**Service-by-Service Breakdown**

**1. API Gateway (Port 8000)**

**Role**: Central entry point and traffic orchestrator**Key Features**:

* **Intelligent Routing**: Routes requests to appropriate microservices
* **JWT Authentication**: Token validation and user authorization
* **Rate Limiting**: Multi-level protection (global, per-user, per-endpoint)
* **Load Balancing**: Round-robin, least connections, fastest response
* **Circuit Breaker**: Prevents cascade failures
* **Request/Response Transformation**: Protocol adaptation
* **CORS Management**: Cross-origin request handling
* **Metrics Collection**: Request/response analytics

**Technologies**: Express.js, Redis, Circuit Breaker Pattern

**2. Authentication Service (Port 3001)**

**Role**: Identity and access management**Core Capabilities**:

* **Multi-factor Authentication (MFA)**: TOTP, SMS, Email verification
* **OAuth 2.0 Integration**: Google, Microsoft SSO
* **Session Management**: Redis-backed distributed sessions
* **Password Policies**: Complexity requirements, rotation
* **Role-Based Access Control (RBAC)**: 5 user roles with granular permissions
* **Account Security**: Lockout policies, suspicious activity detection
* **JWT Management**: Access/refresh token lifecycle

**Advanced Features**:

* Biometric authentication support
* Device fingerprinting
* Audit logging
* Password breach detection

**Database Schema**: hrms\_auth - Users, roles, sessions, permissions

**3. Employee Service (Port 3002)**

**Role**: Employee lifecycle and organizational management**Modules**:

* **Employee Profile Management**: Personal info, contact details, emergency contacts
* **Department Management**: Hierarchical org structure
* **Position Management**: Job roles, salary bands, responsibilities
* **Document Attachments**: Profile photos, ID documents, contracts
* **Audit Trail**: All changes tracked with timestamps and user attribution

**Key Features**:

* **CSV Import/Export**: Bulk employee operations
* **Photo Management**: Profile picture processing with Sharp.js
* **Organizational Hierarchy**: Manager-subordinate relationships
* **Employee Directory**: Searchable company directory
* **Analytics Dashboard**: Employee metrics and insights

**Event Publishing**: Publishes employee lifecycle events via Kafka**Database Schema**: hrms\_employee - Employees, departments, positions, audit logs

**4. Performance Service (Port 3004)**

**Role**: Performance management and goal tracking**Advanced Features**:

* **Performance Reviews**: 360-degree feedback, peer reviews
* **Goal Management**: OKRs, SMART goals, cascading objectives
* **Competency Assessments**: Skill-based evaluations
* **Development Plans**: Career growth roadmaps
* **Performance Metrics**: KPI tracking and analytics
* **Calibration Sessions**: Manager alignment processes

**Enterprise Capabilities**:

* **Distributed Tracing**: Jaeger integration
* **Metrics Collection**: Prometheus metrics
* **Secrets Management**: HashiCorp Vault integration
* **API Versioning**: v1/v2 support
* **Circuit Breaker**: Resilience patterns

**Database Schema**: hrms\_performance - Reviews, goals, metrics, assessments

**5. Learning & Development Service (Port 3005)**

**Role**: Training and skill development platform**Comprehensive Features**:

* **Course Management**: Video courses, assessments, certifications
* **Content Processing**: Video transcoding, thumbnail generation
* **Progress Tracking**: Completion rates, time spent, learning paths
* **Skill Tracking**: Competency mapping and gap analysis
* **Certificate Generation**: PDF certificates with verification
* **Social Learning**: Discussions, peer recommendations
* **Gamification**: Points, badges, leaderboards

**Integrations**:

* **External Video Platforms**: YouTube, Vimeo integration
* **Content Storage**: AWS S3/MinIO for video files
* **Analytics**: Learning effectiveness metrics
* **Recommendations**: AI-powered course suggestions

