Agent Assist Platform

AI-Powered Contact Center Intelligence

TECHNICAL DOCUMENTATION

Engineering Reference for Frontend, Backend & DevOps Teams

Table of Contents

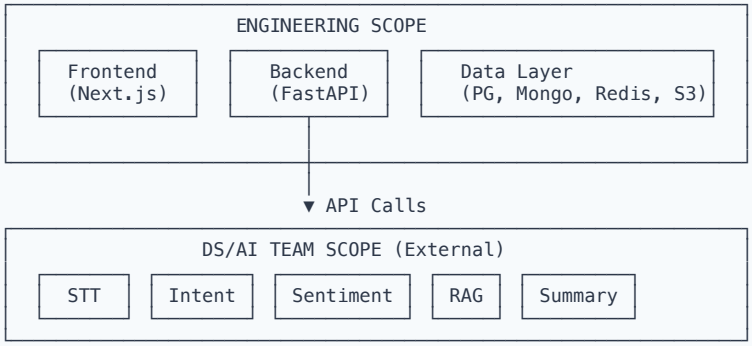
1. Platform Overview

Engineering Scope & Responsibilities

Engineering Scope

This documentation covers **Engineering-owned systems only**. DS/AI services (STT, Intent, Sentiment, RAG, Summarization, Guidance) are **external APIs** provided by the Data Science team. Engineering consumes them.

Scope Boundary



Responsibility Matrix

WHAT ENGINEERING BUILDS	WHAT DS/AI TEAM PROVIDES
User interfaces (Agent, Manager, Admin)	Speech-to-Text API
Data storage & retrieval	Intent Detection API
Real-time event streaming	Sentiment Analysis API
User management & RBAC	Chat/RAG API
Notifications & webhooks	Summarization API
CRM/Telephony integrations	Guidance/SOP API
Analytics aggregation	Embedding/Vector search

Important Rule

Frontend never calls DS/AI APIs directly. All DS/AI calls go through Backend.

2. System Architecture

High-Level System Block Diagram

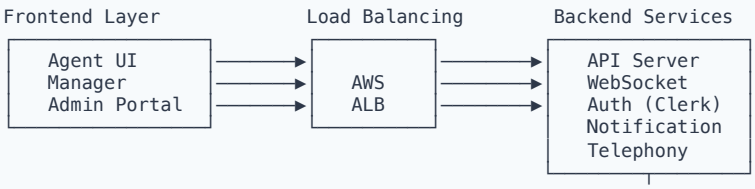
Architecture Overview

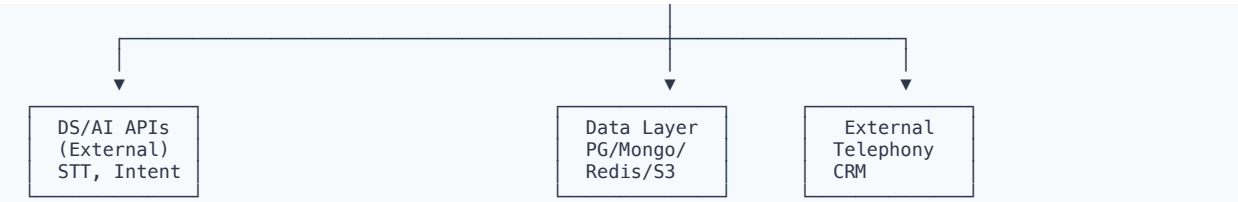
The Agent Assist Platform follows a microservices architecture with three main layers: Frontend, Backend Services, and Data Layer. Each layer is independently scalable and communicates through well-defined APIs and message queues.

System Components

LAYER	COMPONENT	TECHNOLOGY	PURPOSE
Frontend	Agent UI	Next.js	Widget/Standalone for agents
	Manager Dashboard	Next.js	Web Portal for managers
	Admin Portal	Next.js	Client configuration
Backend	API Server	FastAPI	REST API endpoints
	WebSocket Server	FastAPI/Socket.io	Real-time events
	Notification Service	Python/Celery	Alerts & notifications
	Telephony Service	Python	Call integration
Data Layer	PostgreSQL	AWS RDS	Config & metadata
	MongoDB	Self-hosted/Atlas	Sessions & documents
	Redis	ElastiCache	Cache & state
	TimescaleDB	Self-hosted	Metrics & analytics
	S3	AWS	Audio & files

Data Flow Architecture



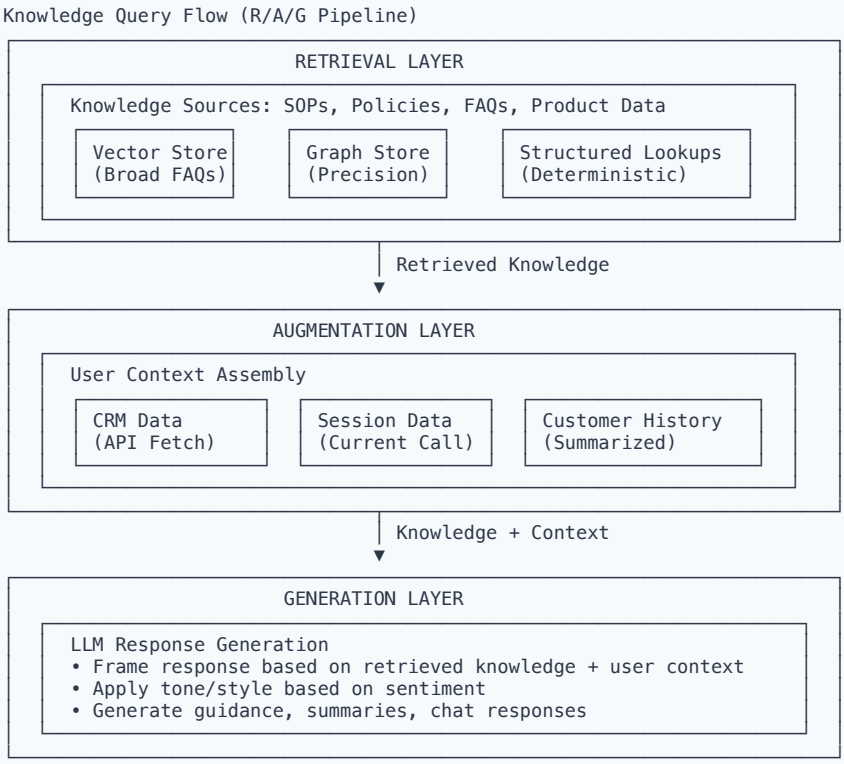


Key Architecture Decisions

- **Separation of Concerns:** Frontend apps are independent and communicate only through Backend APIs
- **Real-time via WebSocket:** All live updates (intent, sentiment, transcript) pushed through WebSocket
- **DS/AI as External:** All ML/AI processing delegated to external DS team APIs
- **Multi-database Strategy:** Right tool for the job - relational, document, cache, time-series

Knowledge Retrieval Architecture (RAG Pipeline)

The knowledge retrieval system follows a strict **Retrieval → Augmentation → Generation (R/A/G)** separation. This separation is critical: if Retrieval or Augmentation is weak, Generation will hallucinate regardless of prompt engineering.



Knowledge Representation Strategy

PHASE	APPROACH	USE CASE	TRADE-OFFS
Phase 0-1	Parametric (Predefined Partitions)	Known clients, defined industries	Fast to ship, controllable, requires manual bucket definition
Phase 2+	Graph-based Retrieval	High-precision scenarios (similar items)	Deterministic relationships, harder to build/maintain
Future R&D	Bayesian Non-parametric (IBP/CRP)	Multi-industry scale, auto-discovery	Automatic node creation, requires significant R&D

Retrieval Strategy Selection

SCENARIO	RECOMMENDED APPROACH	RATIONALE
Generic guidelines, broad FAQs	Vector Search	Semantic similarity works well for distinct concepts
Similar items (handsets, plans, roaming packs)	Graph + Structured Query	Vectors fail when items are semantically close
Deterministic lookups (pricing, features)	Structured Database	Precision requirements cannot be met with probabilistic retrieval

Precision Requirement

Precision is non-negotiable for production calls. Approximate answers may work in demos but fail in real customer conversations. Wrong roaming pack, wrong handset model, wrong plan detail = lost trust. This precision gap is the key differentiator.

3. Infrastructure

Docker Containers & Cloud Services

Current Deployment: Docker Containers

The platform is deployed using Docker containers. All services are containerized for consistency across development, staging, and production environments.

COMPONENT	SERVICE	STATUS
Container Runtime	Docker	Current
Container Orchestration	Docker Compose / AWS ECS	Current
Cloud Provider	AWS	Current
Load Balancer	AWS ALB	Current
Object Storage	AWS S3	Current
Message Queue	AWS SQS	Current
Logging	AWS CloudWatch	Current

Future Enhancements

- **Kubernetes (EKS):** Migration to AWS EKS for advanced orchestration
- **Horizontal Scaling:** Auto-scaling based on load metrics
- **Terraform:** Infrastructure as Code for reproducible deployments

Docker Architecture

```
Docker Host(s)
├── Frontend Containers
│   ├── agent-ui:latest
│   ├── manager-dashboard:latest
│   └── admin-portal:latest
│
├── Backend Containers
│   ├── api-server:latest
│   ├── websocket-server:latest
│   ├── telephony-worker:latest
│   ├── notification-worker:latest
│   └── background-worker:latest
│
├── Database Containers (Dev/Staging)
│   ├── postgres:16
│   ├── mongo:7
```

```
│   ├── redis:7
│   └── timescaledb:latest
│
└── Networking
    ├── frontend-network
    ├── backend-network
    └── data-network
```

AWS Services Used

SERVICE	PURPOSE	CONFIGURATION
AWS ALB	Load balancing	HTTPS termination, path-based routing
AWS RDS	PostgreSQL database	Multi-AZ for production
AWS ElastiCache	Redis caching	Cluster mode for HA
AWS S3	File storage	Audio recordings, documents
AWS SQS	Message queue	Async job processing
AWS SES	Email service	Transactional emails
AWS CloudWatch	Monitoring & logs	Centralized observability

4. Database Architecture

Data Storage Strategy & Schema Design

> **Extended Documentation**
>
> For complete DDL scripts, ER diagrams, and migration files, see `01-SYSTEM-ARCHITECTURE.md`

4.1 Database Overview

DATABASE	TYPE	PURPOSE	HOSTING
PostgreSQL 16+	Relational	Users, orgs, config, RBAC, audit	AWS RDS
MongoDB 7+	Document	Sessions, transcripts, documents	Self-hosted / Atlas
Redis 7+	In-Memory	Cache, real-time state	AWS ElastiCache
TimescaleDB 2.14+	Time-Series	Call metrics, analytics	Self-hosted on EKS
S3	Object Store	Audio, files	AWS S3

4.2 PostgreSQL Schema

Core Entities

TABLE	PRIMARY KEY	KEY FIELDS	INDEXES
organizations	id (UUID)	name, industry, region, settings (JSONB)	-
users	id (UUID)	org_id (FK), email, clerk_id, role_type, is_active	org_id, clerk_id
roles	id (UUID)	name, description	name (unique)
permissions	id (UUID)	role_id (FK), resource, action, scope	role_id
processes	id (UUID)	org_id (FK), name, config (JSONB)	org_id
queues	id (UUID)	process_id (FK), name, language	process_id

feature_configs	id (UUID)	org_id (FK), feature_key, enabled, settings (JSONB)	(org_id, feature_key) unique
webhooks	id (UUID)	org_id (FK), url, events[], auth_type, is_active	org_id
audit_logs	id (UUID)	org_id, user_id, action, resource_type, old/new_value	(org_id, created_at), partitioned monthly

Entity Relationships

```
ORGANIZATION (1) ──────────┬──── (*) USER
                           ├──── (*) PROCESS ──────── (*) QUEUE
                           ├──── (*) FEATURE_CONFIG
                           ├──── (*) WEBHOOK
                           └──── (*) AUDIT_LOG

USER (*) ──── (*) ROLE ──── (*) PERMISSION
```

