Payroll Management System Documentation

Introduction:

The Payroll Management System is a web application designed to automate payroll processing operations, reduce manual errors, and improve transparency in managing employee salary records, tax deductions, leave tracking, and salary slip generation. It implements role-based access with ADMIN and EMPLOYEE roles, ensuring secure operations using JWT authentication and password encryption.

System Scope:

Admin Role:

- Employee Management (Add, update, delete, view employee details)
- Payroll Processing
- Leave Management(Approve/Reject leave)
- Department & Job role management

Employee Role:

- Profile Management(view/update personal details)
- Leave Requests(Apply, check leave status)
- Salary slips(view/download monthly salary slips)

Security:

- JWT-based Authentication
- Role-based Access Control(Admin, Employee)

Technologies Used:

- **Backend:** Java, Spring Boot, Spring Security (JWT), MySQL, JPA/Hibernate.
- **Frontend:** React (Vite), Axios, Bootstrap 5, React Toastify, Formik and Yup.
- **API Testing:** Swagger UI and Postman.

Technology Stack :

Backend:

- Framework: Spring Boot 3.5.5
- Security: Spring Security with JWT authentication
- Database: MySQL
- Build Tool: Maven
- Testing: Postman, Swagger UI
- Dependencies: o Spring Boot Starter Web
- Spring Boot Starter Data JPA
- Spring Boot Starter Security
- Springdoc OpenAPI
- JWT (JWT implementation)
- Jakarta Validation API

Frontend :

- Framework: React 19.1.1
- UI Library: Bootstrap 5.3.8
- Form Handling & Validation: Formik & Yup
- HTTP Requests Axios
- JWT Handling: jwt-decode
- Routing: react-router-dom
- Icons: react-icons 5.5.0

System Architecture

- Authentication: JWT-based token authentication
- Role-Based Access Control: Separate routes and features for Admin and Employee
- Data Persistence: Hibernate JPA manages ORM between backend entities and MySQL

API Documentation:

The backend exposes RESTful APIs, secured with JWT authentication, with access controlled by role.

Work Process:

Day 1 – Backend Setup and Structured folders

1. Project Setup:

- Created a Spring Boot project using Spring Initializer.
- Added required dependencies: Spring Web, Spring Data JPA, Spring Security, MySQL Driver, JWT, Lombok, Validation.

2. Database Configuration:

- Created **MySQL database** payroll_db.
- Configured application.properties:
- springdoc.swagger-ui

3. Modules:

- Created entities for User, Employee, Department, JobRole, SalaryStructure, PayrollRun, PayrollProcess, Leaves.
- Implemented repository interfaces for CRUD.

.

4. Authentication Flow:

- Implemented Registration (only for Admin for the first time).
- Login with username & password which generates JWT token and navigates to the respective dashboard based on users' role.
- Integrated Swagger UI for API testing.

Challenges & Solutions

- Swagger UI showing 403 Unauthorized with login prompt.
- Cause: Security config didn't allow Swagger paths.
- Fix: Added /swagger-ui

Day 2 – Implementation:

- Employee: Request leave, view all requested leaves.
- Admin: Get pending leave requests and Approve/reject them.
- Defined salary structure (basic, bonus, deductions) by linking employees by their id.
- Created payroll runs.
- Employees can view their monthly payroll

Challenges & Solutions

- Issue: Initially, both Admin & Employee were using the same leave API which caused authorization
- Fix: Splitted endpoints by role and restricted access with correct HttpMethod.

Day 3 – Implementation:

1.Project Setup:

- Initialized React project with Vite.
- Installed dependencies: React-router-dom, JWT Decode, Axios, Formik, Yup, Bootstrap, React Toastify
- Ensured Admin cannot access Employee URLs and vice versa.
- Used React Router and JWT validation for route protection

2. Features:

- Welcome page: welcome note along with the login and register button.
- Login Page: Validates credentials and navigates to dashboard based on role.
- Admin Dashboard: View profile and manage employees, departments, jobs, leaves, payroll and get reports for a specified period.
- Employee Dashboard: Profile, payroll history, leave management.
- Form Validation: Used Formik & Yup for login & input forms.
- Styling: Bootstrap 5.

3.Testing:

- Verified all APIs in Swagger.
- Tested frontend workflows for both Admin and Employee

Challenges & Solutions:

- Navigation issues from one page to another page- I write the navbar page to remove that issues
- Creating the tables, and connecting them.

Conclusion:

In conclusion, a Payroll Management System is essential for any business, regardless of its size or industry. The system streamlines payroll processing, reduces errors, and enhances data security, making it a valuable asset for any business.