Requirement 2 - Build Frontend React App for Employee Management Module

User should able perform below CRUD operations:

- Add New Employee
- List All Employees
- Update Existing Employee
- Delete Existing Employee

Command to create react app => **npm create vite@latest ems-frontend**

App start run on http://localhost:5173/, we can change the port in vite.config.js file –

```
lcome    vite.config.js ×    inde
rontend > vite.config.js > ...
import { defineConfig } from "v
import react from "@vitejs/plug

// https://vite.dev/config/
export default defineConfig({
  plugins: [react()],
  server: {
    port: 3000,
  },
});
```

Command to start App => npm run dev

Node_modules – whenever we run npm install then npm will create this folder and keep all the downloaded JS libraries and packages inside it.

Public folder – we keep all images and static files like images.

Index.css – styles for index.html app.css – styles applied globally for all components

Main.jsx – main entry point for Js related code. Whenever we run our application, then index.html will get served in browser and it internally call main.jsx file –

```
main.jsx ×    index.html
ems-frontend > src > main.jsx
5
6   createRoot(document.getElementById("root")).render(
7     <StrictMode>
8       <App />
9     </StrictMode>
10  );
```

getting the div with id = “root” and render App component within it.

App.jsx – this file contains code for App component (root component)

Adding bootstrap in project => npm install bootstrap –- save

import "bootstrap/dist/css/bootstrap.min.css" in main.jsx file

Create React ListEmployeeComponent and Display Data

Development Steps

1. Create React Functional Component - **ListEmployeeComponent**
2. Prepare Dummy Data (List of Employees) to Display in an HTML Table
3. Write JSX Code to Display List of Employees in HTML Table
4. Import and Use **ListEmployeeComponent** in **App** Component
5. Run and Test React App

Keys in React are special attributes used to identify the items in a list.

```
const numbers = [1, 2, 3, 4, 5];
const updatedNums = numbers.map((number, index) =>
    <li key={index}>
        {number}
    </li>
);
```

```
ListEmployeeComponent.jsx ×
omponents > ListEmployeeComponent.jsx > ListEmployeeComponent
import React from "react";

const ListEmployeeComponent = () => {
  const dummyData = [
    { ...
    },
    { ...
    },
    { ...
    },
  ];
  return (
    <div className="container">
      <h2 className="text-center">List of Employees</h2>
      <table className="table table-striped table-bordered">
        <thead>
          <tr>
            <th>Employee Id</th>
            <th>Employee First Name</th>
            <th>Employee Last Name</th>
            <th>Employee Email Id</th>
          </tr>
        </thead>
        <tbody>
          {dummyData.map((employee) => (
            <tr key={employee.id}>
              <td>{employee.id}</td>
              <td>{employee.firstName}</td>
              <td>{employee.lastName}</td>
              <td>{employee.email}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};
```

```
App.jsx ×   ListEmployeeComponent.jsx
src > App.jsx > App
1   import "./App.css";
2   import ListEmployeeComponent fro
3
4   function App() {
5     return (
6       <>
7         <ListEmployeeComponent />
8       </>
9     );
10  }
```

Connect React App with Get All Employees REST API

Development Steps

1. Install **axios** Library
2. Create **EmployeeService.js** File
3. Write REST Client code to make a REST API call using **axios** API
4. Change **ListEmployeeComponent** to Display Response of the REST API (List of Employees)

Inorder to hold the response of REST Api we have to use State variable. In functional component we use useState hook to define state variables.

Inorder to make REST Api call in functional component we use useEffect hook. It takes 2 arguments – callback function and dependencies list. The useEffect hook is triggered automatically **after the component is rendered**.

App.jsx | EmployeeService.js | ListEmployeeComponent.jsx

src > services > EmployeeService.js > ...

```
import axios from "axios";

const REST_API_BASE_URL = "http://localhost:8080/api/employees";

export const listEmployees = () => axios.get(REST_API_BASE_URL);

```

```
const ListEmployeeComponent = () => {
  const [employees, setEmployees] = useState([]);

  useEffect(() => {
    listEmployees()
      .then((reponse) => {
        setEmployees(reponse.data);
      })
      .catch((error) => {
        console.error(error);
      });
  }, []);

  return (
    <div className="container">
      <h2 className="text-center">List of Employees</h2>
      <table className="table table-striped table-bordered">
        <thead>
          <tr>···
          </tr>
        </thead>
        <tbody>
          {employees.map((employee) => (
            <tr key={employee.id}>
              <td>{employee.id}</td>
              <td>{employee.firstName}</td>
              <td>{employee.lastName}</td>
              <td>{employee.email}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};
```

On running app, we get error – “Access to XMLHttpRequest at 'http://localhost:8080/api/employees' from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.”

