**Create Project & Components**

Welcome to the first installment of the Ravenous project! Over the next couple of weeks, you’ll build a website called “Ravenous”, a Yelp-like clone.

In total, there will be four parts to this project:

* Creating Static Components
* Passing Information to Components
* Setting the State of Ravenous Components
* Interacting with the Yelp API

Today, you will start by building the first part of Ravenous: **Creating Static Components**.

Here’s a quick overview of how Ravenous should function:

* As a user, I should be able to search for restaurants
* As a user, I should be able to view a list of restaurants returned by the Yelp API
* As a user, I should be able to sort through restaurants using a filter

The four projects will test your knowledge of JavaScript and React, all with the goal of building a Yelp-like clone. If you want to get a feel for what Ravenous can be, visit the Yelp website and search for restaurants in your area.

Finally, a few notes before getting started:

* In each project, you’ll be presented with the intended, final outcome (of that project) in the Codecademy browser component.
* You should expect to spend more than 1 day on this specific project. It’s the base of the entire app, and it will likely take a few days to complete this 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.
* You should expect to complete all four Ravenous projects on your personal computer using your preferred tools (terminal, text editors, etc.).

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**

**0/49Complete**

Mark the tasks as complete by checking them off

**Create a React App**

**1.**

Start by creating a React app using the create-react-app package in your preferred terminal. The name of your app should be ravenous.

Hint

create-react-app ravenous

**2.**

In another terminal window, cd into the Ravenous directory and type the following command:

npm start

This command will start a development server and open up a preview of your app in a browser tab. As you build, the preview will automatically update when you save your work. This is a great way of seeing your progress in real-time.

**3.**

Replace the current favicon located at **public/favicon.ico** with [this icon](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/ravenous_favicon.ico).

Make sure you save the new image as **favicon.ico**.

**Add a CSS Reset**

**4.**

By default, create-react-app will generate a sample application. We’ll need to add a reset.css file to ensure our app is styled the same in every browser.

cd into the newly created Ravenous directory. Take a look at the default folder structure generated by create-react-app. Use the command line to create a new file called **reset.css** in the **public/** directory. Copy and paste [this CSS](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/reset.css) into **reset.css**.

**5.**

Link to **reset.css** in the **index.html** file that’s also located in the **public/** directory.

Keep **index.html** open.

Hint

<link href="./reset.css" rel="stylesheet" type="text/css">

**Add Google Fonts**

**6.**

You’ll also need to add a couple of Google fonts that the app will use. In **index.html**, add the following links to add the necessary Google fonts:

<link href="https://fonts.googleapis.com/css?family=Work+Sans:300,500,600" rel="stylesheet">

<link href="https://fonts.googleapis.com/css?family=Poppins:600" rel="stylesheet">

**Create Business.js**

**7.**

Great, now let’s move on to some actual React code! Ravenous will be composed of four different components interacting with each other:

* <Business />
* <BusinessList />
* <SearchBar />
* <App /> (created by default with create-react-app)

You’ll create all four components in this project, one by one. You can expect them to be static components for now — you’ll add additional functionality later.

First, you’ll need to structure the **Ravenous** directory differently.

cd into the **src/** directory. Create a new folder called **components**. You’ll store all of your components in this directory.

**8.**

Each component should have its own folder inside of **components/**. Inside of the **components/** directory create a new folder called **App**.

Move **App.js** and **App.css** to the **App/** folder and update the path in **index.js** accordingly.

Remove **App.test.js** from the **src/** folder, as you will not use it in this project.

Hint

In **index.js** our filepath for importing **App** will change.

...

import App from './components/App/App';

...

**9.**

Next, let’s create the <Business /> component.

Inside of **components/**, create a new folder called **Business**. cd into the **Business/** directory and create two files:

* **Business.js**
* **Business.css**

**Create a Sample Business**

**10.**

Great! Now you’re ready to begin creating your first React component. The purpose of the <Business /> component is to represent how a business (a restaurant) in Ravenous will be formatted and styled. For now, we’ll hard code a single business listing. Later, you’ll update it dynamically using the Yelp API.

Open **Business.js** in your text editor.

Create a JavaScript object called business. Copy the following keys and values and add them to business:

imageSrc: 'https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/pizza.jpg',

name: 'MarginOtto Pizzeria',

address: '1010 Paddington Way',

city: 'Flavortown',

state: 'NY',

zipCode: '10101',

category: 'Italian',

rating: 4.5,

reviewCount: 90

**Import React**

**11.**

Perfect. The information in this object will be consumed by the <Business /> component.

To build the component using React, you’ll need to import the React library.

At the very top of your **Business.js** file (line 1), import React.

Hint

import React from 'react';

**Create the <Business /> Component**

**12.**

Next, below the business object, create a React component called Business. The component should extend React.Component.

Hint

class Business extends React.Component {

}

**13.**

Add a .render() method to the Business component.

Hint

class Business extends React.Component {

render() {

}

}

**14.**

Inside of the .render() method, add a return statement with JSX that renders [this](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/business.txt) HTML.

