MedShare Hub

Healthcare Data Exchange Platform

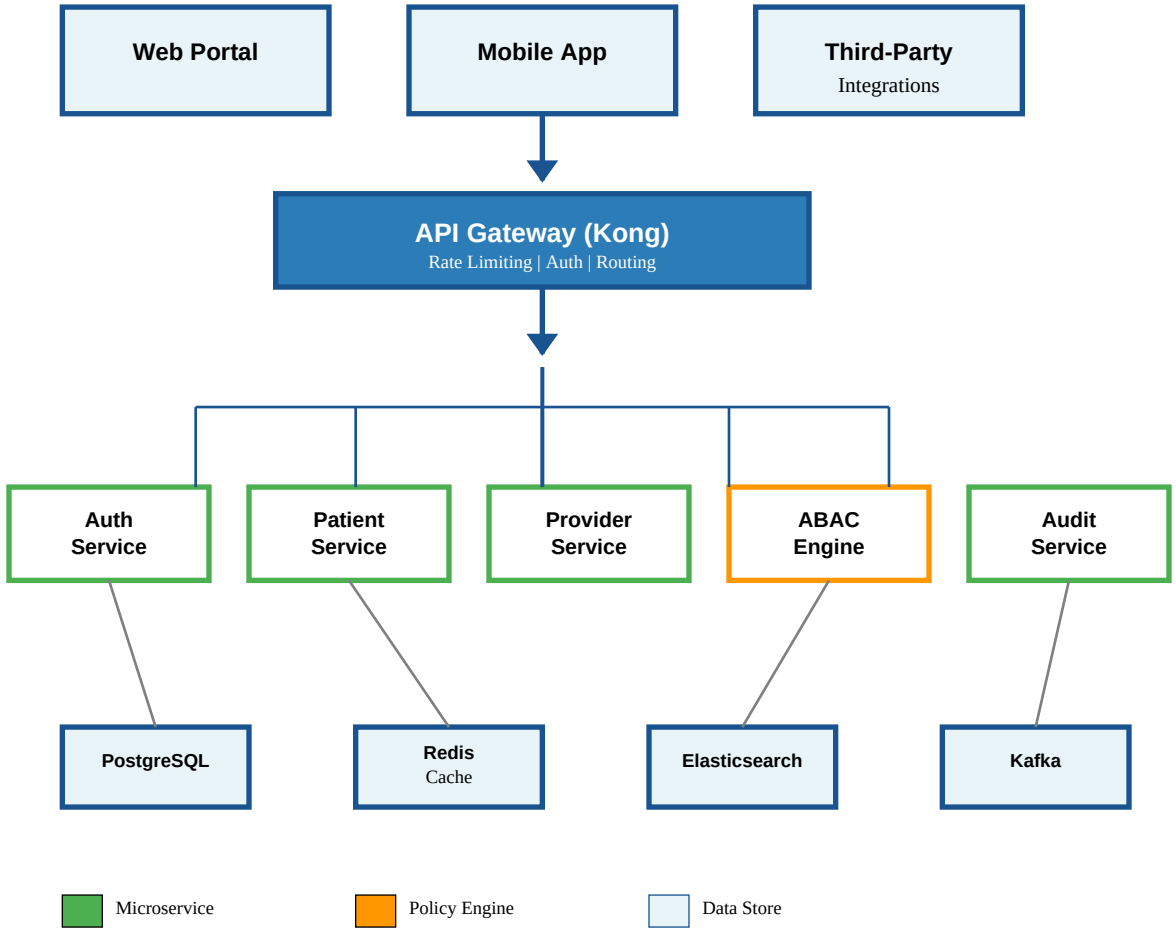
System Design Document

Version: 1.0
Date: January 30, 2026
Status: Technical Design
Architecture: Microservices with ABAC

1. System Architecture Overview

MedShare Hub employs a microservices architecture with centralized policy management through Open Policy Agent (OPA). The system is designed for high availability, scalability, and security, with each component serving a specific purpose in the data exchange ecosystem.

