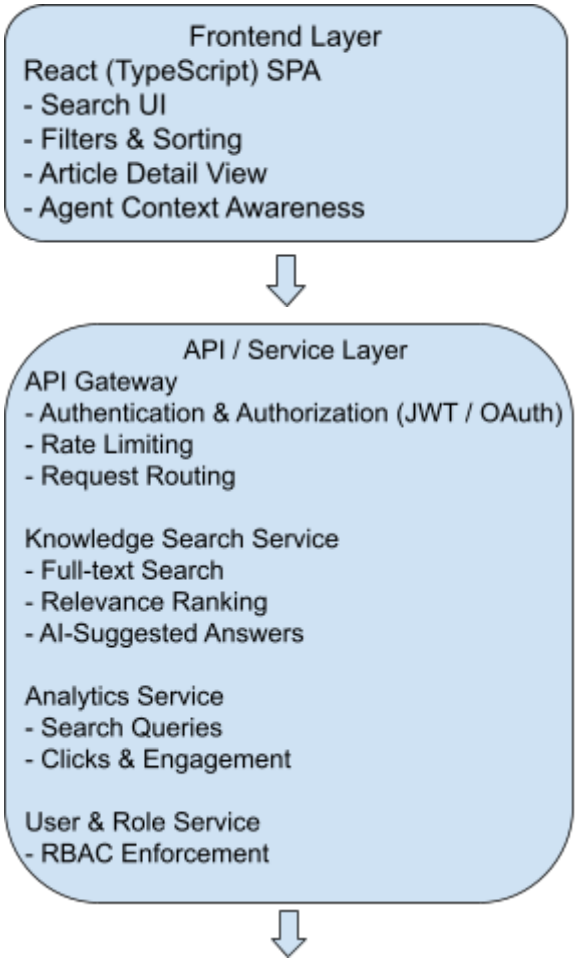
# AI-Powered Knowledge Search Platform

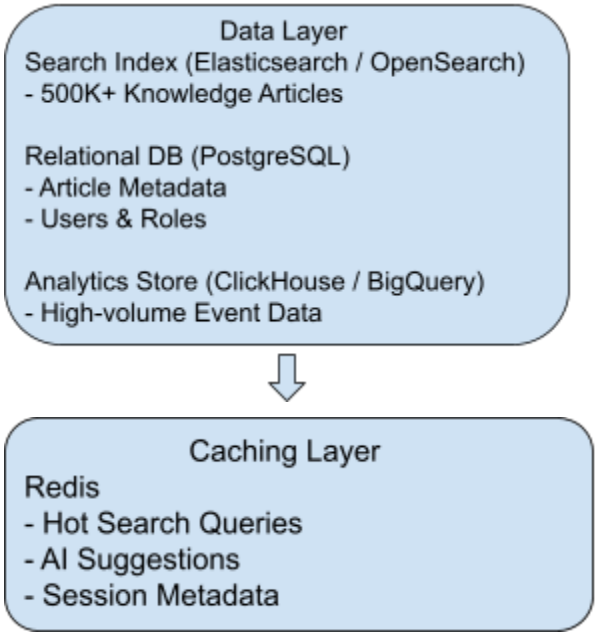
## Comprehensive System Design

---

## 1. High-Level Architecture

### 1.1 High-Level Architecture Diagram (Logical View)






---

## 1.2 Frontend Layer (Technology Choices)

- **React + TypeScript**
  - Component-based architecture
  - Strong type safety for large teams

- **SPA served via CDN**
  - Low latency across multiple time zones

- **Responsive Design**
  - Optimized for desktop and mobile agents

- **Client-side Optimizations**
  - Debounced search
  - Memoization of result rendering

---

## 1.3 API / Service Layer

- **API Gateway**
  - Central entry point
  - Authentication, rate limiting, and request validation

- **Knowledge Search Service**

    - Executes full-text and semantic search
    - Calls AI model for suggested answers

- **Analytics Service**
    - Asynchronous ingestion of search and interaction events

- **User & Role Service**
    - Enforces role-based access control (RBAC)

---

## 1.4 Data Layer

- **Search Index (Elasticsearch / OpenSearch)**
    - Stores searchable article content
    - Supports relevance scoring and filtering

- **Relational Database (PostgreSQL)**
    - Stores article metadata, user accounts, roles

- **Analytics Store**
    - Optimized for high-volume, write-heavy workloads

---

## 1.5 Caching Strategy

- **Redis**
    - Cache frequent search queries
    - Cache AI-generated answer suggestions
    - Reduce load on search index

- **Client-side Cache**
    - Short-lived search result caching for improved UX

---

## 1.6 Security Considerations

- OAuth 2.0 / JWT authentication
- Role-based access control (Agent, Admin, Manager)
- API rate limiting per user
- Encryption in transit (TLS)
- Audit logging for sensitive actions

# 2. Design Details

## 2.1 Technology Stack Choices & Rationale

| Layer | Technology | Rationale |
|---|---|---|
| Frontend | React + TypeScript | Scalable UI, maintainable codebase |
| API | REST + API Gateway | Clear contracts, easy integration |
| Search | Elasticsearch | Proven large-scale search performance |
| Cache | Redis | Low-latency, high-throughput |
| Analytics | ClickHouse / BigQuery | Optimized for large event volumes |

## 2.2 Scalability Approach

- Stateless services scale horizontally
- Search index sharded by tenant or category
- Analytics ingestion is asynchronous
- CDN used for global frontend delivery
- Cache reduces repetitive expensive queries

## 2.3 Performance Optimization Strategies

- Debounced search requests
- Partial document retrieval for previews
- Lazy loading of article details
- Cached AI suggestions
- Optimized relevance scoring pipeline

## 2.4 Security Implementation

- JWT tokens validated at API Gateway
- RBAC enforced at service layer
- Rate limiting prevents abuse

- Secure secrets management
- Audit trails for compliance

---

### 2.5 API Design Principles

- RESTful, resource-oriented endpoints
- Versioned APIs (`/v1`)
- Pagination, filtering, sorting via query parameters
- Idempotent write operations
- Consistent error responses

---

### 2.6 Monitoring & Observability

- Metrics: request latency, error rate, search success
- Distributed tracing across services
- Centralized logging
- Alerts for SLA breaches
- Analytics dashboards for product insights

---

## 3. Trade-Offs Analysis

### Trade-Off 1: Elasticsearch vs Relational Search

**Decision:** Use Elasticsearch for search
**Pros:**

- Fast full-text and semantic search
- Advanced relevance scoring

**Cons:**

- Operational complexity
- Additional infrastructure cost

---

### Trade-Off 2: Client-Side vs Server-Side Filtering

**Decision:** Combine both
**Pros:**

- Faster UI for small result sets
- Reduced server calls

**Cons:**

- Client filtering not viable for very large datasets

---

### Trade-Off 3: REST vs GraphQL APIs

**Decision:** REST
**Pros:**

- Simpler caching
- Easier observability
- Clear ownership boundaries

**Cons:**

- Possible over-fetching
- Less flexible for complex queries

---

# Summary

This design supports:

- **10,000+ concurrent users**
- **500,000+ knowledge articles**
- **AI-powered answer suggestions**
- **Enterprise-grade security**
- **Scalable, observable, and maintainable design**

It balances performance, scalability, and operational complexity while remaining extensible for future AI and analytics capabilities.