**React (client) and Node.js (ListAllUsers): Creating a React Component**

**Code:** <https://github.com/benjaminrittenhouse/CIS4282-tutorials/tree/main/listAllUsers-frontend-component>

1. **Clone** the code above and open it in your IDE. Then, run the **React** app and the **Node** app as mentioned in the previous tutorial:
   1. One terminal window runs **server.js** via: nodemon server.js
   2. Another terminal window runs **App.js** via: npm start
      1. Ensure you are “CD”ed, or in the right directory, for each of these files.
   3. Open the react app at localhost port 3000
2. The newest difference from this tutorial vs the last one (listAllUsers api call) is that we are now putting the information we got from our api call into a React component and displaying it to the user.
   1. Try clicking on the **listAllUsers** link in the navbar. You should see users displayed with some information about them.
   2. Navigate to **Display.js** within the **components folder** on the **client side.**
      1. Note: We have created a state variable, userList, to keep track of the users we have gotten in JSON to be converted to User objects
   3. Note the function **fetchAllUsers()**
      1. This is an **asynchronous** function that is awaiting the JSON to return from the webApi call.
      2. Once we retrieve the data, we set the **state** variable accordingly
      3. We again are using our **environment client side variable** to fill in the url that the web app is running on. If you cannot find why you are having specific errors with either your client or server side, **debug** and ensure your environment variables are working properly.
   4. Inside our return, we are returning a **div** that maps, or loops over, all of the users and puts them into **User components.** With that being said, navigate to **User.js** inside components.
   5. Note the ability to sort by ID and reset
      1. This is referencing the getById call we have on our server, as well as listAllUsers on the server that is called when we click reset. The previous tutorial gave detail about how getById works, and how we are simply specifying a user ID within the URL to insert into a sql statement and get a single record.
3. **User.js**
   1. Inside User.js you will notice that we have a react component taking in the props we passed it
      1. The props we passed are the user data from Display.js that we were just looking at. We will be populating our div with this information.
      2. This promotes code reusability and creates an object that we can fill in with whatever information we prefer from the **backend API call**
      3. All of these users are rendered within **Display.js** and shown in a list.
   2. The props are treated as an **object** where we can access **web\_user** fields using the dot notation, such as **props.web\_user\_id**