**Interacting with the Yelp API**

Welcome to the fourth and final part of the Ravenous project! Take a minute to review what you accomplished in the third part of this project. You:

* Set the state of search bar elements
* Simulated a search query

In this project, you’ll complete the fourth part of Ravenous: **Interacting with the Yelp API**.

So far, Ravenous is mostly static. Clicking on the “Let’s Go” button doesn’t retrieve results from the Yelp API. We’ll need that functionality to complete Ravenous. This project will ask that you build this functionality.

A few notes before getting started:

* Portions of this project will need to be completed on the Yelp website (developer documentation). You should expect to use the documentation independently to help you complete this project.
* You should expect to spend more than 1 day on this specific project. It’s OK if it is not all finished in one day.
* If you don’t understand how to implement a certain part of the project, we’ll provide guidance as needed. However, you should expect to search Codecademy for the exercises that will provide you with the relevant information.

Let’s get started!

If you get stuck during this project or would like to see an experienced developer work through it, click “**Get Help**“ to see a **project walkthrough video**.

**Tasks**

**1/28Complete**

Mark the tasks as complete by checking them off

**Create a util Directory**

**1.**

We’ll store the Yelp functionality in its own directory.

Create a folder called **util** in the **src/** directory.

**Create Yelp.js**

**2.**

cd into the **util/** directory and create a file called **Yelp.js**.

**Retrieve Yelp API v3 Credentials**

**3.**

APIs require that you retrieve and use valid credentials (provided by them) in order to use the API.

Start by navigating to the “Manage App” section of the [documentation page.](https://www.yelp.com/login?return_url=%2Fdevelopers%2Fv3%2Fmanage_app) Note that you’ll have to log in (or create an account if you don’t already have one).

**4.**

Once you login, create a new app using the menu to the left. Fill out the required fields for the app (you can use “Ravenous” for “App Name”). Agree to the terms and conditions and create the app.

**5.**

You should see an “API Key” presented to you. Take note of this, you’ll use it in **Yelp.js**.

**Add Credentials to Yelp.js**

**6.**

Open **Yelp.js**. At the top of the file, create a constant called apiKey. Inside it, store a string of your API key from the Yelp API.

**Add a fetch() Polyfill**

**7.**

Throughout the rest of this project, we will use the fetch() browser API to make our requests.

Since fetch() is a browser API, older browsers may not support it. To increase the accessibilty of Ravenous to a wider audience of users, we’ll need to add a fetch() polyfill to support older browsers.

Within the Ravenous directory in your terminal, run npm install whatwg-fetch --save to install the whatwg-fetch polyfill and add it to your **package.json** file.

**Create a Yelp Module**

**8.**

Create an empty object called Yelp. This object will store the functionality needed to interact with the Yelp API.

**Add search()**

**9.**

Create a method called search. This is the method we’ll use to retrieve search results from the Yelp API.

Pass in term, location, and sortBy parameters to the method.

**10.**

Inside of the method, begin the method with a return which will return a promise that will ultimately resolve to our list of businesses. After return begin your chain of calls by calling fetch().

Hint

return fetch();

**11.**

To retrieve businesses, you’ll have to hit the /businesses endpoint of the Yelp API.

Pass in the following path as the first argument to fetch():

https://api.yelp.com/v3/businesses/search?term=TERM&location=LOCATION&sort\_by=SORT\_BY

Use interpolation to replace TERM, LOCATION, and SORT\_BY with the correct variables in the path above.

Hint

return fetch(`https://api.yelp.com/v3/businesses/search?term=${term}&location=${location}&sort\_by=${sortBy}`);

**12.**

Your fetch() will currently not function correctly due to [CORS](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) restrictions.

We can bypass this restriction with an API called [CORS Anywhere](https://cors-anywhere.herokuapp.com/). CORS Anywhere will take requests sent to its API endpoint, make them for the requesting app with the proper CORS permissions, and then return the response back to the requesting app.

Prepend the URL path you passed to the first argument in fetch() with the following:

https://cors-anywhere.herokuapp.com/

Hint

return fetch(`https://cors-anywhere.herokuapp.com/https://api.yelp.com/v3/businesses/search?term=${term}&location=${location}&sort\_by=${sortBy}`);

**13.**

When we make requests to the Yelp API, we have to present a form of identification for the browser. This is because the Yelp API wants to know that we are authorized to access the API.

This identification is presented using our API key as a [browser header](https://en.wikipedia.org/wiki/User_agent).

Pass in a second argument to the fetch() call. It should be an object with a key of headers.

The value of headers should be another object.

This object should have Authorization as a key. The value of this key should be:

`Bearer ${apiKey}`

Hint

Your current structure should be:

return fetch(..., {

headers: {

Authorization: `Bearer ${apiKey}`

}

});

**14.**

We will need to convert the returned response to JSON for us to be able to effectively utilize our list of businesses.

Chain a call using then() to the end of the fetch() method.

