Payroll Management System

1. Introduction

Payroll management is one of the most important functions of any organization. The process includes maintaining employee details, calculating salaries, bonuses, deductions, generating pay slips, and handling leave approvals. Traditionally, payroll management has been done manually, which is time-consuming and prone to errors.

The Payroll Management System automates these tasks, making payroll processing more efficient, accurate, and secure. This project is developed using React (Frontend), Spring Boot (Backend), and MySQL (Database).

2. Objectives

- To automate the payroll process of employees.
- To manage employee profiles, departments, and job roles.
- To provide employees access to their profiles, leaves, and salary slips.
- To provide administrators with tools to manage employees, payroll, leaves, and reports.
- To generate salary slips and department-wise cost reports automatically.

3. Business Problem Solved

Traditional payroll management suffers from:

• Manual calculation errors leading to financial discrepancies

• Time-consuming leave approval processes

• Lack of real-time visibility into employee data

• Inefficient communication between HR and employees

4. Scope of the System

Admin Role

- Manage employees (CRUD operations).
- Process payroll (generate salaries, maintain salary history).
- Manage departments & job roles.
- Approve/reject leave requests.
- Generate reports (salary, department cost).

Employee Role

- Manage personal profile.
- Apply and track leave requests.
- View salary slips and payroll history.

5. Technical Stack

Frontend Architecture:

- **Framework:** React 19 with Functional Components
- **State Management:** Context API + useState/useEffect
- **HTTP Client:** Axios with Interceptors
- **UI Framework:** Bootstrap 5 + Bootstrap Icons

Backend Architecture:

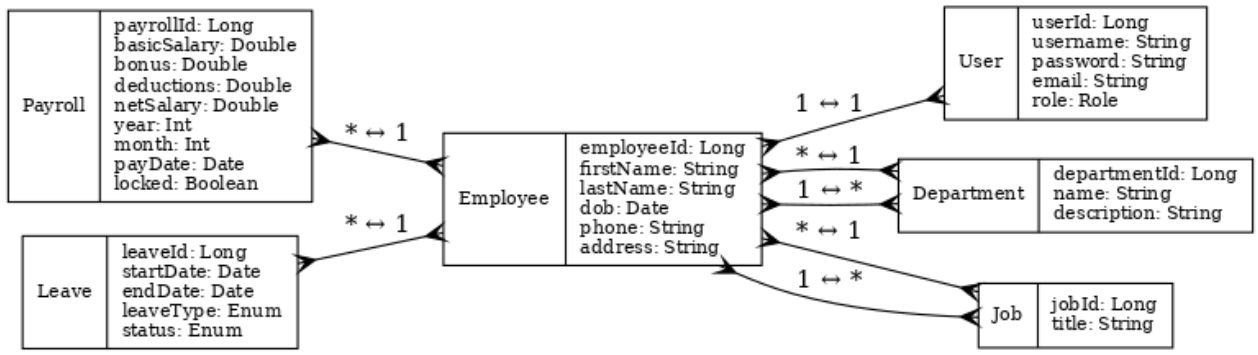
- **Framework:** Spring Boot 3.5 with Spring Security
- **Authentication:** JWT with Bearer tokens
- **Persistence:** Spring Data JPA with Hibernate
- **Documentation:** Springdoc OpenAPI 3
- **Testing:** JUnit 5 + Mockito

6. Database Design

Main Entities:

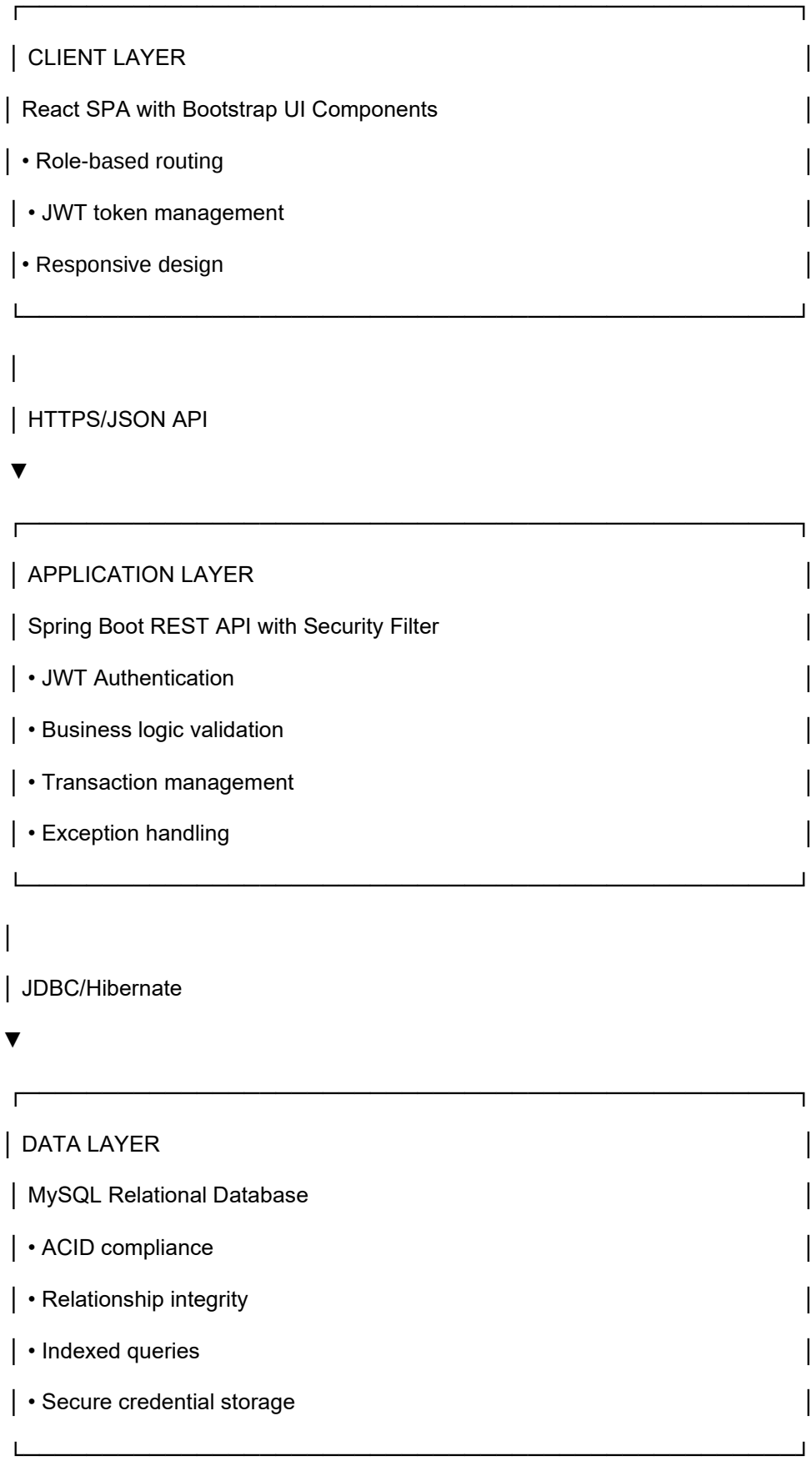
- User (user_id, username, password, email, role).
- Employee (employee_id, user_id, firstname, lastname, dob, phone, address, department_id, job_id etc.).
- Payroll (payroll_id, employee_id, basicSalaty, bonus, deduction, netSalary, year, month).
- Leave (leave_id, employee_id, leave_type, start_date, end_date, status).

7. UML Class Diagram



System Architecture

Three-Tier Architecture Overview



8. Complete Rest API Endpoints

1. Authentication

POST /auth/register -> Register a new employee (role: EMPLOYEE)

POST /auth/login -> Login with email & password (returns JWT token)

2. Employees

GET /employees → Get all employees (Admin only)

GET /employees/{id} → Get details of a specific employee

GET /employees/me/profile → Get logged-in employee's profile (Employee)

POST /employees → Add a new employee (Admin)

PUT /employees/{id} → Update employee details

DELETE /employees/{id} → Delete an employee

3. Departments

GET /departments → Get all departments

POST /departments → Add a new department

PUT /departments/{id} → Update department details

DELETE /departments/{id} → Delete a department

4. Jobs

GET /jobs → Get all jobs

POST /jobs → Add a new job role

PUT /jobs/{id} → Update job details

DELETE /jobs/{id} → Delete a job

5. Payroll

GET /payroll → Get all payroll records (Admin)

GET /payroll/my/{employeeId}/{year}/{month} → Get salary slip for logged-in employee

POST /payroll/generate/{employeeId} → Generate payroll for an employee (Admin)

PUT /payroll/{id} → Update payroll details

DELETE /payroll/{id} → Delete a payroll record

6. Leaves

GET /leaves/pending → Get all pending leave requests (Admin)

