Capstone Project-Payroll Management System

Introduction

Payroll is one of the most important functions in any organization. A manual process can be error-prone, time-consuming, and inefficient. This Payroll Management System automates the calculation of salaries, tax deductions, and leave management. It provides role-based access for Admin and Employees, making payroll operations streamlined, accurate, and secure**.**

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.

A Payroll Management System is required to automate payroll operations, provide role-based access (Admin and Employee), and ensure secure interactions using JWT Authentication.

Objectives

- Automate payroll calculation and salary disbursement.
- Maintain employee records including jobs, departments, and leave history.
- Provide secure login & authentication for different roles.
- Generate reports for payroll, employees, and departments.
- Deliver a modern, responsive React UI for easy interaction.

Scope of the System

Admin Role

- Employee Management – Add, update, delete, view employee details.
- Payroll Processing – Generate monthly salary based on employee details.
- Leave Management – Approve/reject employee leave requests.
- Salary History – Track payroll history for employees.
- Departments & Jobs – Define departments and job roles with base salaries.

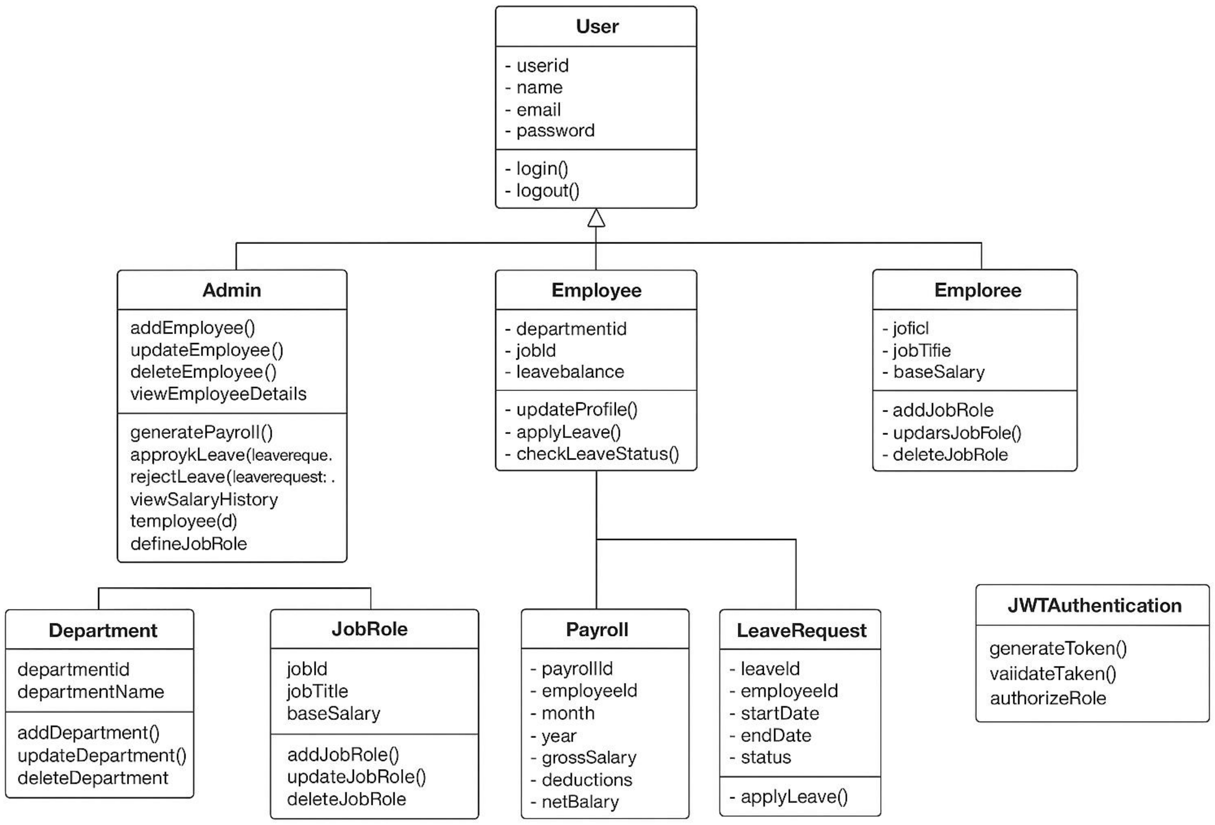
Employee Role

- Profile Management – View/update personal details.
- Leave Requests – Apply for leave and check leave status.
- Salary Slip – View monthly salary slips.