4.3 MongoDB Collections

Sessions Collection

FIELD	TYPE	DESCRIPTION
_id	ObjectId	MongoDB document ID
call_id	String	Unique call identifier from telephony
org_id, agent_id	UUID	Foreign keys to PostgreSQL
queue, language	String	Call routing metadata
status	Enum	active | completed | failed
start_time, end_time	ISODate	Call timestamps
duration_seconds	Number	Calculated call duration
transcript[]	Array	{ speaker, text, timestamp, confidence }
insights.intents[]	Array	{ intent, confidence, timestamp }
insights.sentiment_timeline[]	Array	{ customer, agent, timestamp }
insights.final_sentiment	Object	{ customer: float, agent: float }
summary	Object	{ brief, key_points[], action_items[], resolution }
metadata	Object	{ telephony_provider, audio_s3_key, crm_ticket_id }

ttl_expires_at	ISODate	For TTL-based retention

Sessions Indexes

INDEX	FIELDS	PURPOSE
Primary lookup	org_id, start_time DESC	Session list queries
Agent lookup	org_id, agent_id, start_time DESC	Agent session queries
Customer lookup	org_id, metadata.customer_phone	Repeat caller detection
Call ID	call_id (unique)	Telephony integration
TTL	ttl_expires_at	Automatic document expiry

Documents Collection

FIELD	TYPE	DESCRIPTION
_id	ObjectId	Document ID
org_id	UUID	Organization reference
type	Enum	sop | kb | qa | governance
name, version	String, Number	Document identity
status	Enum	pending | processing | active | failed | archived
process_id, queue_id	UUID	Mapping hierarchy
content	Object	{ original_file (S3 key), file_size, mime_type }
ds_processing	Object	{ submitted_at, completed_at, error_message }

4.4 Redis Data Structures

KEY PATTERN	TYPE	FIELDS / MEMBERS	TTL
`call:{call_id}`	Hash	agent_id, org_id, queue, current_intent, sentiment_customer, sentiment_agent, last_update	4 hours
`session:{user_id}`	Hash	token, org_id, role, permissions (JSON)	24 hours
`floor:{org_id}`	Sorted Set	Members: call_id, Score: start_timestamp	12 hours
`cache:api:{hash}`	String	Cached API response (JSON)	5 minutes

4.5 TimescaleDB Analytics

call_metrics Hypertable

COLUMN	TYPE	DESCRIPTION
time	TIMESTAMPTZ	Metric timestamp (partition key)
org_id	UUID	Organization identifier
agent_id	UUID	Agent identifier (nullable for org-wide)
queue	TEXT	Queue/campaign name
call_count	INTEGER	Number of calls in bucket
total_duration_sec	INTEGER	Sum of call durations
avg_sentiment	FLOAT	Average customer sentiment
avg_sop_adherence	FLOAT	Average SOP score
compliance_events	INTEGER	Compliance flag count
upsell_opportunities	INTEGER	Upsell prompts shown
upsell_conversions	INTEGER	Successful upsells

Continuous Aggregates

VIEW	BUCKET SIZE	AGGREGATIONS
call_metrics_hourly	1 hour	SUM(calls), AVG(sentiment), AVG(sop_score)
call_metrics_daily	1 day	SUM(calls), AVG(sentiment), AVG(sop_score)

4.6 S3 Storage Structure

BUCKET	PATH PATTERN	CONTENTS
agent-assist-recordings	`{org_id}/{year}/{month}/{day}/{call_id}.opus`	Call recordings
agent-assist-recordings	`{org_id}/{year}/{month}/{day}/{call_id}.meta.json`	Recording metadata
agent-assist-documents	`{org_id}/documents/{doc_id}/original.{ext}`	Uploaded documents
agent-assist-documents	`{org_id}/documents/{doc_id}/processed.json`	Extracted text

4.7 Data Retention Policies

DATA TYPE	STORAGE	DEFAULT TTL	CONFIGURABLE RANGE	ENFORCEMENT
Audio Recordings	S3	90 days	30 - 365 days	S3 Lifecycle Rules
Full Transcripts	MongoDB	180 days	30 - 730 days	TTL Index
Call Summaries	MongoDB	365 days	90 - 730 days	TTL Index
Call Metrics	TimescaleDB	180 days	90 - 365 days	Retention Policy
Audit Logs	PostgreSQL	2 years	1 - 7 years	Partition Drop

4.8 Multi-Tenancy Enforcement

DATABASE	ISOLATION METHOD	IMPLEMENTATION
PostgreSQL	Row-level	org_id column + RLS policies on all tables
MongoDB	Row-level	org_id field required in every document, enforced by API layer
Redis	Key prefix	org_id included in key pattern where applicable
TimescaleDB	Row-level	org_id column, queries always filtered

S3	Path prefix	org_id as first path segment

5. Technology Stack

Latest Stable Versions

Frontend Technologies

COMPONENT	TECHNOLOGY	PURPOSE
Framework	**Next.js 15+**	React framework with App Router
UI Library	**React 19+**	Component library
Language	**TypeScript 5.5+**	Type safety
Styling	**Tailwind CSS 4+**	Utility-first CSS
State Management	**Zustand 5+**	Client state
Server State	**TanStack Query 5+**	API caching
WebSocket	**Socket.io Client 4+**	Real-time communication
Charts	**Recharts 2+**	Data visualization
Forms	**React Hook Form 7+**	Form handling
Validation	**Zod 3+**	Schema validation

Backend Technologies

COMPONENT	TECHNOLOGY	PURPOSE
Language	**Python 3.12+**	Backend services
API Framework	**FastAPI 0.115+**	REST API server
WebSocket	**FastAPI WebSocket / Socket.io**	Real-time events
ORM	**SQLAlchemy 2+**	PostgreSQL ORM
MongoDB Driver	**PyMongo 4+**	MongoDB client
Redis Client	**redis-py 5+**	Redis client
Task Queue	**Celery 5+**	Background jobs

Validation	**Pydantic 2+**	Data validation

Database Versions

DATABASE	VERSION
PostgreSQL	16+
MongoDB	7+
Redis	7+
TimescaleDB	2.14+

DevOps & Tools

COMPONENT	SERVICE	STATUS
CI/CD	GitHub Actions	Current
Containerization	Docker	Current
IaC	Terraform	Future
Source Control	GitHub	Current
Package Manager (FE)	pnpm	Current
Package Manager (BE)	uv / pip	Current

