CMPS 356 – Fall 2022
Web Applications Design and Development

Lab 02
React Basics

Objective

- Practice using basic features of React to build a web application
- Use JSX to define user interface elements
- Define and render components
- Set and update component state and handle user-interface events
- Monitor state changes and apply side effects using hooks

Prerequisites

- React app creation utility: https://create-react-app.dev
- React tutorials: https://reactjs.org/tutorial and https://beta.reactjs.org/learn
- React Developer Tools browser extension: https://blog.openreplay.com/an-introduction-to-react-dev-tools
- Visual Studio Code: https://code.visualstudio.com

1. Hello, React!

- Under your private lab repository, create a directory for this lab and open it in Visual Studio Code.
- Create a directory 01-hello-react under the lab directory and use it for this exercise.
- Create an index.html file and, inside it, a Hello component that returns 'Hello, React!'
- Render the Hello component inside the root div. Your page content will be similar to the following code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CMPS 356</title>
    <script
      crossorigin
      src="https://unpkg.com/react/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom/umd/react-dom.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/@babel/standalone/babel.min.js"
    ></script>
    <script type="text/babel">
```

```
      const Hello = (props) => (
        <h1>
          Hello, React!
        </h1>
      );
      const root = ReactDOM.createRoot(document.querySelector('#root'));
      root.render(<Hello />);
    </script>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

- Extend the Hello component to accept a name parameter.
- Redo the above example by creating a new React app using the create-react-app utility.
- Explore the files and directories that are generated. What is part of this toolkit?
- Create a Hello component that returns "Hello, React!"
- Display the Hello component in index.js.
- Experiment with rendering a couple of elements using JSX, e.g., an unordered list of a few elements.
- Create a User component with first name and last name properties.
- Render two User instances.

First Name: John
Last Name: Doe
First Name: Jane
Last Name: Doe

Commented [AE1]: Provide a lot more details to guide the students to deliver the task. Provide UI of final app.

Commented [GY2R1]: done

2. A Clock with a Difference

- Create a directory 02-clock under the lab directory and use it for this exercise.
- Create a React application using the create-react-app utility.
- Design a Clock component that displays the current time. Use a state variable to keep track of the time.
- Update your component to keep the rendered time in sync with the current time
- Augment your component with an hour offset property and an associated state variable. This difference is used to shift the time by that number of hours, similarly to a time-zone difference.
- Add two buttons to increment and decrement, respectively, the hour offset by ½-hour steps.
- Log a message to the console with the current value every time the date or hour offset are updated.

8/30/2022, 4:13:09 PM
+ -
2-hour difference.

Commented [AE3]: Vague, lot more details needed + add illustrative UI

Commented [GY4R3]: done

3. Country Facts Application

- Explore and run the Country Facts application provided under base/country-facts that was developed using plain JavaScript.
- Recreate application using React and use at least two components.
- Fetch the list of continents when the application is first loaded.
- Fetch the list of countries whenever the continent is updated.
- Display the table of facts whenever a country is selected and update whenever the country selection changes.

Commented [AE5]: Add some illustrative screenshots of the app UI and further detailed steps to guide the students.

4. Todo List Application

- Explore and run the Todo List application provided under base/todo-list that was developed using plain JavaScript.
- Recreate the application using React and use at least two components.
- Load the list of todos from local storage when the application is first loaded.

Commented [AE6]: Add some illustrative screenshots of the app UI and further detailed steps to guide the students.