This is due to Cross-Origin Resource sharing policy which is enforcesd by browser for security. This happens when your React application (running on http://localhost:3000) tries to make a request to your Spring Boot backend (http://localhost:8080), which are considered different origins.

We can handle such CORS issue at springboot backend –

EmployeeController.java

```java
@CrossOrigin(origins = "http://localhost:3000")
@AllArgsConstructor
@RestController
@RequestMapping("/api/employees")
public class EmployeeController {
```

“*” means all the clients/origin can call employee controller REST APIs

We can also specify origins like - @CrossOrigin(origins = "http://localhost:3000")

Adding Header and Footer to React App

Development Steps

1. Create HeaderComponent (functional component)
2. Import and Use HeaderComponent in App Component
3. Create FooterComponent (functional component)
4. Import and Use FooterComponent in App Component

```jsx
const HeaderComponent = () => {
  return (
    <div>
      <header>
        <nav
          className="navbar bg-dark border-bottom border-body"
          data-bs-theme="dark"
        >
          <a className="navbar-brand" href="#">
            Employee Management System
          </a>
        </nav>
      </header>
    </div>
  );
};
```

```jsx
function App() {
  return (
    <>
      <HeaderComponent />
      <ListEmployeeComponent />
      <FooterComponent />
    </>
  );
}
```

```
const FooterComponent = () => {
  return (
    <div>
      <footer className="footer">
        <span>All Rights Reserved 2024 by Arjun</span>
      </footer>
    </div>
  );
};
```

Configure Routing in a React App

Development Steps

1. Install **react-router-dom** library using NPM
2. Configure Routing in **App** Component
3. Configure Route for **ListEmployeeComponent**

Inorder to configure routing, we have to use routing related components from react-router-dom

Steps – 1) enclose all the components within BrowserRouter component.

2) Routes is basically a container or a parent for all the individual routes. Within this Routes, we can define the individual Routes –

```
import { BrowserRouter, Routes, Route } from "react-router-dom";

function App() {
  return (
    <>
      <BrowserRouter>
        <HeaderComponent />
        <Routes>
          {/* http://localhost:3000 */}
          <Route path="/" element={<ListEmployeeComponent />}></Route>
          {/* http://localhost:3000/employees */}
          <Route path="/employees" element={<ListEmployeeComponent />}></Route>
        </Routes>
        <FooterComponent />
      </BrowserRouter>
    </>
  );
}
```

Create EmployeeComponent and Configure Route

Development Steps

1. Create React Functional Component - **EmployeeComponent**
2. Add "**Add Employee**" button in **ListEmployeeComponent**
3. Configue Route for **EmployeeComponent**

In order to navigate user form one page to another, we can use useNavigate() hook.

```
const EmployeeComponent = () => {
  return <div>EmployeeComponent</div>;
};

export default EmployeeComponent;
```

```
import { useNavigate } from "react-router-dom";

const ListEmployeeComponent = () => {
  const [employees, setEmployees] = useState([]);
  const navigator = useNavigate();

  useEffect(() => {…
  }, []);

  function addNewEmployee() {
    navigator("/add-employee");
  }

  return (
    <div className="container">
      <h2 className="text-center">List of Employees</h2>

      <button className="btn btn-primary mb-2" onClick={addNewEmployee}>
        Add Employee
      </button>

      <table className="table table-striped table-bordered">…
      </table>
    </div>
  );
};
```

```
function App() {
  return (
    <>
      <BrowserRouter>
        <HeaderComponent />
        <Routes>
          {/* http://localhost:3000/add-employee */}
          <Route path="/add-employee" element={<EmployeeComponent />}></Route>
```

Add employee Form Handling

Development Steps

1. Define state variables (firstName, lastName and email) in **EmployeeComponent** using **useState** Hook.
2. Design **Add Employee Form** using HTML and Bootstrap
3. Create JavaScript Function to handle onClick Event (Form submit)

Connect React App to Add Employee REST API

Development Steps

1. In **EmployeeService**, write a code to call **Add Employee REST API** using axios.
2. Change **EmployeeComponent** to call **EmployeeService** method
3. Navigate to List Employees Page After Form Submission Done

Add Employee Form Validations

Development Steps

1. Use the **useState** hook to initialize state variables that will hold validation errors
2. Write a validation function that checks the form data and returns validation errors
3. Validate Form on Submission
4. Display Validation Errors

The useState() hook is commonly used for managing form input state in React. It allows you to declare a state variable and a corresponding setter function to update the value of the state variable. By using useState(), you can keep track of the form input values and update them when the user interacts with the form.

