Capstone Project: Payroll Management System

**Author:** R Susanna  
**Email:** susannarupanagudi@gmail.com

**1. Problem Statement**

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.

**Solution:**  
A Payroll Management System automates payroll operations, provides role-based access (Admin and Employee), and ensures secure interactions using JWT Authentication.

**2. Scope of the System**

**Admin Role**

* Employee Management: Add, update, delete, and view employee details
* Payroll Processing: Generate monthly salaries
* Leave Management: Approve or reject leave requests
* Salary History: Track payroll history
* Departments & Jobs: Define departments and job roles with base salaries

**Employee Role**

* Profile Management: View and update personal details
* Leave Requests: Apply for leave and check status
* Salary Slip: View monthly salary slips

**Security**

* JWT Token Authentication for login and API authorization
* Role-based access control (Admin vs. Employee)

**3. Project Development Guidelines**

**Backend (Spring Boot + MySQL)**

**Technology Stack:**

* Spring Boot (REST APIs)
* Spring Security with JWT Authentication
* MySQL Database (Employee, Payroll, Leave)
* JPA/Hibernate for ORM

**Modules:**

1. Authentication – Login and role assignment
2. Employee – CRUD operations for employee details
3. Payroll – Salary calculation and history
4. Leave – Leave request and approval workflow
5. Profile – Employee personal data updates
6. Reports – Payroll summary and department cost reports

**Frontend (React.js)**

**Technology Stack:**

* React.js (Functional Components + Hooks)
* React Router for navigation
* Axios for API calls
* JWT Token storage (localStorage/sessionStorage)
* Bootstrap and custom CSS

**Modules:**

* Authentication – Login with JWT handling
* Admin Dashboard – Manage employees, payroll, leave, departments, jobs
* Employee Dashboard – View profile, request leave, view salary slip
* Role-Based Routing – Conditional navigation for Admin and Employee

**API Guidelines**

* Base URL: /api/v1
* All APIs secured with JWT
* Role-based access applied to each endpoint

**Key Endpoints:**

* Authentication and User Management
* Employee Management
* Payroll Processing
* Leave Management
* Departments and Job Roles
* Reports

**Database Guidelines**

* Normalize to at least 3rd Normal Form
* One-to-one mapping between users and employees
* One department → Many employees; One job → Many employees
* Historical salary tracking with effective dates

**Tables:**

1. **users:** user\_id (PK), username, password, email, role
2. **employees:** employee\_id (PK), user\_id (FK), first\_name, last\_name, dob, phone, address, designation, department, salary
3. **payroll:** payroll\_id (PK), employee\_id (FK), basic\_salary, deductions, bonus, net\_salary, pay\_date
4. **leave:** leave\_id (PK), employee\_id (FK), start\_date, end\_date, leave\_type, status

**UX Guidelines**

| **Principle** | **Implementation** |
| --- | --- |
| Consistency | Uniform color schemes, fonts, buttons, table layouts, forms |
| Clarity & Simplicity | Clear labels, simple forms, highlight salary, leave status, deductions |
| Feedback & Response | Confirmation messages and loading indicators |
| Error Prevention & Handling | Input validation, clear error messages, prevent invalid operations |

**Expected Outcomes**

* Automated payroll reduces errors and improves efficiency
* Role-based access ensures security and transparency
* Employees can view salary slips and leave status anytime
* Modular design allows scalability and future integration
* Reporting and analytics track salary history, department costs, and leave trends

**Project Structure Guidelines**

* Separate folders for frontend, backend, and documentation
* Separate configuration files, source code, and assets
* Consistent coding standards, naming conventions, and version control

**Challenges Faced**

1. **Implementing Secure Authentication**  
   Managing secure login and role-based access using JWT tokens was complex, requiring proper token generation, validation, and role restrictions.
2. **Accurate Payroll Calculations**  
   Payroll calculation involves multiple components such as basic salary, deductions, bonuses, and leave adjustments. Ensuring accuracy and reliability was a major challenge.
3. **Frontend-Backend Synchronization**  
   Keeping the React frontend synchronized with backend APIs, handling asynchronous calls, managing state, and displaying role-based dashboards accurately was challenging.