Leave Management System – HLD

**1. High-Level Design (HLD) — Leave Management System**

**1.1 Purpose**

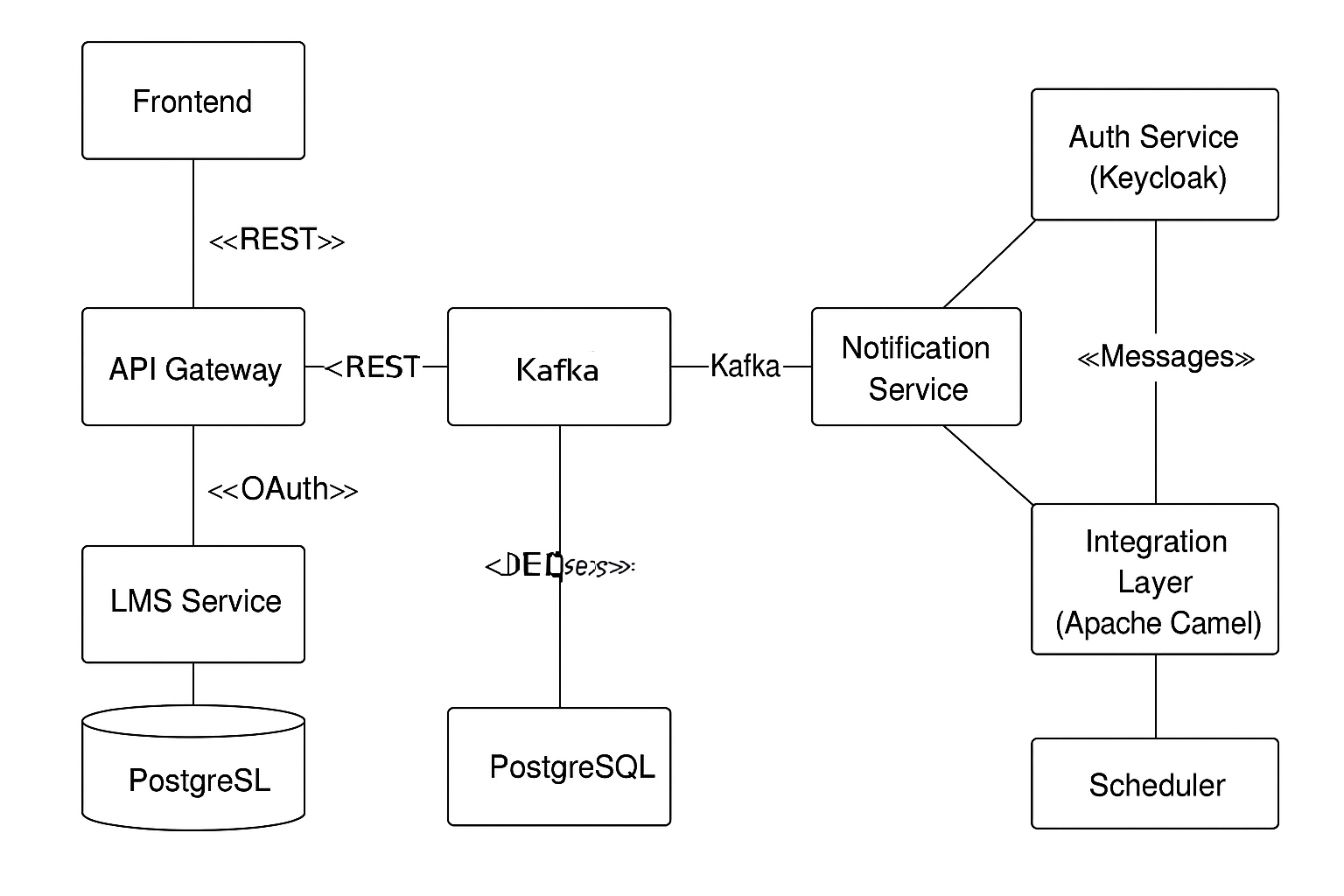
The Leave Management System automates the process of leave requests, approvals, balance tracking, and notifications. It integrates with external systems like authentication providers (Keycloak), payroll, and notification services, and supports multi-role workflows for employees, managers, and HR.

**1.2 System Architecture**

**Core Components**

1. **Frontend UI** (Web & Mobile)
   * React/Angular/Vue frontend (out of scope for backend HLD, but interfaces via REST/GraphQL)
2. **API Gateway**
   * Routes client requests to backend microservices
   * Handles authentication & authorization via Keycloak
3. **LMS Service (Spring Boot)**
   * REST APIs for leave requests, approvals, balances, and reports
   * Business logic for accrual, carryover, and balance validations
   * Kafka producer to publish leave events
4. **Notification Service**
   * Consumes leave events from Kafka
   * Sends notifications via Email, SMS, WhatsApp
5. **Auth Service (Keycloak)**
   * OAuth2/OpenID Connect for authentication
   * RBAC (roles: Employee, Manager, HR, Admin)
6. **Database (PostgreSQL)**
   * Stores employee details, leave types, requests, balances, audit logs
7. **Integration Layer (Apache Camel)**
   * Routes events to notification channels
   * Handles payroll/calendar sync jobs
8. **Scheduler**
   * Runs periodic jobs for accruals and carryover

**1.3 High-Level Component Diagram**



**1.4 Data Flow**

1. **Employee requests leave**
   * API receives request
   * Validates balance and leave type
   * Stores in leave\_request table
   * Publishes leave.requested event to Kafka
2. **Manager approves/rejects**
   * API updates status and balance
   * Publishes leave.approved or leave.rejected event
3. **Notification Service**
   * Consumes events, sends messages
4. **Scheduler**
   * Accrues balances monthly/yearly
5. **Integration Layer**
   * Exports approved leaves to payroll

**1.5 Technology Stack**

* **Backend**: Spring Boot, Java 21
* **Database**: PostgreSQL
* **Messaging**: Kafka
* **Integration**: Apache Camel
* **Authentication**: Keycloak
* **Containerization**: Docker & Docker Compose
* **Deployment**: Kubernetes/VMs (optional)

**1.6 Non-Functional Requirements**

* **Scalability**: Horizontal scaling for API and Kafka consumers
* **Security**: OAuth2, HTTPS, RBAC
* **Performance**: ≤ 200ms API response for standard requests
* **Reliability**: Kafka-based async communication for resilience
* **Auditability**: All changes logged in audit\_log