Front-End Academy

React Capstone

"Stay Organized" - React Rework

Project Description

In this capstone, you will rebuild the “Stay Organized” application you developed in your previous JavaScript capstone project using React's component-based architecture. You will also implement new functionality that was not included in the initial version.

This project will help you deepen your understanding of React while iterating on your previous work. Be sure to read this document closely for hints and tips on rebuilding your application with React, as well as guidelines for implementing additional features.

Setting up the REST API

You will clone another local instance of the same Node.js REST API used in your previous capstone.

```
git clone
https://github.com/DevelopIntelligenceBoulder/stay-organized-workshop-express-s
erver
```

For more detailed directions on cloning and using this Node.js server, refer to the JavaScript Capstone document used in your prior project.

Setting Up the React Project

Follow the steps below to create the new React application.

- Create a New React Project:
    - Open your terminal or command prompt
    - Navigate to the directory where you want to create your new project
    - Run the following command to create a new React project using Create React App:

      ```
      npx create-react-app stay-organized-react
      ```

    - This will create a new directory named stay-organized-react with all the necessary files and dependencies
- Navigate to Your New Project Directory
    - Change to the newly created project directory:
      ```
      cd stay-organized-react
      ```
- Install Additional Dependencies
    - Depending on the requirements of your project, you might need to install additional packages. For example, to use React Router for navigation, run:
      ```
      npm install react-router-dom
      ```
- Start Your React Development Server
    - Launch the development server to ensure everything is set up correctly:
      ```
      npm start
      ```
    - Open your browser and navigate to http://localhost:3000 to see the default React welcome page
- Begin Rebuilding Your Application
    - Start by setting up the basic structure of your application
    - Create components for the different pages you built in your previous application, such as Home, View ToDos, Add ToDo, and Register New User
    - Use React Router to manage navigation between these components
- Integrate the Rest API
    - Use the same Node.js REST API from your previous project
    - Implement API calls within your React components to fetch and manipulate data

Ensure you handle the API responses and update the state of your components accordingly

Capstone Page Requirements

Your React app should include the pages/components below:

- A home page that highlights our "Stay Organized" website
- A View ToDos page that displays all ToDos for a user by picking the user from a dropdown
- **NEW:** A ToDo Details page that allows the user to view and edit ToDo details. See details below.
- An Add ToDo page that allows the user to add a ToDo
- A Register New User page that allows a new user to register for our services

React Tips

When transitioning to a React application, it's important to approach the design strategically to leverage React's robust ecosystem effectively. Below are key tips that will guide you through setting up and optimizing your React application:

- Start by setting up your pages with React Router early in the development process
- Plan how to break down the UI into reusable and independent components. This helps in managing the complexity of the application as it grows
- Ensure you change all class attributes to className in your JSX code

A good grasp of React's fundamental concepts such as JSX, components, state, and props are crucial to your success with this capstone!

View ToDos Page

This page allows the user to view all of the ToDo tasks for a specific registered user by picking that user from a dropdown.

When the page loads, it will display a dropdown listing all of the registered users. You can find that information by sending a GET request to the REST API endpoint `api/users`.

When a registered user is selected from the dropdown list, the page should fetch and display that user's ToDo tasks. The REST API allows you to find the ToDo Tasks of a single user by sending a GET request to `api/todos/byuser/1` where 1 is the `id` of the user whose ToDos you want.

```
 [
  {
    "id": 1,
    "userid": 5,
    "category": "Personal Task",
    "description": "Finish studying for ENG 211 exam",
    "deadline": "2020-11-15",
    "priority": "Medium",
    "completed": false
  },
  ...
]
```

Display the ToDo tasks in some reasonable format.

ToDos Detail Page

This page allows users to view all details for ONE ToDo item and to ***make changes*** to it such as updating the description, priority, deadline, completion status, etc.

NOTE: Users access the ToDos Details page by selecting an individual ToDo item in the View ToDos page. The ToDos Detail page is NOT available in the Navigation Bar.

New ToDo Page

This page allows a user to enter a new ToDo task. You must collect the following information for a ToDo task:

userid	- a number that identifies the user (see notes below)
category	- a type of ToDo that comes from a dropdown displaying values retrieved from the API by calling `/api/categories`
description	- a potentially long string describing the task. Use a `<textarea>` to let user enter a long description.
deadline	- a date deadline (when you test, use the format YYYY-MM-DD but you don't have to enforce it via code)
priority	- an urgency that comes from a dropdown with hard-coded the options "Low", "Medium" or "High"

You should use a dropdown list to allow the user to select the person to whom the ToDo task is assigned.

Register New User Page

The site already has some users registered. If you choose to implement the register page, you will collect 3 pieces of information from the user:

name
username
password

Under the hood, recall the `users.json` file resembles:

```
[
  {
    "id":1,
    "name": "Ian Auston",
    "username": "gamer04",
    "password": "gamer04!"
  },
  ...
]
```

To add a user, you must send a POST request to `api/users`. The field names for the user data must match what is shown in `users.json` EXCEPT you will not send an `id`. It will be auto-generated.

[Advanced Challenges]: Searching and Filtering

Enhance the user experience by allowing users to quickly locate and manage their ToDo tasks. By implementing one or more of the following functionalities, users can efficiently find tasks based on specific criteria, making task management more intuitive and effective.

- Add a Search Bar:
  - Integrate a search bar within the ToDo list page that allows users to type in keywords. This search functionality should filter the displayed tasks in real-time based on the keywords entered, matching against task descriptions.
- Implement Category Filters:
  - Provide users with the ability to filter tasks by their categories (e.g., Work, Personal, Errands). This can be implemented using dropdown menus or checkboxes, allowing users to select one or multiple categories to narrow down the displayed tasks.
- Sort by Priority:
  - Add a sorting feature that lets users organize their tasks based on priority levels (e.g., Low, Medium, High). This will help users to focus on high-priority tasks and manage their workload more effectively
- Filter by Completion Status:
  - Allow users to filter tasks based on their completion status (e.g., Completed, Pending). This will enable users to easily track pending tasks and mark them as completed when done.

What Makes a Good Capstone?

You should:

- Build a consistent look-and-feel throughout the site with intuitive navigation
- Implement at least the required pages
- Have a responsive user interface

You should adhere to best practices such as:

- Have a good directory structures

- Have good file naming conventions)
- Have well- formatted HTML, CSS and JavaScript
- Use good names for your React components and JavaScript variables/functions
- Use HTML, CSS and JavaScript comments effectively

Make sure that:

- You use a ESLint tool if you can find one to ensure you've written good JavaScript!
- There are no JavaScript errors at run time

Build a **PUBLIC** GitHub Repo for your code.

- Use an appropriate branch structure and have a commit history with meaningful comments
- Include a README.md file that describes your project and includes screenshots!
- Make sure it is on the first page of your GitHub projects!

Class Demonstrations

On the last day of the capstone week, you will present your capstone to your peers, managers, and mentors. The ITC staff will provide more details.

Typically, you will be expected to:

- Show off your website and the pages within it
- Show one interesting piece of React you wrote
- Answer questions from the audience if time permits