High-Level System Architecture



1.1 Architecture Principles

• **Separation of Concerns:** Authorization logic is decoupled from business logic through OPA

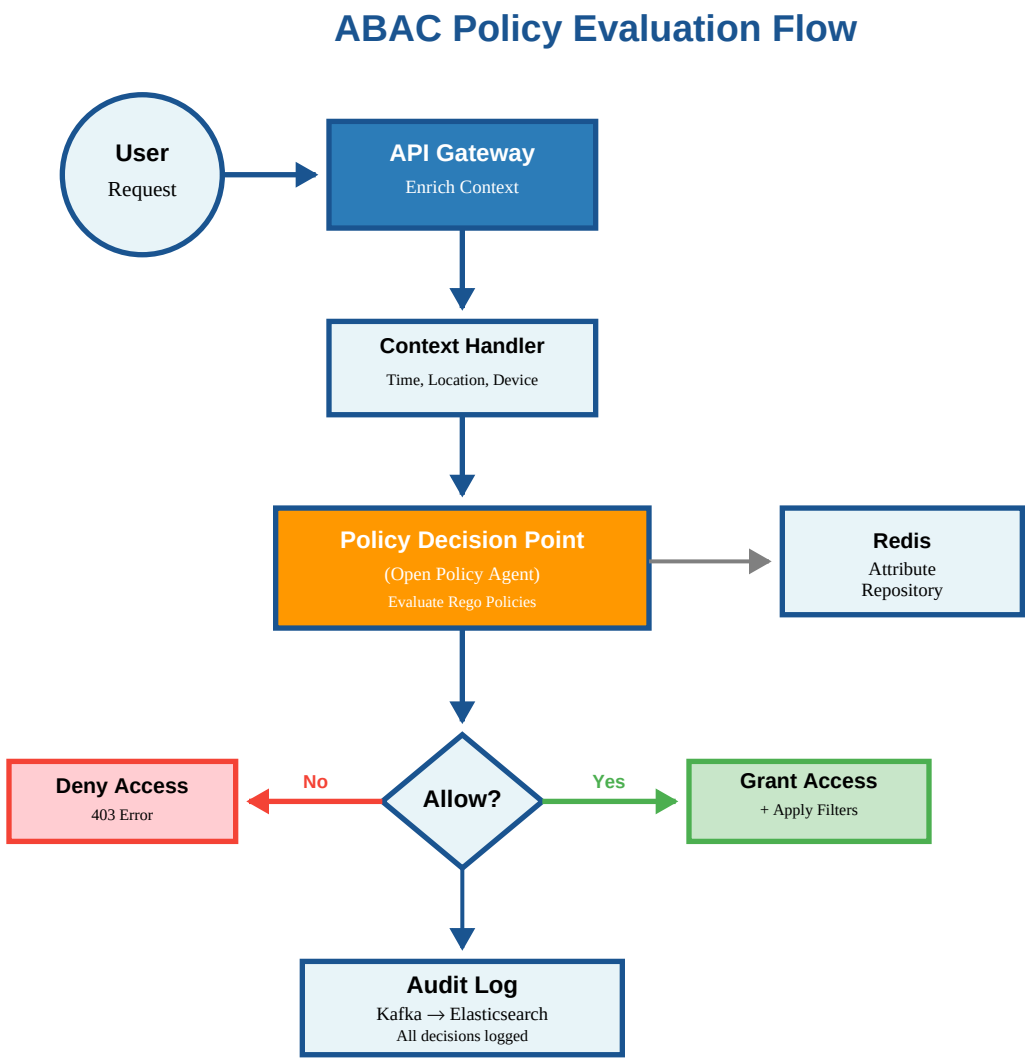
• **Defense in Depth:** Multiple security layers including API gateway, service-level auth, and database row-level security

• **Scalability:** Stateless services enable horizontal scaling to handle increasing load

- **Resilience:** Circuit breakers and fallback mechanisms ensure graceful degradation
- **Observability:** Comprehensive logging, monitoring, and distributed tracing

2. ABAC Policy Evaluation Flow

Every access request flows through a standardized policy evaluation process. This ensures consistent, auditable, and context-aware authorization decisions across the entire platform.