We have to dynamically add the css on input tag.

src > services > EmployeeService.js > ...

```
export const createEmployee = (employee) =>
  axios.post(REST_API_BASE_URL, employee);
```

src > components > EmployeeComponent.jsx > EmployeeComponent

```
const EmployeeComponent = () => {
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [email, setEmail] = useState("");
  const [errors, setErrors] = useState({
    firstName: "",
    lastName: "",
    email: "",
  });

  const navigator = useNavigate();
```

```
function saveEmployee(e) {
  e.preventDefault();

  if (validateForm()) {
    const employee = { firstName, lastName, email };

    createEmployee(employee).then((response) => {
      console.log(response);
      navigator("/employees");
    });
  }
}

function validateForm() {
  let valid = true;
  const errorsCopy = { ...errors };

  if (firstName.trim()) {
    errorsCopy.firstName = "";
  } else {
    errorsCopy.firstName = "First name is required";
    valid = false;
  }

  if (lastName.trim()) {···
  } else {···
  }

  if (email.trim()) {···
  } else {···
  }

  setErrors(errorsCopy);
  return valid;
}
```

```
return (
  <div className="container">
    <br /> <br />
    <div className="row">
      <div className="card col-md-6 offset-md-3 offset-md-3">
        <h2 className="text-center">Add Employee</h2>
        <div className="card-body">
          <form>
            <div className="form-group mb-2">
              <label className="form-label">First Name :</label>
              <input
                type="text"
                placeholder="Enter Employee First Name"
                name="firstName"
                className={`form-control ${
                  errors.firstName ? "is-invalid" : ""
                }`}
                onChange={(e) => setFirstName(e.target.value)}
              ></input>

              {errors.firstName && (
                <div className="invalid-feedback">{errors.firstName}</div>
              )}
            </div>
            <div className="form-group mb-2">···
            </div>
            <div className="form-group mb-2">···
            </div>

            <button className="btn btn-success" onClick={saveEmployee}>
              Submit
            </button>
          </form>
        </div>
      </div>
    </div>
  </div>
);
};
```

Update Employee Feature – Adding Update button, Title and Route

Development Steps

1. Add **Update** button to list employees page
2. Add Route for Update Employee in **App** component
3. Change Page Title Dynamically (**EmployeeComponent** supports both Add and Update)

```
function App() {
  return (
    <>
      <BrowserRouter>
        <HeaderComponent />
        <Routes>
          {/* http://localhost:3000/edit-employee/1 */}
          <Route
            path="/edit-employee/:id"
            element={<EmployeeComponent />}
          ></Route>
```

useParams() hook from react-router-dom library is used to get the query parameters from url. This returns object with key value pair.

src > components > ListEmployeeComponent.jsx > ListEmployeeComponent

```jsx
const ListEmployeeComponent = () => {

  function updateEmployee(id) {
    navigator(`/edit-employee/${id}`);
  }

  return (
    <div className="container">
      <h2 className="text-center">List of Employees</h2>

      <button className="btn btn-primary mb-2" onClick={addNewEmployee}>
        Add Employee
      </button>

      <table className="table table-striped table-bordered">
        <thead>
          <tr>
            <th>Employee Id</th>
            <th>Employee First Name</th>
            <th>Employee Last Name</th>
            <th>Employee Email Id</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {employees.map((employee) => (
            <tr key={employee.id}>
              <td>{employee.id}</td>
              <td>{employee.firstName}</td>
              <td>{employee.lastName}</td>
              <td>{employee.email}</td>
              <td>
                <button
                  className="btn btn-info"
                  onClick={() => updateEmployee(employee.id)}
                >
                  Update
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};
```

src > components > EmployeeComponent.jsx > EmployeeComponent > pageTitle

```jsx
const EmployeeComponent = () => {
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [email, setEmail] = useState("");
  const { id } = useParams();
  const [errors, setErrors] = useState({
    firstName: "",
    lastName: "",
    email: "",
  });

  const navigator = useNavigate();

  function saveEmployee(e) {…
  }

  function validateForm() {…
  }

  function pageTitle() {
    if (id) {
      return <h2 className="text-center">Update Employee</h2>;
    } else {
      return <h2 className="text-center">Add Employee</h2>;
    }
  }

  return (
    <div className="container">
      <br /> <br />
      <div className="row">
        <div className="card col-md-6 offset-md-3 offset-md-3">
          {pageTitle()}
          <div class="card-body">
            <form>
```