6. System Flows & Diagrams

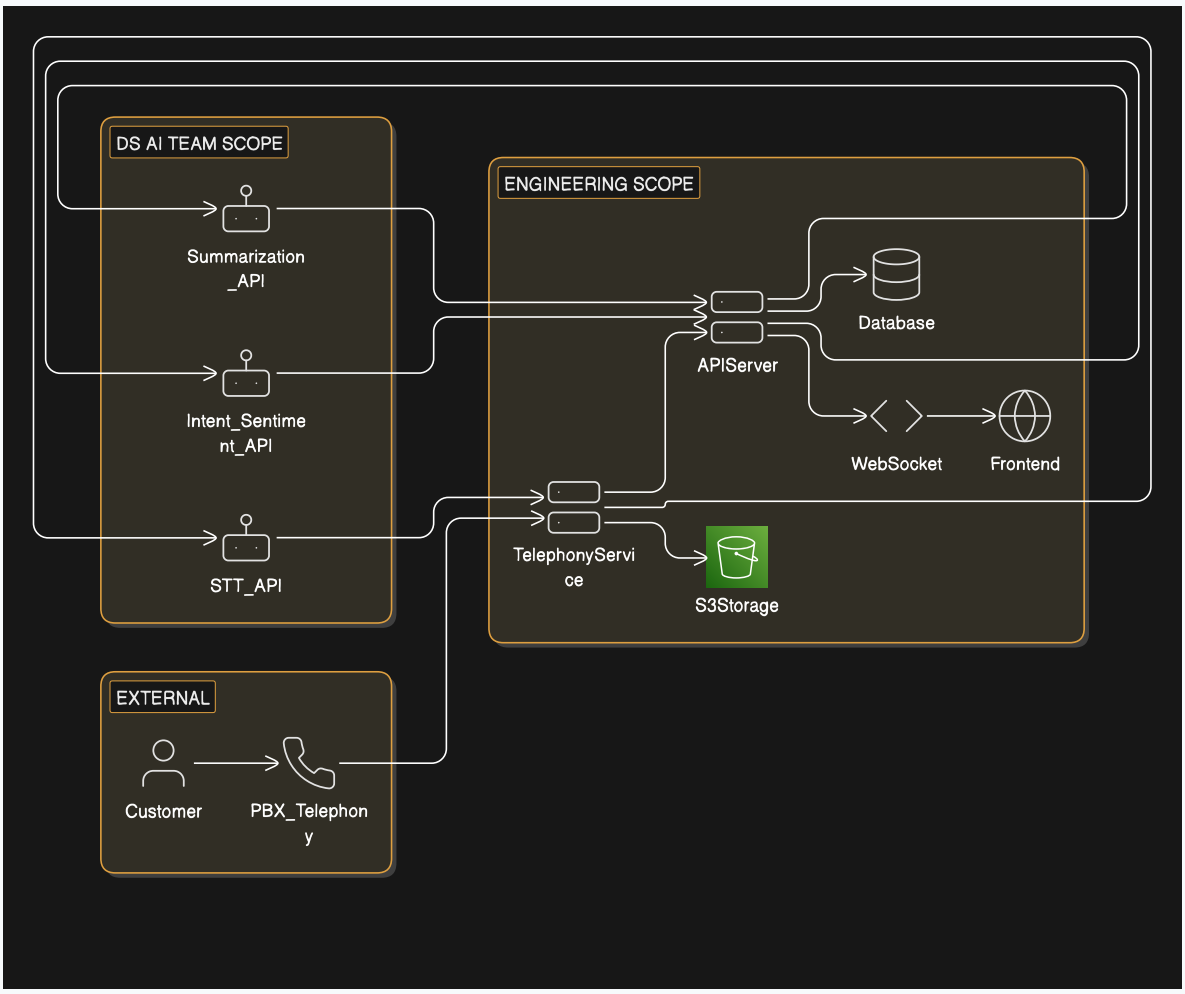
Process Flows & Data Pipelines

Extended Documentation

For Mermaid sequence diagrams and code examples, see `02-SYSTEM-FLOWS.md`

6.1 Call Lifecycle Flow

Complete Call Lifecycle from Incoming Call to Post-Call Summary

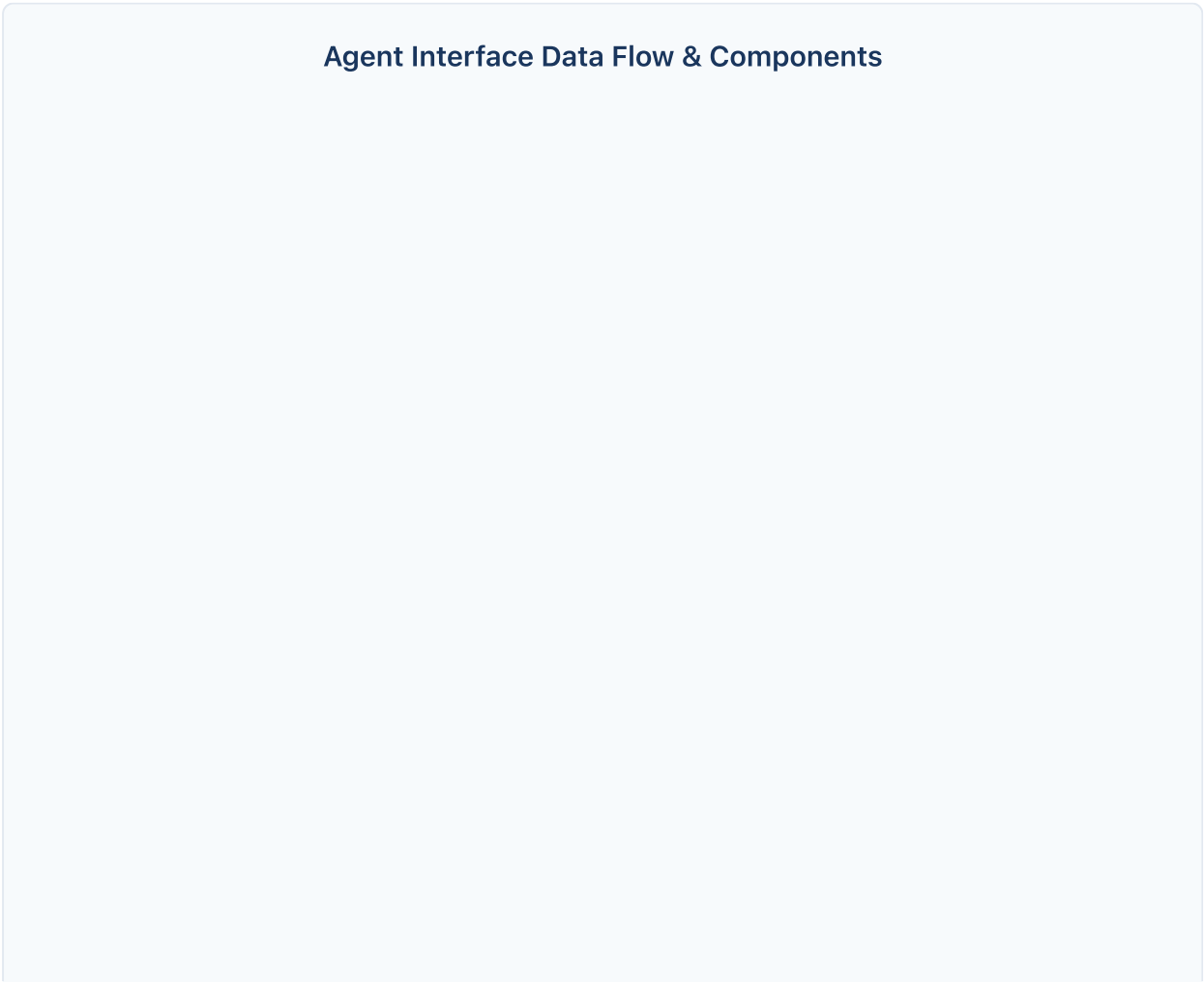


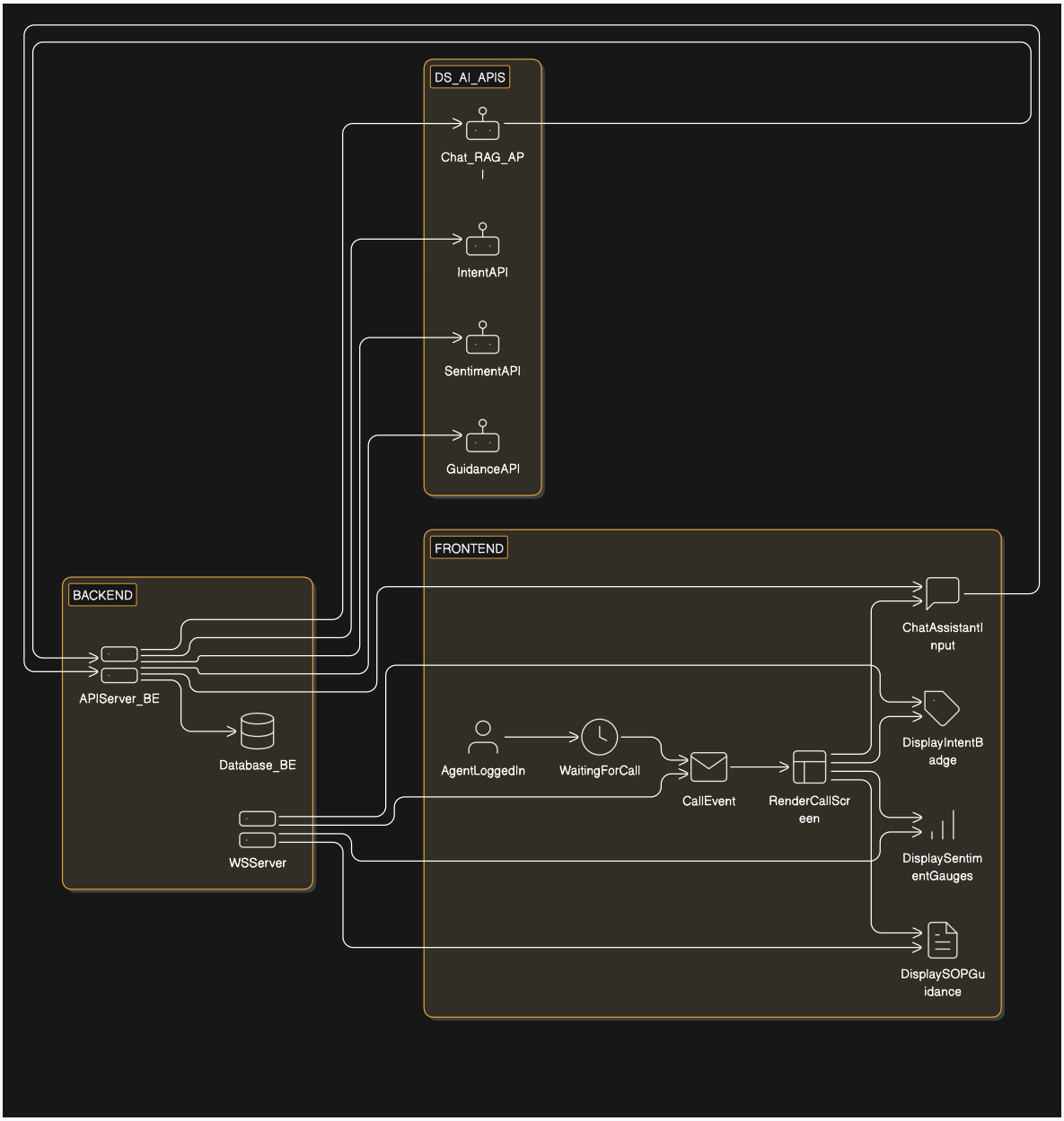
Stage-by-Stage Responsibilities

STAGE	ENGINEERING RESPONSIBILITY	DS/AI API CALLED	DATA WRITTEN

Call Start	Extract metadata (ANI, DNIS, queue), create session	-	MongoDB: session (status: active)
Recording Start	Initialize S3 upload stream	-	S3: audio chunks
Audio Streaming	Forward audio to STT API	STT API	-
Transcript Received	Store chunk, trigger analysis	Intent + Sentiment APIs	MongoDB: transcript[], Redis: call state
Real-time Push	Push to WebSocket	-	-
Call End	Finalize recording, calculate duration	-	S3: finalize, MongoDB: end_time
Post-Call	Request summary, store results	Summarization API	MongoDB: summary, TimescaleDB: metrics
Notification	Trigger webhooks, notify frontend	-	-

6.2 Agent UI Data Flow





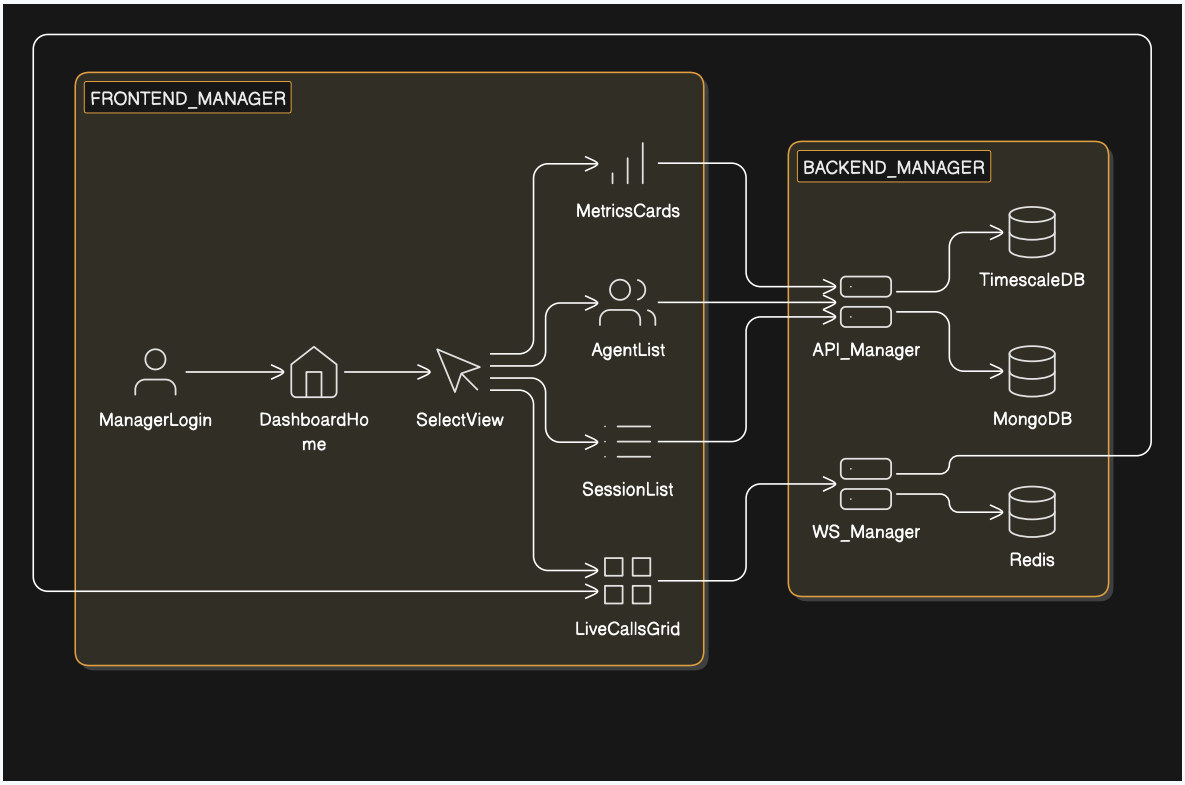
UI Component Data Sources

COMPONENT	TRANSPORT	EVENT/ENDPOINT	PAYLOAD	UPDATE FREQUENCY
Intent Badge	WebSocket	`intent.updated`	{ intent, confidence }	Every 3-5s
Sentiment Gauges	WebSocket	`sentiment.updated`	{ customer, agent } (-1 to +1)	Every 5s
SOP Panel	WebSocket	`guidance.updated`	{ sop, step, total_steps, next_action }	On step change
Coaching Prompt	WebSocket	`coaching.prompt`	{ type, message }	On trigger
Chat Assistant	REST	`POST /chat/message`	Request: { query }, Response: { answer,	On user query

			sources[] }	
Repeat Caller	REST	`GET /sessions?customer_id=`	Previous sessions list	On call start

