Capstone Project Documentation

Payroll Management System

Introduction

Payroll is one of the most critical operations in any organization. It involves calculating salaries, managing employee benefits, deducting taxes, approving leaves, and generating accurate payslips. Traditionally, these tasks are handled manually, which is prone to errors, time-consuming, and lacks transparency.

The Payroll Management System (PMS) is designed to automate these operations and provide a secure, efficient, and user-friendly platform for both administrators and employees. With automation, organizations can ensure timely salary processing, accurate deductions, and easy access to payroll history.

This project uses Spring Boot for the backend, MySQL as the database, and React.js for the frontend, with JWT authentication to ensure secure access. The system supports role-based access, where administrators can manage employees, payroll, and departments, while employees can manage their own profiles, view payslips, and request leaves.

Problem Statement

Organizations face challenges in maintaining salary records, handling tax deductions, tracking employee leave, and generating accurate salary slips. Manual payroll management is inefficient, prone to human errors, and lacks security and transparency. An automated system is required to address these challenges.

Objectives

- **Automate Payroll Processing** – Eliminate manual errors by automatically calculating salaries, deductions, and bonuses.
- **Provide Role-Based Access** – Ensure secure and controlled access for administrators and employees through JWT authentication.
- **Efficient Employee Management** – Enable administrators to add, update, delete, and manage employee records effectively.
- **Leave Management** – Allow employees to apply for leave and administrators to approve or reject requests seamlessly.
- **Generate Reports** – Provide payroll history, salary slips, and department cost reports for transparency and decision-making.
- **Ensure Security & Transparency** – Protect data with encryption and allow employees to access their payroll and leave information anytime.

Functional Requirements

Admin Role:

- The Admin is the central authority who manages all employee and payroll-related operations in the system.
- In Employee Management, the admin can add new employees, update their personal or job details, delete records of inactive staff, and view all employee profiles.
- Through Payroll Processing, the admin calculates monthly salaries by considering basic pay, deductions, and allowances.
- The system ensures payroll accuracy, and the admin can lock payroll runs once finalized to prevent modifications.
- In Leave Management, the admin reviews employee leave requests and has the authority to approve or reject them based on company policies.
- Admin also manages leave balances, ensuring leave deductions reflect accurately in payroll.
- With Salary History tracking, the admin can monitor previous payroll runs, verify past payments, and resolve disputes.
- Admin is responsible for setting up Departments (e.g., HR, Finance, IT) to organize employees properly.
- Under Job Roles, the admin defines positions (like Manager, Developer) and assigns base salaries to each role.
- Overall, the admin ensures smooth functioning, transparency, and compliance in payroll and employee management.

Employee Role:

