HR Payroll and Employee Management System

Phase 1: Synopsis & ERD (Entity-Relationship Diagram)

Team Members:

A011: Srushti Dewalekar

A018: Tanvi Halankar

A030: Mahi Khoriya

Table Of Contents

1. Introduction

The HR Payroll and Employee Management System is a cloud-ready, modular application built using the microservices architecture pattern. It aims to automate and streamline various Human Resource (HR) processes such as employee record keeping, salary computation, attendance tracking, and leave management. By decomposing these business functions into independent services, the system ensures high scalability, maintainability, and flexibility.

The application is designed for medium to large organizations that handle a growing number of employees and need to manage HR tasks efficiently and securely. Each microservice is responsible for a specific business domain and communicates with other services through RESTful APIs, ensuring

separation of concerns and fault isolation. This modularity also enables teams to develop, deploy, and scale services independently.

2. Problem Statement

Most traditional HR systems are monolithic and suffer from the following limitations:

- Lack of Modularity: Any update to one functionality (like payroll) requires deployment of the entire system.
- Poor Scalability: As the organization grows, these systems struggle to manage increasing data and user loads.
- Manual Processing: Many HR tasks such as salary generation, leave approval, and attendance tracking are still performed manually or via spreadsheets, increasing the likelihood of errors and delays.
- Difficult Maintenance: Tight coupling of features leads to challenges in debugging, scaling, and upgrading components.
- Security Risks: Centralized systems lack fine-grained access controls and often expose sensitive data across all users.

To overcome these issues, the proposed solution uses microservices to ensure decoupling, better data integrity, real-time processing, and secure access to HR functionalities.

3. Objectives:

- Automate HR Operations: Streamline the management of employee data, attendance, payroll, and leave requests.
- Microservices Architecture: Design modular services for key HR functions like Employee Management, Payroll, Attendance, and Leave.
- Data Security & Role-Based Access: Ensure only authorized personnel (e.g., HR Managers) can access sensitive data.
- Scalability & Maintainability: Facilitate rapid scaling and independent deployment of services.
- Integrate REST APIs: Build robust RESTful interfaces for cross-service communication and frontend integration.
- Provide Admin Insights: Optionally include dashboards or reports for HR personnel to gain insights on payroll, leaves, etc.

4. Scope of the Project

The system will be developed using Spring Boot microservices, with each service focused on a specific HR domain. The initial scope includes the following services:

1. Employee Service
    - Manage employee details (Create, Read, Update, Delete operations)
    - Associate employees with specific departments and roles
2. Payroll Service
    - Calculate net salary using base salary, applicable taxes, and deductions
    - Generate and store monthly payroll records for each employee
3. Attendance Service
    - Record daily attendance including check-in and check-out times
    - Track attendance status such as present, absent, half-day, etc.

4. Leave Management Service
    - Allow employees to apply for various types of leaves
    - Facilitate leave approval or rejection by managers or HR
    - Track leave balances and history for each employee
5. Authentication Service
    - Secure login and role-based access (e.g., Admin, HR, Employee).
    - Login with JWT token

5. Technologies Used

Category	Technology
Frontend (optional)	React
Backend	Java, Spring Boot
Database	MySQL
Version Control	GitHub
API Testing	Postman

6. Expected Outcome

- A fully functional HR system with modular microservices that independently manage payroll, attendance, employee data, and leave applications.
- Independent deployment pipelines for each microservice, supporting continuous development, testing, and integration.
- A normalized, scalable, and secure relational database schema to ensure data integrity and optimal performance.
- A clean and well-organized codebase with clearly defined service boundaries and thoroughly documented REST APIs.
- A publicly available GitHub repository containing the complete project source code, ERD diagram, and supporting documentation.