6.3 Manager Dashboard Data Flow

Manager Dashboard Architecture & Data Sources



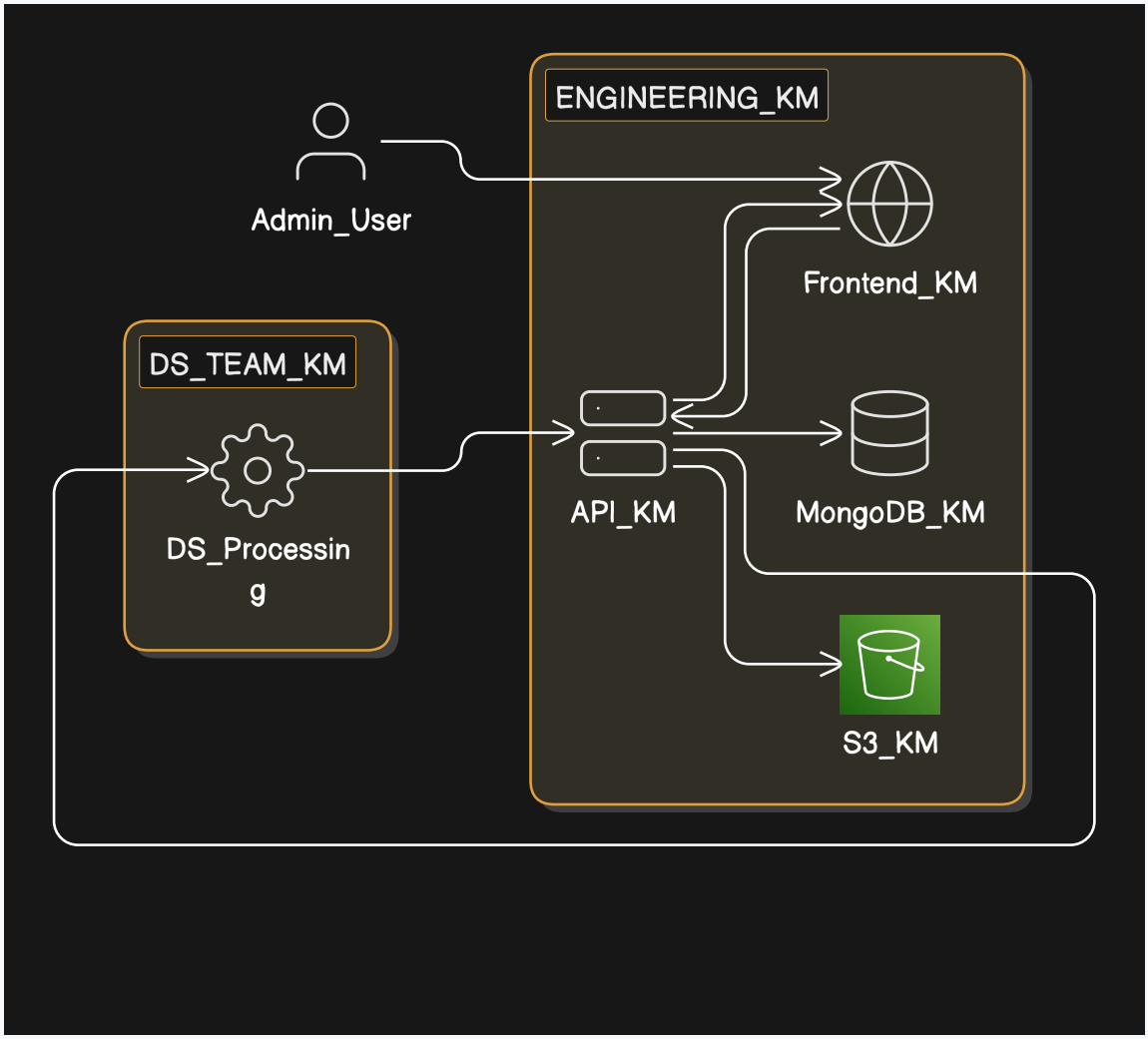
Dashboard Query Patterns

VIEW	DATA SOURCE	QUERY TYPE	CACHE TTL
Overview Metrics	TimescaleDB	Aggregate from continuous aggregates	5 min
Trend Charts	TimescaleDB	Time-bucket queries with filters	5 min
Agent List	PostgreSQL + MongoDB	Join users with session aggregates	1 min
Session List	MongoDB	Filtered query with pagination	No cache
Session Detail	MongoDB	Single document lookup	No cache

Floor View	Redis	WebSocket subscription to `floor.update`	Real-time

6.4 Knowledge Management Flow

Document Upload & Processing Pipeline



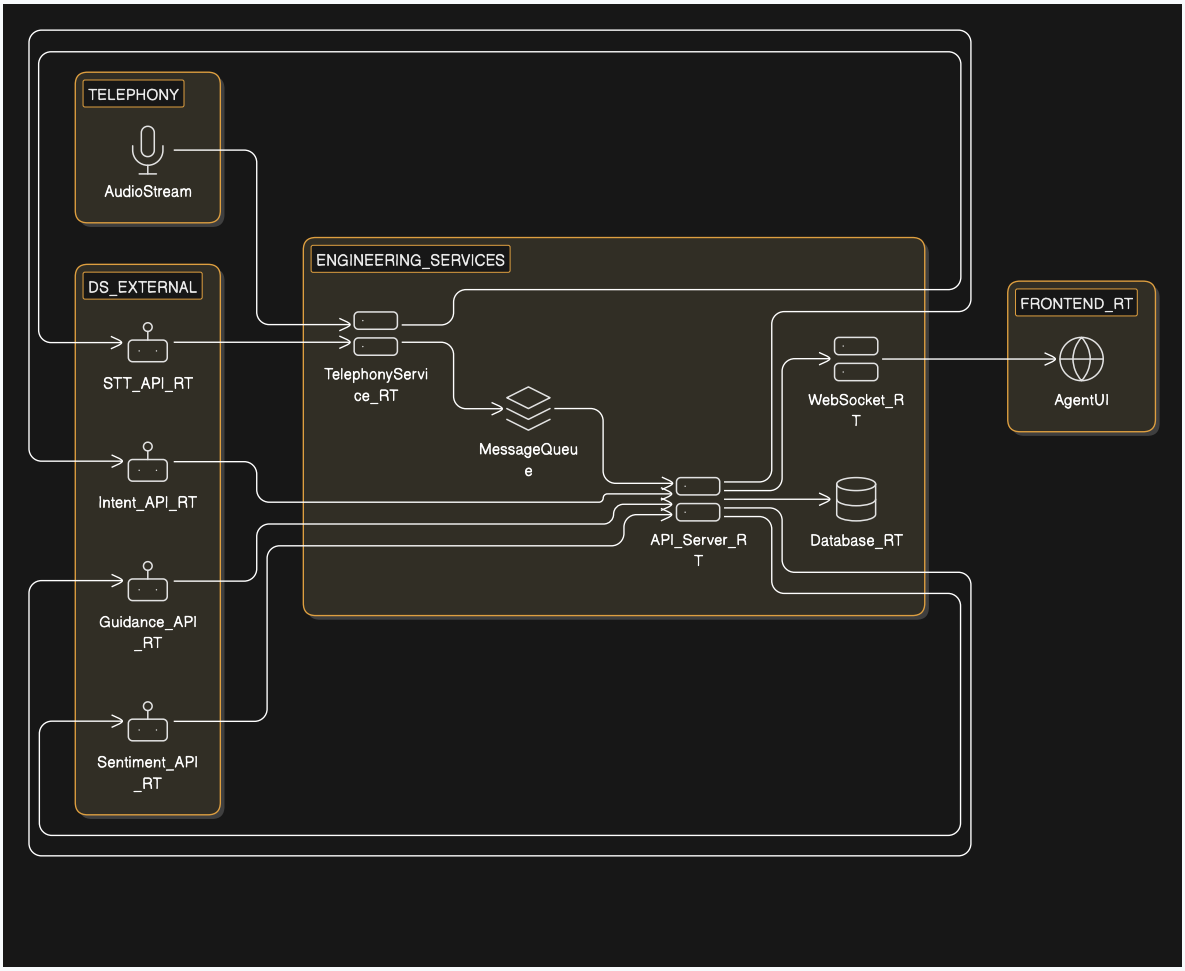
Document Processing Pipeline

STEP	SYSTEM	ACTION	STATUS
1. Upload	Frontend → API	Validate file type/size, upload to S3	-
2. Record	API → MongoDB	Create document record	pending
3. Queue	API → SQS	Submit processing job	pending
4. Process	Worker → DS API	Call DS processing endpoint	processing
5. DS Work	DS Team	Parse, chunk, embed, index in vector store	processing
6. Callback	DS → API (Webhook)	Notify processing complete	-

7. Activate	API → MongoDB	Update status, notify admin	active

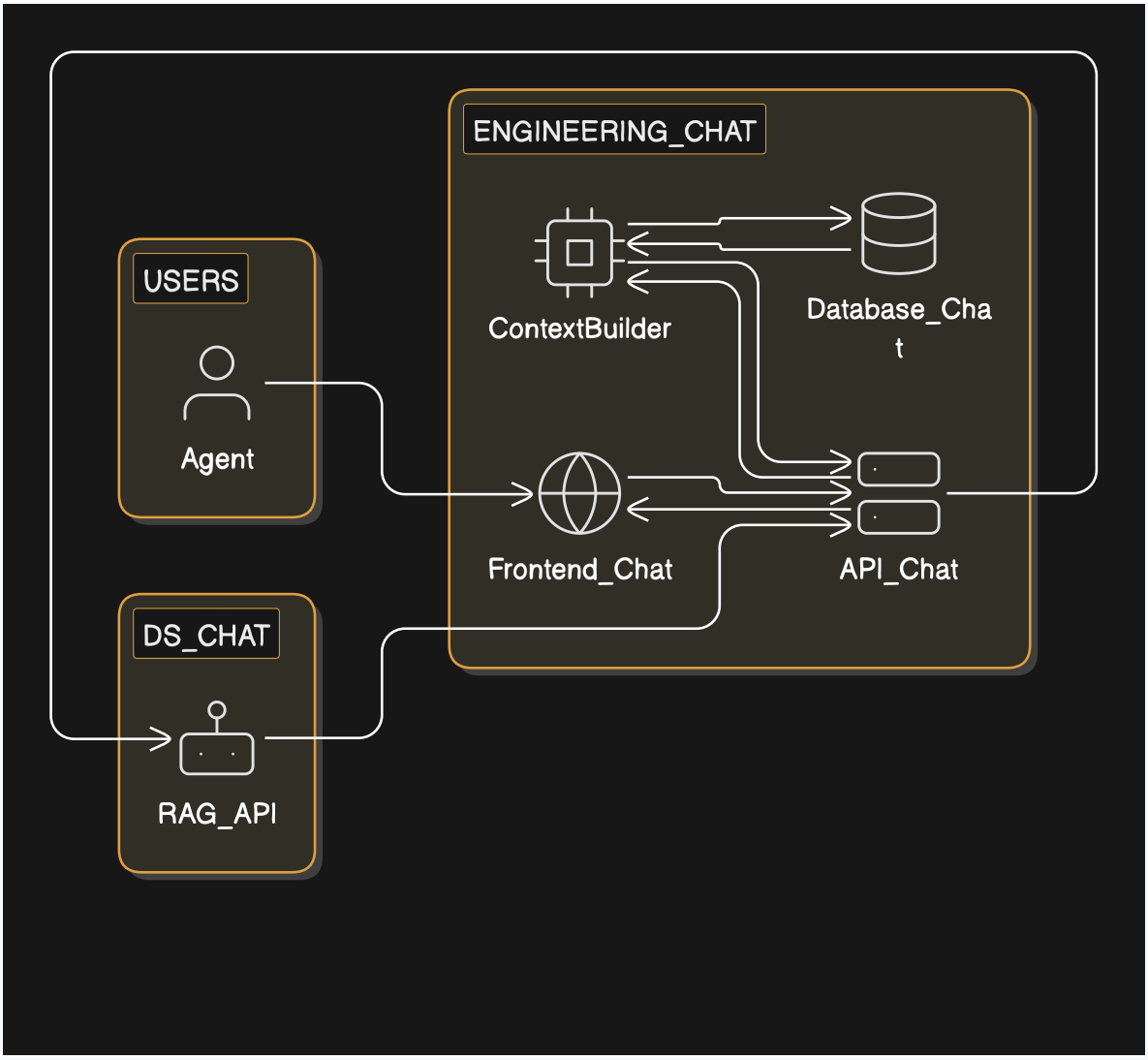
6.5 Real-Time Processing Pipeline

Real-Time Data Processing & Event Flow



6.6 Chat Assistant Flow (R/A/G Pipeline)

Chat/RAG Query Processing Flow



The Chat Assistant follows the **Retrieval → Augmentation → Generation** pipeline. Each layer is distinct and failure in early layers cannot be compensated by later layers.