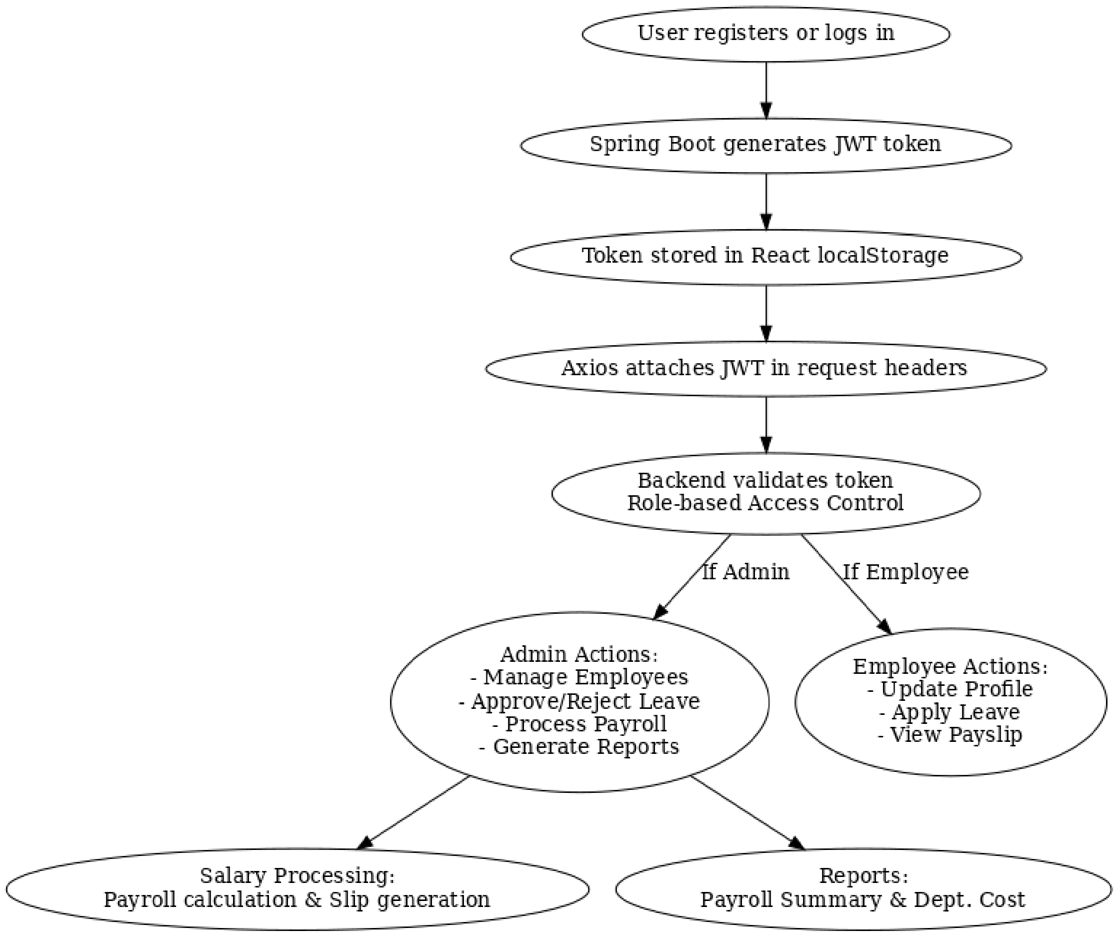
- The Employee role is designed for individual staff members to manage their own information and payroll transparently.
- Through Profile Management, employees can view their personal details such as name, contact info, department, and job role.
- Employees are also allowed to update certain fields like phone number, address, or emergency contact for accuracy.
- The system ensures that sensitive details like salary or role changes can only be modified by the admin.
- In the Leave Requests module, employees can apply for different leave types such as sick leave, casual leave, or earned leave.
- Each leave request is submitted digitally, reducing paperwork and ensuring faster approval workflows.
- Employees can track the status of their leave (Pending, Approved, or Rejected) directly from their dashboard.
- The Salary Slip module allows employees to view their monthly payslips securely.

- Payslips include details like gross salary, deductions, net pay, and payment date for full transparency.
- Overall, the Employee role empowers staff with self-service capabilities, reducing dependency on HR/Admin for routine tasks.

Security:

- The system uses JWT (JSON Web Token) Authentication to ensure only valid users can log in and access APIs.
- When a user (Admin or Employee) logs in successfully, the backend generates a JWT token containing user details and their role.
- This token is stored securely in the frontend (React) using localStorage/sessionStorage for session management.
- For every API call made through Axios, the JWT token is attached in the Authorization header.
- The Spring Boot backend verifies the token before processing the request to confirm the user's identity.
- Role-based access control (RBAC**)** ensures that actions are limited based on roles.
- For example, only Admins can add employees, process payroll, or generate reports.
- Employees, on the other hand, can only update their profile, apply for leave, and view salary slips.
- Unauthorized access attempts (like an Employee trying to access Admin APIs) return 401 Unauthorized or 403 Forbidden errors.
- Together, JWT authentication and RBAC provide data security, integrity, and controlled access, protecting sensitive payroll information.

