

Payroll Management System - Documentation

Introduction:

The **Payroll Management System** is a web application designed to automate payroll processing and leave management. It implements **role-based access** with **ADMIN** and **EMPLOYEE** roles, ensuring secure operations using **JWT authentication** and password encryption.

- **Admin Role:** Manages employees, departments, jobs, payroll runs, leaves, and reports.
- **Employee Role:** Views personal profile, payroll and requests leave.
- **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.

Work Process:

Day 1 – Backend Setup and Structured folders

1. Project Setup:

- Created a Spring Boot project using Spring Initializr.
- 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:

- `spring.jpa.hibernate.ddl-auto=update`: Automatically updates DB schema.
- `jwt.secret`: Secret key for token generation.
- `logging.level`: Helps debug security flow.
- `springdoc.swagger-ui.*`: Enables Swagger for API testing.

3. Core Modules:

- Created entities for **User, Employee, Department, JobRole, SalaryStructure, PayrollRun, PayrollProcess, Leaves**.
- Implemented repository interfaces for CRUD.
- Worked on services and controllers for managing users, employees, departments and job role CRUD operations.

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.
- **Password Encryption** using BCrypt.
- 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/**` and `/v3/api-docs/**` to `permitAll()` at the initial stage to test the apis' in swagger.

Day 2 – Implemented other Modules

1. Leaves Management:

- Employee: Request leave, view all requested leaves.
- Admin: Get pending leave requests and Approve/reject them.

2. Payroll Module:

- Defined salary structure (basic, bonus, deductions) by linking employees by their id.
- Created payroll runs.
- Employees can view their monthly payroll.

3. Reports:

- Payroll summary (of all the employees combined).
- View department-wise payroll cost.

4. Security Configuration:

- Defined role-based access with SecurityFilterChain.

Ex:

```
requestMatchers("/api/v1/leaves/**").hasAnyRole("ADMIN","EMPLOYEE")
```

Challenges & Solutions

- **Issue:** Initially, both Admin & Employee were using the same leave API which caused authorization (403) issue .
- **Fix:** Splitted endpoints by role (Admin vs Employee) and restricted access with correct HttpMethod rules.

Day 3 – Frontend Development (React + Vite)

1. Project Setup:

- Initialized React project with **Vite**.
- Installed dependencies: React-router-dom, JWT Decode, Axios, Formik, Yup, Bootstrap, React Toastify.

2. Folder Structure:

- src/pages/auth for Login.
- src/pages/admin for Admin dashboard & management.
- src/pages/employee for Employee dashboard, leaves and view payroll.
- src/components for shared components (Navbars, Footer, InputForms for validations).

3. Features:

- **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.

Challenges & Solutions

- **Issue:** CORS and Responsiveness of tables.

- **Fix:** Implemented **CORS** in **security config** along with the **SecurityFilterChain** in the backend and gone through the documentation of bootstrap to know the classnames for tables and div tag.

Day 4 – Testing, Integration & Security

1. Protected Routes:

- Ensured Admin cannot access Employee URLs and vice versa.
- Used React Router and JWT validation for route protection.

2. Logout Functionality:

- Cleared JWT from local storage.
- Redirects to login page.

3. Testing:

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

Conclusion:

The Payroll Management System provides seamless management of employee data, payroll, leave requests and reports. It ensures security through role-based access, encrypted passwords, and JWT authentication.

- **Admins:** Full control over employee management, departments, job roles, leave approvals, payroll, and reports.
- **Employees:** View profile, request leave, and view payroll.