Lab 6

Due Nov 21, 2023 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 Comic 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 collection/Marvel Comic states 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 **Comics (https://gateway.marvel.com/v1/public/comics?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 Comics 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)

```
import md5 from '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/comics';
const url = baseUrl + '?ts=' + ts + '&apikey=' + publickey + '&hash=' + hash;
```

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

/api/comics/page/:pagenum

This route will respond with JSON data of a paginated list of Marvel Comics 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 comics in the list, the route will respond with a 404 status code. Redis Cache: You will check to see if the results for that pagenum are in the cache. If they are, your route will respond with the data from the cache, if it's not in the cache, you will query the API, store it in the cache and respond with the returned data from the API.**

/api/comics/:id

This route will send more detailed JSON data about a single comic, to be used in the corresponding frontend route. **If the comic does not exist, the route will respond with a 404 status code. Redis Cache: You will check to see if that comic data is in the cache. If it is, your route will respond with the data from the cache, if it's not in the cache, you will query the API, store it in the cache and respond with the returned data from the API.**

Routes You Will Need for Your Frontend.

/

This route will explain the purpose of the site.

/marvel-comics/page/:pagenum

This route will display a paginated list of comics. It will use the :pagenum param to determine what page to request from the backend /api/comics/page/:pagenum route. 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 comic will have the option to "collect" or "give-up" that comic from the currently selected sub-collection (if a comic is already in the selected sub-collection, you will not show the collect button for that comic, just the give-up button). A sub-collection can have at most 20 comics in the collection at a time. If a sub-collection's collection is full, then the sub-collection is unable to "collect" any additional comics until they "give-up" a comic in their "sub-collection". Clicking on a comic should take you to the corresponding /marvel-comics/:id page. **If the Page does not contain any more comics in the list, the backend will respond with a 404 status code, which should be displayed on the frontend application.**

/marvel-comics/:id

This route will show details about a single comic. It will use the :id param to determine what comic to request from the backend /api/comics/:id route. You should also be able to "collect" or "give-up" the comic from the currently selected sub-collection's "collection". (if a comic is already in the selected sub-collection, you will not show the collect button for that comic, just the give-up button) **If the comic does not exist, the backend will respond with a 404 status code, which should be displayed on the frontend application.**

/marvel-comics/collections

This route will render a single, scrollable list of all the current sessions' created sub-collections and their comic "collections". On this page, a user can add and delete sub-collections to a list of collections in the redux store. When a sub-collection is created, the user should have the ability to enter a name for the sub-collection. i.e. "Spider-Man Collection", "X-Men Collection", "Super Villain Collection" etc.

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

HTML, Look, and Content

Besides the specified settings, as long as your HTML is valid, 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 **Material UI (https://mui.com/material-ui/)** to make your life easier if you need any UI elements.

As for the data that you choose to display on the comic details pages, that is up to you. You should show as much of the data as possible that is important about the comic's details. At minimum, every comic's individual page should show its title, an Image, the description, their onSaleDate, and their printPrice.

Redux/Context API

You will use either Redux or the Context API to store the state of all the session's current sub-collections, and their comic "collections". You should be able to create/delete sub-collections, assign a sub-collection as the "selected" sub-collection, and collect/give-up comics from a sub-collection's collection. This entails creating the action creators, actions, reducers and dispatch the actions. A comic should only be able to be added to a sub-collection once but different sub-collections can have the same comic in it.

Keep in mind, that as Redux/Context API states are associated with an individual session, if your page refreshes, all created sub-collections, and collected comics 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 comics, you will get and additional 5 points.
3. If you implement your own custom GraphQL backend instead of an Express backend, you will get 10 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!
3. Remember to have fun with the content.

4. Remember to practice usage of async / await!