Employee Management System Project Documentation

1. Skill Description

- **Technologies Used: Java, Spring Boot, Spring Cloud, Spring Data JPA, MySQL/MariaDB, RESTful APIs**

2. Problem Statement

Develop a distributed Employee Management System using Spring Microservices where:

- **Admin can manage employees, departments, and their roles.**
- **Employees can manage their profiles, view assigned tasks, and track performance.**

3. Microservices Architecture

3.1 Service Registry & Discovery

- **Eureka Server:**
  - **A centralized service registry where all microservices register themselves.**
  - **Enables dynamic discovery of services.**

3.2 API Gateway

- **Spring Cloud Gateway:**
  - **Acts as the entry point for all client requests.**
  - **Routes requests to the appropriate microservice.**
  - **Provides cross-cutting concerns like security, rate limiting, and logging.**

3.3 Authentication Service

- **User Authentication & Authorization:**
  - **Manages user registration, login, and JWT token generation.**
  - **Handles roles for Admin and Employee to secure access to different parts of the application.**

3.4 Employee Management Microservice

- **Employee Directory:**

- **Manages employee details, including personal information, job roles, and departments.**
- **Provides APIs for adding, editing, deleting, and fetching employee information.**
- **Handles the assignment of employees to departments and roles.**

3.5 Department Management Microservice

- **Department Catalog:**
  - **Manages departments within the organization.**
  - **Provides APIs for creating, editing, and deleting departments.**
  - **Organizes employees within departments and manages department-specific information.**

3.6 Task Management Microservice

- **Task Assignment:**
  - **Manages the creation and assignment of tasks to employees.**
  - **Provides APIs for task creation, updating, tracking, and completion.**
  - **Monitors task progress and deadlines.**

3.7 Performance Management Microservice

- **Employee Performance Tracking:**
  - **Tracks employee performance based on completed tasks and other metrics.**
  - **Provides APIs for performance reviews, feedback, and evaluation.**
  - **Generates performance reports for review by Admin.**

4. Project Flow

4.1 Admin Module

- **Admin Dashboard:**
  - **A centralized interface for the Admin to manage employees, departments, tasks, and performance.**
  - **Provides analytics and reports on employee performance and departmental efficiency.**
- **Employee Management:**

    - **Admin can perform CRUD operations on employees through the Employee Management Microservice.**
    - **Assign employees to specific roles and departments.**
- **Department Management:**
    - **Admin can create, view, edit, and delete departments via the Department Management Microservice.**
    - **Manage employee assignments within departments.**
- **Task Management:**
    - **Admin can create and assign tasks to employees through the Task Management Microservice.**
    - **Monitor task completion and update task statuses as needed.**
- **Performance Management:**
    - **Admin can track and evaluate employee performance through the Performance Management Microservice.**
    - **Schedule and manage performance reviews and feedback sessions.**

4.2 Employee Module

- **Employee Registration & Authentication:**
    - **Employees can register and log in using the Authentication Service.**
    - **JWT tokens are used to secure API access.**
- **Profile Management:**
    - **Employees can view and update their personal information through the Employee Management Microservice.**
    - **View assigned roles and departments.**
- **Task Management:**
    - **Employees can view and manage their tasks through the Task Management Microservice.**
    - **Update task statuses and track progress.**
- **Performance Tracking:**
    - **Employees can view their performance metrics and feedback through the Performance Management Microservice.**

- **Participate in performance reviews and track career development.**

5. Testing and Refinement

- **Unit Testing:**
  - **Each microservice is tested independently using JUnit and Mockito.**
- **Integration Testing:**
  - **End-to-end testing is performed to ensure seamless communication between microservices.**
- **Validation:**
  - **Implement validation for all user inputs at both client and server sides.**
- **Bug Fixing:**
  - **Continuously monitor and address bugs identified during testing.**
- **UI/UX Refinement:**
  - **Regularly update the user interface based on user feedback and testing results**