**Employee Management System**

**Overview**

This is a Spring Boot application designed to manage employees and their departments. It provides CRUD (Create, Read, Update, Delete) operations for both entities and demonstrates the relationship between employees and their respective departments. Data is stored in a MySQL database, and APIs accept JSON input/output.

**Entities**

* **Employee**: Represents an employee with attributes such as ID, name, job title, salary, and associated department.
* **Department**: Represents a department with attributes such as ID and name.

**Database Relationship**

An **Employee** belongs to a **department** using a @ManyToOne relationship, enabling joins between the two tables.

**Employee Table Structure:**

Long id // EmployeeID, Primary Key

name varchar // EmployeeName

jobTitle varchar // EmployeeName

salary double // EmployeeName

Long department\_id // Foreign Key Dapartment

**Department Table Structure:**

Long id // Department ID, Primary Key

name varchar // Department Name

**APIs (Postman Collection saved in Postman folder of project)**

The following RESTful endpoints are available for managing employees.

**Get All Employees**

URL: http://localhost:8080/api/employees

Method: GET

Description: Retrieves a list of all employees with their department details.

**Get Employee by ID**

URL: http://localhost:8080/api/employees/{id}

Method: GET

Description: Fetches details of a specific employee by ID.

**Create Employee**

URL: http://localhost:8080/api/employees

Method: POST

Request Body (JSON): {

    "name": "Vini",

    "jobTitle": "Senior Talent Acq",

    "salary": 50000.0,

    "department": {

        "name": "HR"

    }

}

Description: Creates a new employee. Department is specified by its name.

**Update Employee**

URL: http://localhost:8080/api/employees/{id}

Method: PUT

Request Body (JSON): {

    "name": "John Doe",

    "jobTitle": "Senior Java Engineer",

    "salary": 90000.0,

    "department": {

        "name": "IT"

    }

}

Description: Updates an existing employee's details.

**Delete Employee by ID**

URL: http://localhost:8080/api/employees/{id}

Method: DELETE

Description: Deletes a specific employee by ID.

**Delete All Employees**

URL: http://localhost:8080/api/employees/deleteAll

Method: DELETE

Description: Deletes all employees.

**Execution and Test process:**

The Application can be executed by Running the EmployeeApplication file in src/main/java.

Database credentials need to be updated and, department data need to be added in database before execution.

For Running the test, run the file as Junit test under src/test/java

**Note: Success test scenarios have been passed and getting NullPointerException in Failure scenarios which I am working on to resolve it.**