Lab 8

Start Assignment

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

CS-546 Lab 8

Template Time

For this lab, you will be using HTML, CSS, and Handlebars to make your first simple templated web application! You will be building a form that allows you to search through the following dataset (https://gist.githubusercontent.com/robherley/5112d73f5c69a632ef3ae9b7b3073f78/raw/24a7e1453e65a26a8aa12cd0fb266ed9679816aa/people.json) .

YOU MUST use the directory and file structure in the code stub or points will be deducted. You can download the starter template here: **Lab8_Stub.zip** (https://sit.instructure.com/courses/61549/files/10464240/download?download_frd=1)

PLEASE NOTE: THE STUB DOES NOT INCLUDE THE PACKAGE.JSON FILE. YOU WILL NEED TO CREATE IT! DO NOT ADD ANY OTHER FILE OR FOLDER APART FROM PACKAGE.JSON FILE.

You will be making three pages in your application:

- `http://localhost:3000/` the main page of this application will provide a search form to start a search on the dataset
- `http://localhost:3000/searchpeople` this page will search through the `people.json` dataset and return up to 20 matching results (Sorted by ID), where either the `firstName` or `lastName` contains the provided request form param, `searchPersonName`
- `http://localhost:3000/persondetails/:id` this page will show all the details of the person with the id matching the provided URL param, `id`

All other URLS should return a 404

`GET http://localhost:3000/`

This page will respond with a valid HTML document. The title of the document should be "*People Finder*". You should have the title set as the `<title>` element of the HTML document and as an `h1` in

your document.

Your page should reference a CSS file, `/public/main-styles.css`; this file should have *at least 10 rulesets* that apply to this page; these 10 rules can also apply to elements across all of your pages, or be unique to this page.

You should have a `main` element, and inside of the `main` element have a `p` element with a brief (2-3 sentence description) of what your website does.

Also inside the `main` element, you will have a `form`; this `form` will `POST` to `/searchpeople`. This `form` will have an `input` and a `label`; the `label` should properly reference the same `id` as the `input`. You should also have a `button` with a type of `submit` that submits the form. The `input` in your `form` should have a `name` of `searchPersonName`.

`POST http://localhost:3000/searchpeople`

This page will respond with a valid HTML document. The title of the document should be "*People Found*". You should have the title set as the `<title>` element of the HTML document and as an `h1` in your document. In an `h2` element, you will print the supplied `searchPersonName`.

Your page should reference a CSS file, `/public/main-styles.css`; this file should have *at least 10 rulesets* that apply to this page; these 10 rules can also apply to elements on `/`, or be unique to this page.

You should have a `main` element, and inside of the `main` element have a `ul` tag that has a list of up to 20 people (Sorted by ID) matching the `searchPersonName` found in the request body in the following format (after searching `Rob`).

```
<ul>
    <li>
        <a href="/persondetails/1">Rob Herley</a>
    </li>
    <li>
        <a href="/persondetails/99">Dolph Robottham</a>
    </li>
    <li>
        <a href="/persondetails/270">Grady Orrobin</a>
    </li>
</ul>
```

You must also provide an `a` tag that links back to your `/` route with the text `Make another search`.

If no matches are found, you will print the following HTML paragraph:

```
<p class="not-found">We're sorry, but no results were found for {{searchPersonName}}.</p>
```

If the user does not input text into their form, make sure to give a response status code of 400 on the page, and render an HTML page with a paragraph class called `error`; this paragraph

should describe the error.

`GET http://localhost:3000/persondetails/:id`

This page will respond with a valid HTML document. The title of the document should be "*Person Found*". You should have the title set as the `<title>` element of the HTML document and as an `h1` in your document. In an `h2` element, you will print the name of the person found.

Your page should reference a CSS file, `/public/main-styles.css`; this file should have *at least 10 rulesets* that apply to this page; these 10 rules can also apply to elements on `/`, or be unique to this page.

You should have a `main` element, and inside of the `main` element have a `dl` tag that has a definition list of all the properties of the matching person in the following HTML structure.

Matching Person:

```
{
  "id": 1,
  "firstName": "Robert",
  "lastName": "Herley",
  "address": "1 Castle Point",
  "zip": "07030",
  "phone": "(631) 8967161",
  "ssn": "123-45-6789"
}
```

HTML Printed:

```
 <dl>
  <dt>ID</dt>
  <dd>1</dd>
  <dt>First Name</dt>
  <dd>Robert</dd>
  <dt>Last Name</dt>
  <dd>Herley</dd>
  <dt>Address</dt>
  <dd>1 Castle Point</dd>
  <dt>Zip Code</dt>
  <dd>07030</dd>
  <dt>Phone</dt>
  <dd>(631) 8967161</dd>
  <dt>Social Security Number</dt>
  <dd>123-45-6789</dd>
</dl>
```

If the user does not input text into their form, make sure to give a response status code of 400 on the page, and render an HTML page with a paragraph class called `error`; this paragraph should describe the error.

`http://localhost:3000/public/main-styles.css`

This file should have 10 rulesets that apply to the `/` route, and 10 rulesets that apply to all of your pages. Rulesets may be shared across both pages; for example, if you styled a `p` tag, it would count as 1 of the 10 for both pages.

You may include more than 10 rulesets if you so desire.

References and Packages

Basic CSS info can easily be referenced in the MDN CSS tutorial (https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started) .

Hints

You can use variables in your handlebars layout, that you pass to res.render. For example, in your layout you could have:

```
<meta name="keywords" content="{{keywords}}" />
```

And in your route:

```
res.render("someView", {keywords: "dogs coffee keto"});
```

Which will render as:

```
<meta name="keywords" content="dogs coffee keto" />
```

Or, perhaps, the title tag.

Requirements

1. You **must not submit** your node_modules folder
2. You **must remember** to save your dependencies to your package.json folder
3. You must do basic error checking in each function
4. Check for arguments existing and of proper type.
5. Throw if anything is out of bounds (ie, trying to perform an incalculable math operation or accessing data that does not exist)
6. If a function should return a promise, instead of throwing you should return a rejected promise.
7. You **must remember** to update your package.json file to set `app.js` as your starting script!
8. Your HTML must be valid (https://validator.w3.org/#validate_by_input) or you will lose points on the assignment.
9. Your HTML must make semantical sense; usage of tags for the purpose of simply changing the style of elements (such as `i`, `b`, `font`, `center`, etc) will result in points being deducted; think in

terms of content first, then style with your CSS.

10. **You can be as creative as you'd like to fulfill front-end requirements**; if an implementation is not explicitly stated, however you go about it is fine (provided the HTML is valid and semantical). Design is not a factor in this course.
11. All inputs must be properly labeled!
12. All previous requirements about the `package.json` author, start task, dependenices, etc. still apply