R/A/G Pipeline Stages

STAGE	LAYER	ACTION	FAILURE IMPACT
1	**Retrieval**	Pull relevant SOPs, policies, knowledge from vector/graph store	Wrong/missing knowledge → hallucination
2	**Augmentation**	Attach user context (CRM data, session data, summarized history)	Missing context → generic/irrelevant response
3	**Generation**	Frame response using retrieved knowledge + context	Only impacts phrasing if R/A are solid

Retrieval Context Assembly

CONTEXT FIELD	SOURCE	PURPOSE
org_id	Session	Scope document search to organization

queue	Session	Prioritize queue-specific documents
current_intent	Redis	Guide retrieval strategy selection

Augmentation Context Assembly

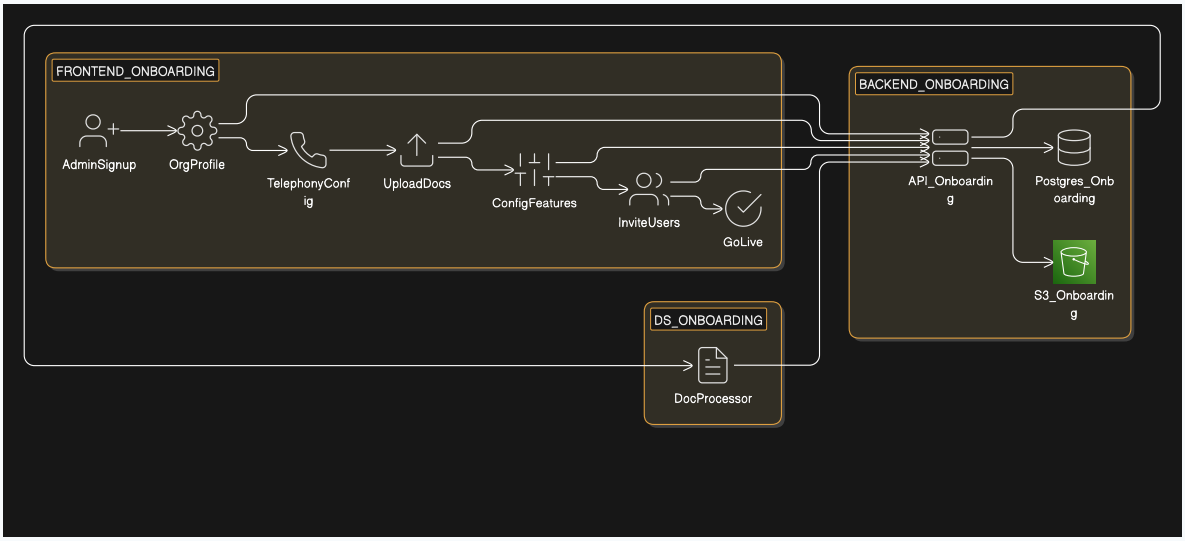
CONTEXT FIELD	SOURCE	PURPOSE
user_summary	CRM API (summarized)	Customer profile and attributes
recent_transcript (last 10)	MongoDB	Current conversation context
customer_sentiment	Redis	Tone adjustment for response
relevant_history	CRM API (filtered)	Recent orders/interactions (summarized, not raw)

Precision Strategy

For queries involving similar items (device models, plans, roaming packs), the system uses **graph-based retrieval** instead of pure vector search. Vector search works for broad FAQs but fails when semantic neighborhoods are tight.

6.7 Client Onboarding Flow

New Client Onboarding Process

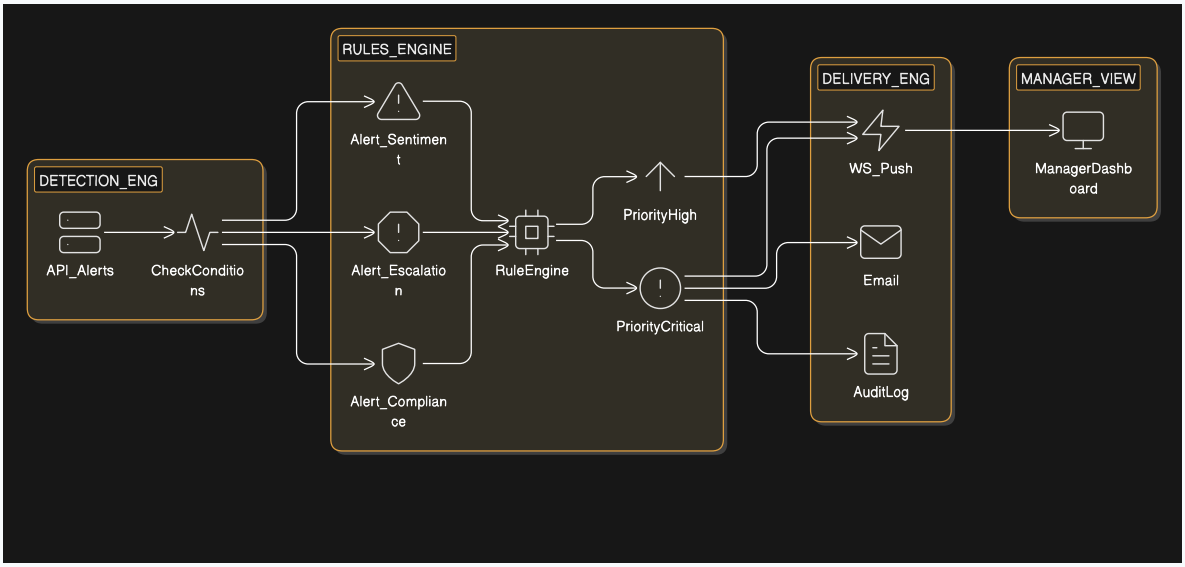


Onboarding Steps

STEP	ROUTE	API ENDPOINT	DATA WRITTEN
1. Signup	/signup	Clerk API	Clerk user + PostgreSQL user
2. Org Profile	/onboarding/profile	POST /organizations	PostgreSQL: organizations
3. Telephony	/onboarding/telephony	PUT /organizations/:id/telephony	PostgreSQL: org settings
4. Documents	/onboarding/documents	POST /documents	S3 + MongoDB + SQS
5. Features	/onboarding/features	PUT /features	PostgreSQL: feature_configs
6. Users	/onboarding/users	POST /users/invite	Clerk invite + PostgreSQL
7. Activate	/onboarding/complete	POST /organizations/:id/activate	PostgreSQL: org status

6.8 Alerts & Notifications Flow

Alert Detection & Notification System



Alert Rules

ALERT TYPE	TRIGGER CONDITION	SEVERITY	CHANNELS
Sentiment Negative	Customer sentiment < -0.5	High	WebSocket
Escalation Risk	Intent = "escalation" AND confidence > 0.7	Critical	WebSocket + Email
Compliance Flag	DS API returns compliance flag	Critical	WebSocket + Email + Audit
Long Duration	Call duration > configured threshold	Medium	WebSocket
SOP Deviation	Adherence score < 50%	Medium	WebSocket

Notification Channels

CHANNEL	TECHNOLOGY	USE CASE	LATENCY
WebSocket	Socket.io	Real-time dashboard alerts	< 100ms
Email	AWS SES	Critical alerts, summaries	1-5s
Webhook	HTTP POST	External integrations	Async (queued)

6.9 Post-Call Processing

STEP	ACTION	DATA SOURCE	DATA WRITTEN

1	Receive call.ended event	Telephony Service	-
2	Finalize S3 recording	S3 multipart	S3: complete upload
3	Fetch full transcript	MongoDB	-
4	Call Summarization API	DS API	-
5	Store summary	DS response	MongoDB: summary
6	Calculate final metrics	Transcript + insights	MongoDB: final_sentiment, duration
7	Insert analytics record	Session data	TimescaleDB: call_metrics
8	Trigger webhooks	Webhook config	-
9	Notify frontend	-	WebSocket: session.completed

7. Technical Specifications

Data Models, APIs & Implementation Details

7.1 Deployment Modes

MODE	DELIVERY	AUTHENTICATION	INTEGRATION POINTS
Embedded CRM	iframe or JavaScript SDK	JWT passthrough from CRM	CRM call events, customer context
Standalone	Separate URL	Clerk SSO	Telephony adapter only

Embedded Mode Configuration

SETTING	TYPE	OPTIONS
embed_type	Enum	iframe | widget
trigger	Enum	call_event | manual
position	Enum	right | left | floating
dimensions	Object	{ width, height }
auth_method	Enum	jwt_passthrough | sso_redirect

7.2 Knowledge Management System

Document Mapping Hierarchy

```
Organization
└── Industry (e.g., Telecom)
    └── Process (e.g., Customer Support)
        └── Queue/Campaign (e.g., Billing, Roaming)
            └── Language (e.g., en, es)
                └── Documents (SOPs, KB articles)
                └── Structured Data (Products, Plans, Devices)
```

Document Types

TYPE	CODE	PURPOSE	RETRIEVAL STRATEGY

SOP	sop	Standard Operating Procedures	Vector (broad semantic match)
Knowledge Base	kb	FAQs, troubleshooting	Vector (semantic similarity)
Product Catalog	product	Devices, plans, packs, pricing	Graph + Structured (precision required)
QA Framework	qa	Quality scoring criteria	Structured lookup
Governance	governance	Compliance rules, guardrails	Structured lookup

Knowledge Partition Strategy (Phase 0-1)

For initial deployment, knowledge is organized into predefined parametric partitions per client/industry:

INDUSTRY	EXAMPLE PARTITIONS	PRECISION REQUIREMENTS
Telecom	Network Issues, Pricing, Device Setup, Roaming, Billing	High (similar plan names, device variants)
E-commerce	Orders, Returns, Shipping, Products	Medium (order-specific lookups)
Banking	Accounts, Transactions, Cards, Loans	High (account-specific precision)

Version Control

FEATURE	IMPLEMENTATION
Version tracking	version field (integer), auto-incremented on upload
Active version	is_active flag, only one per (org, process, queue, language)
Rollback	POST /documents/{id}/activate to switch active version
Audit	All uploads/activations logged to audit_logs

Processing Status Flow

STATUS	DESCRIPTION	NEXT STATUS
pending	Uploaded, awaiting processing	processing
processing	DS team processing (parse, embed, index)	active | failed
active	Available for live queries	archived

failed	Processing error	pending (re-upload)
archived	Superseded by newer version	-

7.3 Real-Time Call Data

WebSocket Event Schemas

EVENT	FIELDS	UPDATE TRIGGER
`call.started`	call_id, agent_id, queue, customer_id, timestamp	Telephony call event
`call.ended`	call_id, duration, timestamp	Telephony hangup
`transcript.chunk`	call_id, chunk_id, speaker, text, timestamp, confidence, is_final	STT API response
`intent.updated`	call_id, intent, confidence, previous	Intent API response
`sentiment.updated`	call_id, customer (-1 to +1), agent, trend	Sentiment API response
`guidance.updated`	call_id, sop_name, current_step, total_steps, next_action	Guidance API response
`coaching.prompt`	call_id, type, message, severity	Alert rules engine

Redis Call State Fields

FIELD	TYPE	UPDATED BY
agent_id	UUID	Call start
org_id	UUID	Call start
queue	String	Call start
start_time	Timestamp	Call start
current_intent	String	Intent API
intent_confidence	Float	Intent API
sentiment_customer	Float	Sentiment API
sentiment_agent	Float	Sentiment API
transcript_count	Integer	STT API

last_update	Timestamp	Any update

7.4 Session Data Structure

Transcript Entry

FIELD	TYPE	DESCRIPTION
chunk_id	String	Unique identifier for deduplication
speaker	Enum	agent | customer
text	String	Transcribed text (PII redacted)
timestamp	Integer	Milliseconds from call start
confidence	Float	STT confidence score (0-1)

Summary Structure

FIELD	TYPE	DESCRIPTION
brief	String	1-2 sentence summary
key_points	Array[String]	Main discussion topics
action_items	Array[String]	Follow-up actions identified
resolution	Enum	resolved | escalated | follow_up | unresolved

Insights Structure

FIELD	TYPE	DESCRIPTION
intents[]	Array	{ intent, confidence, timestamp } history
primary_intent	String	Most confident/frequent intent
sentiment_timeline[]	Array	{ customer, agent, timestamp } over time
final_sentiment	Object	{ customer, agent } at call end
sop_adherence_score	Float	0-100 percentage
compliance_flags[]	Array[String]	Any compliance issues detected