**Database Schema**: hrms\_learning - Courses, progress, skills, certificates

**6. Time & Attendance Service (Port 3003)**

**Role**: Time tracking and attendance management**Core Functions**:

* **Time Tracking**: Clock in/out, break tracking
* **Attendance Monitoring**: Late arrivals, early departures
* **Leave Management**: PTO requests, approval workflows
* **Overtime Tracking**: Automatic calculation and alerts
* **Shift Management**: Schedule planning and assignments
* **GPS Tracking**: Location-based attendance (optional)
* **Biometric Integration**: Fingerprint/face recognition support

**Advanced Analytics**:

* **Attendance Patterns**: Trend analysis
* **Productivity Metrics**: Time utilization reports
* **Compliance Reporting**: Labor law adherence
* **Real-time Notifications**: Late check-in alerts

**Database Schema**: hrms\_attendance - Time entries, leaves, shifts, schedules

**7. Recruitment Service (Port 3006)**

**Role**: Complete Applicant Tracking System (ATS)**Full-Featured ATS**:

* **Job Posting Management**: Multi-channel job distribution
* **Candidate Database**: Resume parsing, profile management
* **Application Tracking**: Status pipeline management
* **Interview Scheduling**: Calendar integration, automated reminders
* **Candidate Evaluation**: Scoring, feedback, comparisons
* **Resume Parsing**: AI-powered skills extraction
* **Skill Matching**: Job-candidate compatibility scoring

**Workflow Features**:

* **Onboarding Tasks**: New hire checklists
* **Reference Checks**: Automated verification workflows
* **Duplicate Detection**: Candidate deduplication
* **Analytics**: Hiring metrics, time-to-fill, source effectiveness

**Database Schema**: hrms\_recruitment - Jobs, candidates, applications, interviews

**8. Document Service (Port 3007)**

**Role**: Enterprise document management**Document Lifecycle**:

* **File Upload/Storage**: Multiple storage backends (local, S3, MinIO)
* **Version Control**: Document versioning and history
* **Access Control**: Permission-based document sharing
* **Content Processing**: Text extraction, thumbnail generation
* **Virus Scanning**: Security scanning (configurable)
* **Search & Indexing**: Elasticsearch integration
* **Workflow Management**: Approval processes

**Compliance Features**:

* **Retention Policies**: Automated document lifecycle
* **Audit Logging**: All document activities tracked
* **Encryption**: At-rest and in-transit encryption
* **Document Templates**: Standardized document creation

**Database Schema**: hrms\_document - File metadata, permissions, workflows

**9. Notification Service (Port 3008)**

**Role**: Multi-channel communication hub**Communication Channels**:

* **Email Notifications**: SMTP integration with templates
* **SMS Messaging**: Twilio/other SMS providers
* **Push Notifications**: Mobile app notifications
* **In-App Notifications**: Real-time browser notifications

**Advanced Features**:

* **Template Engine**: Dynamic content generation
* **User Preferences**: Granular notification controls
* **Queue Management**: Redis/Bull queue for reliable delivery
* **Delivery Tracking**: Read receipts, bounce handling
* **Bulk Operations**: Mass communication campaigns

**Event-Driven**: Listens to Kafka events from all services for automated notifications**Database Schema**: hrms\_notification - Templates, preferences, delivery logs

**🗄️ Data Architecture**

**Database-per-Service Pattern**

Each microservice owns its data with dedicated PostgreSQL schemas:

sql

hrms\_auth         - Authentication & user management

hrms\_employee     - Employee profiles & org structure

hrms\_performance  - Reviews, goals & assessments

hrms\_learning     - Courses, progress & certifications

hrms\_recruitment  - Jobs, candidates & applications

hrms\_notification - Templates, preferences & logs

hrms\_document     - File metadata & permissions

hrms\_attendance   - Time tracking & leave records

**Data Consistency Strategy**

* **Eventual Consistency**: Cross-service data sync via Kafka events
* **Saga Pattern**: Distributed transaction management
* **CQRS Implementation**: Separate read/write models where appropriate

**🌐 Frontend Architecture (React)**

**Modern React Stack:**

* **React 18+** with TypeScript
* **Vite** for blazing-fast development
* **Tailwind CSS** for modern styling
* **React Query** for state management and caching
* **React Hook Form** with Zod validation
* **React Router** for navigation

