Capstone Project: Payroll Management System

A "**Payroll Management System"** is mainly used to simplify and automate how an organization handles employee salaries, deductions, and compliance. Managing employee payroll manually is prone to errors, time-consuming, and lacks transparency. Organizations face challenges in maintaining salary records, handling tax deductions, tracking employee leave, and generating accurate salary slips.

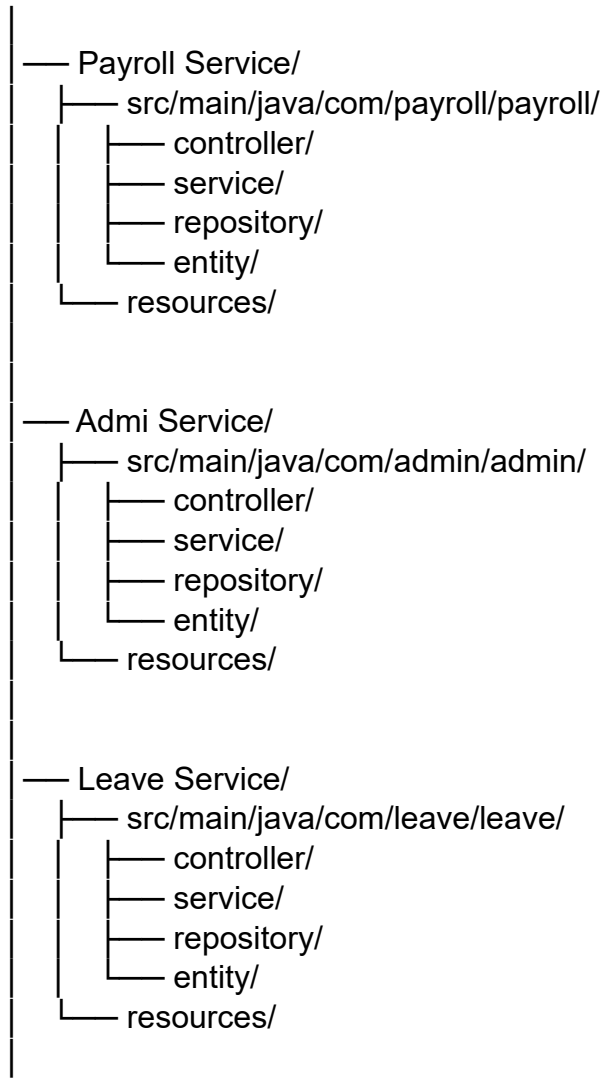
Overview:

- **Name of project:** Payroll Management System
- **Architecture:** Microservices with Spring Boot
- **Security:** JWT & Spring Security for authentication/authorization
- **Database:** MySQL (microservice)
- **Communication:** REST APIs + API Gateway
- **Service Discovery:** Eureka, Config Server

Folder Structure:

- **Employee Service** → manages employee data (CRUD).
- **Payroll Service** → calculates salaries, deductions, net pay.
- **Tax Service** → applies tax rules & compliance.
- **Auth Service** → login, signup, JWT authentication.
- **API Gateway** → single entry point for client requests.
- **Config & Eureka Server** → centralized configuration & service discovery.

```
Payroll-Management-System/
├── Employee Service/
│   ├── src/main/java/com/payroll/employee/
│   │   ├── controller/
│   │   ├── service/
│   │   ├── repository/
│   │   ├── entity/
│   │   └── dto/
│   └── resources/
```



Workflow:

1. User Authentication

- User logs in via the Authentication Service.
- A JWT token is generated and passed to the API Gateway.

2. API Gateway

- Serves as the single entry point.
- Routes requests to the appropriate microservice.

3. **Employee Service**

   - Manages employee records with CRUD operations.

4. **Payroll Service**

   - Fetches employee details.
   - Calculates salary, deductions, and net pay.

5. **Tax Service**

   - Fetches employee details.
   - Calculates salary, deductions, and net pay.

6. **Security**

   - All microservices communicate via JWT tokens.
   - Role-based access control (Admin, HR, Employee)

7. **Databases**
   - Each microservice has its own dedicated database.
8. **Config Server & Eureka Server**
   - Provide centralized configuration management and service discovery**.**

Security:

- JWT tokens for authentication.
- Role-based access (Admin, HR, Employee).
- Encryption for passwords.

Tools and Technologies:

- **Backend:** Java 17, Spring Boot, Spring Security, Spring Cloud
- **Database:** MySQL/PostgreSQL
- **IDE: IntelliJ** for backend, **Visual Studio Code** for frontend
- **Others:** Maven

Challenges:

Implementing Security with JWT

- Configuring Spring Security with JWT for authentication and authorization.
- Managing token expiration, refresh, and invalidation.
- Implementing role-based access control (Admin, HR, Employee) without breaking service communication.

Microservices Communication

- Ensuring smooth REST API communication between Employee, Payroll, and Tax services.
- Avoiding circular dependencies between services.
- Handling failures or downtime in one service without affecting others.

Error Handling Across Services

- Returning consistent and meaningful error messages via the API Gateway.
- Logging exceptions in each microservice for debugging**.**

Service Discovery & Configuration

- Configuring Eureka Server for service registration and discovery.
- Using Config Server for centralized configuration management.
- Resolving issues when a service fails to register or retrieve configs**.**

Database Management

- Maintaining separate databases for each microservice.
- Ensuring data consistency between Payroll and Tax services**.**

Deployment and Testing

- Testing multiple microservices together is more complex than monolithic apps.
- Ensuring all services start correctly and communicate via API Gateway.
- Debugging issues in a distributed environment.

Performance and Scalability

- Optimizing inter-service REST calls to reduce latency.
- Handling concurrent requests for payroll calculations**.**

Logging and Monitoring

- Implementing centralized logging for microservices.
- Monitoring service health and performance**.**

Conclusion:

The Payroll Management System offers a secure, scalable, and efficient way to manage employee salary processing. By using microservices with Spring Boot, it ensures modularity and easy maintenance. With Spring Security and JWT, role-based access and data protection are achieved. Overall, the system reduces manual effort, minimizes errors, and ensures compliance with payroll and tax regulations.