

Data Transformation & LLM Enrichment

(Weaviate Vector Database for Semantic Historical Case Search)

Enhancing CognateAI with Intelligent Case Resolution Search on PC AI



Objective

Build scalable, AI-driven data pipelines on One AI PC AI Infra for intelligent case resolution



Platform

Weaviate Vector Database deployed on PC AI Infrastructure



Scope

Historical support cases with daily trickle feed

End-to-end transformation, PII cleansing, LLM enrichment, embedding generation, and Vector DB indexing



Outcome

Faster case resolution, PII compliance, and improved AI/analytics readiness across the enterprise

Agenda / Table of Contents

- | | |
|---|--|
| <ul style="list-style-type: none">1 Business Challenge (Problem Statement)2 Project Objectives & Success Criteria3 Solution Architecture Overview4 Architecture Deep Dive: Data Sources & Staging5 Architecture Deep Dive: Transformation, VectorDB, Consumption6 Data Model Overview: Weaviate Schema7 SFDC Field Mapping (Detailed)8 Algoleap Data Pipeline: 5-Step Overview9 Pipeline Steps 1–3: Extract, PII Removal, ChatHPE | <ul style="list-style-type: none">10 Pipeline Steps 4–5: Embeddings Generation & Weaviate Load11 Security: OIDC/Keycloak Authentication Flow12 Security: RBAC & Compliance Controls13 Embeddings Strategy: ChatHPE vs Nomic14 12-Week Timeline: Phase Overview15 Team Structure: Roles & Responsibilities (6–7 FTEs)16 Commercial17 Risk Mitigation & Q&A |
|---|--|

Business Challenge (Problem Statement)

Current State

Daily Challenges

-  Engineers spend extra minutes/day searching for similar historical cases
-  Current search is keyword-based only — misses semantically similar cases
-  Inconsistent solutions across teams with duplicated efforts
-  Tribal knowledge lost when engineers leave
 - no knowledge article exists

Business Impact

Quantified Impact

-  1,000+ engineers impacted globally (GRS, DE, Support)
-  Thousand of hours/year wasted on inefficient manual searches
-  Customer satisfaction impacted by slower resolution times
-  Difficult onboarding for new engineers due to knowledge gaps
 - 1.1 Difficulty correlating historical issues and resolutions
 - 1.2 High MTTR with manual cleaning and retries
 - 1.3 Limited AI-readiness across business units

Project Objectives & Success Criteria

Primary Goal

Enterprise Semantic Search

-  Build an enterprise-grade semantic search system for historical support cases
-  Index 1M+ SFDC cases spanning 1 year with a daily 2740 cases trickle feed or 914 cases every 6hrs
-  Enable engineers to find semantically similar cases in seconds
-  Deploy on PC AI (Kubernetes) with OIDC/Keycloak authentication

1.1 Build scalable, AI-driven data pipelines on One AI PC AI Infra for intelligent case resolution

1.2 End-to-end transformation, PII cleansing, LLM enrichment, embedding generation, and VectorDB indexing

Success Criteria

Measurable Metrics

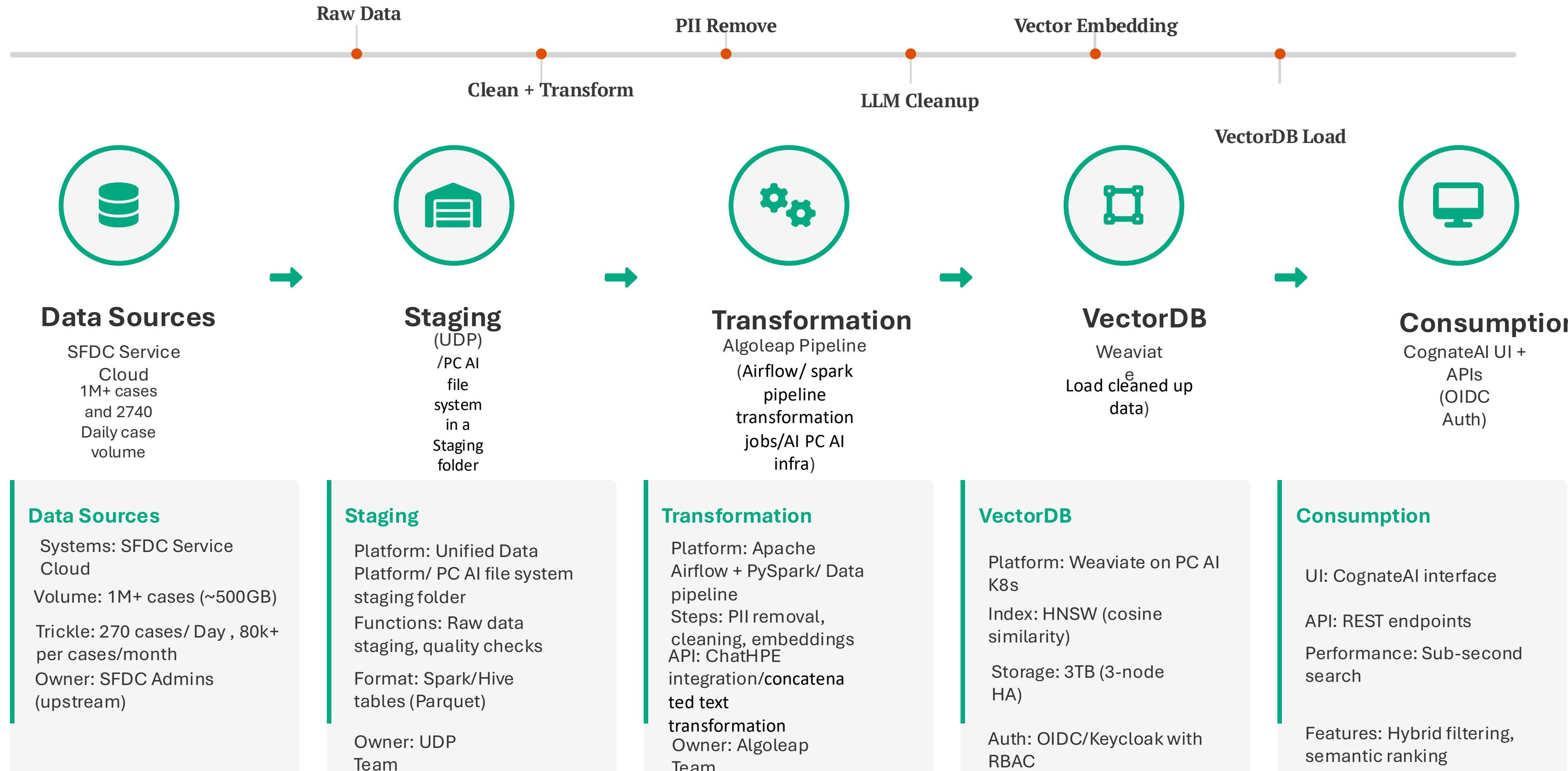
-  Accuracy: >85% precision@5 (validated by GRS team)
-  Performance: Sub-second query response, ≥ 300 cases/min processing
-  Security: Zero PII in loaded data, OIDC/RBAC authentication
-  Reliability: HA cluster, daily backups, observability

