Lab 4

For this lab we will be adapting the previous pages for inserting 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. Please ensure that you have modified the table for employees from Lab 3 as well as added the table required from Lab 3. We will be adding the functionality for the radio buttons and drop down list to the insert employees page. Create a new project named lastname_Lab4, and copy all the files into it from Lab3. Be sure to change the title on the html pages from Lab3 to Lab4, as well as in the package.json file.

We will start by adding the variables needed for the form to the getInitialState function. Add lines 38 and 39 below to your code, making a variable for the employeeMailer as well as the array that will hold the information from the database to populate the drop down. Note your lines numbers may vary slightly from mine.

```
37          employeesalary: "",
38          employeeMailer: "",
39          data: []
40      };
```

Next, the code below the red line below will be added, this will allow the value of the items to be updated when changes have been made. The handleOptionChange function will set the state of the selectedOption to whatever item has been clicked on.

```
39          data: []
40      };
41  },
42  handleOptionChange: function (e) {
43      this.setState({
44          selectedOption: e.target.value
45      });
46  },
```

Next the function to load the employee types from the database will be created. This is similar to the code created in Lab3, note the url is /getemptypes, which will already be in the database due to it being created in Lab3, so that will not have to be recreated.

```
47      loadEmpTypes: function () {
48          $.ajax({
49              url: '/getemptypes',
50              dataType: 'json',
51              cache: false,
52              success: function (data) {
53                  this.setState({ data: data});
54              }.bind(this),
55              error: function (xhr, status, err) {
56                  console.error(this.props.url, status, err.toString());
57              }.bind(this)
58          });
59      },
```

Once the function to load the information from the database is created, it must also be called, which is what the code below will do. If the database is successfully accessed, the function will be called.

```
60      componentDidMount: function () {
61          this.loadEmpTypes();
62      },
```

Additional variables must be added for when the submit button is selected. The two lines below the red line will need to be added to get the values of the mailer and type.

```
71          var employeesalary = this.state.employeesalary;
72          var employeemailer = this.state.selectedOption;
73          var employeetype = emptype.value;
```

The variables will also need to be added to the function that is called when the submit button is hit. The code below adds the 2 lines needed for the mailer and type. Note that this code may have previously all been on one line, it has been expanded so each variable is now on their own line.

```
84          this.props.onEmployeeSubmit({
85              employeeid: employeeid,
86              employeename: employeename,
87              employeeemail: employeeemail,
88              employeephone: employeephone,
89              employeesalary: employeesalary,
90              employeemailer: employeemailer,
91              employeetype: employeetype
92          });
```

Similar to the search page, the radio buttons and drop downs now need to be added to the form itself. The code below creates the table row and table data, then creates the first radio button.

```
<tr>
    <th>
        Join Mailing List
    </th>
    <td>
        <input
            type="radio"
            name="empmailer"
            id="empmaileryes"
            value="1"
            checked={this.state.selectedOption === "1"}
            onChange={this.handleOptionChange}
            className="form-check-input"
        />Yes
```

Once that is created, the second radio button must be created, then the table data and table row will be closed.

```
        <input
          type="radio"
          name="empmailer"
          id="empmailerno"
          value="0"
          checked={this.state.selectedOption === "0"}
          onChange={this.handleOptionChange}
          className="form-check-input"
        />No
    </td>
</tr>
```

The drop down is then created, with a table row and data surrounding it. The SelectList function being called has not been created yet, that will be added in the next step. This code for this part of the form is exactly the same as what was shown on the form for the search page in Lab3.

```
<tr>
    <th>
        Employee Type
    </th>
    <td>
        <SelectList data={this.state.data} />
    </td>
</tr>
```

After the closing tags for the form, the SelectList function will now be created. This will allow up to select the items from the database and populate the select statement and option values contained within. This code is the same as the one from the search page in Lab3, as the drop down for this table will stay the same even on different pages.

```
231  });
232
233  var SelectList = React.createClass({
234      render: function () {
235          var optionNodes = this.props.data.map(function (empTypes) {
236              return (
237                  <option
238                      key={empTypes.dbemptypeid}
239                      value={empTypes.dbemptypeid}
240                  >
241                      {empTypes.dbemptypename}
242                  </option>
243              );
244          });
245          return (
246              <select name="emptype" id="emptype">
247                  {optionNodes}
248              </select>
249          );
250      }
251  });
```

That is the last of the code required for the insertemployees.js page, there are some adaptations that will need to be made on the server.js page to add those 2 variables. Depending on your server.js page, your line numbers may vary, but you will be making changes to the /employee function on the server.js page. The lines below the red line below will be added so that the variables passed from the form can be utilized.

```
119  app.post('/employee', function (req, res,) {
120
121      var eid = req.body.employeeid;
122      var ename = req.body.employeename;
123      var ephone = req.body.employeephone;
124      var eemail = req.body.employeeemail;
125      var esalary = req.body.employeesalary;
126      var emailer = req.body.employeemailer;
127      var etype = req.body.employeetype;
128
```

Previously the insert statement did not utilize a prepared statement, so the entire SQL statement will need to be changed, with an array added to hold the variables. The code between the lines will need to be added, with the previous SQL statement removed. In addition, on line 136, since a prepared statement is being used, the con.query that was previously there needs to be changed to a con.execute. Since these fields are required, the process of checking to see if the variables are there are not required as they were in the search page.

```
128
129     var sqlins = "INSERT INTO employeetable (dbemployeeid, dbemployeename, " +
130         "dbemployeeemail, dbemployeephone, dbemployeesalary, " +
131         "dbemployeemailer, dbemployeetype) VALUES(?, ?, ?, ?, ?, ?, ?)";
132     var inserts = [eid, ename, ephone, eemail, esalary, emailer, etype];
133
134     var sql = mysql.format(sqlins, inserts);
135
136     con.execute(sql, function (err, result) {
137         if (err) throw err;
```

This will complete the insert employees page. The second part of the lab is to utilize the insert 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 the table you created in Lab3 called customerewards. 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 insert customer page, similar to how fields were added to the employee page. The code to make the insert 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.