Backend/controller/employeeHandlers.js

const Employees = require("../models/employee")

const handlers = {

getEmployees: async (req, res, next) => {

let data = [];

try{

data = await Employees.find();

res.status(200).json(data);

} catch(err) {

res.status(500).json({

error: err

});

}

},

getEmployee: async (req, res, next) => {

try{

await Employees.findOne({ssn: req.params.ssn})

.then((employee) => {

if(employee !== null)

res.json(employee);

else

res.json({

"message": "No employee found with SSN: " + req.params.ssn

});

});

} catch (err) {

res.status(500).json({

error: err

});

}

},

createEmployee: async (req, res, next) => {

await Employees.create(req.body)

.then((data) => {

res.json(data);

})

.catch((err) => {

if(err.errors)

res.json({

error: ""

});

else

res.json({

error: "SSN must be unique"

});

})

},

deleteEmployee: async (req, res, next) => {

try{

await Employees.findOne({ssn: req.params.ssn})

.then((employee) => {

if(employee !== null)

Employees.deleteOne({ssn: req.params.ssn})

.then(async (data) => {

let emps = [];

emps = await Employees.find();

res.json(emps);

});

else

res.json({

"message": "No employee found with SSN: " + req.params.ssn

});

});

} catch(err) {

res.status(500).json({

error: err

});

}

},

updateEmployee: async (req, res, next) => {

try{

await Employees.findOne({ssn: req.body.ssn})

.then((employee) => {

if(employee !== null)

Employees.updateOne({ssn: req.body.ssn}, req.body)

.then((data) => {

res.json({

message: "Edited the employee with SSN: " + req.body.ssn + " successfully"

});

});

else

res.json({

"message": "No employee found with SSN: " + req.params.ssn

});

});

} catch(err) {

res.status(500).json({

error: err

});

}

}

}

module.exports = handlers;

**models/employee.js**

const { Schema, default: mongoose } = require("mongoose");

const Employee = new Schema({

ssn: {type: String, required: true, unique: true},

firstname: {type: String, required: true},

lastname: {type: String, required: true},

contact: {type: Number, required: true}

});

module.exports = mongoose.model("employee", Employee);

**backend/index.js**

const express = require("express");

const mongoose = require("mongoose");

const cors = require("cors");

const bodyParser = require("body-parser");

const handlers = require("./controller/employeeHandlers");

const server = express();

server.use(cors());

server.use(bodyParser());

server.get("/", (req, res, next) => {

res.send("hello world");

})

server.get("/emps", handlers.getEmployees);

server.get("/emp/:ssn", handlers.getEmployee);

server.post("/emp", handlers.createEmployee);

server.put("/emp", handlers.updateEmployee);

server.delete("/emp/:ssn", handlers.deleteEmployee);

mongoose.connect("mongodb://127.0.0.1:27017/empDB")

.then((d) => {

server.listen(8080, () => {console.log("Server listening at port: 8080")});

})

.catch((err) => {

console.log("ERROR: " + err);

});

**Frontend/src/app.js**

import './App.css';

import { Button, createTheme, Dialog, DialogActions, DialogContent, DialogTitle, Grid, OutlinedInput, Table, TableBody, TableCell, TableHead, TableRow, ThemeProvider, Typography } from '@mui/material';

import { useEffect, useState } from 'react';

import axios from 'axios';

function App() {

const theme = createTheme({

palette: {

primary: {

main: "#148cbb"

},

secondary: {

main: "#ffffff"

},

info: {

main: "#444444"

}

}

});

const [emps, setEmps] = useState([]);

const [edit, setEdit] = useState(false);

const [create, setCreate] = useState(false);

const [emp, setEmp] = useState(null);

//for create

const [ssn, setSSN] = useState("");

const [firstName, setFirstName] = useState("");

const [lastName, setLastName] = useState("");

const [contact, setContact] = useState(0);

useEffect(() => {

axios.get("http://localhost:8080/emps")

.then((res) => {

setEmps(res.data);

})

.catch((err) => {

console.log(err);

})

}, []);

const handleCreate = () => {

setEmp(null);

setCreate(true);

}

const createCloseHandler = () => {

setCreate(false);

}

const handleEdit = (e) => {

setEmp(e);

setEdit(true);

}

const handleDelete = (ssn) => {

axios.delete("http://localhost:8080/emp/"+ssn)

.then((res) => {

setEmps(res.data);

})

.catch((err) => {

console.log(err);

})

}

const editCloseHandler = () => {

setEdit(false);

}

const handleEditSubmit = () => {

axios.put("http://localhost:8080/emp", emp)

.then((res) => {

axios.get("http://localhost:8080/emps")

.then((res) => {

setEmps(res.data);

editCloseHandler();

})

.catch((err) => {

console.log(err);

})

})

.catch((err) => {

console.log(err);

})

