Lab 3

For this lab we will be adapting the previous pages for searching and adding radio buttons and a drop down list to the pages. The drop down list will be populated from a table from the database as well. The first item that needs to be done is the database needs to have tables changed and a new one added. The first change is to add the fields shown below in red to the employee table, which is one field for a mailing list, and one to show the employee type. The mailer will be the radio buttons; the type will be the drop down list.

☐	1	**dbemployeekey** 🔑	int(11)		No	*None*
☐	2	**dbemployeeid**	varchar(10)	utf8_unicode_ci	No	*None*
☐	3	**dbemployeename**	varchar(50)	utf8_unicode_ci	No	*None*
☐	4	**dbemployeeemail**	varchar(50)	utf8_unicode_ci	No	*None*
☐	5	**dbemployeephone**	varchar(12)	utf8_unicode_ci	No	*None*
☐	6	**dbemployeesalary**	double(10,2)		No	*None*
☐	7	**dbemployeemailer**	tinyint(1)		No	*None*
☐	8	**dbemployeetype**	tinyint(4)		No	*None*

Next, a new table will need to be created named employeetypes. The fields needed are shown below, note the id field is auto-increment and the primary key.

☐	1	**dbemptypeid** 🔑	int(11)		No	*None*	AUTO_INCREMENT
☐	2	**dbemptypename**	varchar(30)	utf8_unicode_ci	No	*None*	

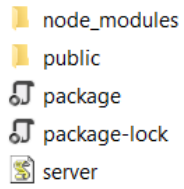
You will need to add a couple records to this table so that items show up, the two records below were added.

☐	Edit Copy Delete	1	Manager
☐	Edit Copy Delete	2	Staff

Finally, you need to edit some of the records in your employee table to give values to the 2 new fields that were added, I have an example of the values given below

dbemployeesalary	dbemployeemailer	dbemployeetype
123123.00	0	1
12345111.00	0	1
123456.00	0	1

From here we can start working with the code. Go into the lab 2 folder and select all the folders and files below and copy into a new folder called lastname_lab3.



Once the files are copied, open the lastname_lab3 folder in VS Code. Go into the package file and change the lab2 to lab3. Also, be sure to go into all the html pages and change the title from lab2 to lab3. We will start on the searchemployees.js page. Since we will have a radio button, we will need code to let us determine when the page is selected which one of the radio buttons is checked. Add the code between the red lines below, our line numbers may not match exactly, but this will be right near the top of the program. This creates a variable called emailervalue which will hold the value of the radio button. It is set to 2 by default until it is determined which value has been selected. The if statements check the variables we will give to each radio button to see if they are checked. If they are checked, then it sets them to 1 for yes and 2 for no respectively.

```
6       console.log(employeeid.value);
7       var emailervalue = 2;
8       if (empmaileryes.checked) {
9           emailervalue = 1;
10      }
11      if (empmailerno.checked) {
12          emailervalue = 0;
13      }
14
15      $.ajax({
```

Since we will have additional variables in this program, we will next need to add variables to the data array we create. Add the lines in red below, which will add the mailer and type variables with the given values to the array

```
23              'employeesalary': employeesalary.value,
24              'employeemailer': emailervalue,
25              'employeetype': emptype.value
26          },
```

We will also need to add the new variable names to the header of our output table, which is done in the code below.

```
56                  <th>Salary</th>
57                  <th>Mailing List</th>
58                  <th>Type</th>
59              </tr>
```

Next, the two new variables will need to be added to the form. The code below will add a blank value for the employeeMailer variable. Where this changes from the usual variable, is that the drop down list will require a set of values gained from the employeetypes table that was created earlier. That is the reason for the data array variable that will be utilized on line 78. Do not forget to add the comma at the end of line 76.

```
76          employeesalary: "",
77          employeeMailer: "",
78          data: []
79      };
```

Since we will have a drop down and radio button now, there is a new object we will create that can handle what will happen when options change inside the program. The function below will handle what occurs when an option is changes. We set the state of the selected option to whatever the value is of the item that was clicked on.

```
80      },
81      handleOptionChange: function (e) {
82          this.setState({
83              selectedOption: e.target.value
84          });
85      },
```

