- ⏱ 4 hours

React Advanced Practice - Surveys

This project is designed to let you stretch your skills as a developer. Rather than a step-by-step guide, you will find a specification as if it came from a client you might be working for.

The goal is to allow you to prepare to show off your skills in these areas:

React Basics Learning Objectives

- Explain how React uses a tree data structure called the virtual DOM to model the DOM.
- Create virtual DOM nodes using JSX.
- Describe how JSX transforms into actual DOM nodes.
- Use the `ReactDOM.render` method to have React render your virtual DOM nodes under an actual DOM node.
- Attach an event listener to an actual DOM node using a virtual node.
- Use `create-react-app` to stand up a new React application and import needed assets.
- Construct a custom 'create-react-app' template and use it to start a new application.

React Component Learning Objectives

- Provide default values for a React component's props.
- Pass props into a React component.
- Use debugging tools to determine when a component is rendering.
- Destructure props.

React Router Learning Objectives

- Create routes using components from the `react-router-dom` package.
- Generate navigation links using components from the `react-router-dom` package.
- Use React Router params to access path variables.
- Use React Router history to programmatically change the browser's URL.
- Redirect users by using the `<Redirect>` component.

React Hooks Learning Objectives

- Use the `useState` hook to manage a component's state.
- Use the `useEffect` hook to trigger an operation as a result of a *side effect*.
- Initialize and update state within a function component.
- Describe the three ways in which a re-render can be triggered for a React component.
- Optimize your application's performance by using `useCallback` and `useRef`.
- Use debugging tools to understand and resolve issues with re-renders.

React Forms Learning Objectives

- Create a React component containing a form.
- Define a single event handler method to handle `onChange` events for multiple form inputs.
- Construct a form that can capture user data using common form inputs.
- Describe a controlled input.
- Implement form validations.
- Handle form submission.

React Context Learning Objectives

- Create a React wrapper component.
- Share and manage global information within a React application using Context.
- Create a React provider component that will manage the value of a Context.
- Retrieve values from a Context throughout your application.
- Describe the relationships between provider, consumer, and context.

Remember, this is your chance to practice your React skills. Don't get bogged down or spend too much time in one area. Check your understanding and skills in each area to best prepare for the real assessment.

Specifications

The a/A Forms company has engaged your services to replace their existing survey website with a new *React* application. Initially, they would like to release a minimum of two surveys. They are also requesting a home page for the site and navigation to the available surveys.

- Home page
- Navigation
- The Sample Survey they use with new prospective customers
  - Has a variety of question types
    - Multiple choice question (`mcq`) means use *radio button* inputs
    - Constructed response (`cr`) on a single-line means a standard text input
    - Constructed response (`cr`) with multiple lines means a text area
    - You can choose to ignore `size`, `charlimit` and `pagebreak`, as these are considered bonus features
  - A simple "Thank you" message shows when submitted (no redirection)
  - Validation should be used to prevent the form from submitting if any of the non-optional (a.k.a. required) fields don't have an answer
  - Questions and survey information are included in the *sample.json* asset file
  - **Bonus**
    - Include one more question in the sample survey for "Employee Number", and obscure the entered text so no one can see the characters that are being typed.

      Hint: Think about password fields.

- A Sensory Preferences Survey
  - A 15 question assessment they allow teachers to share with their students for free which helps them better understand their learning style
  - All questions have the same set of answers:
    - Strongly Disagree (1 point)
    - Disagree (2 points)
    - Neutral (3 points)
    - Agree (4 points)
    - Strongly Agree (5 points)
  - Questions and survey information are included in the *spi.json* asset file
  - You can choose to ignore `pagebreak`, as this is considered a bonus feature

- Answers should be processed, and a summary of the results displayed automatically on submit of the survey (redirect to a report page on submission)
  - Every third question has the same topic
  - Add up the points for each topic and calculate a percentage relative to the total points selected for all three categories
  - Show the highest percentage item first, next highest in the middle, and lowest at the bottom (see wireframe *SPI Report WF.png* in the *_assets* folder)

Provided assets

Clone the starter repo at the `Download` link below. The repo consists of this README and an *_assets* folder that includes the following files:
1) *logo.png* - Company logo 2) *sample.json* - JSON file exported from their current system with the first survey on their list 3) *spi.json* - JSON file from their current system with the inventory questions 4) *bonus.json* - Another JSON file in case you want to do more (see below) 5) *Home Page WF.png* - Wireframe showing the client's general concept; use "Lorem Ipsum" for the text (search online if you don't remember what that is) 6) *SPI Report WF.png* - Wireframe showing the client's general concept for the Sensory Preferences Inventory report

You can choose to use to use the JSON files as mock data, if you'd like. However, the client is fine with copy and paste as well.

The approach

The first step is to use `create-react-app` to get the framework of your application set up quickly. The goal is more important than the order of the steps, so feel free to switch which part of the application you are working on if you get stuck. A few suggestions are provided here.

Tool suggestions (not a complete list, and you may find better ways!)

- Function components with hooks
- `react-router-dom` - `BrowserRouter` and its related pieces, `NavLink`
- `useState` and possibly `useRef` and `useEffect` for form validation
- `useState` for showing thank you message on sample form submit
- `useHistory` for navigating to the report page on sensory preferences survey submission

- `useContext` for sharing answers between sensory preferences survey and the corresponding report
- `Redirect` for navigating away from the report page when there are no results

Option Red - Go Wide

Create components with minimal content quickly, setup `react-router-dom` to navigate to them. Then go through each one and put in basic content, go back to each and add form validation, repeat until all features are complete.

Option Blue - Go Deep

Create one component. Import it into the *App.js*. Build out all its features, e.g., survey questions, then validation, then submit. Replace the component in *App.js* with another one and repeat the process. At the end (or in the middle, if you prefer), bring in `react-router-dom` and configure it for navigation.

Bonus Survey

If you want to go above and beyond, then research the "Likert scale" online and add another page for the survey questions found in *bonus.json*. Include a simple thank-you message on submit, if you'd like.