HackCville Launch Pre-Challenge

## Problem Statement

A software developer, named Mike Gillin, has just moved to Charlottesville. He’s lived in Northern Virginia for the last few years, meaning he doesn’t know all the gems of Charlottesville. Help Mike get to know the best city in Virginia by making him a guide to Charlottesville’s restaurant scene.

**Create a web application that displays nearby restaurants and bars in Charlottesville that are currently open**. It should have a list of restaurants / bars. Each restaurant should display its **rating and price** so that Mike can only go to the best. Initially, we just want you to use React to render this info as a bunch of bullet points, or <**li>’s in React-speak**, but if you figure everything out and want an extra challenge, try using the **Leaflet package** to mark these restaurants on a map.

Restaurant:

-Rating

-Price

Table:

Restaurants

## Technologies

API’s (Application Program Interfaces) are the lifeblood of the modern web. They are what let one website or server communicate with another. Any given page might fire off dozens of API calls to ask various servers for information to display to the user, pull ads to show, or to update servers on how a user has interacted with elements on the page. Learning how to effectively interface with API’s is one of the most important things a budding web developer can do. For this project, you’ll need to **fetch results for Charlottesville from the Google Place Search API.** It’s publically available and free, though you’ll need to sign up for an account to access it. Use an HTTP client, **axios**, to make requests after reading up on requesting data from HTTP API’s. **Axios is a Node package that you can import directly into your React code**. Feel free to ask us about this part – it can be a little tricky the first time.

ReactJS is an exciting new web framework, released by Facebook, that’s changing the way developers create web applications. You can read more about how it works on the React homepage – we highly suggest getting the examples running on your computer and thoroughly understanding them before moving on. After you understand the basics, **create an application with Facebook’s tool, create-react-app**, and start work on your application. You’ll spend this summer coding in React, so we suggest playing with some of the many high quality React tutorials for beginners.

Learning to pick up and use open source libraries as needed is an invaluable tool in a developer’s kit, especially when prototyping. React itself is a Node library, and the way you import and use it is just like any other Node library. Create-react-app will automatically install the “react” package into the folder it creates, but you’ll need to install other Node packages like “axios” into the same folder before you can use them (you’ll do so using **npm**).

A developer’s life is spent in a text editor. We highly suggest using **Visual Studio Code**, a new editor released by Microsoft, as it’s the preferred editor by many frontend developers (including the three instructors of this course). For syntax highlighting, we recommend installing the “**Sublime Babel**” package (by Josh Peng) in Visual Studio Code, and then at the bottom of your .js files you can select the language **JavaScript React (Babel)** to get the syntax highlighting specific to React. We also recommend the **package Prettier** by Esben Petersen, as this will reformat your code/spacing whenever you save (you might also need to go into the **VS settings and switch “editor.formatOnSave” to true**).

## Schedule and Specific Advice

You should spend **Week 1** trying to go over the fundamentals of Node and React. We have lots of resources on Node in the Source (HackCville’s SWE program) Google Drive folders listed below – take a look there to familiarize yourself with what Node and NPM are. To actually install them, if you’re on Mac, you can install [homebrew](https://www.howtogeek.com/211541/homebrew-for-os-x-easily-installs-desktop-apps-and-terminal-utilities/) (only use the first three commands here, once you’ve run “brew doctor” and it’s working, you have homebrew!) and then install node from the Terminal with “brew install node”, which will let you use **“node” and “npm”** from the Terminal. If you’re on Windows, we recommend you install the terminal [GitBash](https://gitforwindows.org/) and then download node directly from their website. At that point, you should be able to use “node” and “npm” with the GitBash terminal. Whether you’re on Mac or Windows, running “node -v” and “npm -v” at this point should display the version of node/npm you downloaded – as long as those commands work, node and npm should be working! You can proceed [to installing create-react-app](https://github.com/facebook/create-react-app) with npm and trying it out.

After using npm to install create-react-app, “cd” into whatever folder you want to create your first app in, and run something like “create-react-app project,” which would create a new folder there called “project” that you can “cd” into (of course, you could have replaced “project” with anything). If you’re not familiar with “cd,” Google around for some basic Unix commands – you’ll need some familiarity with the command line for this. Once you’re in this “project” folder, run **“npm start**” and your web browser should be able to render this React project at “localhost:3000”. Get all this to work, and then start going through React tutorials to familiarize yourself with state, props, components, and React in general. I personally recommend reading through the **official React docs listed** below, and then checking out their tutorial for building **tic-tac-toe**. It’s a nice tutorial that walks you through lots of the fundamental concepts, especially the idea of “lifting up state” as you refactor your code. I recommend writing this in your own React project on your computer instead of their online environment – that way you’ll work through getting the code to work on your machine.