**Key Components:**

**typescript**

/src

├── components/          # Reusable UI components

│   ├── auth/           # Authentication forms

│   ├── layout/         # Layout components (Sidebar, Header)

│   └── ui/             # Generic UI components

├── pages/              # Route-specific pages

│   ├── employees/      # Employee management

│   ├── departments/    # Department management

│   └── auth/           # Authentication pages

├── contexts/           # React contexts

├── store/              # Zustand state management

├── lib/                # Utility functions and API clients

└── types/              # TypeScript type definitions

**Role-Based Navigation:**

**typescript**

*// Sidebar menu with role-based access control*

const menuItems = [

  { title: 'Dashboard', roles: ['ALL'] },

  { title: 'Employees', roles: ['SUPER\_ADMIN', 'HR\_MANAGER', 'HR\_SPECIALIST'] },

  { title: 'Performance', roles: ['SUPER\_ADMIN', 'HR\_MANAGER', 'EMPLOYEE'] },

*// ... more items with granular permissions*

]

**🐳 Infrastructure & DevOps**

**Docker Composition**

**yaml**

*# Complete stack with 18+ containers:*

services:

*# Infrastructure (4 containers)*

  - postgres          *# Multi-database PostgreSQL*

  - redis             *# Session & cache store*

  - kafka + zookeeper *# Event streaming*

*# Monitoring (3 containers)*

  - prometheus        *# Metrics collection*

  - grafana          *# Dashboards & visualization*

  - jaeger           *# Distributed tracing*

*# Applications (9 containers)*

  - api-gateway      *# Central routing*

  - auth-service     *# Authentication*

  - employee-service *# Employee management*

  - performance-service

  - learning-service

  - time-attendance-service

  - recruitment-service

  - document-service

  - notification-service

*# Frontend (1 container)*

  - frontend         *# React app with Nginx*

**Kubernetes Deployment**

* **Production-ready** K8s manifests in backend/k8s/
* **Horizontal Pod Autoscaling** configured
* **Health checks** and **readiness probes**
* **ConfigMaps** and **Secrets** management
* **Resource limits** and **requests** defined

**Terraform Infrastructure**

**hcl**

# AWS infrastructure as code:

- VPC and networking setup

- EKS cluster configuration

- RDS PostgreSQL instances

- ElastiCache Redis clusters

- S3 buckets for file storage

- CloudWatch monitoring

- IAM roles and policies

**📊 Observability & Monitoring**

**Three Pillars Implementation:**

**1. Metrics (Prometheus + Grafana)**

* **Application Metrics**: Response times, error rates, throughput
* **Business Metrics**: Active users, document uploads, course completions
* **Infrastructure Metrics**: CPU, memory, disk, network
* **Custom Dashboards**: Service-specific performance views

**2. Logging (Structured JSON)**

* **Centralized Logging**: Winston with JSON format
* **Correlation IDs**: Request tracing across services
* **Audit Logs**: Security and compliance tracking
* **Log Levels**: Debug, info, warn, error with proper filtering

**3. Distributed Tracing (Jaeger)**

* **Request Flow Visualization**: Cross-service request paths
* **Performance Bottlenecks**: Slow query identification
* **Error Propagation**: Failure point detection
* **Dependency Mapping**: Service interaction analysis

**🔄 Event-Driven Architecture**

**Kafka Event Streaming**

**typescript**

*// Event types published across the system:*

EMPLOYEE\_EVENTS: {

  EMPLOYEE\_CREATED,

  EMPLOYEE\_UPDATED,

  EMPLOYEE\_TERMINATED

}

PERFORMANCE\_EVENTS: {

  REVIEW\_COMPLETED,

  GOAL\_ASSIGNED,

  ASSESSMENT\_FINISHED

}

LEARNING\_EVENTS: {

  COURSE\_COMPLETED,

  CERTIFICATION\_EARNED,

  SKILL\_ACQUIRED

}

ATTENDANCE\_EVENTS: {

  CLOCK\_IN,

  CLOCK\_OUT,

  LEAVE\_APPROVED

}

**Asynchronous Processing**

* **Notification Triggers**: Cross-service event notifications
* **Data Synchronization**: Eventual consistency maintenance
* **Audit Trail**: Comprehensive activity logging
* **Analytics**: Real-time metrics calculation