As we get the information from the server for the employee types, we will need to load these employee types from the server. The code below first creates a function to do this. Then is will run an ajax function that accesses a function we will create on the server.js page called getemptypes. This rest of the code below is very similar to the code used previously to load the search from the database.

```
86      loadEmpTypes: function () {
87          $.ajax({
88              url: '/getemptypes',
89              dataType: 'json',
90              cache: false,
91              success: function (data) {
92                  this.setState({ data: data });
93              }.bind(this),
94              error: function (xhr, status, err) {
95                  console.error(this.props.url, status, err.toString());
96              }.bind(this)
97          });
98      },
```

The function above creates the function to load the employee types, the code below will actually mount and run the code above.

```
99      componentDidMount: function () {
100         this.loadEmpTypes();
101
102     },
```

Similar to before, we will now need to add the variables to the program when the submit button is hit. Lines 111 and 112 create the variables and get their basic information. Lines 120 and 121 will allow these values to be passed along to the server.js page. Lines 114 to 121 have previously been all on one line, they have been separated out for ease of visibility.

```
110         var employeesalary = this.state.employeesalary;
111         var employeemailer = this.state.selectedOption;
112         var employeetype = emptype.value;
113
114         this.props.onEmployeeSubmit({
115             employeeid: employeeid,
116             employeename: employeename,
117             employeeemail: employeeemail,
118             employeephone: employeephone,
119             employeesalary: employeesalary,
120             employeemailer: employeemailer,
121             employeetype: employeetype
122         });
123
```

Now the fields need to be added to the form itself. Below the red line, add the code to create a new table row and header, that says join mailing list. The code from lines 172 to 179 create an input field that is a radio button, with an id of emaileryes and value of 1. Line 177 will change the value of selectedOption to 1 if it is clicked. Line 178 will activate the handleOptionChange method from before if clicked. Line 179 is just for css if needed. Line 180 shows the word Yes on the screen.

```
165                         </td>
166                     </tr>
167                     <tr>
168                         <th>
169                             Join Mailing List
170                         </th>
171                         <td>
172                             <input
173                                 type="radio"
174                                 name="empmailer"
175                                 id="empmaileryes"
176                                 value="1"
177                                 checked={this.state.selectedOption === "1"}
178                                 onChange={this.handleOptionChange}
179                                 className="form-check-input"
180                             />Yes
```

The code below does the same as was done for the radio button above, but for the No option. Lines 190 and 191 will close the table data and table row.

```
181                         <input
182                           type="radio"
183                           name="empmailer"
184                           id="empmailerno"
185                           value="0"
186                           checked={this.state.selectedOption === "0"}
187                           onChange={this.handleOptionChange}
188                           className="form-check-input"
189                         />No
190                     </td>
191                 </tr>
```

Next the drop down list will be created. A new table row and header and made once again. For the data output, a SelectList object will be created using the data we will retrieve from the server. The SelectList object will be created further along the program.

```
192                 <tr>
193                     <th>
194                         Employee Type
195                     </th>
196                     <td>
197                         <SelectList data={this.state.data} />
198                     </td>
199                 </tr>
```

The next item is to add the new fields to the employeeList function that will handle output. This creates the variable we will use for output and links it back to the respective field from the database.

```
220                     empsalary={employee.dbemployeesalary}
221                     empmailer={employee.dbemployeemailer}
222                     emptype={employee.dbemptypename}
223                 >
```

Similar to when we send the information to the database from above, we need code to determine if the user wished to be on the mailing list or not. Since the radio buttons give a value of 1 or 0 to the database, we need if statements to see what the value in the database is so we can then output the correct information to the screen. The code below creates a variable to be used for the output called themailer.

```
242     render: function () {
243
244         if (this.props.empmailer == 1) {
245             var themailer = "YES";
246         } else {
247             var themailer = "NO";
248         }
249
250         return (
```

Next, the 2 new variables need to be output to the output for the list of items returned from the database.

```
270                     </td>
271                     <td>
272                         {themailer}
273                     </td>
274                     <td>
275                         {this.props.emptype}
276                     </td>
277                 </tr>
```

We will now create the SelectList function. This will create the options for a select list using the key and value for the items returned from the employeetypes table. This is similar to outputs utilized above for the search outputs. The return statement creates the option tag that goes in the select statement.

```
280 });
281
282 var SelectList = React.createClass({
283     render: function () {
284         var optionNodes = this.props.data.map(function (empTypes) {
285             return (
286                 <option
287                     key={empTypes.dbemptypeid}
288                     value={empTypes.dbemptypeid}
289                 >
290                     {empTypes.dbemptypename}
291                 </option>
292             );
293         });
```