7.5 Analytics Metrics

Overview Dashboard Metrics

METRIC	CALCULATION	SOURCE
Total Sessions	COUNT(*)	TimescaleDB
Average Sentiment	AVG(final_sentiment.customer)	TimescaleDB
CSAT Score	Derived from sentiment + survey data	TimescaleDB
Average Handle Time	AVG(duration_seconds)	TimescaleDB
SOP Adherence	AVG(sop_adherence_score)	TimescaleDB
First Call Resolution	COUNT(resolution=resolved) / COUNT(*)	MongoDB

Leaderboard Metrics

METRIC	RANKING	AGGREGATION
QA Score	Highest first	AVG per agent
SOP Adherence	Highest first	AVG per agent
Compliance Breach Rate	Lowest first	COUNT per agent
Upsell Conversion	Highest first	conversions / opportunities per agent
Customer Sentiment	Highest first	AVG final_sentiment per agent

7.6 Floor View (Live Monitoring)

Active Call Card Data

FIELD	SOURCE	UPDATE
call_id	Redis sorted set	On call start/end
agent_name	PostgreSQL (cached)	Static
duration	Calculated from start_time	Client-side timer
queue	Redis hash	On call start
current_intent	Redis hash	Real-time
customer_sentiment	Redis hash	Real-time
risk_level	Computed from alerts	Real-time

Risk Level Calculation

LEVEL	VISUAL	CONDITION
Critical	Red border	compliance_breach OR escalation_risk alert active
Warning	Yellow border	sentiment_declining OR sop_deviation alert active
Normal	Green/none	No active alerts

7.7 Configuration Storage

Feature Toggles

FEATURE_KEY	DEFAULT	EFFECT WHEN DISABLED
transcription	enabled	No STT API calls, no transcript
intent_detection	enabled	No intent display
sentiment_analysis	enabled	Hide sentiment gauges
sop_guidance	enabled	No guidance panel
chat_assistant	enabled	Chat disabled
upsell_nudges	disabled	No upsell prompts
post_call_summary	enabled	No summary generation
manager_alerts	enabled	No alert WebSocket events
floor_view	enabled	Floor view inaccessible

7.8 RBAC Matrix

ROLE	SESSIONS	TRANSCRIPTS	DASHBOARD	FLOOR VIEW	ADMIN
Agent	Own only	Own only	❌	❌	❌
Team Lead	Team	Team	Basic	Team only	❌
Manager	All org	All org	Full	Full	❌
Admin	All org	All org	Full	Full	✅

8. Frontend Architecture

Application Structure & Technical Patterns

> **Extended Documentation**
>
> For wireframes, component specs, and implementation details, see `04-FRONTEND-SCREENS.md`

8.1 Application Structure

APPLICATION	PURPOSE	PRIMARY ROUTES	USERS
agent-ui	Real-time call assistance	/call/[id], /summary/[id]	Agents
manager-dashboard	Analytics & monitoring	/, /agents, /sessions, /floor, /analytics	Managers
admin-portal	Client configuration	/knowledge, /features, /users, /integrations	Admins

Monorepo Layout

```
/agent-assist-frontend
├── apps/
│   ├── agent-ui/           # Next.js app
│   ├── manager-dashboard/  # Next.js app
│   └── admin-portal/       # Next.js app
└── packages/
    ├── ui/                 # Shared components
    ├── api-client/         # API client library
    ├── hooks/              # Shared React hooks
    └── types/              # TypeScript definitions
```

8.2 Shared Packages

PACKAGE	PURPOSE	KEY EXPORTS
@agent-assist/ui	Component library	Button, Card, Modal, Table, Badge, Charts
@agent-assist/api-client	HTTP client	apiClient, sessions, agents, documents, analytics
@agent-assist/hooks	React hooks	useWebSocket, useAuth, useRoleAccess, useCallState
@agent-assist/types	TypeScript types	Session, Agent, Document, Analytics interfaces

8.3 State Management

STATE TYPE	SOLUTION	SCOPE	EXAMPLES
Global	Zustand	App-wide	User, organization, WS connection, theme
Call State	Zustand	Agent UI only	Transcript, intent, sentiment, guidance
Server State	TanStack Query	API data	Sessions, agents, analytics (cached 5 min)
Form State	React Hook Form	Forms	Document upload, user invite, settings

Call State Structure

FIELD	TYPE	UPDATED BY
activeCall	Call | null	call.started / call.ended events
transcript[]	TranscriptEntry[]	transcript.chunk event
currentIntent	{ intent, confidence }	intent.updated event
sentiment	{ customer, agent }	sentiment.updated event
guidance	{ sop, step, nextAction }	guidance.updated event
chatHistory[]	ChatMessage[]	User queries + API responses

8.4 API Integration

REST Endpoints

ENDPOINT	METHOD	USED BY	CACHE
/api/v1/sessions	GET	Manager	No
/api/v1/sessions/:id	GET	Manager	No
/api/v1/agents	GET	Manager	1 min
/api/v1/agents/:id/metrics	GET	Manager	5 min
/api/v1/analytics/overview	GET	Manager	5 min
/api/v1/floor/active	GET	Manager	No (real-time)
/api/v1/chat/message	POST	Agent	No
/api/v1/documents	GET/POST	Admin	No

/api/v1/features	GET/PUT	Admin	1 min
/api/v1/users	GET/POST	Admin	No
/api/v1/webhooks	GET/POST/DELETE	Admin	No

WebSocket Events Consumed

EVENT	CONSUMER APP	STATE UPDATE
call.started	Agent UI	Set activeCall
call.ended	Agent UI	Clear activeCall, navigate to summary
transcript.chunk	Agent UI	Append to transcript[]
intent.updated	Agent UI	Update currentIntent
sentiment.updated	Agent UI	Update sentiment
guidance.updated	Agent UI	Update guidance
coaching.prompt	Agent UI	Show toast notification
alert.*	Manager Dashboard	Update alerts list, highlight floor card
floor.update	Manager Dashboard	Update active calls grid

8.5 Component Library

Core Components

COMPONENT	PROPS	VARIANTS
Button	variant, size, loading, disabled, onClick	primary, secondary, danger, ghost
Card	title, subtitle, footer, padding	default, bordered, elevated
Modal	open, onClose, title, size	sm, md, lg, fullscreen
Table	columns, data, onSort, pagination	default, compact, striped
Badge	variant, label, size	success, warning, error, info
Tabs	tabs[], activeTab, onChange	default, pills, underline

Domain Components

COMPONENT	PURPOSE	DATA SOURCE
SentimentIndicator	Visual gauge -1 to +1 with color coding	WebSocket: sentiment.updated
IntentBadge	Intent label with confidence bar	WebSocket: intent.updated
CallTimer	Live duration counter (MM:SS)	Local interval from start_time
TranscriptViewer	Scrollable conversation display	WebSocket: transcript.chunk
SopChecklist	SOP steps with progress indicator	WebSocket: guidance.updated
AudioPlayer	Recording playback controls	S3 signed URL
FloorCard	Active call summary card	WebSocket: floor.update
AlertBanner	Dismissible coaching prompt	WebSocket: coaching.prompt

Chart Components

COMPONENT	LIBRARY	USE CASES
LineChart	Recharts	Sentiment trends, call volume over time
BarChart	Recharts	Agent comparison, queue distribution
PieChart	Recharts	Intent distribution, resolution breakdown

GaugeChart	Custom	Single metric display (CSAT, SOP score)

8.6 Authentication

ASPECT	IMPLEMENTATION
Provider	Clerk
Session storage	Clerk-managed (cookie-based)
Token format	JWT in Authorization header
Protected routes	Clerk middleware in middleware.ts
Public routes	/sign-in, /sign-up only
Role storage	user.publicMetadata.role

Role-Based Access

ROLE	AGENT UI	MANAGER DASHBOARD	ADMIN PORTAL
agent	✅ Full	❌	❌
team_lead	✅ Full	✅ Team views only	❌
manager	✅ Full	✅ Full	❌
admin	✅ Full	✅ Full	✅ Full

8.7 Performance Patterns

PATTERN	IMPLEMENTATION	BENEFIT
Code splitting	Next.js App Router automatic	Smaller initial bundle
Query caching	TanStack Query (staleTime: 5 min)	Reduce API calls
Lazy loading	Dynamic imports for charts, modals	Faster page load
Image optimization	Next.js Image component	WebP, lazy load, sizing
WebSocket reconnect	Socket.io auto-reconnect	Resilient connections
Debounced search	useDebouncedValue (300ms)	Reduce search queries

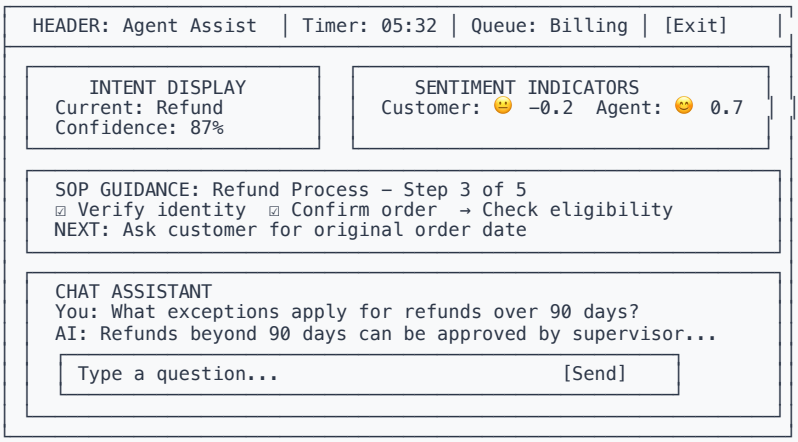
8.8 Error Handling

ERROR TYPE	HANDLING	USER FEEDBACK
API 401	Redirect to /sign-in	Session expired message
API 403	Show permission error	"Access denied" toast
API 404	Show not found page	404 error page
API 500	Show error boundary	Generic error with retry
WebSocket disconnect	Auto-reconnect (3 attempts)	"Reconnecting..." indicator
Network offline	Show offline banner	"No connection" banner

9. Screen Layouts

UI Wireframes

9.1 Agent UI - Live Call Screen



Screen Explanation

The Agent UI Live Call Screen is the primary interface for agents during active calls. It provides real-time AI-powered assistance to help agents handle customer interactions more effectively. The screen is designed to be non-intrusive while delivering critical information at the right moments.

Key Components

- **Header Bar:** Displays the call timer (auto-updating), current queue/campaign name, and an exit button to close the interface. The timer helps agents track call duration.
- **Intent Display:** Shows the current detected intent (e.g., "Refund", "Billing Inquiry") with a confidence percentage. Updates in real-time via WebSocket events (`intent.updated`) as the conversation progresses. Helps agents understand the customer's primary need.
- **Sentiment Indicators:** Dual gauges showing customer and agent sentiment scores ranging from -1 (negative) to +1 (positive). Updates every 5 seconds via `sentiment.updated` WebSocket events. Color-coded visual indicators (red/yellow/green) provide quick status assessment.
- **SOP Guidance Panel:** Displays the active Standard Operating Procedure with step-by-step checklist. Shows completed steps (☑), current step (→), and next action. Updates via `guidance.updated` WebSocket events when the Guidance API detects step progression. Helps ensure agents follow proper procedures.
- **Chat Assistant:** AI-powered knowledge base query interface. Agents can type questions and receive answers based on uploaded documents (SOPs, KB articles). Uses REST API (`POST`

`/api/v1/chat/message`) with RAG (Retrieval-Augmented Generation) to provide contextual answers. Chat history is maintained during the call session.