Work Flow


User registers or logs in
Spring Boot generates JWT token
Token stored in React localStorage
Axios attaches JWT in request headers
Backend validates token
Role-based Access Control
If Admin
If Employee
Admin Actions:
- Manage Employees
- Approve/Reject Leave
- Process Payroll
- Generate Reports
Employee Actions:
- Update Profile
- Apply Leave
- View Payslip
Salary Processing:
Payroll calculation & Slip generation
Reports:
Payroll Summary & Dept. Cost

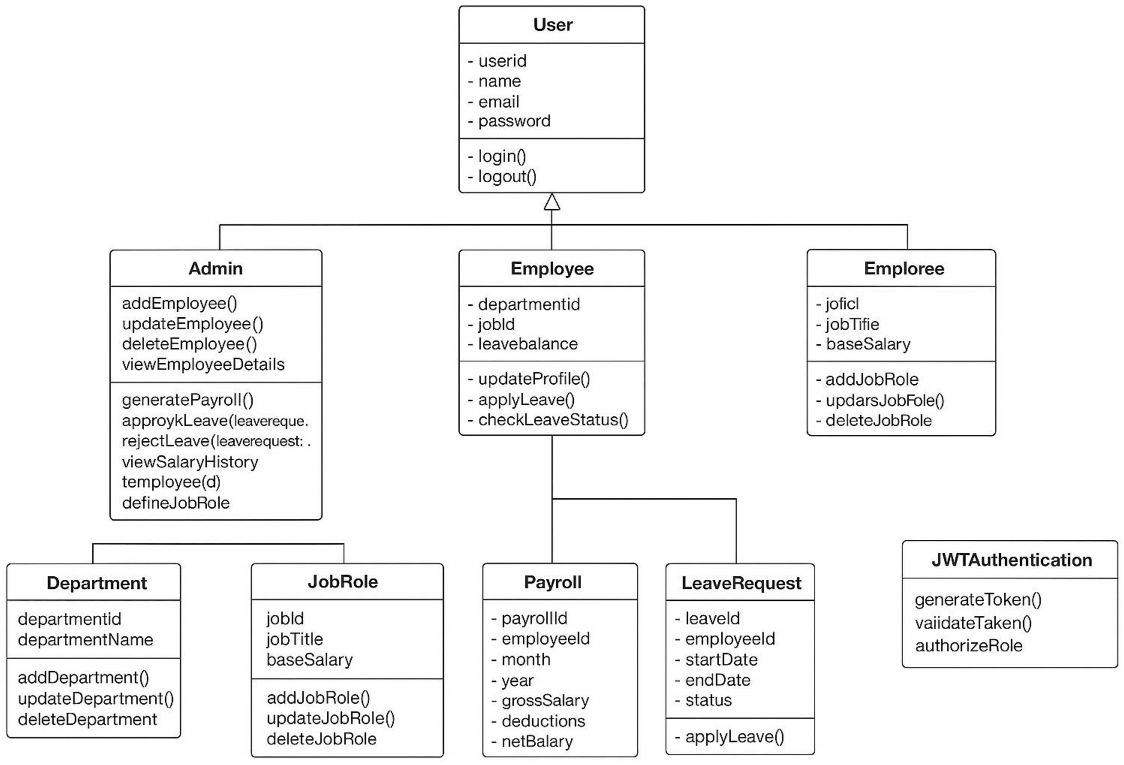
Scope of the System
The Payroll Management System supports two roles: Admin and Employee.

Admin Capabilities:
- Manage employees (add, update, delete, view).
- Process monthly payroll and track salary history.
- Approve/reject leave requests.
- Manage departments and job roles with salary structures.
- Generate payroll and department cost reports.

Employee Capabilities:
- Manage personal profile details.
- Apply for leave and check leave status.
- View monthly salary slips.

Security:
- JWT authentication and role-based access control ensure secure operations.