**🔐 Security Architecture**

**Multi-Layer Security:**

**1. Network Security**

* **API Gateway**: Single entry point with rate limiting
* **CORS Configuration**: Strict origin policies
* **HTTPS/TLS**: End-to-end encryption

**2. Authentication & Authorization**

* **JWT Tokens**: Stateless authentication
* **Role-Based Access Control**: 5-tier permission system
* **Session Management**: Redis-backed sessions
* **MFA Support**: TOTP, SMS, email verification

**3. Data Protection**

* **Password Hashing**: bcrypt with salt rounds
* **Sensitive Data Encryption**: At-rest encryption
* **PII Handling**: GDPR-compliant data processing
* **Audit Logging**: All security events tracked

**4. API Security**

* **Input Validation**: Joi schema validation
* **SQL Injection Prevention**: Prisma ORM protection
* **XSS Protection**: Output sanitization
* **CSRF Protection**: Token-based protection

**🎯 Scalability & Performance**

**Horizontal Scaling**

* **Stateless Services**: Easy horizontal scaling
* **Load Balancing**: Multiple scaling strategies
* **Database Sharding**: Schema-per-service isolation
* **Caching Strategy**: Redis caching at multiple levels

**Performance Optimizations**

* **Connection Pooling**: Efficient database connections
* **Response Compression**: Gzip compression
* **CDN Integration**: Static asset delivery
* **Lazy Loading**: Frontend component optimization

**🚀 Deployment Strategies**

**Environment Management**

**shellscript**

*# Multi-environment support:*

- Development    *# Local Docker Compose*

- Staging        *# Kubernetes staging cluster*

- Production     *# AWS EKS with auto-scaling*

**CI/CD Pipeline**

**yaml**

*# Automated pipeline stages:*

1. Code Commit       *# Git webhook trigger*

2. Unit Tests        *# Jest test execution*

3. Integration Tests *# API endpoint testing*

4. Security Scan     *# Dependency vulnerability check*

5. Docker Build      *# Multi-stage builds*

6. Image Push        *# Container registry*

7. Deployment        *# Rolling updates*

8. Smoke Tests       *# Post-deployment validation*

**📋 Key Architectural Decisions**

**Why Microservices?**

* **Team Independence**: Separate teams can work on different services
* **Technology Diversity**: Each service can use optimal tech stack
* **Scalability**: Scale individual services based on load
* **Fault Isolation**: Failures contained to specific services

**Why Event-Driven Architecture?**

* **Loose Coupling**: Services don't directly depend on each other
* **Eventual Consistency**: Data synchronization without tight coupling
* **Scalability**: Asynchronous processing for better performance
* **Audit Trail**: Complete event history for compliance

**Why Database-per-Service?**

* **Data Ownership**: Each service owns its data
* **Schema Evolution**: Independent database changes
* **Performance**: Optimized for specific service needs
* **Failure Isolation**: Database issues don't affect all services

**🎉 Summary**

Your HRMS is a **state-of-the-art, enterprise-grade system** that demonstrates:✅ **Modern Architecture Patterns**: Microservices, Event-driven, CQRS✅ **Production-Ready Infrastructure**: Docker, Kubernetes, monitoring✅ **Comprehensive Security**: Multi-layer protection and compliance✅ **Excellent Developer Experience**: TypeScript, hot reloading, testing✅ **Operational Excellence**: Logging, metrics, tracing, alerts✅ **Business Completeness**: Full HR lifecycle coverageThis system is ready for enterprise deployment and can handle:

* **1000+ employees**
* **High availability** (99.9%+ uptime)
* **Horizontal scaling** as your organization grows
* **Compliance requirements** (GDPR, SOX, etc.)
* **Integration** with existing enterprise systems

The architecture follows industry best practices and is built for **long-term maintainability** and **continuous evolution**.