

# WEB422 Assignment 4

## Submission Deadline:

Friday, March 8<sup>th</sup> 2019 @ 11:59pm

## Assessment Weight:

9% of your final course Grade

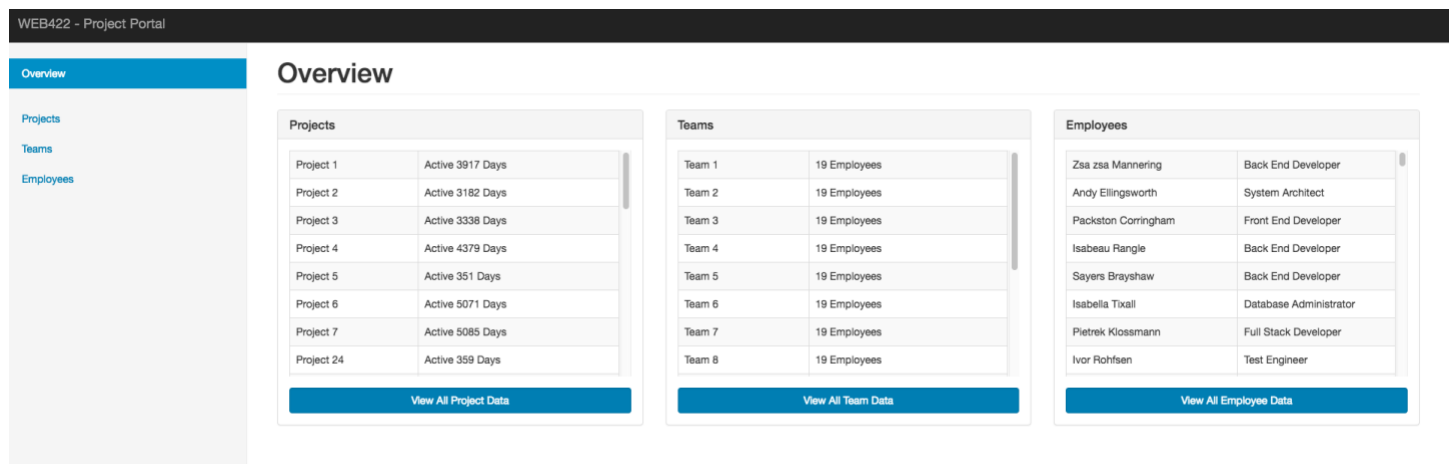
## Objective:

To work with React and practice working with Component based architecture. We will continue to connect to our Teams API as a data source for our app. For an exemplar on how to retrieve Teams API data (using fetch and the "componentDidMount" lifecycle method), refer to the Week 5 example here:

<https://github.com/sictweb/web422/blob/master/Code%20Examples/week5/my-app/src/Employees/Employees.js>

## Specification:

For this application, we will be working entirely in React. Our application will feature 4 routes: Overview, Projects, Teams & Employees. Each route will be responsible for rendering information from our Teams API (often in a <table> format) - we will *not* be updating the data at this time using forms. When complete, the main structure of the app and the "Overview" view will look like the following:



Please Note, once the app is working as expected, please feel free to **add any extra design, Images or CSS to your solution**. Be creative - this is your app.

## Getting Started:

To get started, create a new React app using the command "create-react-app" - if "create-react-app" is not available on your system, you can use the command "npm install -g create-react-app".

Once you have created your new React app - open the folder in Visual Studio Code and run the command "npm start" to get the React development server running. This should open your default web browser and show you <http://localhost:3000>:



At this point, you may go ahead and **delete the files**:

