IMY 220
Assignment 8

React and TypeScript

General Instructions

- This assignment must be completed and submitted by the due date which is available on ClickUP.
- This assignment is a take-home assignment and may take one or two days to complete.
- This assignment should be completed on your own, but you may come to the tutorial session or email the assistant lecturer if you need further assistance.
- No late / email submissions will be accepted.

This assignment focuses on **TypeScript in React**. You will be required to create a simple React application that queries the Pokemon API (PokeAPI) and updates the components based on user input. This project should also use TypeScript for Static Type checking. **TypeScript :any is not allowed (unless otherwise stated) and will not award you any marks**.

Note**: For this assignment you are required to set up a React project using webpack and babel as is taught in the lecture videos. **You may not use a builder such as Create-react-app or Vite. Doing so will result in -10% from your assignment mark.

Part 1 - React App With Webpack and Babel

Create a **simple React App using webpack and babel** as taught in the lecture 19 videos (NodeJs & React).

Add TypeScript to your React app to add static type checking throughout your app. You can learn how to do so here: https://github.com/Microsoft/TypeScript-React-Conversion-Guide#typescript-react-conversion-guide.

Note that in order to use TypeScript in React components, you must name your component files with the **.ts** or **.tsx** extensions (instead of *Component.js* or *.jsx*, you now use *Component.tsx*) **It is recommended to use .tsx for all your component files (including index.js) so the compiler does not complain about your files including JSX in them**. **There are also a few other things you may need to change, so make sure to follow the above tutorial carefully, specifically this section here:**

https://github.com/Microsoft/TypeScript-React-Conversion-Guide?tab=readme-ov-file#minimum-transition-steps.

You are more than welcome to copy and paste in a different project that you have already set up from another practical, assignment or your semester project, just make sure that you have a **working project**, and that you **only** have what is required from you for this assignment (i.e., please remove any extra files when copying over).

You do not have to set up the folder structure like you need to for your project (i.e., with `frontend` and `backend` folders, etc.) but you may do so if you like.

The rest of your assignment depends on this step working, so make sure you have set everything up correctly before moving on to the next step.

Part 2 - API Requests

In a file called ***api.ts***, export an asynchronous function that queries the Pokemon API (PokeAPI): https://pokeapi.co/ (Read their documentation on how to structure a query).

The function should take in a **string**, and query the PokeAPI for a pokemon by **name**. The API request should be made by using one of the methods taught in the lecture videos and slides which returns a **Promise**. Make sure to specify the **parameter** and **return type** of the function (**:any is not allowed**).

You should do adequate **error handling** (for example, if the API request fails or if the response returned is empty) and you should return and display appropriate and informative error messages to the user (not just in the console).

Note: *You should not get a CORS error upon using the PokeAPI. If you do, something else in your project is likely incorrect.*

Part 3 - Pokemon Component

Create a Pokemon class component (***Pokemon.tsx***) that displays the results of the API request that is passed into the component. The component should display the following regarding a Pokemon:

- The **id**
- The **name**
- The **height** and **weight**
- And the **types** of that pokemon

To do this, create an **interface** for the component's **props** with the above data. Make sure to select the correct TypeScript types for each field (**:any is not allowed**). You

should use the PokeAPI documentation (to view what is returned in the response) and the tutorial from Part 1 to help you.

- **Note** that the "*types*" field returned by the PokeAPI is an array of objects specifying the classification types of that pokemon e.g., "Grass", "Fighting", "Fairy", etc.

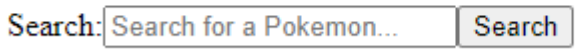
You do not need to include more of the response data you get from the PokeAPI, but you may do so if you like. You must include at least the data above.

Once you have created the interface, you will need to use it in the Pokemon component's **declaration** (see: https://github.com/Microsoft/TypeScript-React-Conversion-Guide?tab=readme-ov-file#add-types) in order to specify the type of the props passed into the component. Also use this in the component's **constructor**.

Part 4 - Search Component

Create a Search class component (***Search.tsx***) that handles the user input for a search. The user can search for a specific Pokemon by name. The search should **not execute** if the input is **empty**, or if the input contains any **numbers**.

The search should look something like this:



You should use the method for grabbing values from HTML inputs, and passing out search output (i.e., passing the search value to the parent component) as discussed in previous assignments, and in the lecture videos/slides. This component should not make API calls, but should just handle user input of a search value that is then passed to the parent component.

1. You will need to create a **property** in your React class for your React ref.
2. You will need to again specify the **types** of the **props** passed in to your Search component (**:any is not allowed**) as you did in Part 3. In this case, the only prop passed in is a **function** from the parent component.

Part 5 - PokemonApp Component

Create a PokemonApp class component (***PokemonApp.tsx***). **This component may have types that are :any**.

This component should act as the parent component for the *Pokemon* and *Search* components, and should handle making requests to the PokeAPI using the function you defined in ***api.ts***.

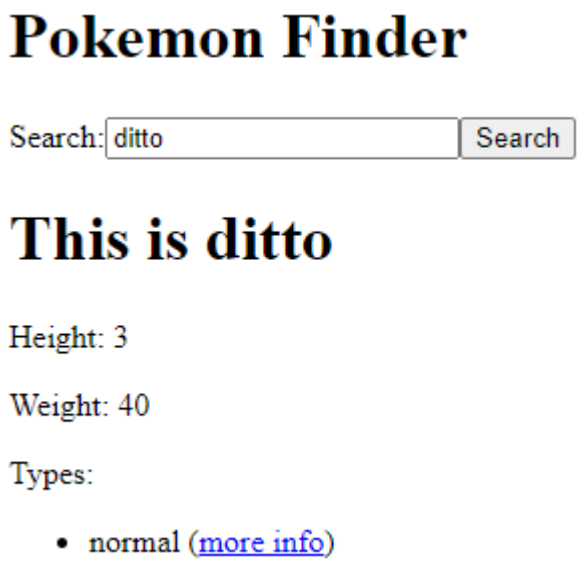
When a user **searches** for a specific Pokemon, a **search** query to the PokeAPI should be made and the Pokemon component updated with the result of the search. The page should not be reloaded.

Remember to display appropriate error messages to the user as described in Part 1.

Render this component in the "**#root**" div in the browser. This should be the root of your application.

The component should look something like this (when the page has initially loaded):

If the user searches for "ditto":



Where "more info" is the url returned by the PokeAPI (https://pokeapi.co/api/v2/type/1/)

Submission Instructions

Place these in a folder named `A8_u12345678` where `12345678` is your student number.

- Submit *all* the files required for this assignment to work (including the config files for babel and webpack) **except** for your node_modules folder and its contents.
- **Do not submit your node_modules folder. This will result in -10% from your assignment mark.**

Zip the **folder** and upload this to ClickUP in the relevant submission slot before the deadline.