Objective:

Build a basic REST API to manage employees in a company, focusing on CRUD operations, RESTful principles, and authentication.

Requirements

○ **Create an Employee**: POST /api/employees/

○ **List all Employees:** GET /api/employees/

○ **Retrieve a Single Employee:** GET /api/employees/{id}/

○ **Update an Employee:** PUT /api/employees/{id}/

○ **Delete an Employee**: DELETE /api/employees/{id}/

Employee Model:

○ **id:**Unique identifier (auto-generated)

○ **name:** String, required

○ **email:** Email field, required and unique

○ **department: S**tring, optional (e.g., "HR", "Engineering", "Sales"**)**

○ **role:** String, optional (e.g., "Manager", "Developer", "Analyst")

○ **date_joined:** Date, auto-generated on creation

Additional Requirements:

○ Validation: Ensure email is unique and valid. name should not be empty.

○ Error Handling: Return appropriate HTTP status codes for different responses:

- ■ 201 Created for successful creation.
- ■ 404 Not Found for invalid employee IDs.
- ■ 400 Bad Request for validation errors.
- ■ 204 No Content for successful deletion.

○ **Filtering:** Allow filtering of employees by department and role (e.g., GET /api/employees/?department=HR).

○ **Pagination:** Limit results per page to 10 employees with pagination support (e.g., GET /api/employees/?page=2).

○ **Authentication**: Use token-based authentication (JWT or simple token) to secure the endpoints. Only authenticated users should access these endpoints.

Summary:

● Brief recap of the CRUD operations.

● Highlight RESTful practices, error handling, and Postman's ease for testing.

Created by ~

ANKIT KUMAR