- App.test.js
- App.css

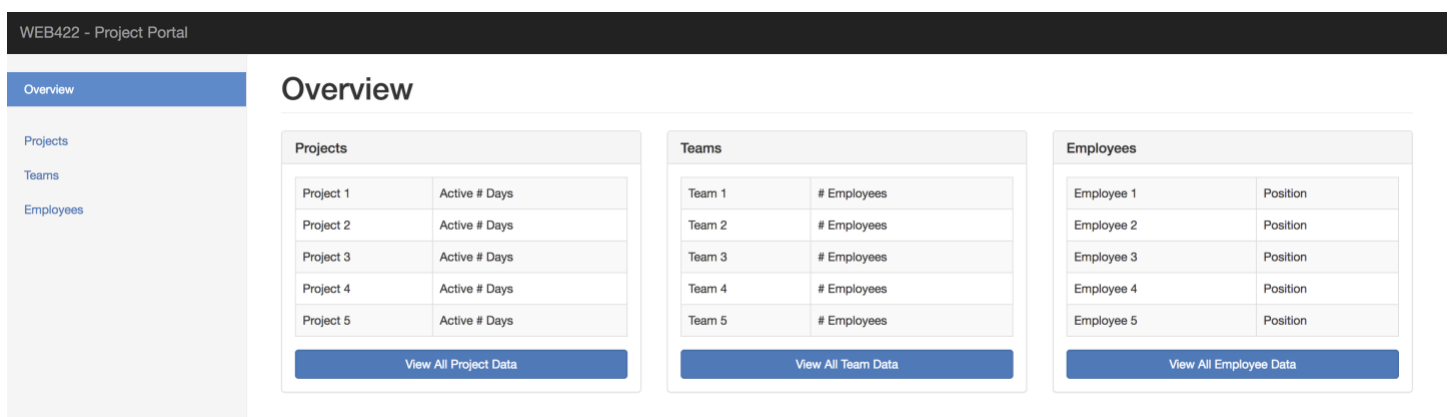
Next, use the code at the following links to **replace** your App.js & index.css files (this will give you a solid starting point and a design to work from):

- <https://scs.senecac.on.ca/~patrick.crawford/shared/winter-2019/web422/A4/App.js>
- <https://scs.senecac.on.ca/~patrick.crawford/shared/winter-2019/web422/A4/index.css>

Once this is complete, you need to update your "public/index.html" file to include the following <link> & <script> tags in their appropriate positions, ie: <link> element **before** '<link rel="manifest" href="%PUBLIC\_URL%/manifest.json">' ) and <script> elements **before** </body>:

- <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
- <script src="https://code.jquery.com/jquery-3.2.1.min.js" integrity="sha256-hwg4gsxgFZzOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4=" crossorigin="anonymous"></script>
- <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IbQib027qvyjSMfHjOMaLkfuWVxZxUPnPnAJA7l2mCWNIpG9mGCD8wGNlCPD7Txa" crossorigin="anonymous"></script>

When you have made the above updates, your app should look like the following:



## Dependencies:

This project will make use of several additional dependencies available from npm including:

- moment: <https://www.npmjs.com/package/moment>
- react-router-dom: <https://www.npmjs.com/package/react-router-dom>

At this point, it makes sense to "npm install --save ..." each of the above modules, since you will need them in your application

## Creating Your Components:

As we know from class, the main building block for all React applications, is the "Component". The following sections of the specification will outline each component that must be built to get your application working properly.

**NOTE:** All Components created for this application must exist **in their own file** (ie: ComponentName.js). The First Step is to create the following components based on the existing code within your App.js. We will shrink down the "App" component to use all of the new components

### NavBar

The NavBar is the simplest component. It simply renders the <nav> element from our current "App".

### SideBar

The SideBar component simply renders the "sidebar" element from our current App, ie: the "<div className='col-sm-3 col-md-2 sidebar'>...</div>" element. Additionally, it accepts the following property:

- "highlight" - this property determines which "nav-sideber" <li> element is "active" (it could be "Overview", "Projects", "Teams" or "Employees". For example, if the "highlight" property is "Projects", then the <li><a href="/projects">Projects</a></li> will get the class "active"

ie: <li class="active"><a href="/projects">Projects</a></li>

To conditionally add the class "active" to an <li> element (for the "Projects" view), you can use the code:

**<li className={this.props.highlight === "Projects" ? 'active' : ''}>**

### MainContainer

The MainContainer component wraps everything that a page needs into its own component. This includes a <NavBar /> and <SideBar /> component. It must account for any "children" that may be added to the component (using this.props.children). Additionally, it accepts the following property:

- "sidebar" - this property determines the value that needs to be passed as the "highlight" property for the <SideBar /> element. This way, we can pass a "sidebar" value to the "MainContainer" component and be assured that the matching "SideBar" will be highlighted.

