UE19CS204 – Web Technologies Mini Project

Project Title – EMPLOYEE

Section – B	Group-13
Team Members	**SRN**
Divyansh Kumar Singh	PES2UG19CS118
Govardhan B S	PES2UG19CS131
B Sai Sujay Reddy	PES2UG19CS080

GITHUB Link -https://github.com/govardhanbs2002/Crud-Operations

Abstract

- CREATE - A new employee is added by entering the details in the form
- READ - The Home button redirects us to the list page of employees
- UPDATE - Employee details can be modified (using Edit option) and is updated when the Submit button is clicked
- DELETE – Clicking on the Delete button deletes the respective employee's details from the database.
- Since we have used MongoDB Compass for the backend database management, any change made in the database is reflected in the browser as well as MongoDB.
-

Description

The website contains the following features-

- ➢Our Website provides basic CRUD operations
- ➢User can input first name ,last name, email id , and phone no inside the text box(uses html forms).
- ➢Submit button is provided to make its entry in the Data-Base
- ➢The navigation bar provides us with the options to move to home page and add new employee page.
- ➢Our website displays all existing employee details on the home screen as well

Technologies Used

- This project makes use of four main JavaScript Technologies.
- MongoDB - No SQL Database documents which acts as a back-end for the web app
- ExpressJS - Back-end web application framework for creating APIs
- ReactJS - JavaScript based library to build a powerful and attractive user interface
- NodeJS - JavaScript runtime environment that enables us to create a JavaScript back-end environment and network applications

Technologies Used - dependencies

- Front-end Dependencies

1. react-router-dom (To work with routing)
2. react-bootstrap (To use bootstrap components in our application)
3. axios (to call the REST API from services)

- Back-end Dependencies

1. mongoose (MongoDB object model for NodeJS)
2. express (runs as a module for NodeJS and useful in routing)
3. body-parser (parse the response data coming from the API and to validate it before using it in our application)
4. cors (to enable cors middleware using NodeJS with various options)

BACKEND

- We have used Express , Body-parse, Mongoose, cors module to build our backend. THESE ARE NPM LIBRARIES.
- Express scaffolding allows us to create a skeleton for the web application . Express is used in building routes pathway and defining functionality of each path .
- Body parse sends middleware requests through express. It helps in accessing frontend elements.
- CORS is used in Cross-Origin Resource Sharing(helps in transferring data from one web page to another like between add employee and home pages).
- Mongoose helps in connecting MongoDB database to our local server. Mongoose also helps in handling queries and CRUD operation.

- Then we come to **Model folder** in there we have **Employee.js file**. We use mongoose to build our schema element. Here we create a mongoose object to access mongodb database. From the object we are creating 4 fields. Schema is the structure in which our elements travelling through our routes .then we create a employee collection in mongodb compass .
- Inside the **Routes folder**, we have **Employee.route.js** . Here we are sending requests to various sections of our site to get the information to mongodb database. Routes were created with express scaffolding. Express module has defined all the possible routes to be taken by our schema element and we have filled the details of each route like add , edit , delete and list employee. The routes would be taken between website and mongodb database.
- First, inside the **Server folder**, we build **server.js file**. This file connects the entire MERN stack. We first import the modules- express , cors , bodyparser, path in server.js file. Then we connect our app with MONGODB database. And then we define the port on which we will listen to other requests.

Frontend Description :

- Add employee in home screen take us to the add employee page using 'axios' here we are able to add all the details and submit after submitting also we can add employees and export the Add employee
- Edit employee also enables to edit or update the details and export it.
- Services file delete the employee and export
- List employee will import delete employee and list the all employees, and create a delete button with proper links and export it.
- In main file it imports list, add ,edit employee files and create a route and exact path and export it.
- In header file we have the navbars to home and to add employees
- In app.js it imports header and main files and create a proper web app

Member Contributions

- Backend- Divyansh Kumar Singh
- Frontend:-

  -Govardhan B S
[AddEmployee.js ,EditEmployee.js, ListEmployee
.js, Services.js(delete an employee)].

  -B Sai Sujay Reddy [Header.js,main.js and app.js].

Thank You