📄 IMPLEMENTATION FILES – EMPLOYEE DIRECTORY SYSTEM

🔷 Overview:

This section documents the files that contain the core logic and functional implementation of the RESTful API endpoints of the Employee Directory System. The files are part of a NodeJS-based backend architecture and are organized modularly by responsibility.

🔷 Directory Structure:

```
backend/
├── app.js
├── config/
│   └── db.js
├── controllers/
│   └── employeeController.js
├── routes/
│   └── employeeRoutes.js
└── package.json
```

🔷 File Descriptions:

📄 app.js

- Entry point for the NodeJS backend server
- Sets up Express, middleware, and routes
- Registers `/api/employees` endpoint and binds it to route handlers
- Listens on defined port and confirms connection to PostgreSQL

📄 config/db.js

- Defines and exports the PostgreSQL connection pool using the `pg` module
- Loads credentials and parameters for DB connection
- Used by all controller modules to perform queries on the database

📄 routes/employeeRoutes.js

- Defines the API route paths for employee operations
- Routes included:
  - `GET /api/employees`
  - `GET /api/employees/:id`
- Imports route handler functions from `employeeController.js`
- Uses Express Router to modularize path registration

📄 controllers/employeeController.js

- Contains business logic for handling employee-related data retrieval
- Function: `getAllEmployees(req, res)` – fetches a summary list of employees from the database
- Function: `getEmployeeById(req, res)` – retrieves a complete profile including:

    - Basic information (name, contact, gender, birthday)
    - HR-assigned fields (designation, role, department, joining/leaving)
    - Skills, certifications, and project list
    - Task statistics and performance rating
    - Auto-calculated fields like `years_in_current_company`
- Executes SQL queries using the DB pool configured in `db.js`

🔷 **Modular Implementation Style:**

- Routing and logic are separated for maintainability
- Each controller can scale independently in future phases
- Database interaction is handled asynchronously with proper error capture
- Follows RESTful principles in structure and response formats

🔷 **Compliance:**

- Follows standard Express-based NodeJS architecture
- Compliant with backend API design practices for modular apps
- Database access uses parameterized queries to avoid SQL injection risks