When completed, your "MainContainer" should include the (structural) code (see comments for hints):

```
<div>
  <NavBar />
  <div className="container-fluid">
    <div className="row">
```

```

        <SideBar /> // Add the correct "highlight" property here
        <div className=" col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
          // be sure to add a reference to the "children" here
        </div>
      </div>
    </div>
  </div>

```

If you've updated your code correctly so far, your `<App>` component's `render()` method should return something like this

```

<MainContainer>
  <h1 className="page-header">Overview</h1>
  <div className="row">
    ...
  </div>
</MainContainer>

```

## ProjectsPanel

You will notice that there's a `<div className="panel panel-default">` that has the "panel-title" of "Projects". This should really be its own "ProjectsPanel" component, since it's dealing with data. For this component, you will need to include the following module

- "moment"

Additionally, this component must adhere to the following specifications:

- Has a "state" that includes a "projects" array
- Uses "fetch" to pull all projects from your Teams API (Heroku) and assign the results to the component state "projects" array (**Hint:** Perform this operation in the "componentDidMount()" lifecycle method)
- Renders a "Panel", where the "panel-title" is "Projects"
- The "panel-body" is a Bootstrap (responsive) table with 2 columns:
  - Project Name
  - Active # Days (HINT: You can use the [moment\(\).diff\(\)](#) method with the 'days' option here)
- Under the table, the panel must continue to include the "View All Project Data" link in the "panel-body"

## TeamsPanel

You will notice that there's a `<div className="panel panel-default">` that has the "panel-title" of "Teams". This should really be its own "TeamsPanel" component, since it's dealing with data.

This component must adhere to the following specifications:

- Has a "state" that includes a "teams" array
- Uses "fetch" to pull all teams from your Teams API (Heroku) and assign the results to the component state "teams" array (**Hint:** Perform this operation in the "componentDidMount()" lifecycle method)
- Renders a "Panel", where the "panel-title" is "Teams"
- The "panel-body" is a Bootstrap (responsive) table with 2 columns:

- Team Name
- # Employees (this is the number of employees on the team)
- Under the table, the panel must continue to include the "View All Team Data" link in the "panel-body"

## EmployeesPanel

You will notice that there's a `<div className="panel panel-default">` that has the "panel-title" of "Employees". This should really be its own "EmployeesPanel" component, since it's dealing with data.

Additionally, this component must adhere to the following specifications:

- Has a "state" that includes a "employees" array
- Uses "fetch" to pull all employees from your Teams API (Heroku) and assign the results to the component state "employees" array (**Hint:** Perform this operation in the "componentDidMount()" lifecycle method)
- Renders a "Panel", where the "panel-title" is "Employees"
- The "panel-body" is a Bootstrap (responsive) table with 2 columns:
  - Full Employee Name (First Name + Last Name)
  - Position Name (ie: "Full Stack Developer")

Under the table, the panel must continue to include the "View All Employee Data" link in the "panel-body"

## Overview

If you have been updating your `<App>` component to use your new components, its "render" method should return the following (this is much more optimized than what was originally provided):

```
<MainContainer>
  <h1 className="page-header">Overview</h1>
  <div className="row">
    <div className="col-md-4">
      <ProjectsPanel />
    </div>
    <div className="col-md-4">
      <TeamsPanel />
    </div>
    <div className="col-md-4">
      <EmployeesPanel />
    </div>
  </div>
</MainContainer>
```

Since this is really the "Overview" view, it should be placed in its own "Overview" component. The only change that needs to be made from the above JSX is that we must provide the text "Overview" to the "MainContainer"'s "sidebar" property, ie: `<MainContainer sidebar="Overview">...</MainContainer>` to ensure that the correct "sidebar" gets highlighted.

