'orange'.