}

const handleCreateSubmit = () => {

console.log(emp);

axios.post("http://localhost:8080/emp", {

ssn: ssn,

firstname: firstName,

lastname: lastName,

contact: contact

})

.then((res) => {

axios.get("http://localhost:8080/emps")

.then((res) => {

setEmps(res.data);

createCloseHandler();

})

.catch((err) => {

console.log(err);

})

})

.catch((err) => {

console.log(err);

})

}

return (

<div className="App">

<ThemeProvider theme={theme}>

<Grid container>

<Grid item alignItems="center" className='width'>

<Typography variant="h4" style={{margin: "1em", fontWeight: "bold"}} color="primary">EMPLOYEE MANAGEMENT SYSTEM</Typography>

</Grid>

<Grid item className='width' alignItems="center">

<Button variant='outlined' color='primary' style={{backgroundColor: "white"}} onClick={handleCreate}>CREATE</Button>

</Grid>

<Grid item className='width' style={{padding: "3em"}} alignItems="center">

<Table style={{backgroundColor: "white", borderRadius: "8 px"}}>

<TableHead>

<TableRow>

<TableCell style={{fontWeight: "bold"}}>SSN</TableCell>

<TableCell style={{fontWeight: "bold"}}>First Name</TableCell>

<TableCell style={{fontWeight: "bold"}}>Last Name</TableCell>

<TableCell style={{fontWeight: "bold"}}>Contact</TableCell>

<TableCell style={{fontWeight: "bold"}}>Edit</TableCell>

<TableCell style={{fontWeight: "bold"}}>Delete</TableCell>

</TableRow>

</TableHead>

<TableBody>

{

emps.map((e) => (

<TableRow>

<TableCell>{e.ssn}</TableCell>

<TableCell>{e.firstname}</TableCell>

<TableCell>{e.lastname}</TableCell>

<TableCell>{e.contact}</TableCell>

<TableCell>

<Button

variant="contained"

color="primary"

onClick={() => handleEdit(e)}

>

EDIT

</Button>

</TableCell>

<TableCell>

<Button variant="contained" color="error" onClick={() => handleDelete(e.ssn)}>

DELETE

</Button>

</TableCell>

</TableRow>

))

}

</TableBody>

</Table>

</Grid>

</Grid>

<Dialog

open={edit}

onClose={editCloseHandler}

>

<DialogTitle>Edit Employee</DialogTitle>

<DialogContent style={{width: "300px"}}>

{

(emp !== null)?

<Grid container spacing="10px">

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="SSN" value={emp.ssn} onChange={(e) => {

setEmp({

ssn: e.target.value,

firstname: emp.firstname,

lastname: emp.lastname,

contact: emp.contact

});

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="First Name" value={emp.firstname} onChange={(e) => {

setEmp({

ssn: emp.ssn,

firstname: e.target.value,

lastname: emp.lastname,

contact: emp.contact

});

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="Last Name" value={emp.lastname} onChange={(e) => {

setEmp({

ssn: emp.ssn,

firstname: emp.firstname,

lastname: e.target.value,

contact: emp.contact

});

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="number" placeholder="contact" value={emp.contact} onChange={(e) => {

setEmp({

ssn: emp.ssn,

firstname: emp.firstname,

lastname: emp.lastname,

contact: e.target.value

});

}}></OutlinedInput>

</Grid>

</Grid>

:

<></>

}

</DialogContent>

<DialogActions style={{paddingRight: "1.6em", marginBottom: "1em"}}>

<Button variant='contained' color='primary' onClick={handleEditSubmit}>EDIT</Button>

<Button variant='outlined' color='info' onClick={editCloseHandler}>CANCEL</Button>

</DialogActions>

</Dialog>

<Dialog

open={create}

onClose={createCloseHandler}

>

<DialogTitle>Add Employee</DialogTitle>

<DialogContent style={{width: "300px"}}>

<Grid container spacing="10px">

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="SSN" onChange={(e) => {

setSSN(e.target.value);

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="First Name" onChange={(e) => {

setFirstName(e.target.value);

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="Last Name" onChange={(e) => {

setLastName(e.target.value);

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="number" placeholder="contact" onChange={(e) => {

setContact(e.target.value);

}}></OutlinedInput>

</Grid>

</Grid>

</DialogContent>

<DialogActions style={{paddingRight: "1.6em", marginBottom: "1em"}}>

<Button variant='contained' color='primary' onClick={handleCreateSubmit}>ADD</Button>

<Button variant='outlined' color='info' onClick={createCloseHandler}>CANCEL</Button>

</DialogActions>

</Dialog>

</ThemeProvider>

</div>

);

}

export default App;

**app.css**

.App {

text-align: center;

}

.App-logo {

height: 40vmin;

pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

.App-logo {

animation: App-logo-spin infinite 20s linear;

}

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

.App-link {

color: #61dafb;

}

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}