NOTE: For this component, you will need to include the following modules

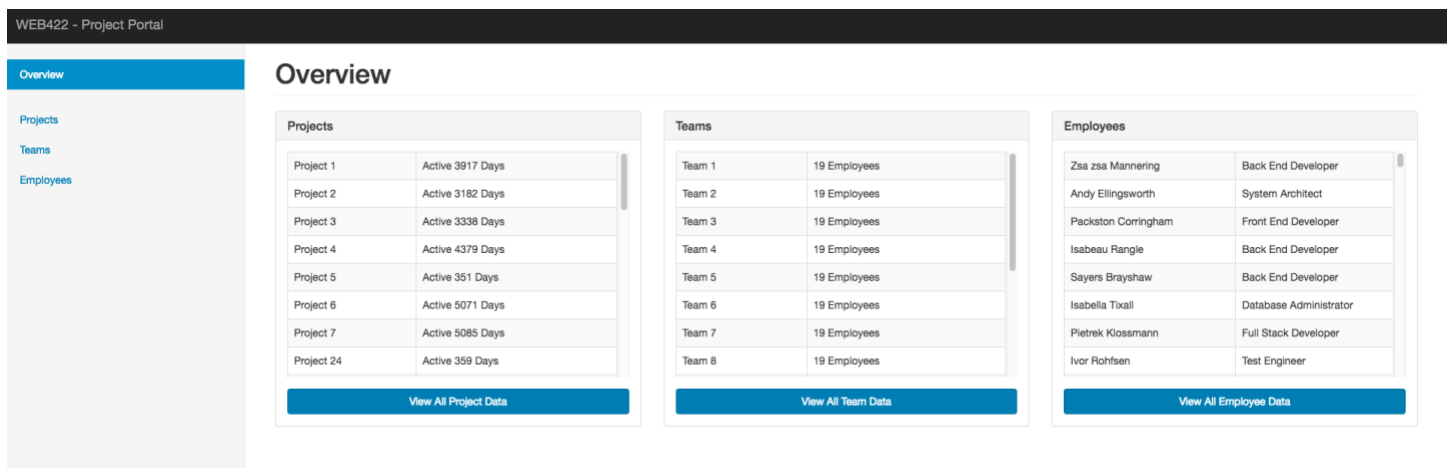
- "MainContainer"
- "ProjectsPanel"

- "TeamsPanel"
- "EmployeesPanel"

Once this is complete, feel free to replace the entire `return()` statement for the `<App>` component, with your new "Overview" component for testing, ie:

```
class App extends Component {
  render() {
    return (
      <Overview />
    );
  }
}
```

Take a look in the browser - this should give you a working "Overview":



## Creating Your Components pt II:

Now that your main building blocks are constructed, it's time to add the high level views (used in your routes), including:

### Projects

Like the "Overview" route - this route's `render()` method will be wrapped in a `<MainContainer>...</MainContainer>` Component. However instead of `sidebar="Overview"`, the sidebar property will instead pass "Projects", ie `<MainContainer sidebar="Projects">...</MainContainer>`.

Also, the "page-header" must read "Projects".

This Component must also adhere to the below specifications:

- For this component, you will need to include the following modules
  - "moment"
  - "MainContainer"
- This component has a "state" that includes a "projects" array

- Uses "fetch" to pull all projects from your Teams API (Heroku) and assign the results to the component state "projects" array (**Hint:** Perform this operation in the "componentDidMount()" lifecycle method)
- It must render a responsive bootstrap table: `<table className="table table-striped table-bordered">...</table>` with the following columns:
  - "Name" (the Project Name)
  - "Description" (the Project Description)
  - "Start Date" (the Project Start Date - HINT: Use "moment" to format the date using "LL")
  - "End Date" (the Project End Date - if this doesn't exist or is null, simply render "n/a", otherwise use "moment" to format the date using "LL")
- Once Complete, your component should look like:

