Lab 7

Due May 11 by 11:59pm **Points** 100 **Submitting** a file upload

CS-554 Lab 7

React Marvel API Revisited

For this assignment, you will take your Marvel API Lab and modify it. You will incorporate Express, Redis and Redux into the application.

1 . You will create an Express server with all the routes needed to query the Marvel API. Your React application will make Axios calls to these routes instead of calling the Marvel 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 to handle the state of your React application.

You will be using the **Marvel API** (https://developer.marvel.com). You will need to register and sign up for an API key. You will not be able to make requests to the API without signing up and getting an **API key** (https://developer.marvel.com/account). You will use the **Characters** (https://gateway.marvel.com/v1/public/characters?ts=1592417963445&apikey=a8f9ccf932bf29fd379ef00e11668673&hash=f061194023791a1593a0ea861a27da67), **Comics** (https://gateway.marvel.com/v1/public/comics?ts=1592417963445&apikey=a8f9ccf932bf29fd379ef00e11668673&hash=f061194023791a1593a0ea861a27da67), and **Series** (https://gateway.marvel.com/v1/public/series?ts=1592417963445&apikey=a8f9ccf932bf29fd379ef00e11668673&hash=f061194023791a1593a0ea861a27da67) listings. Please look at the data returned so you know the schema of the data and the objects it returns (the links to Characters, Comics and Series above work but using my API key. DO NOT use my API key. Please register for your own. You will need to compose the URL with your API key, a ts (time stamp) and a hash.

You can use the following code to construct the URL. (this example is for characters, you would do the same for series and comics) you can read more about AUTHORIZING AND SIGNING REQUESTS from the link below

https://developer.marvel.com/documentation/authorization (https://developer.marvel.com/documentation/authorization)

```
const md5 = require('blueimp-md5');
const publickey = 'your_public_key(API KEY) from Marvel dev portal';
const privatekey = 'your private key from Marvel dev portal';
const ts = new Date().getTime();
const stringToHash = ts + privatekey + publickey;
const hash = md5(stringToHash);
const baseUrl = 'https://gateway.marvel.com:443/v1/public/characters';
const url = baseUrl + '?ts=' + ts + '&apikey=' + publickey + '&hash=' + hash;
```

Routes You Will Need for Your Express Server: All routes Should Return JSON Data.

/characters/page/:page

This route will respond with JSON data a paginated list of Marvel Characters. It will use the :page param to determine what page to request from the API. 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 the Page does not contain any more Characters in the list, the route will respond with a 404 status code and display it on the front end application.**

/characters/:id

This route will show details about a single Character. **If the Character does not exist, the route will respond with a 404 status code and display it on the front end application.**

/comics/page/:page

This route will render a paginated list of comics. It will use the :page param to determine what page to request from the API. **If the Page does not contain any more comics in the list, the route will respond with a 404 status code and display it on the front end application.**

/comics/:id

This route will show details about a single comic. **If the comic does not exist, the route will respond with a 404 status code and display it on the front end application.**

/series/page/:page

This route will render a paginated list of series. It will use the :page param to determine what page to request from the API. You will also show some sort of pagination UI (see below). **If the Page does not contain any more series in the list, the route will respond with a 404 status code and display it on the front end application.**

/series/:id

This route will show details about a single series. **If the series does not exist, the route will respond with a 404 status code and display it on the front end application.**

Pagination

Pagination can be done using an external library from NPM, such as **react-bootstrap's pagination (https://react-bootstrap.github.io/components/pagination/)** or **react-paginate (https://github.com/AdeleD/react-paginate)**.

The minimum you must provide for a pagination UI:

- **If you are on page 0, you will show a button to go to the *next* page.**
- **If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page.**

Pagination HINT: Look at the data.. You can also do the pagination by using the following fields in the data: "offset": 0, "limit": 20, "total": 1493, "count": 20

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. If you want to update it, you may; if you do not, that is fine.

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 chose to display on the details pages, that is up to you. You should show as much of the data as possible. For example, a single character has comics associated with it. It would be nice if you add links to the comics details page for a comic that a character appears in, also perhaps link to the characters that are in a comic on the comic details page.

Redux

You will use Redux to store the state of your entire application. You must store ALL the state of your application in a Redux store, creating the action creators, actions, reducers and dispatch the actions.

Extra Credit

1. If you add search functionality either to characters, comics or series you will get 5 points extra for each that you implement so you have the potential to get 15 points extra credit.
2. If you use GraphQL instead of Express for the server, 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!