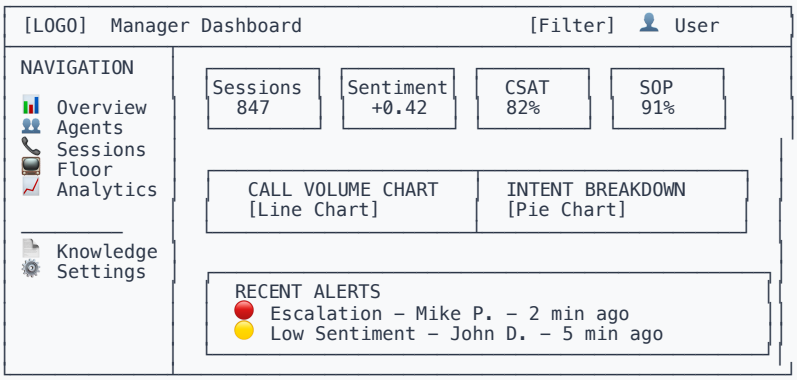
Data Sources

COMPONENT	DATA SOURCE	UPDATE METHOD
Call Timer	Local calculation from call start time	Client-side interval (1s)
Intent	Backend → DS Intent API	WebSocket: `intent.updated`
Sentiment	Backend → DS Sentiment API	WebSocket: `sentiment.updated`
SOP Guidance	Backend → DS Guidance API	WebSocket: `guidance.updated`
Chat Responses	Backend → DS RAG API	REST API (on-demand)

User Interactions

- Agents can type questions in the chat assistant input field and click "Send" to query the knowledge base
- The Exit button closes the Agent UI interface (call continues in telephony system)
- All real-time data updates automatically without user action
- Coaching prompts may appear as dismissible banners when alerts are triggered (e.g., sentiment declining, escalation risk)

9.2 Manager Dashboard - Overview



Screen Explanation

The Manager Dashboard Overview is the main landing page for managers and team leads. It provides a comprehensive view of organizational performance metrics, trends, and alerts. The dashboard aggregates data from multiple sources to give managers actionable insights at a glance.

Key Components

- **Navigation Sidebar:** Persistent navigation menu with icons for Overview, Agents, Sessions, Floor View, Analytics, Knowledge Management, and Settings. Role-based access control determines which menu items are visible.
- **Metric Cards (Top Row):** Four key performance indicators displayed as cards:
  - **Sessions:** Total number of calls handled (count from TimescaleDB)
  - **Sentiment:** Average customer sentiment score across all calls (calculated from TimescaleDB metrics)
  - **CSAT:** Customer Satisfaction score derived from sentiment and survey data
  - **SOP:** Average SOP adherence percentage across all agents

  These metrics are cached for 5 minutes to reduce database load.
- **Call Volume Chart:** Line chart showing call volume trends over time (hourly/daily buckets). Data sourced from TimescaleDB continuous aggregates. Managers can adjust time range (last 24 hours, 7 days, 30 days).
- **Intent Breakdown Chart:** Pie chart displaying distribution of call intents (e.g., Refund, Billing Inquiry, Escalation). Data aggregated from MongoDB sessions collection. Helps identify common customer needs.
- **Recent Alerts Panel:** Chronologically ordered list of recent alerts with severity indicators (🔴 Critical, 🟡 Warning). Shows alert type, affected agent, and timestamp. Updates in real-time via WebSocket `alert.*` events. Clicking an alert navigates to the relevant session or floor view.
- **Filter Controls:** Allows managers to filter data by date range, queue/campaign, agent, or team. Filters apply to all dashboard components.

Data Sources

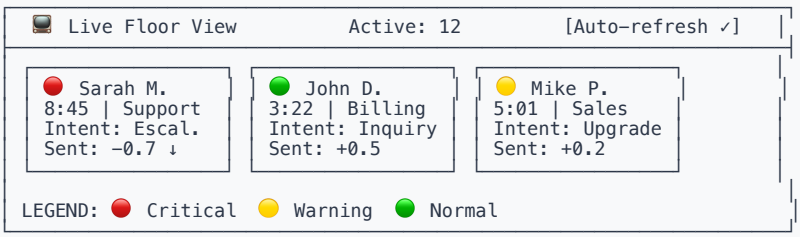
COMPONENT	DATA SOURCE	UPDATE METHOD	CACHE TTL
Metric Cards	TimescaleDB (call_metrics)	REST API: `GET /api/v1/analytics/overview`	5 minutes
Call Volume Chart	TimescaleDB continuous aggregates	REST API: `GET /api/v1/analytics/trends`	5 minutes
Intent Breakdown	MongoDB (sessions collection)	REST API: `GET /api/v1/analytics/intents`	5 minutes
Recent Alerts	WebSocket events + PostgreSQL (audit_logs)	WebSocket: `alert.*` events	Real-time

User Interactions

- Managers can click metric cards to drill down into detailed views (e.g., click "Sessions" to see session list)
- Charts support hover tooltips showing exact values and time points
- Filter controls allow dynamic data filtering without page reload
- Clicking an alert in the Recent Alerts panel navigates to the relevant session detail or floor view

- Navigation menu items route to dedicated pages (Agents, Sessions, Floor, Analytics)

9.3 Live Floor View



Screen Explanation

The Live Floor View provides real-time monitoring of all active calls across the organization. Managers use this screen to monitor agent performance, identify calls requiring intervention, and maintain situational awareness of call center operations. The view updates in real-time as calls start, progress, and end.

Key Components

- **Header Bar:** Displays the total count of active calls and an auto-refresh toggle. The active call count updates dynamically as calls start and end.
- **Active Call Cards:** Each card represents one active call with the following information:
  - **Agent Name:** Name of the agent handling the call (from PostgreSQL users table)
  - **Call Duration:** Elapsed time since call start (MM:SS format, auto-updating)
  - **Queue/Campaign:** The queue or campaign the call belongs to
  - **Current Intent:** Most recent detected intent from the Intent API
  - **Customer Sentiment:** Current customer sentiment score (-1 to +1) with trend indicator (↑, ↓, →)
  - **Risk Level Indicator:** Color-coded border (🔴 Red = Critical, 🟡 Yellow = Warning, 🟢 Green = Normal) based on active alerts

  Cards are arranged in a responsive grid layout that adjusts based on screen size.
- **Legend:** Explains the color coding system for risk levels to help managers quickly identify calls needing attention.
- **Auto-Refresh Toggle:** When enabled, the view automatically updates every 2-3 seconds. When disabled, managers can manually refresh.

Data Sources

DATA ELEMENT	DATA SOURCE	UPDATE METHOD
Active Call List	Redis sorted set (`floor:{org_id}`)	WebSocket: `floor.update`

Call State (intent, sentiment)	Redis hash (`call:{call_id}`)	WebSocket: `intent.updated` , `sentiment.updated`
Agent Information	PostgreSQL (users table, cached)	Initial load + cache refresh
Risk Level	Computed from active alerts (Redis + WebSocket)	WebSocket: `alert.*` events

Risk Level Calculation

The risk level for each call card is determined by active alerts:

- 🔴 **Critical (Red Border):** Triggered when a compliance breach is detected OR an escalation intent is detected with high confidence. These calls require immediate manager attention.
- 🟡 **Warning (Yellow Border):** Triggered when customer sentiment is declining (trending negative) OR SOP adherence score drops below 50%. These calls may need monitoring or intervention.
- 🟢 **Normal (Green/No Border):** No active alerts. Call is proceeding normally.

User Interactions

- Clicking on a call card opens a detailed view or modal showing full call information, transcript preview, and agent actions
- Managers can filter the floor view by queue, team, or risk level
- Sorting options allow ordering by duration, sentiment, or risk level
- Auto-refresh can be toggled on/off to control update frequency
- Cards with critical alerts may have a "Join" or "Monitor" button for manager intervention

Real-Time Updates

The floor view subscribes to multiple WebSocket events to maintain real-time accuracy:

- `call.started` - Adds a new card to the view
- `call.ended` - Removes the card from the view
- `intent.updated` - Updates the intent display on the relevant card
- `sentiment.updated` - Updates sentiment score and trend indicator
- `alert.*` - Updates risk level and border color
- `floor.update` - Batch update of all active calls (sent periodically)

10. Integrations

External Services & APIs

10.1 Authentication (Clerk)

```
User → Frontend → Clerk → JWT Token → API Server → Clerk Verify → PostgreSQL (RBAC) → Response
```

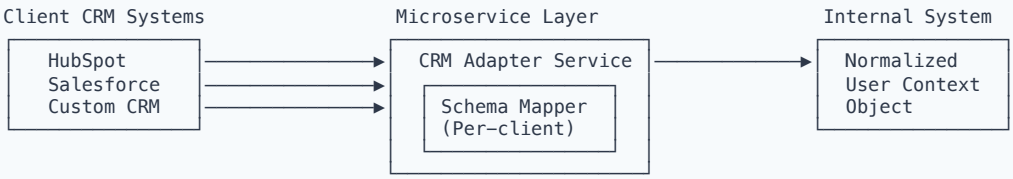
10.2 Telephony Integration

PROVIDER	TYPE	ENGINEERING WORK
Cisco Finesse	Primary	API adapter for call events
Twilio	Secondary	WebSocket adapter

10.3 CRM Integration

Integration Architecture

CRM integration follows an **API-based microservice architecture**. Every client's CRM is different, so the system uses a mapping layer to translate external schemas to internal format.



Supported CRM Systems

CRM	INBOUND DATA	OUTBOUND DATA	INTEGRATION METHOD
HubSpot	Call events, customer ID, contact data	Summary, tickets	REST API + Webhooks
Salesforce	Call events, customer ID, account data	Summary, case records	REST API + Streaming API

Custom CRM	Configurable via mapping	Configurable via mapping	API Adapter Plugin

User Context API

The CRM integration exposes two primary APIs for context assembly during augmentation:

API	PURPOSE	DATA FORMAT
`GET /api/v1/context/user/{identifier}`	Fetch user demographics and profile	Normalized JSON (subscriber number, plan ID, user attributes)
`GET /api/v1/context/history/{identifier}`	Fetch user transaction/interaction history	Summarized history (not raw tables)

> **User Data Handling**
>
> **Raw tabular data is NOT passed directly to LLM.** Large columnar data causes token explosion and high hallucination risk. The system uses industry-specific structured summarization to create "user summary objects" rather than dumping raw history.

Schema Mapping Configuration

COMPONENT	DESCRIPTION	CONFIGURATION
Field Mapping	Map CRM fields to internal schema	JSON mapping config per client
Data Transform	Transform/normalize values	Transformation rules
Summary Builder	Convert raw data to summarized context	Industry-specific templates

> **API & Webhook Reference**
>
> The endpoints and events listed below are **examples of core functionality**. For complete API reference and all available webhook events, refer to the API documentation and OpenAPI specification.

10.4 Webhook Events

EVENT	TRIGGER
`call.started`	Call begins
`call.completed`	Call ends + summary ready
`sentiment.alert`	Sentiment below threshold

`escalation.detected`	Escalation intent detected
`document.processed`	Document processing complete

> **Note:** This is a basic list of webhook events. Additional events will be added as the platform evolves. This section will be continuously updated.

10.5 REST Endpoints

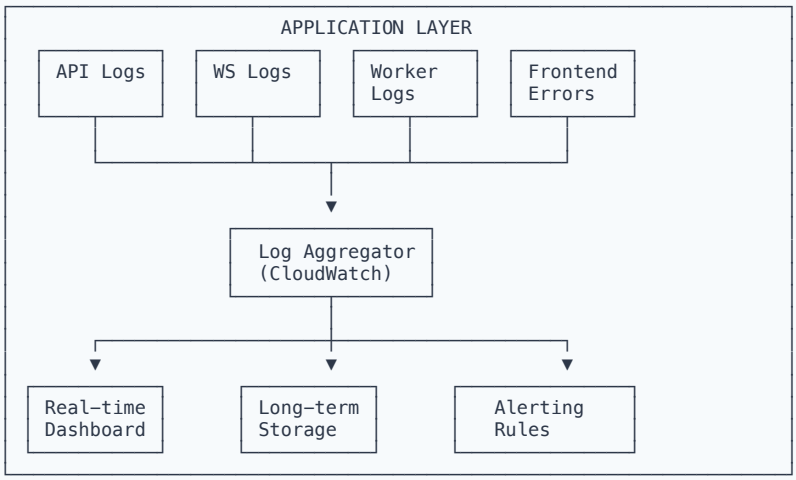
ENDPOINT	METHOD	DESCRIPTION
`/api/v1/sessions`	GET	List sessions
`/api/v1/sessions/:id`	GET	Session detail
`/api/v1/agents`	GET	List agents
`/api/v1/analytics/overview`	GET	Dashboard metrics
`/api/v1/chat/message`	POST	Chat query
`/api/v1/documents`	GET/POST	Manage documents

> **Note:** This is a basic list of REST endpoints. Additional endpoints will be added as the platform evolves. This section will be continuously updated. For complete API reference, see the OpenAPI specification.