UML Diagram



1. User Class

Attributes: userid, name, email, password
Methods:
- login() – Authenticates the user.
- logout() – Ends the session.

This is the base class for all system users. Both Admin and Employee inherit from it.

2. Admin Class (inherits from User)

Methods:
- addEmployee()
- updateEmployee()
- deleteEmployee()
- viewEmployeeDetails()
- generatePayroll()
- approveLeave(leaveRequest)
- rejectLeave(leaveRequest)

• defineJobRole()
Admin has full control over the management functions of payroll, employees, and leave.

3. Employee Class (inherits from User)

Attributes: employeeName, departmentid, jobId
Methods:
- updateProfile()
- applyLeave()
- checkLeaveStatus()

Employees can only manage their own profile, leave, and payroll viewing.

4. Department Class

Attributes: departmentid, departmentName
Methods:
- addDepartment()
- updateDepartment()
- deleteDepartment()

Departments categorize employees (e.g., HR, IT, Finance).

5. JobRole Class

Attributes: jobId, jobTitle, baseSalary
Methods:
- addJobRole()
- updateJobRole()
- deleteJobRole()

Defines different roles (e.g., Manager, Developer) with associated salaries.

6. Payroll Class

Attributes: payrollId, employeeId, month, year, grossSalary, deductions, netSalary
Represents payroll records for each employee.

7. LeaveRequest Class

Attributes: leaveId, employeeId, startDate, endDate, status (Pending/Approved/Rejected)
Methods:
- applyLeave() – Employees apply for leave.

Tracks employee leave requests.

8. JWTAuthentication Class

Methods:
- generateToken() – Creates a JWT token for a user session.
- validateToken() – Verifies the authenticity of a token.
- authorizeRole() – Ensures role-based access (Admin vs Employee).

Provides the security layer for login and access control.

Relationships

Inheritance:
- Admin and Employee inherit from User.

Associations:
- Admin manages Department, JobRole, Payroll, and LeaveRequest.
- Employee is associated with Payroll and LeaveRequest.
- JWTAuthentication is used by both Admin and Employee for secure login and authorization.

In summary:
This UML class diagram models a Payroll Management System where User is the base entity. Admins have management privileges, while Employees have limited self-service functionality. Payroll, job roles, departments, and leave requests are managed as separate entities, while JWTAuthentication ensures secure system access.

System Architecture

Architecture Flow:

1. React Frontend – Employees and Admins interact through a responsive UI built with React.js and Bootstrap.
2. Axios – Handles HTTP requests and communicates securely with backend APIs.
3. Spring Boot REST API – Processes requests, applies business logic (e.g., payroll calculation, leave approval), and interacts with the database.
4. MySQL Database – Stores users, employees, departments, jobs, payroll history, and leave records.
5. JWT Security Layer – Ensures authentication and role-based authorization. Admins have access to management features; employees access only their own data.

Database Design

Entities & Tables:

- User : user_id (PK), username, password, email, role (Admin/Employee).
- Employee : employee_id (PK), user_id (FK), first_name, last_name, dob, phone, address, designation, department, salary.
- Payroll : payroll_id (PK)**,** employee_id **,** basic_salary**,** deductions**,** bonus**,** net_salary, pay_date.

- Leave : leave_id (PK), employee_id, start_date, end_date, leave_type, status.

Relationships:

- One User ↔ One Employee (1:1).
- One Department ↔ Many Employees (1:N).
- One Job ↔ Many Employees (1:N).
- One Employee ↔ Many Payroll Records (1:N).
- One Employee ↔ Many Leave Requests (1:N).

Technology Stack

- Frontend: React.js (Hooks, Axios, React Router, Bootstrap).
- Backend: Spring Boot, Spring Security (JWT), Spring Data JPA.
- Database: MySQL.
- Authentication & Authorization: JWT.
- API Documentation: Swagger UI.
- Version Control: Git & GitHub.