Connect React app to Get Employee REST API

Development Steps

1. In **EmployeeService**, write a code to call Get **Employee REST API** using axios.
2. Use **useEffect** hook to populate the employee data in the form for update

src > services > EmployeeService.js > ...

```
export const getEmployee = (employeeId) =>
  axios.get(REST_API_BASE_URL + "/" + employeeId);
```

src > components > EmployeeComponent.jsx > ...

```
const EmployeeComponent = () => {
  useEffect(() => {
    if (id) {
      getEmployee(id)
        .then((response) => {
          setFirstName(response.data.firstName);
          setLastName(response.data.lastName);
          setEmail(response.data.email);
        })
        .catch((error) => {
          console.log(error);
        });
    }
  }, [id]);

  return (
    <div className="container">
      <br /> <br />
      <div className="row">
        <div className="card col-md-6 offset-md-3 offset-md-3">
          {pageTitle()}
          <div className="card-body">
            <form>
              <div className="form-group mb-2">
                <label className="form-label">First Name :</label>
                <input
                  type="text"
                  placeholder="Enter Employee First Name"
                  name="firstName"
                  value={firstName}
                  className={ form-control ${
                    errors.firstName ? "is-invalid" : ""
                  }`}
                  onChange={(e) => setFirstName(e.target.value)}
                ></input>

                {errors.firstName && (
                  <div className="invalid-feedback">{errors.firstName}</div>
                )}
              </div>
```

Connect React App to Update Employee REST API

Development Steps

1. In **EmployeeService**, write a code to call **Update Employee REST API** using axios.
2. Change **EmployeeComponent.saveOrUpdateEmployee()** method to perform both add and update employee operations

---

src > services > EmployeeService.js > updateEmployee

```
13  export const updateEmployee = (employeeId, employee) =>
14    axios.put(REST_API_BASE_URL + "/" + employeeId, employee);
```

src > components > EmployeeComponent.jsx > EmployeeComponent

```
9   const EmployeeComponent = () => {
73    function saveOrUpdateEmployee(e) {
74      e.preventDefault();
75
76      if (validateForm()) {
77        const employee = { firstName, lastName, email };
78
79        if (id) {
80          updateEmployee(id, employee)
81            .then((response) => {
82              console.log(response);
83              navigator("/employees");
84            })
85            .catch((error) => console.log(error));
86        } else {
87          createEmployee(employee)
88            .then((response) => {
89              console.log(response);
90              navigator("/employees");
91            })
92            .catch((error) => console.log(error));
93        }
94      }
95    }
96
97    return (
98      <div className="container">
99        <br /> <br />
100       <div className="row">
101         <div className="card col-md-6 offset-md-3 offset-md-3">
102           {pageTitle()}
103           <div className="card-body">
104             <form>
105 >             <div className="form-group mb-2">···
121               </div>
122 >             <div className="form-group mb-2">···
137               </div>
138 >             <div className="form-group mb-2">···
151               </div>
152
153               <button
154                 className="btn btn-success"
155                 onClick={saveOrUpdateEmployee}
156               >
157                 Submit
158               </button>
159             </form>
```

Delete Employee feature

Development Steps

1. In **EmployeeService**, write a code to call **Delete Employee REST API** using axios.
2. Add **Delete** button to list employees table
3. Create JavaScript function to handle **Delete** button event

src > services > EmployeeService.js > deleteEmployee

```
export const deleteEmployee = (employeeId) =>
  axios.delete(REST_API_BASE_URL + "/" + employeeId);
```

