**Simple Employees API using Nodejs**

**🚀 Exciting News: Just Launched My Employee API Using Node.js and Express! 🌐**

* **Required Packages –**
  + body-parser – (To access data coming in)
  + express – (nodejs web application framework.)
  + helmet – (To protect our data or API)(required to protect our data)
  + dotenv (To access environment variable saved in ".env” file)(optional)
  + morgan
* **Operations can be performed –** 
  + **CREATE**
  + **READ**
  + **DELETE**
  + **PUT (All change making)**
  + **PATCH (some changes making)**
* **.env file content –**

PORT = 3000

* **Code –**

import bodyParser from "body-parser";

import express from "express";

import helmet from "helmet";

import morgan from "morgan";

import dotenv from "dotenv";

dotenv.config();

*const* employees= [

    {

      "id": 1,

      "name": "John Doe",

      "position": "Software Engineer",

      "department": "Engineering",

      "salary": 80000

    },

    {

      "id": 2,

      "name": "Jane Smith",

      "position": "Marketing Specialist",

      "department": "Marketing",

      "salary": 60000

    },

    {

      "id": 3,

      "name": "Mike Johnson",

      "position": "Financial Analyst",

      "department": "Finance",

      "salary": 70000

    },

    {

      "id": 4,

      "name": "Emily Davis",

      "position": "Human Resources Manager",

      "department": "Human Resources",

      "salary": 75000

    },

    {

      "id": 5,

      "name": "Alex Rodriguez",

      "position": "Sales Representative",

      "department": "Sales",

      "salary": 65000

    },

    {

      "id": 6,

      "name": "Sara Brown",

      "position": "Customer Support Specialist",

      "department": "Customer Support",

      "salary": 55000

    },

    {

      "id": 7,

      "name": "Chris Taylor",

      "position": "Product Manager",

      "department": "Product Management",

      "salary": 90000

    },

    {

      "id": 8,

      "name": "Amanda White",

      "position": "Quality Assurance Engineer",

      "department": "Engineering",

      "salary": 82000

    },

    {

      "id": 9,

      "name": "Mark Johnson",

      "position": "Sales Manager",

      "department": "Sales",

      "salary": 75000

    },

    {

      "id": 10,

      "name": "Olivia Davis",

      "position": "Customer Relations Specialist",

      "department": "Customer Support",

      "salary": 55000

    },

    {

      "id": 11,

      "name": "Brian Miller",

      "position": "Operations Analyst",

      "department": "Operations",

      "salary": 72000

    },

    {

      "id": 12,

      "name": "Jessica Garcia",

      "position": "Marketing Manager",

      "department": "Marketing",

      "salary": 90000

    },

    {

      "id": 13,

      "name": "Daniel Martinez",

      "position": "IT Support Specialist",

      "department": "IT",

      "salary": 68000

    },

    {

      "id": 14,

      "name": "Sophia Johnson",

      "position": "Product Designer",

      "department": "Product Management",

      "salary": 78000

    },

    {

      "id": 15,

      "name": "William Brown",

      "position": "Financial Planner",

      "department": "Finance",

      "salary": 70000

    },

    {

      "id": 16,

      "name": "Ella Taylor",

      "position": "Human Resources Specialist",

      "department": "Human Resources",

      "salary": 75000

    },

    {

      "id": 17,

      "name": "James Anderson",

      "position": "Customer Success Manager",

      "department": "Customer Support",

      "salary": 82000

    },

    {

      "id": 18,

      "name": "Grace Smith",

      "position": "Sales Associate",

      "department": "Sales",

      "salary": 65000

    },

    {

      "id": 19,

      "name": "Benjamin Davis",

      "position": "Software Developer",

      "department": "Engineering",

      "salary": 80000

    },

    {

      "id": 20,

      "name": "Ava Rodriguez",

      "position": "Product Owner",

      "department": "Product Management",

      "salary": 95000

    }

  ];

*let* response; // To send response

*const* app = express();

*const* PORT = process.env.PORT;

app.use(helmet()); // To protecting Data

app.use(morgan("combined")); // For connecting API's public

app.use(bodyParser.urlencoded({extended:true})) // To get data coming in

*let* filteredEmployees;

*let* employeeId;

*let* employeeData;

*let* employeeIndex;

*const* filterMiddleware = (*req*,*res*,*next*) *=>* {

    try{

        if(*req*.query.department && *req*.query.position && *req*.query.salary)

        filteredEmployees = employees.filter((*employee*) *=>* (*employee*.department==*req*.query.department && *employee*.position==*req*.query.position && *employee*.salary==*req*.query.salary))

        else if(*req*.query.department && *req*.query.position)

        filteredEmployees = employees.filter((*employee*) *=>* (*employee*.department==*req*.query.department && *employee*.position==*req*.query.position))

        else if(*req*.query.position && *req*.query.salary)

        filteredEmployees = employees.filter((*employee*) *=>* (*employee*.position==*req*.query.position && *employee*.salary==*req*.query.salary))

        else if(*req*.query.department && *req*.query.salary)

        filteredEmployees = employees.filter((*employee*) *=>* (*employee*.department==*req*.query.department && *employee*.salary==*req*.query.salary))

        else if(*req*.query.department)

        filteredEmployees = employees.filter((*employee*) *=>* (*employee*.department==*req*.query.department))

        else if(*req*.query.position)

        filteredEmployees = employees.filter((*employee*) *=>* (*employee*.position==*req*.query.position))

        else if(*req*.query.salary)

        filteredEmployees = employees.filter((*employee*) *=>* (*employee*.salary==*req*.query.salary))

        else throw {"error":"Specify filters like department,salary,etc."};

        if(filteredEmployees.length==0)

        throw {"error":"Don't have any data"};

        next();