Pass in a callback function. The callback function should:

* Use arrow function syntax
* Take one parameter called response
* Return a call to .json() on response

Hint

return fetch(..., {

headers: { ... }

}).then(response => {

return response.json();

});

**15.**

Chain another .then() call after the previous to retrieve the list of businesses from the converted JSON response.

Pass in a callback function that takes one parameter called jsonResponse.

**16.**

Inside of the callback function, add an if statement that checks to see if jsonResponse has a businesses key (this would represent a valid response returned by the Yelp API).

If it doesn’t, we don’t want our site to crash trying to render a list of businesses that don’t exist.

Hint

if (jsonResponse.businesses) {

}

**17.**

If this key *does* exists in the JSON response, we should return an array that has all of the business’ properties we’ll need (the ones we previously hard coded, like name, address, city, and more).

In the if statement, iterate through jsonResponse.businesses using map().

Hint

return jsonResponse.businesses.map();

**18.**

Pass a callback function to the map() method. The callback function should take a parameter called business.

Hint

return jsonResponse.businesses.map(business => {

});

**Accessing the JSON Response**

**19.**

Inside of the callback function, return an object.

This object will be lengthy and should include all of the attributes needed to display a business in Ravenous.

It will also be up to you to access the jsonReponse object returned by the Yelp API to extract the specific attributes needed. Most of these attributes reside inside at the top-levl of business, but some do not. You’ll have to do some digging to see exactly where they reside in the response object.

We’ll provide the list of keys that should be in the object:

* id
* imageSrc
* name
* address
* city
* state
* zipCode
* category
* rating
* reviewCount

Hint

We’ll start you off with the first couple of values:

Inside of the callback function:

{

id: business.id,

imageSrc: business.image\_url,

// more key/values from the response

}

**20.**

Finally, export your Yelp module.

**Add a key to <Business />**

**21.**

That completes **Yelp.js**.

Now that we have real data, we can modify the Business component that BusinessList renders. Every list item rendered by a React component needs to have a unique key, we were unable to hard code this, but we can now use the unique ID of each business as the key.

Open **BusinessList.js**. Add a key property to the rendered <Business /> component.

Set it equal to the ID of business.

Hint

business.id

**Import Yelp.js**

**22.**

The Yelp API functionality you wrote will need to be used by App.

Open **App.js** and import Yelp at the top of the file (after the first four imports).

**Remove Hard Coded Information**

**23.**

Remove the hard coded business and businesses information from **App.js**.

**Add an App Constructor**

**24.**

Above the searchYelp() method, add a constructor.

Inside of the constructor:

* Set the initial state using this.state
* this.state should be an object with a key of businesses set to an empty array
* Bind the searchYelp() method to the current value of this (since we are about to update it to use this)

Hint

this.state = {

businesses: []

};

this.searchYelp.bind(this);

**Modify searchYelp()**

**25.**

Remove the console.log() message from searchYelp().

Inside of searchYelp(), call Yelp.search(). Pass the same three arguments supplied to .searchYelp() in your Yelp.search() call.

**26.**

Once we retrieve our list of businesses, we will need to update the state.

Chain a then() call to the end of Yelp.search().

Pass it a callback function that takes one parameter called businesses.

Inside of the callback function, update the state using setState().

Pass in an object to setState().

The object should have a key of businesses with a value of businesses (the returned array of businesses).

**27.**

Now that you are querying Yelp for real businesses and passing down the returned list of businesses, your app should work as expected. Try out querying for a few different types of restaurants and zipcodes to make sure it works as expected.

Congratulations you just made a full front-end web application that updates automatically and uses real data! This is a truly awesome achievement~

**Completion and Reflection**

**28.**

Congratulations! You’ve completed the Ravenous project!

You’ve come a long way. You started the project by:

* Hard coding business information
* Simulating a list of businesses
* Simulating a query to the Yelp API

Now, your project is at a state where all of these initial issues have been solved. You can now search the Yelp API for restaurants that match a search option!

Over the course this project, you utilized the React library, importing and exporting components, the Yelp API, props, and more to achieve a fully functioning Yelp-like clone.

You should be proud of how far you’ve come. As you continue towards your goal of building front-end web apps, you’ll continue to use many of these foundational skills as you learn more challenging and advanced concepts.

We suggest that you push your React knowledge by adding features to Ravenous. This will allow you to not only reinforce your new skills, but create something unique and exciting in the process. We’ve provided a few ideas for how to improve Ravenous in the hint below, but now that you know how to code, you can implement any feature you want!

Once again, congratulations on a successful project and thank you for all of the hard work you put into this project. Enjoy!

Hint

Below is a list of some potential features to add to Ravenous:

* Make addresses clickable and have them open the address in Google Maps in a new tab
* Make images clickable and have them open the business’ website in a new tab
* Clicking on a different sorting option automatically requeries the Yelp API, rather than having to manually click “Let’s Go” again
* Implement your own type of sort (for example, by entering a distance or radius from a central location)
* Allow you to search by pressing “Enter” (or “Return”) on your keyboard, as opposed to manually clicking
* Add autocompletion of addresses to the “Location” input