Web-Based Employee Leave Management System

Objective:

To develop an intuitive web application for employees to apply for leaves and for managers to approve or reject them. The system will maintain accurate records of leave balances, leave history, and provide notifications and reports.

Tools &Technologies:

- Frontend: HTML/CSS, JavaScript (React.js or Vue.js)
- Backend: Node.js (Express) or Python (Django/Flask)
- Database: MySQL or MongoDB
- Authentication: JWT or OAuth2
- Hosting: Heroku, Vercel, or AWS

Requirements and Features

User Roles:

- Employee: Apply for leaves, view balance and history
- Manager: Approve/reject leave requests, view team leave calendar
- Admin (Optional): Set leave policies, manage users

Core Features:

- Leave application form with date, type, reason

- Leave approval workflow with status tracking

- Email/SMS notifications on request submission and approval

- Real-time leave balance updates

- Dashboard for leave statistics and history

- Secure login and role-based access

System Design and Architecture

System Components:

1. Frontend Interface: User-friendly forms and dashboards

2. REST API Backend: Handles leave logic, authentication, role-based access

3. Database Layer: Stores user data, leave applications, balance, logs

4. Notification Module: Email/SMS notifications using APIs like Twilio or SendGrid

Architecture Diagram:

Employee/Manager -> Frontend UI -> REST API -> Database -> Notification System

Workflow Example:

Employee submits leave -> Manager reviews request -> Decision sent ->

System updates record and balance.

UI Mockups & User Flow

Employee Dashboard:

- Apply Leave Button

- Leave Balance Overview

- Leave History Table

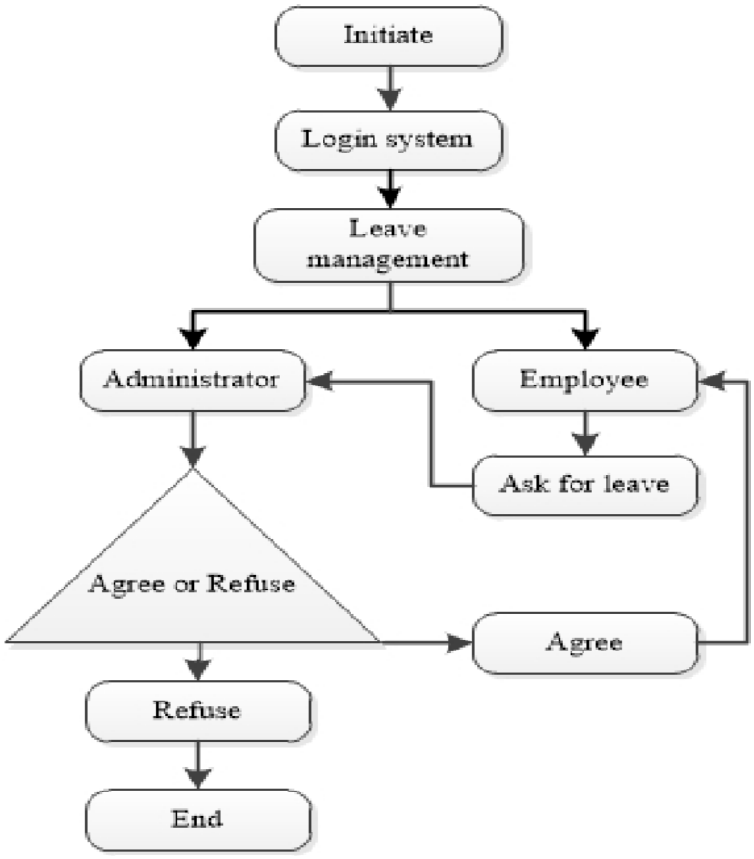
Manager Dashboard:

- Pending Requests Table

- Approve/Reject Buttons

- Team Calendar

User Flow Diagram



Implementation Plan

Phase 1: Planning and Design

- Define leave types, rules, and policies
- Design database schema and API endpoints

Phase 2: Development

- Build frontend and backend modules
- Set up authentication and role access
- Implement leave application and approval logic

Phase 3: Testing

- Unit tests for backend logic
- UI/UX testing with test users
- Fix bugs and refine workflows

Phase 4: Deployment

- Deploy to cloud
- Set up monitoring and analytics
- Train HR/Admin staff for use

Future Scope and Conclusion

Future Enhancements:

- Mobile app integration
- Calendar integration (Google/Outlook)
- AI-based leave pattern analysis
- Slack/MS Teams Chatbot extension

Conclusion:

This Leave Management System streamlines leave application and approval processes, improves record-keeping, and ensures transparency between employees and management. It reduces manual work and is scalable for organizations of all sizes.