src > Components > ListEmployeeComponent.jsx > ListEmployeeComponent

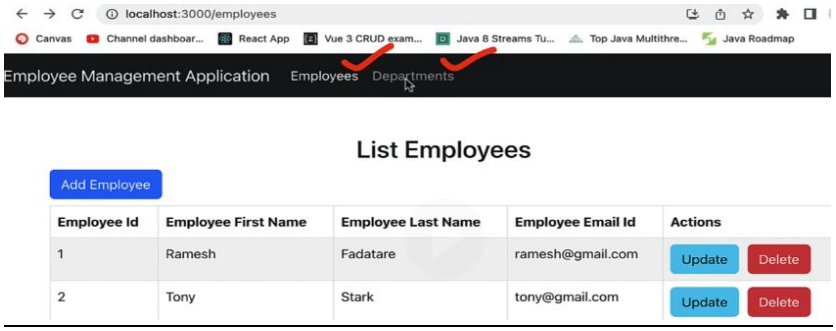
```
const ListEmployeeComponent = () => {

  function removeEmployee(id) {
    console.log(id);
    deleteEmployee(id)
      .then((response) => {
        getAllEmployees();
      })
      .catch((error) => console.log(error));
  }

  return (
    <div className="container">
      <h2 className="text-center">List of Employees</h2>

      <button className="btn btn-primary mb-2" onClick={addNewEmployee}>
        Add Employee
      </button>

      <table className="table table-striped table-bordered">
        <thead>···
        </thead>
        <tbody>
          {employees.map((employee) => (
            <tr key={employee.id}>
              <td>{employee.id}</td>
              <td>{employee.firstName}</td>
              <td>{employee.lastName}</td>
              <td>{employee.email}</td>
              <td>
                <button
                  className="btn btn-info"
                  onClick={() => updateEmployee(employee.id)}
                >
                  Update
                </button>
                <button
                  className="btn btn-danger"
                  onClick={() => removeEmployee(employee.id)}
                  style={{ marginLeft: "10px" }}
                >
                  Delete
                </button>
              </td>
            </tr>
          ))}
        </tbody>
```

Requirement 4 - Build UI for Department Management Module

User should able perform below CRUD operations:

- Add New Department
- List the Departments
- Update existing Department
- Delete existing Department
- Add Employee to a particular Department

Add Navigation Links in Header



Development Steps

1. Add Nav bar link to /employees
2. Add Nav bar link to /departments
3. Create ListDepartmentComponent.js
4. Configure Route for ListDepartmentComponent

We used NavLink component from React-router-dom library to navigate user to the particular page.

src > components > HeaderComponent.jsx > HeaderComponent

```
4   const HeaderComponent = () => {
5     return (
6       <div>
7         <header>
8           <nav
9             className="navbar navbar-expand-lg bg-dark border-bottom border-body"
10            data-bs-theme="dark"
11          >
12            <a className="navbar-brand" href="#">
13              Employee Management System
14            </a>
15
16            <div className="collapse navbar-collapse" id="navbarNavDropdown">
17              <ul className="navbar-nav">
18                <li className="nav-item">
19                  <NavLink className="nav-link" to="/employees">
20                    Employees
21                  </NavLink>
22                </li>
23                <li className="nav-item">
24                  <NavLink className="nav-link" to="/departments">
25                    Departments
26                  </NavLink>
27                </li>
```

src > components > ListDepartmentComponent.jsx > ...

```jsx
import React from "react";

const ListDepartmentComponent = () => {
  return <div>ListDepartmentComponent</div>;
};

export default ListDepartmentComponent;
```

src > App.jsx > App

```jsx
function App() {
  return (
    <>
      <BrowserRouter>
        <HeaderComponent />
        <Routes>
          {/* http://localhost:3000/departments */}
          <Route
            path="/departments"
            element={<ListDepartmentComponent />}
          ></Route>
```

Design List Department Component & Connect with Get All Departments REST API

Development Steps

1. Create DepartmentService.js File
2. Write REST Client code to make a REST API call using axios API
3. Change ListDepartmentComponent to Display Response of the REST API (List of Departments)

src > services > DepartmentService.js > ...

```jsx
import axios from "axios";

const DEPARTMENT_REST_API_BASE_URL = "http://localhost:8080/api/departments";

export const getAllDepartments = () => axios.get(DEPARTMENT_REST_API_BASE_URL);
```

src > components > ListDepartmentComponent.jsx > ListDepartmentComponent

```jsx
const ListDepartmentComponent = () => {
  const [departments, setDepartments] = useState([]);
  useEffect(() => {
    getAllDepartments()
      .then((response) => {
        setDepartments(response.data);
      })
      .catch((error) => console.log(error));
  }, []);
  return (
    <div className="container">
      <h2 className="test-center">List of Departments</h2>
      <table className="table table-striped table-bordered">
        <thead>
          <tr>
            <td>Department Id</td>
            <td>Department Name</td>
            <td>Department Description</td>
          </tr>
        </thead>
        <tbody>
          {departments.map((department) => (
            <tr key={department.id}>
              <td>{department.id}</td>
              <td>{department.departmentName}</td>
              <td>{department.departmentDescription}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};
```

Add Department Feature

Development Steps

1. Create React Functional Component - **DepartmentComponent**
2. Add "**Add Department**" button in **ListDepartmentComponent**
3. Configue Route for **DepartmentComponent**