Follow the guidelines below when you write the HTML (linked above) as JSX:

* Change all class attributes to className.
* Do not change the class values, as we will use them in the next step to add style to the business component.
* Replace the relevant information with references to properties in the business object (i.e. {business.thisProperty}).

**Import Business.css**

**15.**

You added a bunch of className attributes to the HTML elements in the Business component, but they won’t be very useful if we don’t style those elements with CSS! Create a new file called **Business.css** in the **Business/** directory. Add [this CSS](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/Business.css) to to the **Business.css** file.

**16.**

This CSS won’t be very useful if your <Business /> component can’t access it. At the top of **Business.js** (line 2), import the **Business.css** file. Remember that the CSS file lives in the same parent directory as **Business.js**.

Hint

import './Business.css';

**Export the <Business /> Component**

**17.**

You just built your first React component! Congratulations! All that’s left to do is to make this component available to the rest of the Ravenous app. At the bottom of the file (last line), export the Business component.

Hint

export default Business;

**Create BusinessList.js and BusinessList.css**

**18.**

Let’s move on to the next component: <BusinessList />.

In the **components/** directory, create another folder called **BusinessList**.

Inside of **BusinessList**, create two files:

* **BusinessList.js**
* **BusinessList.css**

**19.**

Add the following CSS to **BusinessList.css**:

.BusinessList {

display: flex;

justify-content: space-around;

flex-wrap: wrap;

margin: 4.4rem 10%;

}

**Import React and BusinessList.css**

**20.**

Great! Now you’re ready to start building this component. Start by opening **BusinessList.js**. At the top of the file, import the following (in this order):

* The React library
* **BusinessList.css**

**Import the <Business /> Component**

**21.**

The point of the <BusinessList /> component is to simulate what a returned list of businesses would look like in Ravenous (after querying the Yelp API, for example). To help this simulation, <BusinessList /> will make use of the <Business /> component repeatedly. To use the <Business /> component, you’ll have to import it. On the next line, import the <Business /> component.

Stuck? Get a hint

**Create the <BusinessList /> Component**

**22.**

Let’s start building the component. Use the React library to create a component called BusinessList.

Hint

\_ BusinessList \_ React.Component {

}

Make sure to replace the \_ with the correct keywords.

**23.**

Inside of the component, add a render() method.

**Rendering <Business />**

**24.**

Inside of the .render() method, add a return statement with JSX that renders [this](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/business-list.txt) HTML.

Follow the guidelines below when you write the HTML (linked above) as JSX:

* Change all class attributes to className.
* Do not change the class values, as we will use them in the next step to add style to the business list component.
* Replace each comment with a Business component.

Hint

<div className="BusinessList">

<Business />

<Business />

// Add four more Business components

</div>

**Export the <BusinessList /> Component**

**25.**

That’s it for the BusinessList component! This list component will need to be rendered again by another component, so you’ll need to export it. At the bottom of the file, export BusinessList.

**Create SearchBar.js**

**26.**

To search for businesses (restaurants) in Ravenous, you’ll need a search bar. We’ll implement this with a SearchBar component.

In the **components/** directory, create another folder called **SearchBar**.

Inside of **SearchBar**, create two files:

* **SearchBar.js**
* **SearchBar.css**

**27.**

Add [this CSS](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/SearchBar.css) to the **SearchBar.css** file.

Here are the two images you’ll need for the project:

* [Ravenous (desktop size)](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/background_search_desktop.jpg)
* [Ravenous (mobile size)](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/background_search_mobile.jpg)

The “mobile” version should be used in any media queries you write.

**Import React and SearchBar.css**

**28.**

Great! Now you’re ready to start building this component. Start by opening **SearchBar.js**. At the top of the file, import the following (in this order):

* The React library
* **SearchBar.css**

**Create an Options Object**

**29.**

The search bar will communicate with the Yelp API, but you’ll build the functionality to communicate with the API in a later project. Today, you’ll build part of the structure that’s needed to communicate with the Yelp API. Specifically, requests to the Yelp API must follow [formatting and naming conventions set by the API](https://www.yelp.com/developers/documentation/v3/business_search). For example, the search bar should allow users to search businesses by:

* Best Match
* Highest Rated
* Most Reviewed

To achieve this, you’ll create an object with keys and values that conform to what the API expects to receive (as shown in the documentation provided above). Let’s see what this looks like.

Start by creating an object called sortByOptions.

**30.**

The object should have three keys (in this order):

* Best Match
* Highest Rated
* Most Reviewed

Make sure the keys are strings.

**31.**

Next, set the values. Use [the documentation](https://www.yelp.com/developers/documentation/v3/business_search) to help you set the values of those keys. The sort\_by entry in the table of the “Parameters” section will be helpful. The values should be strings. See if you can find the keys before looking at the hint.

Hint

const sortByOptions = {

'Best Match': 'best\_match',

'Highest Rated': 'rating',

'Most Reviewed': 'review\_count'

}

**Create the <SearchBar /> Component**

**32.**

Let’s start building the search bar component. Use the React library to create a component called SearchBar. Don’t add a render() method just yet.