You should spend **Week 2** trying to write the **React components you’ll need for this assignment**. Don’t worry about making the calls to Google Places yet, instead just pass along “dummy data” that you hardcode in and make sure you can get the bullet points rendering with that test data. **This dummy data should be an array of JavaScript Objects (like JSONs), where each object has the info relevant to the restaurant.** You want to use the **.map()** in your **render()** method to map over this array of objects and **return an array of React components (probably <li>’s),** each of which represents a single bullet point of info, which you can then reference in the return section of your render() method. The function you pass into .map() needs to be using [arrow syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions) for this to work. [This stackoverflow response](https://stackoverflow.com/questions/32157286/rendering-react-components-from-array-of-objects) should be useful.

­­­­­­­­­­­­­

Finally, devote **Week 3** to grabbing the data you need from Google Places and passing it into the component(s) you’ve made. First, you’ll probably want to get a **Google Places API Key**, then figure out the URL to access the data you want using [their docs](https://developers.google.com/places/web-service/search). Once you have the right URL, you can just load it up in a web browser and you’ll see the JSON. In the react folder, you can use “**npm install --save axios**” to install the axios package, which lets you make HTTP/API requests. From there, use **axios.get(<url>)** to grab that same data. You’ll probably want to use this axios request in a **componentDidMount() method** (one of the React component lifecycle methods you can read about in their docs), and once it has the data, update the state of your component using **this.setState()** with that data. Note that **axios.get()** is asynchronous, and so you’ll need to use the .then() and .catch() [Promise-based syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises) to access its response and use it to update the state. This new data should be passed into the components you made in Week 2. You might run into problems with CORS, or Cross-Origin Resource Sharing – [here’s a Chrome extension](https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfghkpbjhddihlkkiljbi?hl=en) to get around this.

## Useful Links

<https://reactjs.org/> (official React docs/tutorial)

<https://github.com/axios/axios> (the node package you’ll use to grab data from Google Places)

<https://code.visualstudio.com/> (the text editor we recommend, although feel free to use another like Sublime if you’re more comfortable)

<https://developers.google.com/places/web-service/search> (the Google Places API docs).

[https://medium.mybridge.co/learn-react-js-from-top-45-tutorials-for-the-past-year-v-2018-28b7f4](https://medium.mybridge.co/learn-react-js-from-top-45-tutorials-for-the-past-year-v-2018-28b7f4d4b2c4)

[D4b2c4](https://medium.mybridge.co/learn-react-js-from-top-45-tutorials-for-the-past-year-v-2018-28b7f4d4b2c4) (big smattering of tutorials/resources)

<https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82> (basic overview of what API’s are)

<https://drive.google.com/open?id=1GBf37wj7UPT4AqWb8WlWnHR1nsKwTn2c> (resources from HackCville’s Source program Spring 2018, including slides on Node, HTTP/API, and React)

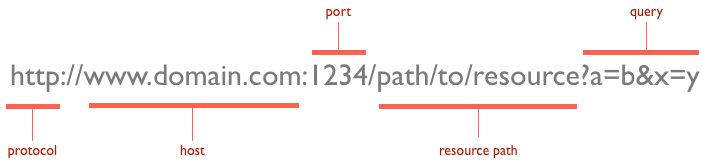
<https://github.com/mdcrawford/source> (GitHub repo from Source program Spring 2018, has plenty of Node examples)

<https://drive.google.com/open?id=0B6SZBCxp8p2hMzBvR1B5MnFYMHM> (resources from HackCville’s Source program Fall 2017, including slides on Node, HTTP/API, and React)

<https://github.com/xiexieeric/source> (GitHub repo from Source Fall 2017, has lots of Node and React examples)

## Final Hints

**HOW HTTP REQUESTS ARE STRUCTURED (you’ll need this for your Google Places search – don’t forget your API key needs to be in the query, should just be key=your-key-here**



1. Protocol 2) host 3) port 4) resource path 5) query

Also, when at your React app in a Google Chrome at localhost:3000, you can right-click and click “Inspect” to pull up the developer tools. Most significantly, the “Console” tab here is where you’ll find anytime your code runs console.log() on something – **this is your best friend for testing and debugging**.

## General Encouragement

Above all else, don’t be afraid to google, copy, paste, delete, rearrange, and tweak until this application is complete. Most of the resources you need should be here, but there’s probably lots you’ll need to Google on your own to figure things out. The goal of this project is exposure, not perfect understanding, so if, at the end of the day, you have an application that works, you should be proud. We’ll dive into many of the details and concepts in further detail this summer.

**“Don’t let perfect get in the way of good.”**