Bottle Rocket Web Test

We are excited to have you complete this web engineering take home test! The purpose of this is to gauge how you would approach a typical web project here at Bottle Rocket. Some of the things we are looking for are: good code that is structured and readable, responsiveness, and that you can work within the constraints of project requirements and translate mock ups into clean UI.

What are you building?

You'll be building a sample weather app, aptly called, "BR Weather." Your app will grab a list of cities from our weather API where you will display the current, daily, and hourly conditions, as well as display an animated gif of the local radar.

What's included?

This app has been *partially implemented* for you so you can dive right into building out features! This project was bootstrapped with Create React App, along with Sass, React Router, and a couple provided components to get you started.

Mockups

You will find a PDF of the annotated mockup of the design in this repo under the `./src/designs` directory.

Note: A blurred element sits above the image of each city. If you've never used the css blur function before, please refer to the documentation. You may use this inconjunction with the `background-color` property.

Font

The font used throughout the design is "Open Sans", which you can find at Google Fonts and include in the `/public/index.html` file.

Requirements

Please be sure to meet all the minimum requirements for this take home test.

- Use the provided mockups as a reference for the layout, font, styling, and colors for each screen.
- Mobile first design with responsive breakpoints:
    - 375px (min) - 744px for small mobile screens
    - 744px - 1200px (max) for larger tablet screens
- Please use Sass for styling. We have provided a `/styles/` directory which includes a `main.scss` file and the predefined breakpoints from a `_variables.scss` file for you:

```
// Small mobile devices
$screen-sm-min: 23.4375rem; // 375px

// Small tablets
$screen-md-min: 48.375rem; // 744px

// Large tablets
$screen-lg-min: 75rem; // 1200px
```

You can use the `@use` directive to include them in your css files:

```
@use '_variables.scss';

@media screen and (min-width: variables.$screen-md-min) {
  width: variables.$screen-md-min;
}
```

- Use the BR Weather API to asynchronously fetch and parse the weather data.
- Display the, "Loading" message and animated spinner while waiting for **any** API response.
- Display the, "No Results" error message if a city's individual data is not returned, the API is currently unavailable, or there are no results.
- Use the provided component `<Icon />` for weather and navigation icons.
- You can stick with one browser (Chrome, Firefox, Safari, or Edge) to test in, but please let us know which one so we can see your work as you intended!

Home Screen Requirements

- Load the following 4 default cities' weather data in their order by using their associated `id` value. Please refer to the Weather API section on how to query for this data, including how to request the correct `icon_mode` to be used with our `<Icon />` component.

  Default Cities

  1. Austin: `F52083D2-841C-4E1A-A4C9-83827B07D8EE`
  2. Boston: `1B4DBE6A-EB69-4357-8BE8-1968DD9ECDF3`
  3. Dallas: `0B66D5E5-039F-4951-A63A-2693464617CB`
  4. Nashville: `54A0CBA3-F4F8-47B8-974C-7FCE641CAD2C`
- Each city's weather data should be displayed in individual slides in a carousel that has a sliding animation.
  - On initial load, the URL should be that of the first city, e.g. `/austin`. When the slide changes to a new city, update the URL.
- Display a max of 10 days from the daily forecast results in a horizontal scroll with the first day highlighted.
  - If the screen size is less than or equal to 390px, only show 4 days.

    - When a user selects a day in the "Weekly Forecast" highlight the selected day and display the sun/moon rise and set data.
- Display a max of 16 hours in the "Hourly Forecast" in a horizontal scroll with the most recent hour highlighted.
    - When a user selects a time under "Hourly Forecast" highlight the selected hour and update the details below it.
    - If the screen size is less than or equal to 390px, only show 8 hours.
- The "Local Radar" button should transition to the Radar Screen screen with the URL being, `/{city}/radar`
- Selecting the trash icon will remove the current viewable city from the list. ***You do not need to maintain this state when refreshing the app!***
    - If all cities are removed show the, "No Results" state.