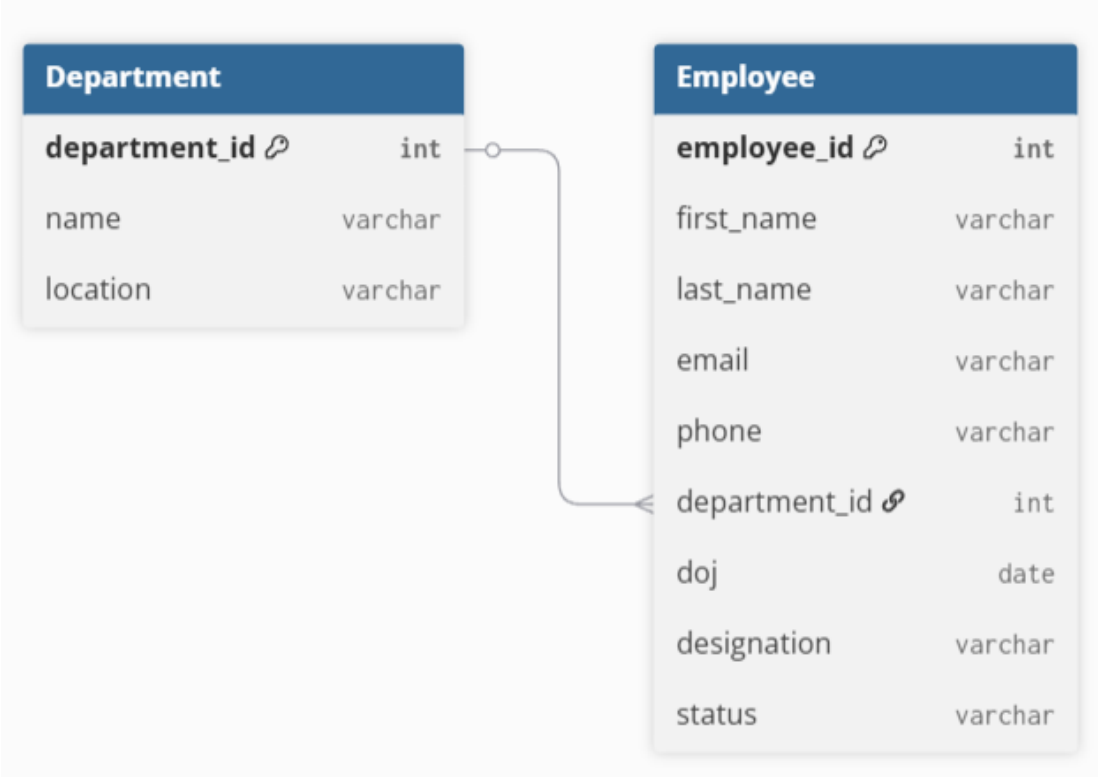
7. GitHub Repository Link

Link: https://github.com/TanviHalankar/CMS

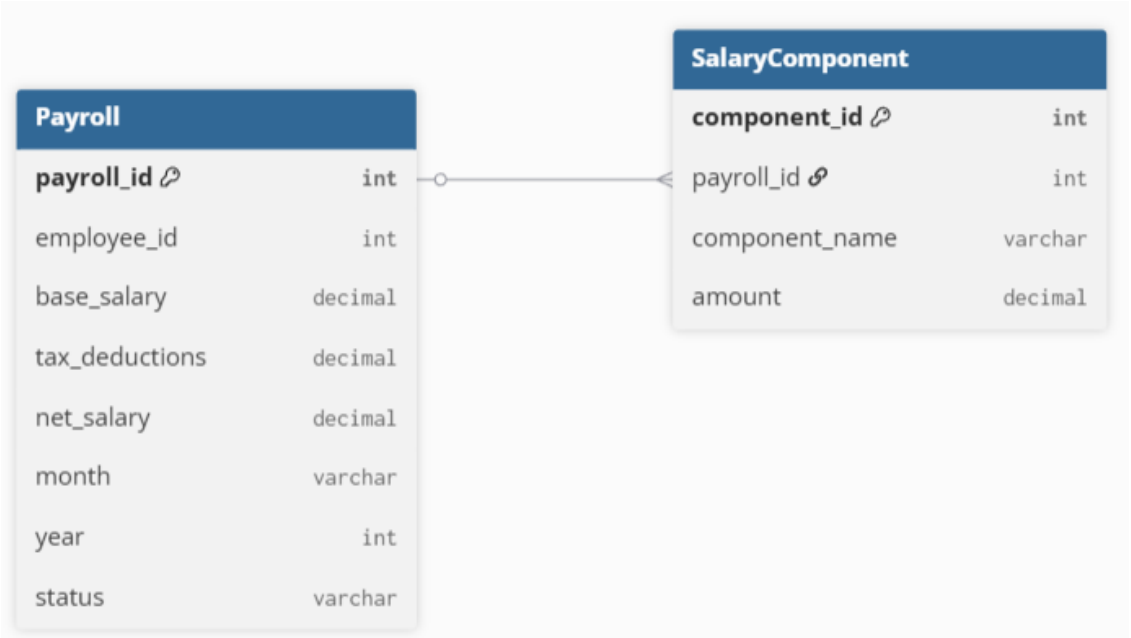
8. ER Diagram (For Each Service)

