Lab 2

For this lab you will be utilizing the code you made in Lab 1. Copy all files from the lastname_lab1 folder you created into a lastname_Lab2 folder. Once complete open a terminal window and run npm update to ensure all libraries have been copied over. Go into the package.json file and change the name and description to lab2. Also, on the html pages, make sure the title is set to Lab 2. The purpose of this lab is to add search pages for the employee and customer tables in the database. You will need to create links on the html of the insert pages for the search pages. The added links, shown on the insertemployees.html page are below. Repeat this process on the insert customer html page.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Lab 2</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/react/0.14.0/react.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/react/0.14.0/react-dom.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.6.15/browser.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/marked/0.3.2/marked.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/classnames/2.1.5/index.min.js"></script>
</head>
<body>
    <nav>
        <a href="insertcustomer.html">Insert Customer</a>
        <a href="insertemployee.html">Insert Employee</a>
        <a href="searchcustomer.html">Search Customer</a>
        <a href="searchemployee.html">Search Employee</a>
    </nav>
    <div id="content"></div>
    <script type="text/babel" src="scripts/insertemployees.js">
    </script>
</body>
</html>
```

Once you have that complete, you will need to create 2 new html pages, one called searchemployee.html and one called searchcustomer.html. The link to the js on these pages will need to be changed to the js for the search functionality. An example of this for the search employee page is below. Perform a similar process for the search customer page.

```
    <div id="content"></div>
    <script type="text/babel" src="scripts/searchemployees.js">
    </script>
```

Once the html pages are created, the js pages will need to be created. For this lab, the searchemployees.js page has been provided. Place that into the scripts directory. The code for this page will be discussed during the class period. However, that will just create the form, the code to actually make the information access the database will be housed in the server.js file. Go to the server.js file, and below where the default page to open is set, add in the code below the red line in the image. Note that your line numbers may be slightly different from mine. The code on line 32 creates a method that will get information, and has the identifier of getemp. Lines 33 to 37 will create local variables that can be accessed from the referring js page.

```
28 app.get('/', function (req, res) {
29     res.sendFile(path.join(__dirname + "/public/insertemployee.html"));
30 });
31
32 app.get('/getemp/', function (req, res) {
33     var eid = req.query.employeeid;
34     var ename = req.query.employeename;
35     var ephone = req.query.employeephone;
36     var eemail = req.query.employeeemail;
37     var esalary = req.query.employeesalary;
38
```

The next set of code will create a SQL statement on lines 39 and 40. Line 41 will create the array of values which will be placed into the ? spots in the SQL statement. Notice the '%', these are the wildcard values from SQL so that if the value is anywhere in the field it will still show up. Line 43 merges the array and the SQL statement and prepares it to be run. Line 45 is not required, it outputs the prepared statement to the console so if there is an issue you can see what the issue may be.

```
39     var sqlsel = 'Select * from employeetable where dbemployeeid Like ? and dbemployeename Like ? and dbemployeephone Like ?'
40         + ' and dbemployeeemail Like ? and dbemployeesalary Like ?';
41     var inserts = ['%' + eid + '%', '%' + ename + '%', '%' + ephone + '%', '%' + eemail + '%', '%' + esalary + '%'];
42
43     var sql = mysql.format(sqlsel, inserts);
44
45     console.log(sql);
46
```

The next set of code is below, this will run the query, throwing an error if there is one. Line 52 sends the data retrieved back to the js page as a data array.

```
47     con.query(sql, function (err, data) {
48         if (err) {
49             console.error(err);
50             process.exit(1);
51         }
52         res.send(JSON.stringify(data));
53     });
54 });
```

From here, the page is now complete and you can run npm start to check to see if the data from your database shows up on the screen. You can also search the page to see if that functions. This will get the employee page running for searches. The second part of this lab is to get the customer page running. You will need to create a searchcustomers.js page and also add code similar to the code created above on the server.js page for getcust. Once complete, zip your files and upload to the dropbox in D2L.

If your pages do not function, remember, ctrl+shift+J in Chrome can show many errors, especially with typos. In addition, errors may also show up on the console as well.