**Create renderSortByOptions()**

**33.**

Create a method called renderSortByOptions() in the body of the component declaration.

**34.**

The purpose of renderSortByOptions() is to dynamically create the list items needed to display the sort options (Best Match, Highest Rated, Most Reviewed). This is to help [future proof](https://en.wikipedia.org/wiki/Future_proof) against potential changes to the Yelp API.

The method should iterate through the keys and values of the sortByOptions object and return a list item. The list item elements should use the keys as an attribute, and the values as content. Let’s start building it out.

Start adding a return statement (no parentheses).

Hint

renderSortByOptions() {

return

}

**35.**

To iterate through the object, you’ll need to start by accessing the keys of the sortByOptionsobject. Call the keys() method on the JavaScript Object library. Pass in sortByOptions as the argument.

Hint

renderSortByOptions() {

return Object.keys(sortByOptions)

}

**36.**

Now that you have access to the keys, you’ll iterate through them using the map() method. Call the map() method by chaining it to the end of the line you just wrote.

Hint

renderSortByOptions() {

return Object.keys(sortByOptions).map();

}

**37.**

Pass a callback function to the map() method as an argument. The callback function should have one parameter called sortByOption. The callback function should also use arrow syntax.

Hint

renderSortByOptions() {

return Object.keys(sortByOptions).map(sortByOption => {

});

}

**38.**

Now let’s store the object values in a variable. Inside of the callback function, access the sortByOptions values using the sortByOption parameter of the callback function. Store values in variable called sortByOptionValue.

Hint

let sortByOptionValue = sortByOptions[sortByOption];

**39.**

On the next line, return a <li> element. The list item should have an attribute called key set to sortByOptionValue (don’t forget to use curly braces to inject JavaScript). The content of the list item should be sortByOption. Again, use curly braces to achieve the JavaScript injection.

Hint

return <li key=?> ? </li>;

Use the instructions to help you fill in the ?s.

**40.**

Great, you’re done with renderSortByOptions()! You’ll use this method when building the structure of the search bar component.

**Render the <SearchBar /> Component**

**41.**

Let’s build the structure of the <SearchBar /> component. Start by adding a render() method.

**42.**

Inside of the .render() method, add a return statement with JSX that renders [this](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/search-bar.txt) HTML.

Follow the guidelines below when you write the HTML (linked above) as JSX:

* Change all class attributes to className.
* Do not change the class values, as we will use them in the next step to add style to the search bar component.
* Replace the comment with a call to .renderSortByOptions().

**Export the <SearchBar /> Component**

**43.**

That’s it for the SearchBar component! This list component will need to be rendered again by another component, so you’ll need to export it. At the bottom of the file, export SearchBar.

**Import React and App.css**

**44.**

It’s time to put all of the components together!

Open the **App.js** file in the default **App/** directory.

At the top of **App.js**, import the the following (in this order):

* The React library
* **App.css**

Make sure that **App.css** has [this CSS](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/App.css).

**Import <BusinessList /> and <SearchBar />**

**45.**

You’ll need to import the <BusinessList /> and <SearchBar/ > components too. Next, import BusinessList and SearchBar. You’ll have to use the correct path to import them.

Hint

import BusinessList from '../?/?';

Take a look at your directory structure to fill in the ? correctly.

**Modify the Default <App /> Component**

**46.**

The create-react-app command creates a default App component for you. It includes a render() method along with a return statement. We won’t need the default App component, so let’s make some modifications. Delete everything inside of the return statement.

**Render Ravenous**

**47.**

Inside of the **App.js** .render() method, return JSX that renders [this](https://s3.amazonaws.com/codecademy-content/programs/react/ravenous/app.txt) HTML.

Follow the guidelines below when you write the HTML (linked above) as JSX:

* Change the class attribute to className.
* Do not change the class values, as we will use them to add style to the website.
* Replace the comments with their corresponding components.

**Export <App />**

**48.**

By default, the create-react-app command adds a default export in the class declaration of the App component, making it look like the following:

export default class App extends Component {

// Code

}

To be consistent across with how components are exported in the Ravenous project, remove the export default from the beginning of the class declaration. Instead, export App at the bottom of the file (you’ve done this for all components so far).

**Completion and Reflection**

**49.**

Congratulations! You completed the first part of the Ravenous project. Let’s briefly review what you built:

* A Business component that simulates a single business
* A BusinessList component that simulates a list of single businesses
* A SearchBar component that will be used in the future to search for business

The App component renders a SearchBar component and a BusinessList component. The final product is a *simulation* of how the Ravenous project will look and work. Some functionality (like querying the Yelp API) is currently missing, but you’ll build it in the upcoming projects.

Reflect on the knowledge you used in this project. As you do so, consider the following question:

* At the moment, some information (like business information) is hard coded into specific components. How might you pass this information dynamically *between* components?

We’ll expand each component through the coming weeks to address the question above (and more).