The next code will create the select statement itself with the name and id of emptype. The optionNodes created above are then added on line 297. This will conclude all items needed for the searchemployees.js page.

```
294         return (
295             <select name="emptype" id="emptype">
296                 <option value = "0"></option>
297                 {optionNodes}
298             </select>
299         );
300     }
301 });
```

We will now need to adapt the server.js page to access the information from the employeetypes table. Line 32 below creates the function getemptypes we used in the js page above. This creates the sql statement to select all from the table on line 34, then formats the query on line 35. Lines 37 through 40 attempt to run the query on the connection, and throw an error if there is one. Line 43 will return the values if the query is successful.

```
30 });
31
32 app.get('/getemptypes/', function (req, res) {
33
34     var sqlsel = 'select * from employeetypes';
35     var sql = mysql.format(sqlsel);
36
37     con.query(sql, function (err, data) {
38         if (err) {
39             console.error(err);
40             process.exit(1);
41         }
42
43         res.send(JSON.stringify(data));
44     });
45 });
46
47 app.get('/getemp/', function (req, res) {
```

We now need to add the variables to the getemp function as well. They are added in the code below, making sure to match the variables from the searchemployees.js page

```
52     var esalary = req.query.employeesalary;
53     var emailer = req.query.employeemailer;
54     var etype = req.query.employeetype;
```

Where the search function will get tricky, is that if something is not selected with input boxes, the like statements will still allow the query to run successfully. However, with drop downs and radio buttons, you cannot run the like command in the same way. So, the query itself will change depending on whether or not someone chose the drop down or radio button. Lines 56 and 57 are just there to ensure the data is coming over correctly. Line 59 is the if statement to see if either of the variables for the radio buttons are set. If so, SQL code is created and placed into a variable on line 60 to add the field to the select statement in SQL. A variable is also created to be used in the associative array on line 61. The else statement will just create a standard like clause since nothing was selected.

```
55
56      console.log("Mailer: "+  emailer);
57      console.log("Type: " + etype);
58
59      if (emailer == 1 || emailer == 0) {
60          var maileraddon = ' and dbemployeemailer = ?';
61          var maileraddonvar = emailer;
62      } else {
63          var maileraddon = ' and dbemployeemailer Like ?';
64          var maileraddonvar = '%%';
65      }
66
```

A similar process is then done for the employee type, this time checking to see if the value is above zero.

```
67      if (etype > 0) {
68          var typeaddon = ' and dbemployeetype = ?';
69          var typeaddonvar = etype;
70      } else {
71          var typeaddon = ' and dbemployeetype Like ?';
72          var typeaddonvar = '%%';
73      }
74
```

The SQL query and array need to be modified like below to add in the additional info on the sql query and the variables in the array. The variables do not need the %% since they are looking for a specific value since they were drop downs or radio buttons.

```
75      var sqlsel = 'Select employeetable.*,  employeetypes.dbemptypename from employeetable ' +
76          ' inner join employeetypes on employeetypes.dbemptypeid = employeetable.dbemployeetype ' +
77      ' where dbemployeeid Like ? and dbemployeename Like ? and dbemployeephone Like ? '
78      + ' and dbemployeeemail Like ? and dbemployeesalary Like ?' + maileraddon + typeaddon;
79
80      var inserts = ['%' + eid + '%', '%' + ename + '%', '%' + ephone + '%', '%'
81          + eemail + '%', '%' + esalary + '%', maileraddonvar, typeaddonvar];
82
83      var sql = mysql.format(sqlsel, inserts);
```

This will complete the search employees page. The second part of the lab is to utilize the customer page you created previously and adapt it to perform a similar task. You will need to add a field for a drop down for the customer to determine their rewards status. This will come from a table you will need to create in your database called customerewards. There should be silver, gold and platinum as the rewards types, though more may be added in the future. For the radio button, you will ask if the user wishes to be a discount club participant, or just a normal participant. The options for the radio buttons should say discount member or standard member. You will then add the fields to the search page, similar to how they were added to the employee page. The code to make the search function on the customer page will also need to be on the server.js page as well. When complete, zip your project and upload to the dropbox in D2L.