API Design

Authentication

- POST /auth/login → Login and generate JWT.
- POST /auth/register → Register a new user (Admin only).

Employees

- POST /api/employees → Add employee (Admin).
- GET /api/employees → Fetch all employees (Admin).
- GET /api/employees/{id} → Get employee details (Admin/Self).
- PUT /api/employees/{id} → Update employee (Admin).
- DELETE /api/employees/{id} → Remove employee (Admin).

Departments & Jobs

- POST /api/departments → Add department (Admin).
- GET /api/departments → View all departments (Admin).
- POST /api/jobs → Add job role (Admin).
- GET /api/jobs → View all jobs (Admin).

Payroll

- POST /api/payroll/process → Process salary for all employees (Admin).
- GET /api/payroll/history/{employeeId} → View salary history (Admin/Employee).
- GET /api/payroll/slip/{year}/{month} → Get payslip (Employee).

Leave Management

- POST /api/leave/request → Apply for leave (Employee).
- GET /api/leave/status/{employeeId} → Check leave status (Employee).
- PATCH /api/leave/approve/{leaveId} → Approve leave (Admin).
- PATCH /api/leave/reject/{leaveId} → Reject leave (Admin).

Reports

- GET /api/reports/payroll-summary → Payroll summary (Admin).
- GET /api/reports/department-cost → Department-wise salary cost (Admin).

Security Implementation

- Spring Security + JWT Authentication
- Login generates JWT, stored in React localStorage.
- Axios interceptors attach JWT to every request.
- Role-based access control:
    - /api/admin/** → Admin only.
    - /api/employee/** → Employee only.

Implementation Details

Frontend (React)

- Pages: Login, Register, Admin Dashboard, Employee Dashboard.
- Axios services for API calls with JWT header.
- Role-based navigation (Admin vs Employee).

Backend (Spring Boot)

- Controllers → Handle API endpoints.
- Services → Contain business logic (payroll calculation, leave approval).
- Repositories → Database operations via JPA.
- SecurityConfig → JWT filter, role-based restrictions.

Testing Phase
Testing ensures the PMS is secure, reliable, and meets requirements.

Sample Test Cases

Test ID	**Scenario**	**Input**	**Expected Output**	**Result**
TC-01	Admin Login	Valid credentials	JWT Token generated	Pass
TC-02	Employee Login	Invalid password	Error: Unauthorized	Pass

Test ID	Scenario	Input	Expected Output	Result
TC-03	Add Employee	Employee JSON	Employee added successfully	Pass
TC-04	Process Payroll	Run payroll for July	Payslips generated	Pass
TC-05	Apply Leave	Sick leave request	Status: Pending	Pass
TC-06	Approve Leave	Leave ID = 21	Status: Approved	Pass
TC-07	Unauthorized Access	Employee hits /api/admin/employees	403 Forbidden	Pass

---

UI Mockups (Sample Screens)

- Login Page.
- Admin Dashboard (Employees, Payroll, Reports).
- Employee Dashboard (Profile, Leave, Payslip).

Benefits

- Centralized payroll automation.
- Reduces manual errors and saves HR time.
- Secure role-based access with JWT.
- Transparent leave tracking.
- Easy API documentation with Swagger.

Expected Outcomes
1. Efficiency & Accuracy – Automated payroll reduces manual errors.
2. Role-Based Access – Admins manage all data, employees access only their records.
3. Transparency – Employees can view salary slips and leave status anytime.
4. Security – JWT ensures secure login and API communication.
5. Scalability – Modular design supports future integration with tax or attendance systems.
6. Reporting – Admins can generate payroll and cost reports for decision-making.

Conclusion

The Payroll Management System automates payroll operations, ensuring efficiency, accuracy, security, and transparency. With modern technologies like Spring Boot, React.js, MySQL, and JWT, the system provides a scalable solution for organizations. Future enhancements such as tax integration, biometric attendance, and mobile support make it adaptable for long-term organizational growth.