📄 API DESIGN DOCUMENT – PHASE 1: EMPLOYEE DIRECTORY SYSTEM

🔷 Overview:

This document outlines the API structure and logic implemented of the Employee Directory System backend. It covers the RESTful endpoints created to support employee data retrieval, filtered data for frontend consumption, and the associated logic layers used in NodeJS with PostgreSQL.

🔷 Module Name:

Employee API – Data Retrieval and Profile View

🔷 Technology Stack:

- Backend Framework: NodeJS with Express.js
- Database: PostgreSQL
- DB Interface: `pg` Node package (PostgreSQL client for NodeJS)
- API Style: RESTful
- Response Format: JSON

🔷 Objective:

To build RESTful APIs to manage employee data and provide filtered data for frontend display.

🔷 Implemented Endpoints:

Endpoint	Method	Description
`/api/employees`	GET	Returns a list of all employees with basic profile data
`/api/employees/:id`	GET	Returns full detailed profile of a specific employee
(Planned) `/api/employees?name=John&department=HR`	GET	Fetch employees filtered by name and department
(Planned) `/api/departments`	GET	Returns the list of available departments for filtering

🔷 Key API Functionalities:

🔹 GET /api/employees

- Fetches a simplified list of employees
- Fields returned: employee_id, first_name, last_name, profile_picture, role_id, designation_id, availability_status

🔹 GET /api/employees/:id

- Fetches full profile of a specific employee
- Includes: personal info, HR-assigned fields, previous experience, project counts, ratings, task summary, skills, certifications, and projects
- Also calculates: years of experience in the current company (based on joining_date)

🔷 Core Tables Referenced:

- `employees`
- `skills`
- `certifications`
- `projects`
- `roles`
- `designations`
- `main_departments`
- `sub_departments`

🔷 Modular Layers Involved:

🔁 Routing Layer

- File: `routes/employeeRoutes.js`
- Purpose: Defines route paths for `/employees` and `/employees/:id`
- Imported Functions: `getAllEmployees`, `getEmployeeById`

🧠 Controller Layer

- File: `controllers/employeeController.js`
- Function: `getAllEmployees()` – fetches all employees
- Function: `getEmployeeById(emp_id)` – fetches full profile using joins and subqueries
- Performs nested queries to attach:
  - `skills`
  - `certifications`
  - `projects`
- Performs calculation logic for:
  - `years_in_current_company` using `joining_date`

🔷 Design Highlights:

- Queries use parameterized SQL to prevent injection
- Data is structured for frontend rendering
- Normalized foreign keys are used for role, department, designation
- Hardcoded demo values like `is_online = true` and `'Available'` status used temporarily for simulation
- Modular design for easy extension in Phase 2 (e.g., filtering, role-based visibility)

🔷 Limitations (Handled in Future Phases):

- Filtering logic via query params not implemented in this phase
- Department listing (`/departments`) not included yet

- No authentication or access control is enforced in this API phase
- Error handling is basic (200, 404, 500 only)

🔷 **Normalization Compliance:**

- API logic is aligned with the 3rd Normal Form database structure
- All foreign-key based lookups (skills, roles, projects, etc.) are resolved at controller level
- Joins are handled using primary-foreign relationships as per schema design document

🔷 **Usage Purpose:**

- This API module supports profile listing, viewing, and frontend population
- Forms the base foundation of the Employee Directory System backend