Developing the *Wireframer* App

HW 2: 2nd Technology Learning Assignment - Front-End APIs like React and Making and API

In this assignment we'll again make the very same Todo List application but this time instead of just using plaing JavaScript we'll make use of ReactJS, a front-end API created by Facebook that makes creating views for Web applications easier. Note that there's a lot to learn concerning React, so to start, watch the React JS Crash Course - 2019 video on YouTube. This is an excellent 98 minute tutorial that will help you understand the full setup of our project. Note that there are many other resources online for using React, including the W3Schools' React Tutorial.

Note that in this assignment, I want you to first go through the process of setting up a Create React App project using Visual Studio Code. To get @todo up an running do the following:

- Download and install Node.js. This tool provides a platform for other tools.
- Download and install npm. This tool helps retrieve other development packages, making project setup easier.
- Optionally install yarn. Note that this is a tool like npm that can alternatively be used. You may use either npm or yarn for things like starting React builds, it is up to you.

- Download and install **React Developer Tools**. This will help with things like debugging.

- Create a local directory called **sandbox_hw2** on your computer where you will do a bit of learning for HW 2.

- Open Visual Studio Code and Open the **sandbox_hw2** directory. In this tool, click on the **Terminal** menu and choose **New Terminal**. This should open and dock a Terminal window to the bottom of your VSC window. This is where you'll be able to enter all necessary commands for starting your react builds and servers.

- Install the Create React App starting project by go to the Terminal and typing **npx create-react-app .**, note there is a dot at the end of the command which means here (i.e. this directory), which is where the initial project will be installed. Once you have this installed you'll be able to see how all the pieces fit together for your basic React app. Note the YouTube video I've listed above uses this basic app as the starting point.
- Start the basic app by going to the VSC Terminal and running the appropriate command in either yarn (i.e. **yarn run start**) or in npm (i.e. **npm start**). These commands both compile the React project and create a runnable version, then start a Web server that serves the content that will be served to a browser on a local URL. Note that your default browser will automatically be opened and any updates that you make to your program will be reloaded each time you save your changes.

  RECOMMENDATION: while you are working, I strongly recommend that you do so with a split screen where Visual Studio Code takes up the left half of the screen and the browser you are using to test your app takes up the right. Once you start the react server, every time you edit code in the left and save, it will immediately be reflected in the browser window on the right. This makes for much easier developer experience with instant feedback regarding errors as you type and save continusously. In addition, I ***strongly*** recommend that you use the **React Components** tab in the Firefox Debugging window. This will let you see what all the state data is for all of your components,

which will make it much easier for you to figure out what is going on a track down problems.

- Download/Clone the **todo_hw2** project. Put the full project into a directory you have chosen to work. Note that you are required to use **Git** for the duration of this assignment. You should make periodic commits with good comments as you make progress. Make sure you keep your repository **private**.

- Run the application and you'll find that it is a mess of a partially working TodoList similar to what we did in HW 1. It is your task to complete the project. Note that you may make changes to the project as you like but it ***must*** remain a React application. Be careful deciding how to manage the application's state in particular in concert with event handling.

Note that you may use any code you like from my mostly working solutions for HW 1. Of course you may also use any code you like from your own solutions.

jsTPS - porting our Transaction Processing Library

Note that you should create a second git project for jsTPS, which will be a JavaScript Transaction Processing System that we can use for some simple Undo/Redo in this assignment. In order to do this, start by porting jTPS to work in JavaScript. Note that you ***must*** provide simple tests identical to that provided inside jTPS, but such that it would run inside a Web browser. For this portion of the assignment the TA will want to run your tests in a browser window and note ***all*** tests should work as they do in the original library. For this, you may create a simple Web form for collecting and presenting user input.

todo_hw2

Complete the React application such that it works just like your project in HW 1 with one exception, control Z and control Y provide Undo/Redo features that work for the following edits:

- List Name Change

- Owner Name Change
- List Item Order Change (i.e. move up/down>
- List Item Removal
- List Item Edit (i.e. result of submit)

Note that deleting a list is *not* undoable.

Handin Instructions

When you are done, zip up the entire **todo_hw2** directory and submit that single ZIP file via **Blackboard**. Note, it *must* be a .zip file, *not* a .rar file. Make sure you provide the URL of your GitHub repository when you submit your file.

Grading

Note that grading will be based on program performance and will only be done by appointment with each student's prescribed Teaching Assistant. This part of your grade will be based on how well your program performs specific required tasks.

Web page created and maintained by