11. Logging & Data Processing

Observability & Data Pipeline

11.1 Logging Architecture



11.2 Log Categories

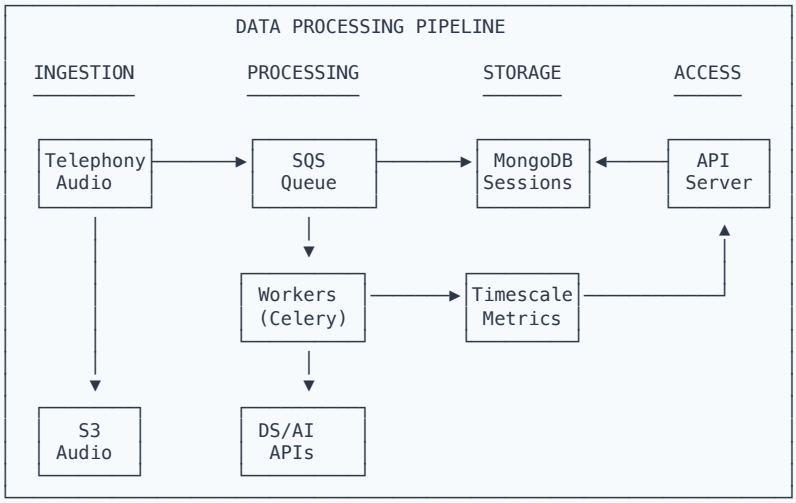
LOG TYPE	PURPOSE	RETENTION	STORAGE
Application Logs	Debugging, errors	30 days	CloudWatch
API Request Logs	Request/response tracking	90 days	CloudWatch
Audit Logs	Security, compliance	2 years	PostgreSQL
Call Event Logs	Real-time events	180 days	TimescaleDB
Error Logs	Exception tracking	90 days	CloudWatch + Sentry

11.3 Structured Log Format

```
{
  "timestamp": "2026-01-20T12:55:00.000Z",
  "level": "INFO",
  "service": "api-server",
  "trace_id": "abc123",
  "org_id": "org-uuid",
  "user_id": "user-uuid",
```

```
  "event": "api.request.completed",
  "method": "POST",
  "path": "/api/v1/sessions",
  "status_code": 200,
  "duration_ms": 145
}
```

11.4 Data Processing Pipeline



11.5 Data Flow Summary

DATA TYPE	SOURCE	PROCESSING	DESTINATION
Audio Stream	Telephony	Telephony Service	S3 + DS STT API
Transcript	DS STT API	API Server	MongoDB (sessions)
Intent/Sentiment	DS APIs	API Server	Redis + MongoDB
Metrics	API Server	Background Worker	TimescaleDB
Documents	Admin Upload	S3 + DS Processing	MongoDB

11.6 Monitoring & Alerts

METRIC	ALERT CHANNEL
API Latency (p99)	Slack + PagerDuty
Error Rate	Slack + PagerDuty
WebSocket Disconnects	Slack

Queue Depth (SQS)	Slack
DS API Errors	Slack
Database CPU	CloudWatch + Slack

12. Glossary

Key Terms & Definitions

TERM	DEFINITION
Intent	Customer's call purpose (from DS Intent API). Examples: refund_request, billing_inquiry, escalation
Sentiment	Emotional score ranging from -1 (negative) to +1 (positive), provided by DS Sentiment API
SOP	Standard Operating Procedure - guided steps for handling specific call types
RAG	Retrieval-Augmented Generation - technique used by DS Chat API to answer questions using knowledge base
STT	Speech-to-Text - converts audio to text transcript
Floor View	Real-time dashboard showing all active calls for manager monitoring
Session	A single call instance with all associated data (transcript, insights, summary)
Queue	Call routing category (billing, support, sales, etc.)
Process	Business process grouping containing one or more queues
RBAC	Role-Based Access Control - permission system based on user roles
WebSocket	Protocol for real-time bidirectional communication
Multi-Tenancy	Multiple organizations on same platform with data isolation
White-Labeling	Customization of branding (logo, colors, domain) per organization
Coaching Prompt	Real-time suggestion shown to agent based on call analysis
Escalation	When a call needs to be transferred to supervisor or higher tier

13. Technical Notes

Implementation Guidelines & Policies

13.1 API Error Codes

HTTP	CODE	DESCRIPTION	CLIENT ACTION
400	INVALID_REQUEST	Malformed request body/params	Check request format
400	VALIDATION_ERROR	Field validation failed	Check field constraints
401	UNAUTHORIZED	Missing/invalid auth token	Re-authenticate
403	FORBIDDEN	Insufficient permissions	Check RBAC
403	ORG_MISMATCH	Resource in different org	Verify org_id
404	RESOURCE_NOT_FOUND	Resource does not exist	Verify ID
409	CONFLICT	State conflict (e.g., duplicate)	Refresh and retry
429	RATE_LIMITED	Too many requests	Backoff and retry
500	INTERNAL_ERROR	Server error	Retry, contact support
502	DS_API_ERROR	DS/AI service failure	Graceful degradation
503	SERVICE_UNAVAILABLE	Service down	Retry with backoff

13.2 DS/AI API Failure Handling

Circuit Breaker Configuration

API	TIMEOUT	FAILURE THRESHOLD	RECOVERY TIMEOUT	FALLBACK
STT	5s	5 failures	30s	Queue for retry
Intent	3s	3 failures	15s	Return "Unclear"
Sentiment	5s	3 failures	15s	Hide sentiment UI
RAG/Chat	8s	3 failures	30s	Return "No results"
Summary	15s	5 failures	60s	Queue for async

Fallback Behavior

FAILURE SCENARIO	SYSTEM BEHAVIOR	AGENT DISPLAY
STT timeout	Queue audio for background retry	No visible change
Intent API down	Return last known intent or "Unclear"	Show "Analyzing..." or stale value
Sentiment API down	Skip sentiment updates	Sentiment section hidden
RAG timeout	Return empty with message	"Unable to find answer - try again"
Summary API fail	Queue for background processing	"Summary generating..."

> **Core Resilience Principle**
>
> The system must **never block the agent from continuing the call**. All AI features degrade gracefully while call handling continues uninterrupted.

13.3 PII Handling

Data Classification

DATA TYPE	CLASSIFICATION	STORAGE TREATMENT	DISPLAY TREATMENT
Phone Numbers	PII	Encrypted at rest	Masked: ***-***-1234
Customer Names	PII	Encrypted at rest	Full for authorized roles
Account Numbers	PII	Encrypted at rest	Masked: ****5678
SSN/Tax IDs	Sensitive PII	Auto-redacted, NOT stored	[REDACTED]
Payment Card	PCI	Auto-redacted, NOT stored	[REDACTED]

Auto-Redaction Patterns

PATTERN TYPE	REGEX PATTERN	REPLACEMENT
SSN	\d{3}-\d{2}-\d{4}	[REDACTED_SSN]
Phone	\d{3}[-.]?\d{3}[-.]?\d{4}	[REDACTED_PHONE]
Credit Card	\d{4}[-\s]?\d{4}[-\s]?\d{4}[-\s]?\d{4}	[REDACTED_CARD]
Email	[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}	[REDACTED_EMAIL]

13.4 Data Ownership & Storage Options

OPTION	DATA LOCATION	USE CASE	ENGINEERING WORK
Shunya-Managed	Shunya AWS infrastructure	Standard deployments	Default setup
Client-Managed	Client's cloud/on-prem	Regulated industries	Custom connectors
Hybrid	Operational: Shunya, Recordings: Client	Cost + compliance balance	Split storage config

13.5 Data Retention Enforcement

DATA TYPE	ENFORCEMENT METHOD	SCHEDULE	AUDIT
Audio (S3)	S3 Lifecycle Rules	Daily evaluation	CloudWatch metrics
Sessions (MongoDB)	TTL Index on ttl_expires_at	Background thread	Deletion logged
Metrics (TimescaleDB)	Retention Policy	Daily chunk drop	TimescaleDB jobs
Audit Logs (PostgreSQL)	Partition Drop	Monthly	Job logs

13.6 Multi-Tenancy Rules

RULE	IMPLEMENTATION
Every query must include org_id	API middleware extracts from JWT, adds to all queries
No cross-org data access	Row-level security (PostgreSQL), query filters (MongoDB)
org_id in all documents	Required field in MongoDB schemas
org_id in S3 paths	First path segment: {org_id}/...
org_id in Redis keys	Key pattern: floor:{org_id}, etc.

13.7 Audit Logging Requirements

EVENT CATEGORY	EVENTS LOGGED	FIELDS CAPTURED

Authentication	Login, logout, failed attempts	user_id, ip, user_agent, timestamp
Data Access	Session view, transcript view, export	user_id, resource_id, action
Configuration	Feature toggle, webhook change, doc upload	old_value, new_value, who, when
User Management	Create, update role, deactivate	target_user, changes, admin_id
Alerts	Alert created, acknowledged	alert_type, call_id, severity

13.8 Implementation Checklist

- **Every DB query includes org_id filter** — No exceptions for tenant isolation
- **All DS API calls go through circuit breaker** — Never call DS APIs directly
- **PII auto-redaction before storage** — Apply regex patterns before MongoDB insert
- **TTL set on all session documents** — Based on org retention policy
- **Audit log all config changes** — Via API middleware
- **Frontend never calls DS APIs** — All AI data flows through Backend

13.9 Knowledge Retrieval Strategy Roadmap

The knowledge retrieval architecture follows a phased approach, evolving from parametric to non-parametric as scale increases.

PHASE	APPROACH	SCOPE	KEY CHARACTERISTICS
Phase 0-1	Parametric Partitions	Known clients (Telecom)	Manual bucket definition per client/industry, vector search for broad FAQs
Phase 2-3	Hybrid (Vector + Graph)	Multiple clients, same industry	Graph for precision scenarios, Cypher query builder for structured retrieval
Phase 4+	Bayesian Non-parametric	Multi-industry scale	IBP/CRP style auto-discovery, automatic node creation based on data patterns

Retrieval Precision Guidelines

DATA TYPE	RECOMMENDED RETRIEVAL	RATIONALE
Generic SOPs & Guidelines	Vector Search	Semantic similarity sufficient

Device Models / Handsets	Graph + Structured	Similar names cause vector mis-retrieval
Pricing / Plans / Packs	Graph + Structured	Tight semantic neighborhoods
Roaming / Regional Data	Graph + Structured	Country/region disambiguation critical
FAQs / Troubleshooting	Vector Search	Broad concepts, semantic search works

Industry-First Strategy

Telecom is the primary target industry. Deep solution for one industry enables 80-90% reuse for similar players. Generic multi-industry solutions appear to work in demos but fail silently on accuracy in production calls.

13.10 Technical Infrastructure Roadmap

ITEM	CURRENT	FUTURE	BENEFIT
Orchestration	Docker Compose / ECS	Kubernetes (EKS)	Auto-scaling, better orchestration
Infrastructure	Manual / CLI	Terraform IaC	Reproducible deployments
Vector Storage	MongoDB (via DS)	Pinecone / Weaviate	Better vector search at scale
Graph Storage	Not implemented	Neo4j / Neptune	Precision retrieval for similar items
API	REST only	REST + GraphQL	Flexible frontend queries

— End of Document —