- Each microservice owns its own database.
- There should be no foreign key constraints across service boundaries.
- Hence each service has a separate ERD

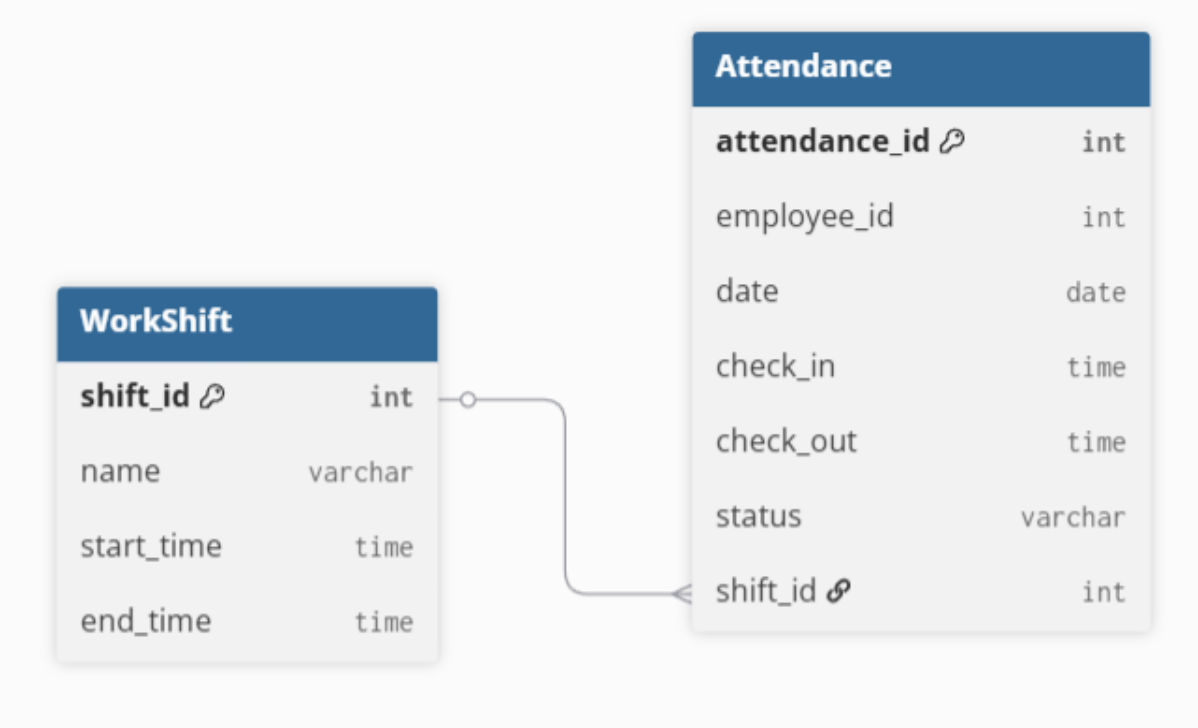