GET /leaves/employee/{employeeId} → Get leave history of an employee

POST /leaves/employee/{employeeId} → Apply for leave (Employee)

PATCH /leaves/{leaveId}/approve → Approve leave request (Admin)

PATCH /leaves/{leaveId}/reject → Reject leave request (Admin)

7. Reports

GET /reports/payroll/{year}/{month} → Get payroll report for a specific month/year

GET /reports/departments → Get department-wise salary cost

9. Day-by Day Development Journey

Day 1: Foundation and Database Design

Database Implementation:

- Designed normalized database schema with 6 main tables
- Established proper foreign key relationships and constraints
- Implemented indexes for frequent query optimization

Day 2: Core Business Logic Implementation

Service Layer Development:

- Implemented 6 core services with complete CRUD operations
- Added business validation rules and error handling
- Created comprehensive DTO pattern for data transfer

Day 3: Comprehensive Testing

Testing Strategy:

- Unit tests for all service methods
- Integration tests for REST controllers
- Mockito for mocking dependencies
- Test containers for database testing

Postman Testing:

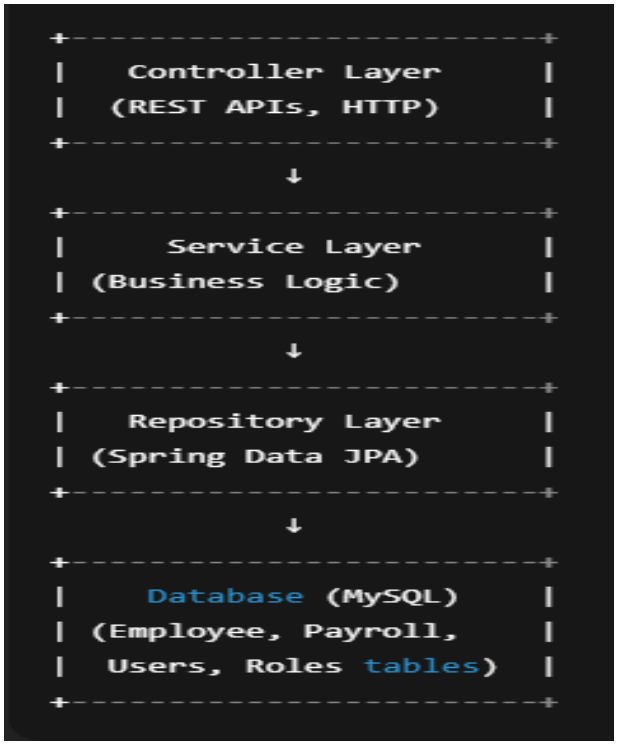
- Created comprehensive collection with 35+ requests
- Tested all authentication scenarios
- Verified error responses and status codes
- Tested edge cases and validation rules
- Used Swagger UI for interactive API testing

Day 4: Frontend Development and Integration

Key Frontend Features:

- JWT token management with automatic refresh
- Role-based routing and component rendering
- Form validation with real-time feedback
- Responsive design with Bootstrap
- Error handling and user notifications

Layer Arcitecture:



Frontend Layer architecture

10. Future Enhancement

Short-term Improvements

- Email Notifications: Automated email alerts for leave approvals and payroll processing
- Advanced Reporting: PDF generation for salary slips and management reports
- Bulk Operations: Mass employee data upload and payroll processing
- Mobile Responsiveness: Enhanced mobile UI/UX optimization

Long-term Features

- Integration APIs: Third-party integrations with banking systems and accounting software
- Advanced Analytics: Predictive analytics for HR planning and cost optimization
- Attendance Integration: Biometric and time tracking system integration
- Performance Management: Employee performance tracking and appraisal workflows

11. Challenges

1. Integration of Frontend and Backend

- Ensuring smooth communication between React (frontend) and Spring Boot (backend).
- Handling CORS issues while connecting frontend API calls with backend services.

2.JWT Authentication & Role-Based Access

- Implementing secure login and role separation (Admin vs. Employee).

3.Time & Resource Management

- Balancing development time across frontend, backend, and database integration.
- Debugging issues in multiple layers (React UI, Spring Boot API, MySQL queries).

12. Conclusion

The Payroll Management System automates and secures salary management with clear separation of roles. It improves efficiency, transparency, and accuracy, and provides scope for future enhancements.

GitHub Link : Payroll_Management_System