Lab 4

Due Mar 31 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip

CS-554 Lab 4

React Ticketmaster API

For this lab, you will create a Ticketmaster API Single Page Application using React.

You will be creating a Single Page Application using React that implements the following routes using **create-react-app (https://create-react-app.dev/)** and **react router 6 (https://reactrouter.com/en/main)** .

Your React application will be making Axios calls to the Ticketmaster API for data.

will be utilizing the **Ticket Master Discovery API (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/)** for data.

The base URL is **https://app.ticketmaster.com/discovery/v2/**

You will have to register a free account and get your own individual API key. After receiving the api key, you would insert the key into the url that you requested with axios.

In addition, when querying events, venues, or attractions for any of your routes, you should filter the data to only be results found in the US. You can do this with the **'countryCode'** url param.

So for example, if you were querying all events, the url should be

https://app.ticketmaster.com/discovery/v2/events?apikey=${API_KEY}&countryCode=US`

The full authorization documentation can be found **here (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#overview)**

You can play with the API and different parameters **here (https://developer.ticketmaster.com/api-explorer/v2/)**

Pages

/

The root directory of your application will be a simple page explaining the purpose of your site (to talk about Ticketmaster, the API, perhaps your favorite event/concert you ever attended etc..). Be creative with this one!

This page will have a `<Link>` to the Events Listing (`/events/page/1`), The Attractions Listing (`/attractions/page/1`), and the Venues Listing (`/venues/page/1`)

/events/page/:page

You will utilize **this endpoint (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#search-events-v2)** from the Ticketmaster API to get the data for this route.

This route will render a paginated list of Events from the API. It will use the :page param to determine what page to request from the API. If you are on page 1, you will show a button to go to the *next* page. 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 the Page does not contain any more events in the list, the SPA will redirect to a 404 page.**

/events/:id

You will utilize **this endpoint (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#event-details-v2)** from the Ticketmaster API to get the data for this route.

This route will show details about a single Event. **If the Event does not exist, the SPA will redirect to a 404 page. This must work for EVERY event in the API**

/attractions/page/:page

You will utilize **this endpoint (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#search-attractions-v2)** from the Ticketmaster API to get the data for this route.

This route will render a paginated list of Attractions. It will use the :page param to determine what page to request from the API. **If the Page does not contain any more Attractions in the list, the SPA will redirect to a 404 page.**

/attractions/:id

You will utilize **this endpoint (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#attraction-details-v2)** from the Ticketmaster API to get the data for this route.

This route will show details about a single Attraction. **If the Attraction does not exist, the SPA will redirect to a 404 page. This must work for EVERY attraction in the API**

/venues/page/:page

You will utilize **this endpoint (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#search-venues-v2)** from the Ticketmaster API to get the data for this route.

This route will render a paginated list of venues. 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 venues in the list, the SPA will redirect to a 404 page.

/venues/:id

You will utilize t**his endpoint (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#venue-details-v2)** from the Ticketmaster API to get the data for this route.

This route will show details about a single venue. **If the venue does not exist, the SPA will redirect to a 404 page**. **This must work for EVERY event in the API**

Pagination

The minimum you must provide for a pagination UI:

- **If you are on page 1, you will show a button to go to the *next* page.**
- **If you are on page 2+, you will show a *next* button and a *previous* button, that will take you to the previous page.**

The Ticketmaster API makes it very easy to calculate the number of pages that you will have. You will use the following fields to calculate the number of pages based on the data in the api.

For example, When we query the Ticketmaster API's search endpoint you will notice the following properties in the JSON return object. `number`: this shows the offset where we are in the list, i.e. the page number. `size` : this allows us to select how many objects we want per page, the default is 20. `totalElements`: this shows us how many total elements are found in the list. `totalPages`: this shows us how many total pages are found in the list. Using these fields, we can easily calculate how many pages we will have in the list.

Again, it is SUPER important when working with an API that you read the documentation and understand how the API works and understand the schema of the data returned by the API. So please, please read the documentation and analyze the data returned.

HTML, Look, and Content

Be creative with this assignment!

I do, however, recommend using **Material-UI (https://mui.com)** 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, the events/page/:page route should show events with their picture, the event name, the price range, and the start date. It would be nice if the event linked to its individual page in events/:id.

So you should at LEAST show the image for the event/attraction/venue, the name of the event/attraction/venue, the start date if there is one, the price range if there is one, and some additional fields. Do NOT just dump raw JSON to the react components, points will be deducted if you do! Depending on how much data you display and how far you go with it, you may get extra points for going the extra mile and displaying as much data as possible.

Extra Credit

If you add search functionality either to events, attractions or venues you will get 5 points extra for each that you implement so you have the potential to get 15 points extra credit.

General Requirements

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