Employee Service



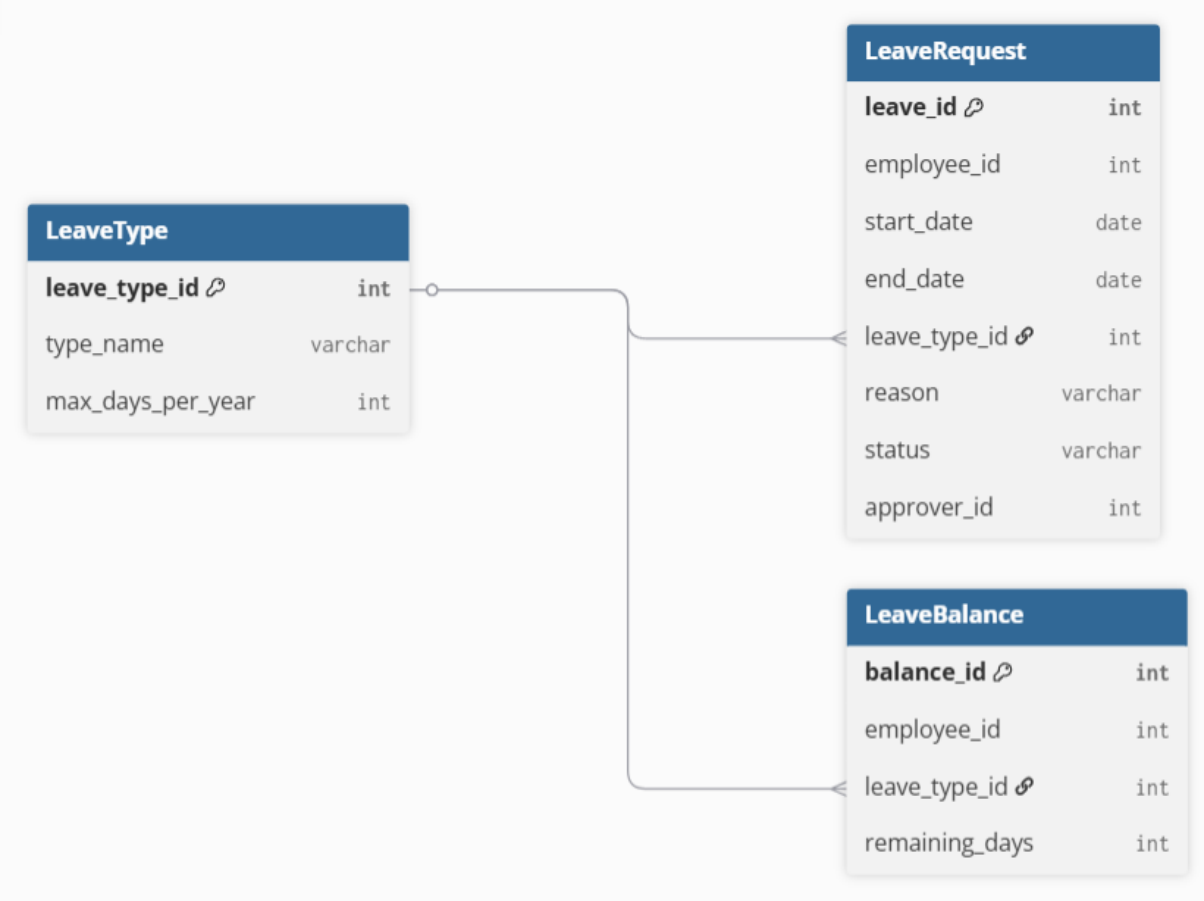
Payroll Service



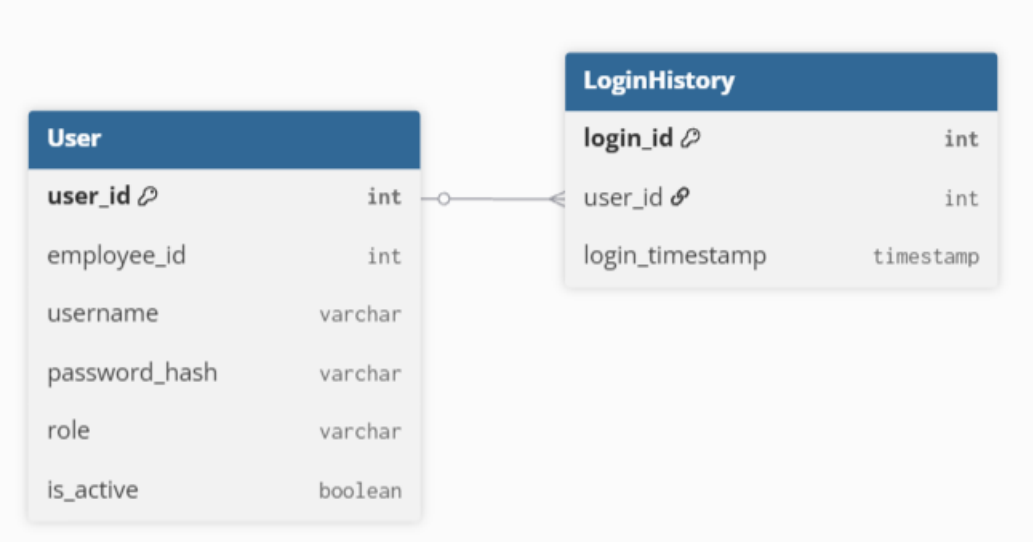
Attendance Service



Leave Management Service



Authentication Service



ER Diagram(Combined)