Inorder to add link, we'll use Link component from react-router-dom library

src > components > DepartmentComponent.jsx > ...

```jsx
import React from "react";

const DepartmentComponent = () => {
  return <div>DepartmentComponent</div>;
};

export default DepartmentComponent;
```

```
src > components > ListDepartmentComponent.jsx > ListDepartmentComponent
5   const ListDepartmentComponent = () => {
6     const [departments, setDepartments] = useState([]);
7     useEffect(() => {…
13    }, []);
14    return (
15      <div className="container">
16        <h2 className="test-center">List of Departments</h2>
17        <Link to="/add-department" className="btn btn-primary mb-2">
18          Add Department
19        </Link>
20        <table className="table table-striped table-bordered">…
38        </table>
39      </div>
40    );
41  };
```

```
src > App.jsx > App
10  function App() {
11    return (
12      <>
13        <BrowserRouter>
14          <HeaderComponent />
15          <Routes>
16            {/* http://localhost:3000/add-department */}
17            <Route
18              path="/add-department"
19              element={<DepartmentComponent />}
20            ></Route>
```

Add Department Form Handling

Development Steps

1. Define state variables (departmentName, departmentDescription) in DepartmentComponent using **useState** Hook.
2. Design Add Department Form using HTML and Bootstrap
3. Create JavaScript Function to handle onClick Event (Form submit)

Connect React to Add Department REST API

Development Steps

1. In DepartmentService, write a code to call add department REST API using axios
2. Change DepartmentComponent to call DepartmentService method
3. Navigate to List Departments Page After Form Submission Done

src > services > DepartmentService.js > createDepartment

```javascript
export const createDepartment = (department) =>
  axios.post(DEPARTMENT_REST_API_BASE_URL, department);
```

src > components > DepartmentComponent.jsx > DepartmentComponent

```jsx
const DepartmentComponent = () => {
  const [departmentName, setDepartmentName] = useState("");
  const [departmentDescription, setDepartmentDescription] = useState("");
  const navigator = useNavigate();

  function saveDepartment(e) {
    e.preventDefault();

    const department = { departmentName, departmentDescription };
    createDepartment(department)
      .then((response) => {
        console.log(response.data);
        navigator("/departments");
      })
      .catch((error) => console.log(error));
  }

  return (
    <div className="container">
      {" "}
      <br />
      <br />
      <div className="row">
        <div className="card col-md-6 offset-md-3 offset-md-3">
          <h2 className="text-center">Add Department</h2>

          <div className="card-body">
            <form>
              <div className="form-group mb-2">
                <label className="form-label">Department Name:</label>
                <input
                  type="text"
                  name="departmentName"
                  placeholder="Enter Department Name"
                  value={departmentName}
                  onChange={(e) => setDepartmentName(e.target.value)}
                  className="form-control"
                />
              </div>
              <div className="form-group mb-2">…
              </div>

              <button
                className="btn btn-success mb-2"
                onClick={(e) => saveDepartment(e)}
              >
                Submit
              </button>
            </form>
```

Adding Update Button, Title and Route

Development Steps

1. Add Update button to list departments page
2. Add Route for Update Department in App component
3. Change Page Title Dynamically (DepartmentComponent supports both Add and Update)

---

src > App.jsx > App

```jsx
function App() {
  return (
    <>
      <BrowserRouter>
        <HeaderComponent />
        <Routes>
          <Route
            path="/edit-department/:id"
            element={<DepartmentComponent />}
          ></Route>
```

src > components > ListDepartmentComponent.jsx > ...

```jsx
const ListDepartmentComponent = () => {
  function updateDepartment(id) {
    navigator(`/edit-department/${id}`);
  }

  return (
    <div className="container">
      <h2 className="text-center">List of Departments</h2>
      <Link to="/add-department" className="btn btn-primary mb-2">
        Add Department
      </Link>
      <table className="table table-striped table-bordered">
        <thead>
          <tr>
            <th>Department Id</th>
            <th>Department Name</th>
            <th>Department Description</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {departments.map((department) => (
            <tr key={department.id}>
              <td>{department.id}</td>
              <td>{department.departmentName}</td>
              <td>{department.departmentDescription}</td>
              <td>
                <button
                  onClick={() => updateDepartment(department.id)}
                  className="btn btn-info"
                >
                  Update
                </button>
              </td>
            </tr>
          ))}
        </tbody>
```

src > components > DepartmentComponent.jsx > DepartmentComponent

```
9   const DepartmentComponent = () => {
12    const navigator = useNavigate();
13    const { id } = useParams();
14
15    function saveDepartment(e) {···
35    }
36
37    function pageTitle() {
38      if (id) {
39        return <h2 className="text-center">Add Department</h2>;
40      } else {
41        return <h2 className="text-center">Edit Department</h2>;
42      }
43    }
44
45    return (
46      <div className="container">
47        <br /> <br />
48        <div className="row">
49          <div className="card col-md-6 offset-md-3 offset-md-3">
50            {pageTitle()}
51            <div className="card-body">
52              <form>
```

