Employee Management System (EMS) – Project Documentation

---

1. Project Overview

The **Employee Management System (EMS)** is a full-stack web application designed to manage employees, their attendance, tasks, leaves, and internal communication efficiently. It serves as a centralized system for both **Managers** and **Employees**, improving transparency and productivity.

---

2. Objective

- Provide a unified portal for HR-related tasks
- Enable seamless manager-employee communication
- Digitize and streamline attendance, task, leave, and ticket tracking
- Improve employee engagement with feedback loops and task visibility

---

3. Users & Roles

➤ Employee

- Login/logout securely
- Mark attendance (Check-in, Check-out)
- Submit daily feedback
- View/update task status
- Raise/view leave requests
- View assigned tickets
- Chat with manager

➤ Manager

- Login/logout securely
- Manage employee records (CRUD)
- Manage leaves (approve, decline, filter by status)
- View/respond to daily feedback
- Oversee attendance logs
- Assign and monitor tasks
- Assign and track support tickets
- Chat with employees

---

4. Modules & Features

Authentication & Authorization

- Role-based access (Manager / Employee)
- JWT-based token authentication

Employee Management

- Add, edit, delete employee records
- View all employees with search/filter
- Assign managers/departments

Attendance Management

- Employees check-in and check-out
- Status auto-calculated (present, half-day, etc.)
- Managers view daily/monthly attendance

Leave Management

- Employees submit leave requests
- Managers approve/decline with comments
- Filters by leave status (Pending, Approved, Declined)

Feedback System

- Employees submit daily feedback
- Managers can reply
- Historical feedback stored per employee

Task Management

- Managers assign tasks to employees
- Task attributes: title, description, priority, deadline
- Task statuses: Pending, Working, Completed
- Logs all task status updates with timestamps

Ticket System

- Managers assign tickets (like support/dev issues)
- Employees view and update status
- Statuses: Open, In Progress, Resolved, Closed

Chat System

- Real-time or async message system
- Manager ↔ Employee communication
- Message read-status, timestamps

---

5. Tech Stack

Layer	Tech
Frontend	React.js, React Router
Backend	Spring Boot, Spring Security, JPA, REST
Database	MySQL
Authentication	JWT (Access + Refresh Tokens)
DevOps	GitHub
Deployment	VPS

---

6. Database Design Overview

Main Tables:

- `users`
- `employees`
- `attendance`
- `leaves`
- `feedback`
- `tasks`
- `task_status_logs`
- `tickets`
- `chat_messages`
- `departments` *(optional)*

(Relational schema shared in previous message.)

---

7. API Endpoints (Sample)

`/api/auth/login`

- POST: `{ username, password }`

`/api/attendance/mark`

- POST (Employee): `{ checkIn | checkOut }`

`/api/leaves/request`

- POST (Employee): `{ startDate, endDate, reason }`

`/api/leaves/all`

- GET (Manager): `?status=approved`

`/api/tasks`

- POST (Manager): Assign task
- PUT (Employee): Update task status

(We'll define the full OpenAPI spec if needed.)

---

8. Key Design Highlights

- Clean separation of concerns with **controller → service → repository** in Spring
- Stateless backend with JWT
- Centralized error handling
- React with protected routes, role-based UI rendering
- Form validation, optimistic UI updates for responsiveness

---

9. Optional Future Enhancements

- Admin role (for analytics, payroll, org-wide settings)
- Integration with biometric device for attendance
- Email notifications for leaves/tickets
- Performance analytics for employees
- Push notifications / WebSockets for live chat
