Back End 1

Overview

1. Create a folder called **server**
2. Inside the server folder create an **index.js** file
3. Use npm to install express, then require it at the top of your index.js file
4. Create a variable called app and set its value equal to express invoked
5. Set your server up to accept JSON object responses
6. Set your express server to listen to requests on port 4000, test with nodemon
7. Create a get request for the endpoint '/api/users' and another to get weather information
8. Start your server up and check its functionality using the given HTML file

Detailed Instructions

Step 1

- download the materials from Frodo and open the folder in VS Code
- create a folder called `server` in your lab exercise folder

Step 2

- inside the `server` folder you just made, create a JavaScript file named `index.js`

Step 3

- in order to get our server working we will need to use Node Package Manager to install two packages.
  - open up your terminal window and install `express` and `cors` using this command:

```
npm i express cors
```

- We will need to import the `express` package into our `index.js` file next
  - inside `index.js`, import `express` like so:

```
const express = require("express");
```

- import the `cors` package next, similar to how you imported `express`
  - inside `index.js`, import `cors` like so:

```
const cors = require("cors");
```

Note: Cors

Cors is a package that allows the client and server to communicate with each other without the need for advanced configuration.

Step 4

Next, to avoid repeating lengthy code, we will create a variable called `app` that we will reuse to initialize express commands. Set `app` equal to the invocation of express

- on the next line in `index.js`, set the invocation of `express` equal to a variable called `app` like this:

```
const app = express();
```

Step 5

Our client and server will be communicating by sending and receiving JSON objects to each other. In this step, we will configure our express server to use JSON objects correctly.

- on the next line in `index.js`, use the `use` method from `express` to invoke `express.json`

```
app.use(express.json());
```

Step 6

Similar to Step 5, we will also need to allow our express server to use the cors package we required.

- on the next line in `index.js`, allow your express app to use cors functionality like this:

  ```
  app.use(cors());
  ```

Step 7

Let's now tell `express` to set our server up to run (or "listen") on `port 4000`

- on the next line, set your `express` server up to listen to requests on `port 4000`

```
app.listen(4000, () => console.log("Server running on port 4000"));
```

- Use the nodemon command in your terminal to get your new server listening on port 4000
- in terminal, use the command to check that your server is working correctly so far

`nodemon server/index`

Step 8

We will create first endpoint and method to handle sending a friends array back to the client (front end)

- in `index.js`, above the listen but below any middleware (`app.use` calls are middleware), create the follow GET endpoint and method as follows:

```
app.get("/api/users", (req, res) => {
  let friends = ["Nitin", "Eric", "Jeddy", "Cameron", "Riley"];
  res.status(200).send(friends);
});
```

- Let's add another endpoint and method that will tell us how the weather is today
- on the next line, create another GET endpoint and method as follows:

```
app.get("/weather/:temperature", (req, res) => {
  const phrase = `<h3>It was ${req.params.temperature} today</h3>`;
  res.status(200).send(phrase);
});
```

With object destructuring, we can make the above code a little easier to read

```
app.get("/weather/:temperature", (req, res) => {
  const { temperature } = req.params;
  const phrase = `<h3>It was ${temperature} today</h3>`;
  res.status(200).send(phrase);
});
```

Step 9

- Launch the index.html file in your browser and test out your endpoints by clicking the "GET Friends List" button or by navigating to either http://localhost:4000/weather/hot or to http://localhost:4000/weather/cold

Submit

Initialize a git repo containing all your work and push it to GitHub.