

Lab 6

Due May 1 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip

CS-554 Lab 6

React-Redux Marvel API

For this assignment, you will make a Marvel Character collecting 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 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 or the Context API (You can choose which you want to use) to handle the collector/Marvel Character states of your React application.

You will be using the [Marvel API \(https://developer.marvel.com/\)](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 \(https://developer.marvel.com/account\)](https://developer.marvel.com/account). You will use the [Characters \(https://gateway.marvel.com/v1/public/characters?ts=1592417963445&apikey=a8f9ccf932bf29fd379ef00e11668673&hash=f061194023791a1593a0ea861a27da67\)](https://gateway.marvel.com/v1/public/characters?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 Stories 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. 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.)

`/marvel-characters/page/:pagenum`

This route will respond with JSON data a paginated list of Marvel Characters from the API, 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 Characters in the list, the route will respond with a 404 status code.**

`/character/:id`

This route will send more detailed JSON data about a single Marvel Character, to be used in the corresponding frontend route. **If the Character 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.

`/marvel-characters/page/:pagenum`

This route will display a paginated list of Marvel Characters. It will use the `:pagenum` param to determine what page to request from the backend. If you are on page 1, you will show a button to go to the *next* page on the client side application. If you are on page 2+, 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 but not the next button. Each Character will have the option of "collect" or "give-up" that character from the currently selected Collector's "collection". An individual Collector can have at most 10 Characters in their "collection" at a time. If a Collector's collection is full, then they are unable to "collect" any additional Characters until they "give-up" a Character in their "collection" Clicking on a Character should take you to the corresponding `/character/:id` page. **If the Page does not contain any more Characters in the list, the backend will respond with a 404 status code, which should be displayed on the frontend application.**

`/marvel-characters/:id`


This route will show details about a single Character. You should also be able to "collect" or "give-up" the Character from the currently selected Collector's "collection". **If the Character does not exist, the backend will respond with a 404 status code, which should be displayed on the frontend application.**

`/collectors`

This route will render a single, scrollable list of all the current sessions' created collectors and their character "collections". On this page, a user can add and delete Collectors to a list of collectors in the redux store. When a collector is created, the user should have the ability to enter a name for the collector. i.e. "collector 1", "collector 2", etc.

You will also be able to "select" a "Collector", whose collection of all collected/give-up Characters will be assigned to. One and only one Collector can be considered the "selected" Collector at any given moment, and you cannot delete the currently selected Collector. Clicking on a Character in a Collector's "collection" should take you to the corresponding `/character/:id` page. The characters should also have "give up" buttons here just like on the `character/:id` and `/marvel-characters/page/:pagenum` routes.

HTML, Look, and Content

Besides the specified settings, as long as your HTML is valid and your page runs passes a [tota11y](https://khan.github.io/tota11y/)  (<https://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/)  (<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 character details pages, that is up to you. You should show as much of the data as possible. At minimum, every character'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 collectors, and their Character "collections". You should be able to create/delete collectors, assign a collector the "selected" status, and collect/give-up Characters from a collector's collection. 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 Collectors, and collected Characters will be reset to their original state. You can look into using the `persist` package in Redux to persist state if you like (if you persist state, we will give you 5 points extra credit as noted below!) As such, you must ensure that your site is a Single Page Application.

Extra Credit

1. If your Redux/Context API state persists on a page refresh, you will get 5 extra credit points.
2. If you implement a search functionality for finding individual Characters, you will get an 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`!

