Weather Application

(https://infinite-plains-69610.herokuapp.com/)

(Source code: https://github.com/shraddhaNeeraj/WeatherApp)

Design Decisions:

I noticed that the Dark Sky API requires latitude and longitude of a place to send back a weather report of that place.

Since coordinates are cumbersome for users to enter, I decided to make an input field in which the user would enter the zipcode of the place (changes detected onBlur) they are interested to know the weather details of. I used the zipcode API (https://www.zipcodeapi.com/) to acquire the city, state, latitude and longitude corresponding to that zipcode. Once the latitude and longitude were available, I could use them to make a Dark Sky API call for that moment (current timestamp) to get the weather details of that place.

Note: The free version of zipcodeapi allows only 10 requests from authorized clients (heroku and my localhost) per hour. So the app may not work after the first 10 requests in any given hour.

The react component hierarchy can be demonstrated with the following diagram:

<table>
<tr><td></td><td>App Component</td><td></td></tr>
<tr><td></td><td>WeatherReport Component</td><td></td></tr>
<tr><td>HourlyWeather Component
(renders a set of HourlyWeatherInfo components)</td><td>DailyWeather Component
(renders a set of DailyWeatherInfo components)</td><td>Alerts Component</td></tr>
</table>

1. App Component is the root component, which starts by rendering the WeatherReport Component.
2. WeatherReport Component displays a zipcode input field and fetches location and weather information for the provided zipcode.
3. The data collected by asyc API calls is passed off to a service called “WeatherReportService'' that abstracts the logic for extraction of useful information that will be eventually rendered in the UI.
4. WeatherReport Component conditionally renders “HourlyWeather”, “Daily Weather” and “Alerts” components that work on the “hourly”, “daily” and “alerts” sections of the API response.
5. HourlyWeather and DailyWeather display the hourly and daily weather summaries respectively and go on to render child components to display weather details for each hour and each day respectively.
6. HourlyWeatherInfo and DailyWeatherInfo components display relevant weather information for given hour and given day respectively.
7. I have made tabs to navigate between hourly and daily weather sections.
8. Installed “weather icons” package to render appropriate icons. Maintained iconMappings enum to translate the icon style returned by the API into corresponding styles provided by that package.

What else would I have done if I had more time and were deploying to production ?

There are a lot of opportunities to improve over the current weather app as it was developed within the given time limit of four hours.

1. I have focussed more on the functionality rather than the UI styles and could have spent more time beautifying the app.
2. I would have liked to push the AJAX calls from WeatherReport to the WeatherReportService, that would have returned back a Promise (any other async equivalent) so as to make the component agnostic of the urls and any other information required to make the calls. The service could handle it all. The component would simply update its state when the Promise is resolved.

3. As of now, HourlyWeather and DailyWeather Components literally duplicate each other. So I could have used just one component and then conditionally rendered *HourlyWeatherInfo and DailyWeatherInfo* from that same component. (<ng-template> can be used in angular to render different types of child components based on type of input data. I wanted to research equivalent patterns in React to do so gracefully.)
   The idea behind keeping those two components separate to begin with, was to display information differently in both of them. But for now, their layouts are exactly the same.)
4. My unit test coverage is not 100% right now and would have written more and meaningful tests if I had more time. I have left some TODOs in code as well. I would have liked to explore Jest and Enzyme a little more to be able to use them in writing tests.
5. I would have added more illegal data checks in code and test cases for them eg: what if alert component gets null data, etc.
6. I would have handled rendering of alerts more nicely. Currently, alerts are rendered as huge blobs of data.
7. I would have liked to not incorporate styles in the JSX, but write separate methods to get the styles so that the logic to determine styles could be independently tested.
8. I work with Observables and Subjects a lot in Angular, and would have liked to use them (or any more preferred ways for React) to handle async calls rather than plain AJAX provided by jQuery.