WEB422 - Project Portal

Overview

Projects

Teams

Employees

## Projects

Name	Description	Start Date	End Date
Project 1	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque lobortis vel nunc non tincidunt. Proin elementum facilisis ipsum id tincidunt. Phasellus ut orci placerat, cursus ante sed, feugiat elit. Nullam at velit metus. Morbi suscipit fringilla tellus, id tristique massa mollis a. Cras non tincidunt diam. Morbi rutrum enim eget facilisis aliquet. Ut mattis euismod fermentum. Vestibulum ut tincidunt purus, et porttitor mi.	January 17, 2007	n/a
Project 2	Nunc at imperdiet purus. Nullam tincidunt orci nibh, eget pellentesque metus rutrum tincidunt. Donec diam purus, dictum non mollis id, facilisis at urna. Nunc euismod bibendum ipsum sed pellentesque. Proin faucibus urna nisi, quis vulputate dolor facilisis vitae. Donec enim magna, posuere id hendrerit sit amet, elementum id dui. Mauris lectus ligula, volutpat eget lacus non, lobortis vestibulum enim. Fusce id pretium risus. Morbi in porttitor arcu, vitae malesuada nunc. Vestibulum mattis bibendum ligula et congue. Pellentesque tempor magna ut magna tempus ullamcorper.	January 21, 2009	n/a
Project 3	Morbi aliquam sodales fringilla. Praesent eget ultricies dolor, vestibulum sollicitudin nulla. Morbi ut luctus magna, vel gravida nunc. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nam eu mollis nunc. Cras ac metus nulla. Quisque non urna id neque tristique vestibulum. Aliquam at sem at sem facilisis molestie a nec turpis. Nulla facilisi. Nulla et risus non urna maximus facilisis. Mauris vulputate est at nunc semper placerat.	August 18, 2008	n/a
Project 4	Nullam sit amet dapibus ligula. Donec convallis neque ut consequat laoreet. Mauris vulputate rhoncus erat, et suscipit ipsum tristique et. Aenean venenatis pretium velit, at semper quam dictum quis. Fusce elementum dictum diam, quis consequat erat feugiat sed. Quisque tortor enim, dignissim ac auctor sit amet, dapibus vitae justo. Suspendisse pharetra feugiat massa, vel fermentum arcu tincidunt a. Donec a placerat enim. Mauris convallis orci sed neque consequat iaculis. Curabitur maximus, leo at euismod mattis, ipsum tortor tristique nulla, ultrices semper eros ipsum nec ex. Nunc vitae tempor urna. Ut tincidunt orci ac laoreet vehicula. Morbi sit amet convallis risus. Quisque sed leo sed justo mattis consectetur at vitae risus. Sed quis tortor porttitor, venenatis mi in, porttitor quam. Integer sit amet dolor bibendum, auctor turpis vel, mattis quam.	October 12, 2005	n/a
Project 5	Interdum et malesuada fames ac ante ipsum primis in faucibus. Sed nec viverra ex. Etiam massa urna, aliquam at malesuada sed, suscipit a neque. Sed laoreet ullamcorper erat ultricies tincidunt. Proin lobortis velit vel sem varius eleifend. Ut vitae purus turpis. Nulla condimentum, ex sed fringilla commodo, feila diam consequat justo, nec efficitur augue dolor ac lacus. Donec dictum auctor mi faucibus consectetur.	October 22, 2016	n/a
Project 6	Nullam faucibus vestibulum lorem. In mauris justo, faucibus in dignissim sit amet, euismod et ligula. Etiam sed elit nec ante cursus pellentesque. Cras arcu tellus, malesuada eu ex et, facilisis lacinia justo. Nam tempor mi dictum, posuere nunc ut, eleifend metus. Phasellus auctor vestibulum elit, at tincidunt nibh vehicula luctus. Donec ac nibh justo. Nullam luctus, lectus vitae maximus egestas, ligula feila suscipit nisi, vel eleifend purus leo a libero. Aliquam posuere facilisis dictum. Etiam eu placerat quam. Maecenas vel efficitur massa, quis rhoncus tortor. Aenean in sollicitudin dolor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur varius venenatis fringilla.	November 20, 2003	n/a
Project 7	Aenean tristique molestie nisi, non posuere orci sollicitudin nec. Vestibulum id nibh augue. Nam iaculis fermentum mauris, ac interdum massa luctus a. Curabitur a rhoncus purus. Vivamus vel urna ac odio viverra egestas. Ut a nisi in nisi aliquam molestie ac in urna. Nunc sit amet justo eget nisi elementum vestibulum. Sed blandit eros ullamcorper arcu placerat rutrum. Praesent dignissim ligula quis nulla porta	November 6, 2003	n/a

## Teams

Like the "Overview" route - this route's "render()" method will be wrapped in a `<MainContainer>...</MainContainer>` Component. However instead of `sidebar="Overview"`, the `sidebar` property will instead pass "Teams", ie `<MainContainer sidebar="Teams">...</MainContainer>`.

