1 Project 5

Due: Dec 4 by 11:59p **No extensions or late submissions**

Important Reminder: As per the course *Academic Honesty Statement*, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

This document first provides the aims of this project. It then lists the requirements as explicitly as possible. It then provides some background information. Finally, it provides some hints as to how those requirements can be met.

1.1 Aims

The aims of this project are as follows:

- To give you experience with programming within the browser.
- To give you some familiarity with using the react.js library.
- To expose you to using a simple bundling tool.

1.2 Virtual Machine Tips

If you are doing this project on your VM:

- Since your VM is not accessible except via a ssh-tunnel, you will need to run a browser locally on your VM and point it to `http://localhost:`*PORT* where *PORT* is the port on which you are running your parcel development server.
- If you experience issues with your VM, please follow the instructions for *Restarting your VM* in the *Course VM Setup* document.

1.3 Requirements

You must check in a `submit/prj5-sol` directory in your gitlab project such that typing `npm ci` within that directory followed by `npm start` is sufficient to start up a server running on port 2347.

Accessing that port using a browser should display a **setup page** which contains a form which allows the user to specify the URL at which the sensors web services are running (this should default to `http://zdu.binghamton.edu:2345`).

Submitting the form on the setup page should display an **app page**. This app page should constitute a *single-page app* which will allow the user to access 4 tabs. These 4 tabs should have the same functionality as the **Sensor Types**

Search, **Sensor Types Add**, **Sensors Search** and **Sensors Add** pages from the previous project except:

- Blurring a form field should result in validating the value in that field. This includes validating that the model number provided for a sensor specifies a valid sensor type id.
- Blurring any form field on a search page should lead to the search results being updated as per the specified fields, provided there are no errors.

  Your solution is also subject to the following implementation constraints:

  - You must use reactjs for setting up your UI.
  - You must use *Project 3* web-services (possibly modified) directly from within the browser.
  - You must use axios for calling the web services.

1.4 Underlying Web Services

You will need a server running the underlying web services at the URL specified on the set up page. Alternatives:

1. Deploy your solution to *Project 3*.
2. Deploy the *provided solution* to *Project 3*.
3. Use the solution to *Project 3* running on *<http://zdu.binghamton.edu:2345>*. This site is accessible only from within the CS department network.

1.5 Provided Files

The prj5-sol directory contains a start for your project. Some of the files provided include:

sensors-setup.html An entry HTML page which allows you to specify the URL at which *Project 3 Web Services* are running.

sensors-app.html The only HTML page for your single-page app.

sensors-app.js The top-level JavaScript loaded by your app.

sensors-ws.js A wrapper which calls the *Project 3 Web Services*.

components/app.jsx The top-level `app` React component.

components/tab.jsx The `tab` React component used for displaying tabs.

styles CSS style sheets included by `sensors-app.html` to give the app a reasonable look-and-feel.

You may change any of the above as needed.

1.6 Hints

The following steps are not prescriptive in that you may choose to ignore them as long as you meet all project requirements.

1. Read the project requirements thoroughly. Understand the *users app* covered in class.

2. Start your project by creating a `work/prj5-sol` directory. Change into that directory and initialize your project by running `npm init -y`. This will create a `package.json` file; this file should be committed to your repository.

3. Install necessary dependencies:

   ```
   $ npm install --save-dev parcel-bundler \
        '@babel/plugin-transform-runtime'
   $ npm install react axios
   ```

   The first command installs development dependencies which includes the parcel bundler. Additionally, we need `@babel/plugin-transform-runtime` to translate `async`/`await`.

   The second command installs runtime dependencies.

4. Copy in the files you have been provided with:

   ```
   $ cp -r ~/cs544/projects/prj5/prj5-sol/.gitignore \
           ~/cs544/projects/prj5/prj5-sol/.babelrc \
   ~/cs544/projects/prj5/prj5-sol/* .
   ```

   The `.gitignore` is set up to ignore `node_modules` as well as the `.cache` and `dist` directories created by parcel.

   The `.babelrc` file is used for initializing `babel`.

5. Add the following `start` script to the `scripts` section of your `package¬.json` file:

   ```
   "start": "parcel sensors-setup.html  --port 2347"
   ```

6. You should be able to start the parcel development server:

   ```
     $ npm start
   ```

   If you get errors like

   ```
     cannot read property 'type' of undefined
   ```

   and the server hangs, `^C` out of it and start the server once more; that should work.

7. Point your browser to port 2347 on your host. You should see the sensors app load with four empty tabs.

8. Add components for the four tabs to meet all project requirements.

It is a good idea to commit and push your project periodically whenever you have made significant changes. When it is complete please follow the procedure given in the *github setup directions* to submit your project using the `submit` directory.