Lab 6

Due Apr 25 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip

CS-554 Lab 6

React-Redux Pokémon API

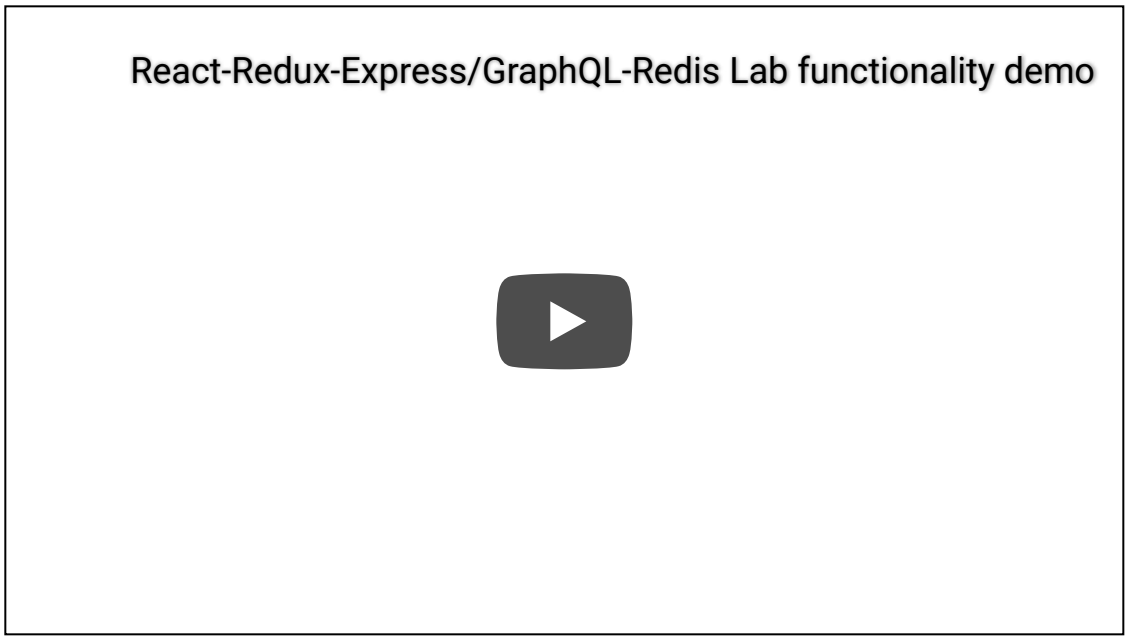
For this assignment, you will make a miniature Pokémon "catching" application. You will incorporate Express, Redis and Redux/Context API into the application.

1 . You will create an Express server with all the routes needed to query the Pokémon API. Your React application will make Axios calls to these routes instead of calling the Pokémon API end-points directly. Your routes will check to see if you have the data being requested in the Redis cache, if you do, your routes will respond with the cached data. If the data is not in the cache, you will then query the API for the data, then store it in the cache and respond with the data.

2. You will use Redux or the Context API (You can choose which you want to use) to handle the trainer/Pokémon states of your React application.

You will be using the **Pokémon Api** **(https://pokeapi.co/docs/v2)** .

Here is a video demo of the lab that our TA Loughlin created:



Routes You Will Need for Your Express Server. All routes should return JSON.

/pokemon/page/:pagenum

This route will respond with JSON data a paginated list of Pokémon, to be used in the corresponding frontend route. It will use the :pagenum param to determine what page to request from the API. **If the Page does not contain any more Pokémon in the list, the route will respond with a 404 status code.**

/pokemon/:id

This route will send more detailed JSON data about a single Pokémon, to be used in the corresponding frontend route. **If the Pokémon does not exist, the route will respond with a 404 status code.**

Routes You Will Need for Your Frontend.

/

This route will explain the purpose of the site.

/pokemon/page/:pagenum

This route will display a paginated list of Pokémon. It will use the :pagenum param to determine what page to request from the backend. If you are on page 0, you will show a button to go to the *next* page on the client side application. If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page. If you are on the last page, then you will show a *previous* button. Each Pokémon will have the option of "catching" or "releasing" that Pokémon from the currently selected Trainer's "team". An individual Trainer can have at most 6 Pokémon in their team at a time. If a Trainer's team is full, then they are unable to "catch" any additional Pokémon until a Pokémon from the team is "released." Clicking on a Pokémon should take you to the corresponding /pokemon/:id page. **If the Page does not contain any more Pokémon in the list, the backend will respond with a 404 status code, which should be displayed on the frontend application.**

/pokemon/:id

This route will show details about a single Pokémon. You should also be able to "catch" or "release" the Pokémon from the currently selected Trainer's "team." **If the Pokémon does not exist, the backend will respond with a 404 status code, which should be displayed on the frontend application.**

/trainers

This route will render a single, scrollable list of all the current sessions' created trainers and their Pokemon "teams". On this page, a user can add and delete trainers to a list of trainers in the redux store. You will also be able to "select" a "Trainer", whose team all caught/released Pokémon will be assigned to. One and only one trainer can be considered the "selected" Trainer at any given moment, and you cannot delete the currently selected Trainer. Clicking on a Pokémon in a Trainer's "team" should take you to the corresponding /pokemon/:id page.

HTML, Look, and Content

Besides the specified settings, as long as your HTML is valid and your page runs passes a **tota11y (http://khan.github.io/tota11y/)** test, the general look and feel is up to you. Feel free to re-use the same general format as one of your previous labs, if you'd like.

I do, however, recommend using **React-Bootstrap (https://react-bootstrap.github.io/getting-started/introduction/)** to make your life easier if you need any UI elements.

As for the data that you choose to display on the Pokemon details pages, that is up to you. You should show as much of the data as possible. At minimum, every Pokémon's individual page should show its name, an Image, and their type(s).

Redux/Context API

You will use either Redux or the Context API to store the state of all the session's current trainers, and their Pokémon "teams". You should be able to create/delete trainers, assign a trainer the "selected" status, and catch/release Pokémon from a trainer's team. This entails creating the action creators, actions, reducers and dispatch the actions.

Keep in mind, that as Redux/Context API states are associated with an individual session, if your page refreshes, all created Trainers, and caught Pokémon will be reset to their original state. As such, you must ensure that your site is a Single Page Application.

Extra Credit

1. If your pagination displays each Pokémon's "official-artwork" (WITHOUT performing an additional axios/backend query for every Pokémon), you will get 5 extra credit points.
2. If you implement a search functionality for finding individual Pokémon, you will get and additional 5 points.
3. If you implement your own custom GraphQL backend instead of an Express backend, you will get 15 extra credit points.

General Requirements

1. Remember to submit your `package.json` file but **not** your `node_modules` folder.
2. Remember to run HTML Validator and tota11y tests!
3. Remember to have fun with the content.
4. Remember to practice usage of async / await!