Also, the "page-header" must read "Teams".

This Component must also adhere to the below specifications:

- For this component, you will need to include the following module
  - "MainContainer"
- This component has a "state" that includes a "teams" array
- Uses "fetch" to pull all teams from your Teams API (Heroku) and assign the results to the component state "teams" array (**Hint:** Perform this operation in the "componentDidMount()" lifecycle method)
- It must render a responsive bootstrap table: `<table className="table table-striped table-bordered">...</table>` with the following columns:
  - "Name" (the Team Name)
  - "Projects" (an "unordered list" of all projects that the team is responsible for)
  - "Employees" (The number of employees on the team)
  - "Team Lead" (The full name (First & Last) of the Team Lead)

- Once Complete, your component should look like:

WEB422 - Project Portal

Overview

Projects

**Teams**

Employees

## Teams

Name	Projects	Employees	TeamLead
Team 1	<ul style="list-style-type: none"> <li>Project 21</li> <li>Project 22</li> </ul>	19 Employees	Zsa zsa Mannering
Team 2	<ul style="list-style-type: none"> <li>Project 23</li> <li>Project 25</li> </ul>	19 Employees	Rex Anster
Team 3	<ul style="list-style-type: none"> <li>Project 26</li> <li>Project 27</li> </ul>	19 Employees	Lonee Kilbourn
Team 4	<ul style="list-style-type: none"> <li>Project 28</li> <li>Project 30</li> </ul>	19 Employees	Rafi Malicki
Team 5	<ul style="list-style-type: none"> <li>Project 1</li> <li>Project 2</li> </ul>	19 Employees	Cleveland Jacob
Team 6	<ul style="list-style-type: none"> <li>Project 3</li> <li>Project 4</li> </ul>	19 Employees	Meade Zuker
Team 7	<ul style="list-style-type: none"> <li>Project 5</li> <li>Project 6</li> </ul>	19 Employees	Maxi Cowperthwaite

## Employees

Like the "Overview" route - this route's "render()" method will be wrapped in a `<MainContainer>...</MainContainer>` Component. However instead of `sidebar="Overview"`, the `sidebar` property will instead pass "Employees", ie `<MainContainer sidebar="Employees">...</MainContainer>`.

Also, the "page-header" must read "Employees".

This Component must also adhere to the below specifications:

- For this component, you will need to include the following modules
  - "moment"
  - "MainContainer"
- This component has a "state" that includes an "employees" array
- Uses "fetch" to pull all employees from your Teams API (Heroku) and assign the results to the component state "employees" array (**Hint:** Perform this operation in the "componentDidMount()" lifecycle method)
- It must render a responsive bootstrap table: `<table className="table table-striped table-bordered">...</table>` with the following columns:
  - "Name & Position" (The full name and position of the employee)
  - "Address" (The full address of the employee)
  - "Phone Num" (The phone number + extension of the employee)
  - "Hire Date" (The hire date of the employee (HINT: Use "moment" to format the date using "LL"))
  - "Salary Bonus" (The employee's salary bonus, including a "\$")
- Once Complete, your component should look like:



