Human Resource Management System(HRMS) - Full Stack Development Assignment

Ch Tagore

backend/package.json

```
{
  "name": "hrms-backend",
  "version": "1.0.0",
  "main": "src/server.js",
  "type": "module",
  "scripts": {
    "dev": "nodemon src/server.js"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.0",
    "pg": "^8.10.0"
  }
}
```

db.js — PostgreSQL Pool

```
import pkg from "pg";
const { Pool } = pkg;

export const pool = new Pool({
  connectionString: process.env.DATABASE_URL
});
```

Middleware: auth.js

```
import jwt from "jsonwebtoken";

export function auth(req, res, next) {
  const token = req.headers.authorization?.split(" ")[1];
  if (!token) return res.status(401).json({ message: "No token" });

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch {
```

```
    res.status(401).json({ message: "Invalid token" });
  }
}
```

Middleware: logger.js

```
import { pool } from "../config/db.js";

export async function logAction(userId, action) {
  await pool.query(
    "INSERT INTO logs (user_id, action) VALUES ($1, $2)",
    [userId, action]
  );
}
```

Database Schema

Users

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  email TEXT UNIQUE NOT NULL,
  password TEXT NOT NULL
);
```

Employees

```
CREATE TABLE employees (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL,
  role TEXT,
  created_at TIMESTAMP DEFAULT NOW()
);
```

Teams

```
CREATE TABLE teams (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT NOW()
);
```

Team_assignments

```
CREATE TABLE team_assignments (
  id SERIAL PRIMARY KEY,
```

```
  employee_id INT REFERENCES employees(id) ON DELETE CASCADE,
  team_id INT REFERENCES teams(id) ON DELETE CASCADE
);
```

Logs

```
CREATE TABLE logs (
  id SERIAL PRIMARY KEY,
  user_id INT REFERENCES users(id),
  action TEXT,
  created_at TIMESTAMP DEFAULT NOW()
);
```

Auth Route — register/login (auth.js)

```
import express from "express";
import bcrypt from "bcryptjs";
import jwt from "jsonwebtoken";
import { pool } from "../config/db.js";
import { logAction } from "../middleware/logger.js";

const router = express.Router();

router.post("/register", async (req, res) => {
  const { email, password } = req.body;
  const hashed = await bcrypt.hash(password, 10);

  await pool.query(
    "INSERT INTO users (email, password) VALUES ($1, $2)",
    [email, hashed]
  );

  res.json({ message: "User registered" });
});

router.post("/login", async (req, res) => {
  const { email, password } = req.body;

  const result = await pool.query(
    "SELECT * FROM users WHERE email = $1",
    [email]
  );

  const user = result.rows[0];
  if (!user) return res.status(400).json({ message: "Invalid email" });

  const match = await bcrypt.compare(password, user.password);
  if (!match) return res.status(400).json({ message: "Wrong password" });
```

```
  const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET);

  await logAction(user.id, "User logged in");
  res.json({ token });
});

export default router;
```

Employee CRUD
/routes/employees.js

```
import express from "express";
import { pool } from "../config/db.js";
import { auth } from "../middleware/auth.js";
import { logAction } from "../middleware/logger.js";

const router = express.Router();

router.get("/", auth, async (req, res) => {
  const result = await pool.query("SELECT * FROM employees ORDER BY id");
  res.json(result.rows);
});

router.post("/", auth, async (req, res) => {
  const { name, role } = req.body;

  const result = await pool.query(
    "INSERT INTO employees (name, role) VALUES ($1, $2) RETURNING *",
    [name, role]
  );

  await logAction(req.user.id, `Created employee ${result.rows[0].id}`);
  res.json(result.rows[0]);
});

router.put("/:id", auth, async (req, res) => {
  const { name, role } = req.body;
  const { id } = req.params;

  await pool.query(
    "UPDATE employees SET name=$1, role=$2 WHERE id=$3",
    [name, role, id]
  );

  await logAction(req.user.id, `Updated employee ${id}`);
  res.json({ message: "Updated" });
```

```
});

router.delete("/:id", auth, async (req, res) => {
  const { id } = req.params;

  await pool.query("DELETE FROM employees WHERE id=$1", [id]);
  await logAction(req.user.id, `Deleted employee ${id}`);

  res.json({ message: "Deleted" });
});

export default router;
```

app.js

```
import express from "express";
import cors from "cors";

import authRoutes from "./routes/auth.js";
import employeeRoutes from "./routes/employees.js";
import teamRoutes from "./routes/teams.js";
import assignmentRoutes from "./routes/assignments.js";

const app = express();
app.use(cors());
app.use(express.json());

app.use("/auth", authRoutes);
app.use("/employees", employeeRoutes);
app.use("/teams", teamRoutes);
app.use("/assignments", assignmentRoutes);

export default app;
```

server.js

```
import app from "./app.js";
import "dotenv/config";

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log("Server running on " + PORT));
```

FRONTEND (React + Vite)

src/api.js

```
import axios from "axios";

export const api = axios.create({
  baseURL: "http://localhost:5000"
});

export function setToken(token) {
  api.defaults.headers.common["Authorization"] = "Bearer " + token;
}
```

src/pages/Login.jsx

```
import { useState } from "react";
import { api, setToken } from "../api";

export default function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  async function submit(e) {
    e.preventDefault();
    const res = await api.post("/auth/login", { email, password });
    setToken(res.data.token);
    window.location.href = "/employees";
  }

  return (
    <form onSubmit={submit}>
      <h2>Login</h2>
      <input placeholder="email" onChange={e => setEmail(e.target.value)} />
      <input placeholder="password" type="password" onChange={e =>
setPassword(e.target.value)} />
      <button>Login</button>
    </form>
  );
}
```

src/pages/Employees.jsx

```
import { useEffect, useState } from "react";
import { api } from "../api";

export default function Employees() {
  const [employees, setEmployees] = useState([]);
  const [form, setForm] = useState({ name: "", role: "" });
```

```
  async function load() {
    const res = await api.get("/employees");
    setEmployees(res.data);
  }

  async function submit(e) {
    e.preventDefault();
    await api.post("/employees", form);
    load();
  }

  useEffect(() => { load(); }, []);

  return (
    <div>
      <h1>Employees</h1>

      <form onSubmit={submit}>
        <input placeholder="Name" onChange={e => setForm({ ...form, name: e.target.value })}
/>
        <input placeholder="Role" onChange={e => setForm({ ...form, role: e.target.value })} />
        <button>Add</button>
      </form>

      <ul>
        {employees.map(e => (
          <li key={e.id}>{e.name} — {e.role}</li>
        ))}
      </ul>
    </div>
  );
}
```