1 Project 5

Due: May 10 by 11:59p **No 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 install` 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 URL shortening 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 shorten

URLs in arbitrary text and display information about specified long/short urls. Specifically, the app page should display 2 tabs:

Shortener tab This tab should display a form which allows the user to type in arbitrary text which may be of any length. When the form is submitted, it should redisplay the tab with the user input retained. Additionally, it should display the user input elsewhere on the page with all URLs in the text replaced by HTML links to the corresponding short URLs returned by the shortener web service.

Info tab This tab should display a form containing an input widget which allows the user to specify a URL. When the form is submitted or the input widget looses focus, the tab should redisplay with information about the input url which includes:

- The short URL.
- The long URL.
- A count of the number of times the short or long URL have been accessed in the underlying web services.
- The activation status of the URL (this will always be **true** in this project).

Form submission should check for errors. When errors are detected, the action for the form submission should not be taken. Instead the form should be redisplayed **with all user input retained** and suitable error messages displayed.

Any styling of the content is acceptable as long as the content meets the functionality described above.

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 Example Operation

A running version of this project is available at *Shortener Set up*. As usual, this site is accessible only from within the CS department network. Note that since the underlying web services only run on `http`, `https` redirects are not handled.

1.5 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>*. Note that the underlying database is cleared every hour. This site is accessible only from within the CS department network.

The solution provided for *Project 3* has an additional service running at `/x-subst` which returns short url substitutions. Specifically, it expects a JSON body with property `text` giving the text which contains zero-or-more url's to be shortened. It returns a JSON response with a property `value` which is a list of substitutions for the url's found in text. Each substitution is a JSON object having the following properties:

`lo` The inclusive starting index of a url in `text` (the first character of the url is at `text[lo]`).

`hi` The exclusive ending index for the above url (the last character of the url is at `text[hi - 1]`).

`shortUrl` The short url corresponding to the `text` at indexes `lo` (inclusive) to `hi` (exclusive).

Note that unlike the `x-text` service, this service increments the counts for every url which is shortened.

1.6 Provided Files

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

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

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

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

shortener-ws.js A wrapper which calls the *Project 3 Web Services*. Note that the `subst()` function provides an interface to the additional substitute service documented above.

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 `shortener-app.html` to give the app a reasonable look-and-feel.

1.7 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. Look at the sample *sample web site*. 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 the `.cache` and `dist` directories created by parcel (the `node_modules` directory should already be ignored via a `.gitignore` in an ancestor directory).

   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 shortener-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 shortener app load with two empty tabs.

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

   [When displaying the shortened results for the text shortening tab, react strongly discourages the insertion of raw HTML (the *DOM Elements* workaround is called dangerouslySetInnerHTML!). The newly added substitute service returns only a mapping from the input text to the short urls, allowing you to display the shortened text as a sequence of separate jsx components.]

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 *gitlab setup directions* to submit your project using the `submit` directory.