Overview

Projects

Teams

Employees

## Employees

Name & Position	Address	Phone Num	Hire Date	Salary Bonus
Zsa zsa Mannering - Back End Developer	7471 Burning Wood Crossing, Santa Monica CA, 90410	1-(310)552-1997 ex: 1	November 7, 2010	\$ 24901
Andy Ellingsworth - System Architect	947 Lake View Parkway, Fresno CA, 93715	1-(559)533-3179 ex: 2	June 23, 2008	\$ 11219
Packston Corringham - Front End Developer	9619 Lakewood Gardens Avenue, North Hollywood CA, 91606	1-(818)413-1751 ex: 3	July 7, 2014	\$ 12955
Isabeau Rangle - Back End Developer	3 American Parkway, San Diego CA, 92160	1-(619)381-2818 ex: 4	December 20, 2015	\$ 15146
Sayers Brayshaw - Back End Developer	8 Grim Parkway, San Francisco CA, 94105	1-(415)683-0534 ex: 5	March 31, 2007	\$ 438
Isabella Tixall - Database Administrator	1705 Rusk Point, San Diego CA, 92170	1-(619)503-8372 ex: 6	November 26, 2006	\$ 2787
Pietrek Klossmann - Full Stack Developer	14996 Swallow Hill, San Francisco CA, 94142	1-(415)352-8964 ex: 7	April 20, 2007	\$ 19241
Ivor Rohlsen - Test Engineer	160 Wayridge Lane, Van Nuys CA, 91411	1-(826)934-1269 ex: 8	February 9, 2003	\$ 20632
Marya Springins - Database Administrator	37935 Morningstar Pass, Los Angeles CA, 90101	1-(213)918-4029 ex: 9	February 7, 2005	\$ 7456
Cherilyn Paula - Front End Developer	86 Rigney Hill, Pasadena CA, 91131	1-(826)376-3431 ex: 10	January 29, 2014	\$ 4791
Adrian Fromant - Test Engineer	4 Springs Crossing, Fullerton CA, 92635	1-(714)348-7536 ex: 11	June 20, 2005	\$ 7823
Helene Burchill - Test Engineer	3 Thierer Place, Fresno CA, 93778	1-(559)798-7463 ex: 12	November 9, 2013	\$ 20531
Verise Oak - Front End Developer	9592 Forest Run Alley, Ventura CA, 93005	1-(805)860-0667 ex: 13	October 29, 2003	\$ 8387

## NotFound

The "NotFound" is a simple route that we will use whenever a route is not identified within the app. Like the "Overview" route - this route's "render()" method will be wrapped in a `<MainContainer>...</MainContainer>` Component (without a "sidebar" property).

Also, the "page-header" must read "Not Found" and the text below should simply be a `<span>` element with the text: "Page Not Found".

Once Complete, your component should look like:

Overview

Projects

Teams

Employees

## Not Found

Page Not Found

## Creating Routes

All of our custom components are finally complete. Now we must register the routes so that our application knows how to handle navigation between routes.

- Update your "index.js" `ReactDOM.render()` line to wrap your `<App />` component in a `<BrowserRouter>` element, ie:

```
<BrowserRouter>
  <App />
</BrowserRouter>
```

- (Don't forget to import `{BrowserRouter}` from 'react-router-dom')

Next, we must update our "App" component to register the routes according to the following specification (NOTE: Refer to the [Week 5 notes on "React Routing"](#) for further details on how to wire the below routes):

- `"/"` will render `<Overview />`

- `"/projects"` will render `<Projects />`
- `"/teams"` will render `<Teams />`
- `"/employees"` will render `<Employees />`
- Any other Route will render `<NotFound />`

## Updating all internal Links

The last step is to ensure that all `<a href="..."></a>` elements use the `<Link>` component to prevent the whole page from reloading between routes. This can be done by simply updating any anchor elements that refer to internal routes, ie:

- Everywhere you see `<a href="..."></a>`, replace the code with `<Link to="..."></Link>`

This should eliminate any full-page reloads between routes.

**HINT:** Do not forget to add the line: `"import {Link} from 'react-router-dom';"` in any components that require the "Link" component

## Assignment Submission:

- Add the following declaration at the top of your App.js file:

```

/*****
* WEB422 – Assignment 04
* I declare that this assignment is my own work in accordance with Seneca Academic Policy. No part of this
* assignment has been copied manually or electronically from any other source (including web sites) or
* distributed to other students.
*
* Name: _____ Student ID: _____ Date: _____
*
*****/

```

- Compress (.zip) the all files in your Visual Studio code folder **EXCEPT the node\_modules folder** (this will just make your submission unnecessarily large, and all your module dependencies should be in your package.json file anyway).
- Submit your compressed file (without the node\_modules folder) to My.Seneca under **Assignments -> Assignment 4**

## Important Note:

- **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a **grade of zero (0)**.
- After the end (11:59PM) of the due date, the assignment submission link on My.Seneca will no longer be available.
- Submitted assignments must run locally, ie: start up errors causing the assignment/app to fail on startup will result in a **grade of zero (0)** for the assignment.