2.1 Request Flow Steps

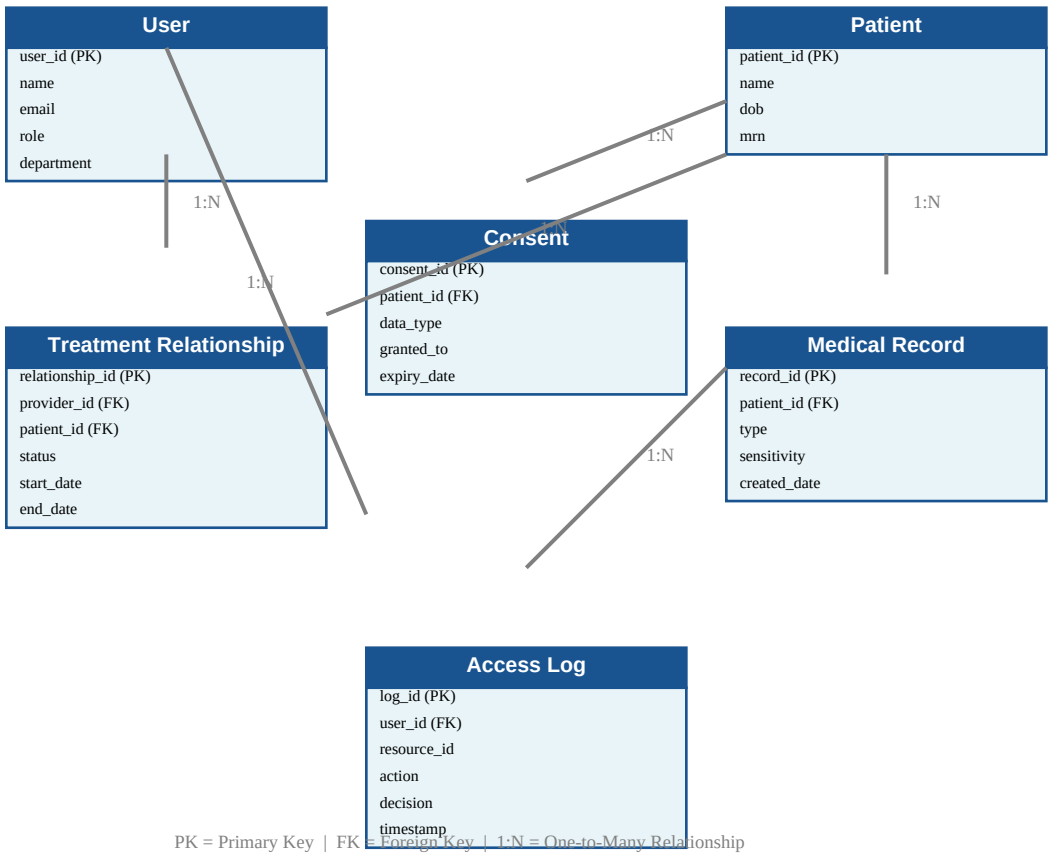
Step	Component	Action	Latency
1	User	Initiates API request with JWT token	-

2	API Gateway	Validates token, extracts user claims	< 10ms
3	Context Handler	Enriches request with time, location, device info	< 20ms
4	OPA (PDP)	Evaluates policies against attributes	< 50ms
5	Redis	Retrieves cached user/resource attributes	< 5ms
6	Decision	Returns allow/deny + obligations	< 10ms
7	Service	Processes request or returns 403 error	< 200ms
8	Audit Log	Writes event to Kafka for compliance	async

3. Data Model & Relationships

The platform's data model is designed to support complex healthcare relationships while maintaining referential integrity and audit trails. All entities are linked to enable comprehensive policy evaluation.

Entity Relationship Diagram



3.1 Key Entities

User: Healthcare providers, administrators, insurance personnel, and patients. Each user has roles, department affiliations, and certifications stored as attributes.

Patient: Individuals whose medical records are managed in the system. Linked to medical records, treatment relationships, and consent preferences.

Medical Record: Documents, lab results, imaging, prescriptions, and clinical notes. Each record has a sensitivity classification that affects access control.

Treatment Relationship: Temporal relationship between providers and patients during active care. Includes start/end dates and relationship type (primary, consulting, emergency).

Consent: Patient-defined rules for data sharing. Specifies what data types can be shared, with whom, and for how long.

Access Log: Immutable audit trail of every access attempt, including user, resource, action, decision, and contextual information.

4. Core Components

API Gateway (Kong)

Purpose: Single entry point for all client requests

Key Responsibilities:

• JWT token validation and refresh

• Rate limiting per user/organization (1000 req/min default)

• Request routing to appropriate microservices

• TLS termination and certificate management

• Request/response transformation

Technology Stack: Kong Gateway with OPA plugin

Authentication Service

Purpose: User authentication and identity management

Key Responsibilities:

• SSO integration (SAML, OAuth2, OIDC)

• Multi-factor authentication (SMS, TOTP, biometric)

• Session management and token issuance

• Password policy enforcement

• Account lockout and breach detection

Technology Stack: Keycloak with PostgreSQL backend

ABAC Policy Engine (OPA)

Purpose: Centralized policy decision point

Key Responsibilities:

• Policy evaluation using Rego language

• Attribute aggregation from multiple sources

• Policy caching and optimization

• Break-glass emergency override logic

• Policy versioning and A/B testing

Technology Stack: Open Policy Agent with REST API

Patient Service

Purpose: Patient data and consent management

Key Responsibilities:

• FHIR R4 patient resource management

• Consent preference storage and retrieval

• Patient portal API endpoints

• Demographic information updates

• Patient matching and de-duplication

Technology Stack: Java Spring Boot with PostgreSQL

Audit Service

Purpose: Comprehensive access logging and compliance

Key Responsibilities:

• Real-time access event logging

• Anomaly detection using ML models

• Compliance report generation

• SIEM integration for security monitoring

• Retention policy enforcement (7 years)

Technology Stack: Kafka + Elasticsearch + Kibana

5. Technology Stack

Layer	Technology	Purpose	Version
Frontend	React + TypeScript	Web application UI	18.x
Frontend	Material-UI	Component library	5.x
API Gateway	Kong	API management	3.x
Backend	Java Spring Boot	Core services	3.x
Policy Engine	Open Policy Agent	ABAC decisions	0.50+
Database	PostgreSQL	Primary data store	15+
Cache	Redis	Attribute caching	7.x
Search/Analytics	Elasticsearch	Audit logs & search	8.x
Message Queue	Apache Kafka	Event streaming	3.x
Identity	Keycloak	Authentication & SSO	21.x
Container	Docker	Application packaging	24.x
Orchestration	Kubernetes	Container orchestration	1.28+
Monitoring	Prometheus + Grafana	Metrics & dashboards	Latest
Tracing	Jaeger	Distributed tracing	Latest

6. Sample ABAC Policy (Rego)

Below is a simplified example of a Rego policy that controls access to medical records based on treatment relationships and data sensitivity.

```
package healthcare.access

# Default deny all access
default allow = false

# Allow physicians to read medical records of their patients
allow {
input.action == "read"
input.resource.type == "medical_record"
input.subject.role == "physician"

# Check active treatment relationship
relationship := data.relationships[input.subject.id][input.resource.patient_id]
relationship.type == "treating_physician"
relationship.status == "active"
```

```
# Verify relationship hasn't expired
time.now_ns() < time.parse_rfc3339_ns(relationship.end_date)

# Department-specific restrictions
not deny_by_sensitivity
}

# Deny access to psychiatric records unless in psychiatry dept
deny_by_sensitivity {
input.resource.sensitivity == "psychiatric"
input.subject.department != "psychiatry"
not input.context.emergency
}

# Emergency override with obligations
allow {
input.context.emergency == true
input.subject.emergency_certified == true
} {
# Set obligations for enhanced logging
output.obligations := ["enhanced_audit", "supervisor_notification"]
}

# Patients always have access to their own records
allow {
input.action == "read"
input.subject.type == "patient"
input.subject.id == input.resource.patient_id
}
```

7. Deployment Architecture

Cloud Provider: AWS (primary) with multi-region deployment for disaster recovery

Kubernetes Clusters: Production, staging, and development environments

Database: Amazon RDS PostgreSQL with Multi-AZ deployment and automated backups

Cache Layer: Amazon ElastiCache for Redis with cluster mode enabled

Message Queue: Amazon MSK (Managed Kafka) with 3+ broker cluster

Load Balancing: Application Load Balancer with SSL termination

CDN: CloudFront for static asset delivery and DDoS protection

Secrets Management: AWS Secrets Manager integrated with Kubernetes

7.1 Scalability Strategy

• Horizontal pod autoscaling based on CPU/memory and custom metrics (requests/second)

• Database read replicas for reporting and analytics queries

• Redis cluster sharding for distributed attribute caching

• Kafka partitioning by organization ID for parallel processing

• Edge caching of policies using OPA's bundle distribution

• CDN caching of static resources and public-facing content

8. Security & Compliance

Control	Implementation	Standard
Data Encryption at Rest	AES-256 encryption for all databases and file storage	HIPAA, GDPR
Data Encryption in Transit	TLS 1.3 with perfect forward secrecy	HIPAA
Authentication	Multi-factor authentication required for all users	NIST 800-63B
Authorization	ABAC with fine-grained policies and context evaluation	NIST ABAC
Audit Logging	Immutable audit trails with 7-year retention	HIPAA §164.312
Network Security	VPC isolation, security groups, WAF protection	AWS Well-Architected
Data Backup	Automated daily backups with 30-day retention	HIPAA
Incident Response	24/7 SOC monitoring and automated alerting	NIST CSF
Vulnerability Management	Weekly security scans and quarterly penetration testing	OWASP Top 10
Access Control	Least privilege principle with regular access reviews	HIPAA

9. Summary

The MedShare Hub system design prioritizes security, scalability, and compliance while maintaining high performance. The microservices architecture enables independent scaling and deployment of components, while the centralized ABAC policy engine ensures consistent, auditable access control across the platform.

By leveraging industry-standard technologies like Open Policy Agent, Kafka, and Kubernetes, the platform is built on proven foundations while remaining flexible enough to adapt to future healthcare data exchange requirements and regulatory changes.