Security

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

UML Diagram



Technology Stack

Backend:

- Spring Boot (REST APIs)
- Spring Data JPA / Hibernate
- Spring Security + JWT (Authentication & Authorization)
- MySQL (Database)

Frontend:

- React JS (UI framework)
- Bootstrap (UI styling)
- Axios (API calls)

Tools:

- Maven (dependency management)
- Postman (API testing)
- GitHub (version control)

Modules & Features

Authentication Module

- Login with JWT Token
- Role-based access (ADMIN, EMPLOYEE)
- Session management

Employee Management

- Add / Update / Delete Employees
- Assign Department & Job
- View employee details

Payroll Management

- Generate salary slips
- Apply tax & deductions
- Store salary history

Leave Management

- Employees apply for leave
- Admin approves/rejects leave
- Leave balance calculation

Reports

- Payroll reports by month
- Employee reports by department/job
- Leave reports

Database Design

Main Tables:

- users → login credentials, roles
- employees → employee profile, department, job
- departments → department master
- jobs → job title & salary band
- salary_structure → base pay, allowances, deductions

- payroll → salary history, net pay
- leave → leave requests & status

Backend Implementation

- **Entities**: Employee, Department, Job, Payroll, Leave
- **Repositories**: JPA repositories for CRUD operations
- **Services**: Business logic for payroll calculation & leave approval
- **Controllers**: REST APIs for frontend communication
- **Security**: JWT Authentication, role-based authorization

Frontend Implementation

- **React Components**: Login, Dashboard, Employees, Payroll, Reports
- **State Management**: React useState & useEffect hooks
- **API Integration**: Axios calls to Spring Boot backend
- **UI**: Bootstrap tables, forms, and modals

Sample User Roles

- **Admin**: Manage employees, payroll, leave approvals, and reports
- **Employee**: View salary slips, request leave, view personal details

Testing

- **Integration Tests**: Spring Boot Test for API endpoints
- **API Testing**: Postman collection with sample requests

Challenges Faced During Implementation

Backend Challenges

API Implementation Issues

- Developing REST APIs with Spring Boot was challenging, particularly ensuring proper request/response handling across different modules (Employee, Payroll, Leave, etc.).

- At times, inconsistent request mappings or missing validations led to unexpected errors.

Authentication & Authorization Errors

- During API testing in Postman and Swagger UI, frequent 401 Unauthorized and 403 Forbidden responses occurred.
- These issues were primarily due to incorrect JWT token handling, missing Authorization headers, or misconfigured role-based access control.

JWT Security Token Handling

- Implementing JWT authentication introduced difficulties such as token expiration handling, token validation errors, and ensuring secure communication.
- Misalignment between frontend token storage and backend validation also contributed to failures.

Registration API Failures

- While building the user registration API, errors arose from password encoding, duplicate usernames/emails, and validation checks.
- Handling proper exception responses and providing meaningful error messages to the client required multiple iterations.

Frontend Challenges

Integration with Backend APIs

- The frontend frequently faced issues due to incorrect or failed backend API responses.
- Login and registration forms often failed because of 401/403 errors, which were directly linked to JWT misconfigurations on the backend.

Form Handling and Validation

- Ensuring smooth data flow from React forms to backend APIs posed challenges.
- Missing validations or incorrect request payloads caused registration/login failures.

UI/UX Responsiveness

- In some cases, buttons and navigation components did not respond as expected.
- This was mainly due to improper event handling, missing state updates in React components, or delays in API response handling.

Challenges and Fixes

Backend – Challenges & Fixes

- **API Implementation Issues** → Fixed by standardizing request mappings, adding validations, and global exception handling.
- **401/403 Unauthorized Errors** → Fixed by correctly passing JWT in Authorization: Bearer <token> and adjusting role-based access.
- **JWT Token Handling Errors** → Fixed by proper token generation/validation, expiry checks, and aligning frontend-backend token flow.
- **Registration API Failures** → Fixed by using unique username/email checks, and clear error messages.

Frontend – Challenges & Fixes

- **API Integration Failures** → Fixed by adding Axios interceptors for JWT and handling unauthorized responses.
- **Form Handling & Validation Issues** → Fixed by adding React form validations and matching request payloads with backend DTOs.
- **UI/UX Responsiveness Problems** → Fixed by correcting event bindings, using React hooks properly, and adding loading indicators.

Future Enhancements

- Add email notifications for payslip generation
- Integration with payment gateways (direct salary transfer)
- Mobile App (React Native)
- Analytics Dashboard with charts

Conclusion

The Payroll Management System successfully automates the payroll process, ensuring **accuracy, transparency, and security**. It eliminates manual calculations, reduces errors, and provides a scalable solution for organizations of any size.