Connect React App to Get Department REST API

Development Steps

1. In DepartmentService, write a code to call Get Department REST API using axios
2. Use **useEffect** hook to populate the department data in the form for update

src > services > DepartmentService.js > getDepartmentById

```
10  export const getDepartmentById = (departmentId) =>
11    axios.get(DEPARTMENT_REST_API_BASE_URL + "/" + departmentId);
```

src > components > DepartmentComponent.jsx > ...

```
8   const DepartmentComponent = () => {
34    useEffect(() => {
35      if (id) {
36        getDepartmentById(id)
37          .then((response) => {
38            setDepartmentName(response.data.departmentName);
39            setDepartmentDescription(response.data.departmentDescription);
40          })
41          .catch((error) => console.log(error));
42      }
43    }, [id]);
44
45    return (···
89    );
90  };
```

Connect React App to Update Department REST API

Development Steps

1. In DepartmentService, write a code to call Update Department REST API using axios.
2. Change **DepartmentComponent.saveDepartment()** method to perform both add and update department operations

src > services > DepartmentService.js > updateDepartment

```
13  export const updateDepartment = (departmentId, department) =>
14    axios.put(DEPARTMENT_REST_API_BASE_URL + "/" + departmentId, department);
```

src > components > DepartmentComponent.jsx > DepartmentComponent

```
9   const DepartmentComponent = () => {
34    function saveOrUpdateDepartment(e) {
35      e.preventDefault();
36
37      const department = { departmentName, departmentDescription };
38
39      if (id) {
40        updateDepartment(id, department)
41          .then((response) => {
42            console.log(response.data);
43            navigator("/departments");
44          })
45          .catch((error) => console.log(error));
46      } else {
47        createDepartment(department)
48          .then((response) => {
49            console.log(response.data);
50            navigator("/departments");
51          })
52          .catch((error) => console.log(error));
53      }
54    }
55
56    return (
57      <div className="container">
58        <br /> <br />
59        <div className="row">
60          <div className="card col-md-6 offset-md-3 offset-md-3">
61            {pageTitle()}
62            <div className="card-body">
63              <form>
64 >            <div className="form-group mb-2">…
74              </div>
75 >            <div className="form-group mb-2">…
85              </div>
86
87              <button
88                className="btn btn-success mb-2"
89                onClick={(e) => saveOrUpdateDepartment(e)}
90              >
91                Submit
92              </button>
93            </form>
```

Implement Delete Department Feature

Development Steps

1. In DepartmentService, write a code to call Delete Department REST API using axios.
2. Add **Delete** button to list departments table
3. Create JavaScript function to handle button event

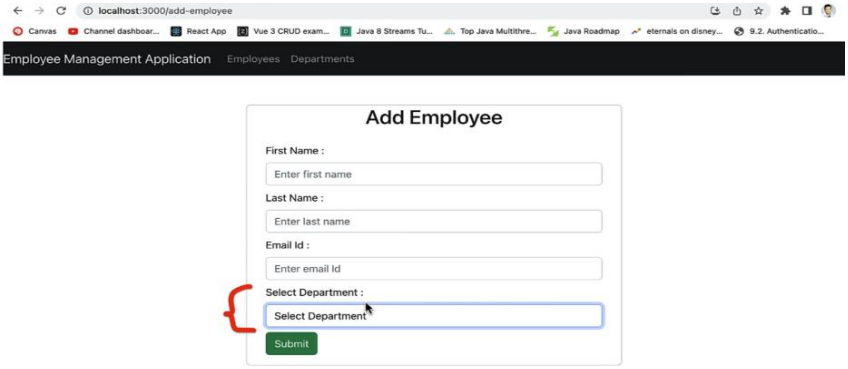
src > services > DepartmentService.js > ...

```
export const deleteDepartment = (departmentId) =>
  axios.delete(DEPARTMENT_REST_API_BASE_URL + "/" + departmentId, departmentId);
```