Radar Screen Requirements

- Display the radar image returned from the API for the current city.
- Selecting the 'Back Arrow' icon will take the user back to the current city with an updated URL of `/{city}`.

Submitting Your Test

Once you are finished, please provide a .zip file of your project (**without `node_modules`**) and send it back to the recruiter with whom you are working.

We ask that you please do not post or share your work on Github or any other publicly accessible site.

Also, don't forget to let us know which browser you tested in!

BR Weather API

The API is hosted on Heroku and may need time to "warm up" and become available. Typically less than a minute.

API Documentation: https://bottlerocketstudios.stoplight.io/docs/all-the-weather/YXBpOjM4NDY0Njl0-all-the-weather.

You can view all the available endpoints and associated schemas, as well as making sample or production requests.

- Base URL: https://all-the-weather.herokuapp.com/
- The APIs you should focus on using are below. Here is where you would pass in the city `id` value from the default cities.
    - `/cities/{id}/forecast`
    - `/forecasts/{id}/radar`
- Be sure to pass along the `icon_mode=name` parameter in order to use the provided `<Icon />` component properly! Available query parameters.

Provided Components and Helpers

We have provided a number of available components to help you.

<Router />

We have included a small scaffold of React Router (v6) which will be your starting point. You can find this in, `./src/components/App/App.jsx`.

Some of the routes have already been provided, including their respective components. You don't need to worry about a 404 route.

```
<Routes>
  <Route
    path="/"
    element={
      <Layout>
        <>Shared Header component</>
      </Layout>
    }
  >
    {['/', '/:city'].map((path) => (
      <Route index path={path} element={<Home />} />
    ))}
    <Route path=":city/radar" element={<Radar />} />
  </Route>
</Routes>
```

<Icon />

It won't be sunny everyday, so we have provided all the necessary weather and navigation icons for you to use. You can find all the available icons in `./src/components/Icons`.

Props

- `name`: (required)
    - type: `string`
- `color` (optional) | Hexidecimal value to be applied to the stroke or fill property of the svg.
    - type: `string` | Default = `#000000`

Weather Icons: You will need to pass the `icon` value found in the `weather` object from the `current, daily, hourly` objects in order to show the correct icon. **Remember to pass in the `icon_mode=name` parameter to the API.**

```
"current":  {
  "humidity":  48,
  "uvIndex":  6.86,
  "visibility":  10000,
```

```
      "weather":  {
        "icon":  "01d",
        "description":  "Clear"
      },
    "timestamp":  "2022-03-03T18:34:56Z",
    ...
  }

  // usage
  <Icon name={'01d'} color={'#ff0000'} />
```

Common Icons: `br-logo`, `search`, `trash`, `loading`, `no-results`, `close`, `sun`, `moon`, `arrow-left`, `arrow-right`, `precipitation`

```
  <Icon name={'search'} color={'#c1c1c1'} />

  <Icon name={'trash'} color={'#e1e1e1'} />
```

Helper Function

You will need to convert the wind direction value in the hourly forecast from degrees to compass direction: `NE`, `NW`, `SE`, `SW`, etc. We have provided you an exported `convertWindDirection` function in the `./src/utils` directory for you to use.

```
  // usage

  convertWindDirection(220); // returns 'SW'
```

Bonus features

If you feel confident that you have successfully completed all the minimum requirements, you are welcome to add your own creativity. When you submit your exercise, let us know what additional things you added and why.

Running and building your app

If you are unfamiliar with Create React App, these are the commands that you will need to use for development and building your app. You can find additional information about CRA scripts here.

`npm install`

Installs all the necessary dependencies.

`npm start`

Runs the app in the development mode.

Open http://localhost:3000 to view it in your browser.

The page will reload when you make changes.

You may also see any lint errors in the console.

`npm run build`

Builds the app for production to the `build` folder.

It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.

Your app is ready to be deployed!