    }catch(error){

*res*.send(error)

    }

}

*let* getEmployeeMiddleware = (*req*,*res*,*next*) *=>* {

    try{

        employeeId =*req*.params.id;

        if(!*Number*.isInteger(*Number*.parseInt(employeeId)))

        throw {"error":"Specify Id only"};

        employeeData = employees.find((*employee*)*=>* *employee*.id==employeeId)

        if(!employeeData)

        throw {"error":`Data not exists with given id ${employeeId}`};

        next();

    }catch(error){

*res*.send(error)

    }

}

*let* postEmployeeMiddleware = (*req*,*res*,*next*) *=>* {

   try{

    if(!(*req*.body.name && *req*.body.position && *req*.body.department && *req*.body.salary))

    throw {"error":"Incomplete details"};

    if(!*Number*.isInteger(*Number*.parseInt(*req*.body.salary)))

    throw {"error":"Salary cannot be in the form of string"};

    next();

   }catch(error){

*res*.send(error)

   }

}

*let* putEmployeeMiddleware = (*req*,*res*,*next*) *=>* {

    try{

        employeeId =*req*.params.id;

        if(!*Number*.isInteger(*Number*.parseInt(employeeId)))

        throw {"error":"Specify Id only"};

        employeeIndex = employees.findIndex((*employee*)*=>* *employee*.id==employeeId)

        if(employeeIndex==-1)

        throw {"error":"Data not found"};

        if(*req*.body.salary && !*Number*.isInteger(*Number*.parseInt(*req*.body.salary)))

        throw {"error":"Salary cannot be in the form of string"};

        employeeData = employees[employeeIndex];

        next()

    }catch(error){

*res*.send(error)

    }

}

*let* deleteEmployeeMiddleware = (*req*,*res*,*next*) *=>* {

    try{

        employeeId =*req*.params.id;

        if(!*Number*.isInteger(*Number*.parseInt(employeeId)))

        throw {"error":"Specify Id only"};

        employeeIndex = employees.findIndex((*employee*)*=>* employee.id==employeeId)

        if(employeeIndex==-1)

        throw {"error":`Data not exists with given id ${employeeId}. hence data cannot be deleted`};

        next();

    }catch(error){

*res*.send(error)

    }

}

app.get("/",(*req*,*res*)*=>*{

*res*.send(employees)

})

app.get("/random",(*req*,*res*)*=>*{

    employeeData = employees[Math.floor(Math.random()\*employees.length)];

*res*.send(employeeData)

})

app.get("/filter",filterMiddleware,(*req*,*res*)*=>*{

*res*.send(filteredEmployees)

})

app.get("/:id",getEmployeeMiddleware,(*req*,*res*)*=>*{

*res*.send(employeeData)

})

app.post("/employee",postEmployeeMiddleware,(*req*,*res*)*=>*{

    employeeData = {

        "id":employees.length + 1,

        "name":*req*.body.name,

        "position":*req*.body.position,

        "department":*req*.body.department,

        "salary":*req*.body.salary

    }

    employees.push(employeeData)

*res*.send(employees[employees.length-1])

})

app.put("/employee/:id",putEmployeeMiddleware,(*req*,*res*)*=>*{

    employeeData.name = *req*.body.name || employeeData.name;

    employeeData.position = *req*.body.position || employeeData.position;

    employeeData.department = *req*.body.department || employeeData.department;

    employeeData.salary = *Number*.parseInt(*req*.body.salary) || employeeData.salary;

    employees[employeeIndex] = employeeData;

*res*.send(employees[employeeIndex])

})

app.patch("/employee/:id",putEmployeeMiddleware,(*req*,*res*)*=>*{

    employeeData.name = req.body.name || employeeData.name;

    employeeData.position = req.body.position || employeeData.position;

    employeeData.department = req.body.department || employeeData.department;

    employeeData.salary = Number.parseInt(req.body.salary) || employeeData.salary;

    employees[employeeIndex] = employeeData;

    res.send(employees[employeeIndex])

})

app.delete("/employee/:id",deleteEmployeeMiddleware,(*req*,*res*)*=>*{

    employeeData = employees.splice(employeeIndex,1)

    res.send(employeeData)

})

app.listen(PORT,(*req*,*res*)*=>*{

    console.log(`Server is running on port ${PORT}`)

})