src > components > ListDepartmentComponent.jsx > ListDepartmentComponent

```
const ListDepartmentComponent = () => {
  useEffect(() => {
    listOfDepartments();
  }, []);

  function listOfDepartments() {
    getAllDepartments()
      .then((response) => {
        setDepartments(response.data);
      })
      .catch((error) => console.log(error));
  }

  function removeDepartment(id) {
    deleteDepartment(id)
      .then((response) => {
        console.log(response.data);
        listOfDepartments();
      })
      .catch((error) => console.log(error));
  }

  return (
    <div className="container">
      <h2 className="text-center">List of Departments</h2>
      <Link to="/add-department" className="btn btn-primary mb-2">
        Add Department
      </Link>
      <table className="table table-striped table-bordered">
        <thead> ···
        </thead>
        <tbody>
          {departments.map((department) => (
            <tr key={department.id}>
              <td>{department.id}</td>
              <td>{department.departmentName}</td>
              <td>{department.departmentDescription}</td>
              <td>
                <button ···
                </button>
                <button
                  onClick={() => removeDepartment(department.id)}
                  className="btn btn-danger"
                  style={{ marginLeft: "10px" }}
                >
                  Delete
                </button>
```

Change Add & Update Employee Feature to use Department



Development Steps

1. In EmployeeComponent, define state variables - departmentId and departments
2. Get All Departments to populate as select box options
3. Add select box to add employee form
4. Pass departmentId in Add Employee REST API Request

src > components > EmployeeComponent.jsx > EmployeeComponent

```
10  const EmployeeComponent = () => {
16    const [departmentId, setDepartmentId] = useState("");
17    const [departments, setDepartments] = useState([]);
18
19    useEffect(() => {
20      getAllDepartments()
21        .then((response) => {
22          setDepartments(response.data);
23        })
24        .catch((error) => {
25          console.log(error);
26        });
27    }, []);
28
29    const [errors, setErrors] = useState({
30      firstName: "",
31      lastName: "",
32      email: "",
33      department: "",
34    });
35
36    function validateForm() {
37      let valid = true;
38      const errorsCopy = { ...errors };
39
40      if (firstName.trim()) {···
42      } else {···
45      }
46
47      if (lastName.trim()) {···
49      } else {
50        errorsCopy.lastName = "Last name is required";
51        valid = false;
52      }
53
54      if (email.trim()) {···
56      } else {···
59      }
60
61      if (departmentId) {
62        errorsCopy.department = "";
63      } else {
64        errorsCopy.department = "Select Department";
65        valid = false;
66      }
67
68      setErrors(errorsCopy);
69      return valid;
70    }
```

```
10  const EmployeeComponent = () => {
80    useEffect(() => {
81      if (id) {
82        getEmployee(id)
83          .then((response) => {
84            setFirstName(response.data.firstName);
85            setLastName(response.data.lastName);
86            setEmail(response.data.email);
87            setDepartmentId(response.data.departmentId);
88          })
89          .catch((error) => {
90            console.log(error);
91          });
92      }
93    }, [id]);
94
95    function saveOrUpdateEmployee(e) {
96      e.preventDefault();
97
98      if (validateForm()) {
99        const employee = { firstName, lastName, email, departmentId };
100
101       if (id) {
102         updateEmployee(id, employee)
103           .then((response) => {
104             console.log(response);
105             navigator("/employees");
106           })
107           .catch((error) => console.log(error));
108       } else {
109         createEmployee(employee)
110           .then((response) => {
111             console.log(response);
112             navigator("/employees");
113           })
114           .catch((error) => console.log(error));
115       }
116     }
117   }
```

```
119   return (
120     <div className="container">
121       <br /> <br />
122       <div className="row">
123         <div className="card col-md-6 offset-md-3 offset-md-3">
124           {pageTitle()}
125           <div className="card-body">
126             <form>
127 >             <div className="form-group mb-2">···
143               </div>
144 >             <div className="form-group mb-2">···
159               </div>
160 >             <div className="form-group mb-2">···
173               </div>
174
175               <div className="form-group mb-2">
176                 <label className="form-label">Select Department:</label>
177                 <select
178                   value={departmentId}
179                   className={`form-control ${
180                     errors.department ? "is-invalid" : ""
181                   }`}
182                   onChange={(e) => setDepartmentId(e.target.value)}
183                 >
184                   <option value="Select Department">Select Department</option>
185                   {departments.map((department) => (
186                     <option key={department.id} value={department.id}>
187                       {department.departmentName}
188                     </option>
189                   ))}
190                 </select>
191                 {errors.department && (
192                   <div className="invalid-feedback">{errors.department}</div>
193                 )}
194               </div>
```