1.1 Idempotent jobs created unique Data sets

1.2 Un-Fragmented data pipelines with consistent auto cleansing processes enabled

Solution Architecture Overview

End-to-End Flow from Data Sources to Consumption



Architecture Deep Dive: Data Sources & Staging

Data Sources Layer

SFDC Service Cloud Cloud

-  1M+ initial cases spanning 1 year of historical historical data (~500GB)
-  70k-80K new cases/month via trickle feed feed for ongoing updates
-  Case data objects include tickets, customer info, info, resolutions, products
-  Metadata fields include status, priority, categories, timestamps, IDs, product number, product family etc

Staging Layer Layer

Staging folder on the Private Cloud AI

Cloud AI

-  SF Source pushed to Kafka, the UDP pulls the data into Hadoop/Vertica, again, UDP push the volume to the Private Cloud AI Staging folder
-  Data quality checks for completeness, schema validation, and consistency
-  Secure access controls with role-based permissions and encryption
-  Lineage tracking for full audit trail of data movement and transformations



Architecture Deep Dive: Transformation, VectorDB, Consumption

Transformation



Apache Airflow PySpark

Data Transformation Pipeline

- PII Removal: Deterministic rules to strip emails, phones, and names from all case data/ Compute to process the daily volume
- Text Normalization: Clean formatting, standardize units, reduce noise in technical descriptions, and Concatenation
- ChatHPE Summarization: Generate concise summaries to improve embedding quality
- Data Quality Checks: Schema validation, completeness checks, error handling

VectorDB



Weaviate HNSW

Vector Database Storage & Indexing

- HNSW Index: Hierarchical navigable small world algorithm for efficient vector search
- Cosine Similarity: Semantic matching based on vector angles, not keywords
- High Availability: 3+ pod cluster with coordinator and data nodes, auto-scaling
- Multi-tenancy: tenant_cognate isolation, replicas and shards for scalability

Consumption



CognateAI RBAC

Interface & Access Controls

- User Interface: CognateAI semantic search with embedded case resolutions
- REST API: Programmatic access with rate limiting, pagination, and versioning
- Hybrid Filters: Combine vector similarity search with metadata filtering
- RBAC & Audit: Role-based access control with comprehensive audit trails

Data Model Overview: Weaviate Schema

Weaviate Class Definition

Case Vectorized Class



Class Name: CaseVectorized



Tenant: tenant_cognate (multi-tenancy enabled)



Type: Document Vector Object

Vector Configuration



Model: ChatHPE text-embedding-3-large



Dimensions: 3,072



Distance Metric: Cosine similarity



Indexing: HNSW (Hierarchical Navigable Small World)

Core Properties

Property	Type	Indexed	Purpose
caseNumber	text	✓	Unique case ID (e.g., "5007T000002AbcD")
caselid	text	✓	SFDC internal ID
accountId	text	✓	Customer account reference
status	text	✓	Case status (New, In Progress, Closed)
priority	text	✓	Priority level (High, Medium, Low)
product	text	✓	Product category or line
createdDate	dateTime	✓	Case creation timestamp
closedDate	dateTime		Case resolution timestamp
title	text	✓	Case subject line (vectorized)
description	text		Full case description (vectorized)
resolutionSummary	text		Solution description (vectorized)

SFDC Field Mapping (Detailed)

Core Fields Mapping

SFDC Field	Weaviate Property	Type & Indexing
CaseNumber	caseNumber	text indexed
Id	caseld	text indexed
AccountId	accountId	text indexed
Status	status	text indexed
Priority	priority	text
Product__c	product	text
Category__c	category	text

Content & Date Fields

SFDC Field	Weaviate Property	Type & Indexing
CreatedDate	createdDate	dateTime
ClosedDate	closedDate	dateTime
Subject	title	text vectorized
Description	description	text vectorized
Resolution__c	resolutionSummary	text vectorized

PII Handling Protocol



All emails, phone numbers, and personal names stripped before vector embedding/Data would be labelled "HPE Confidential/PII" data must be encrypted



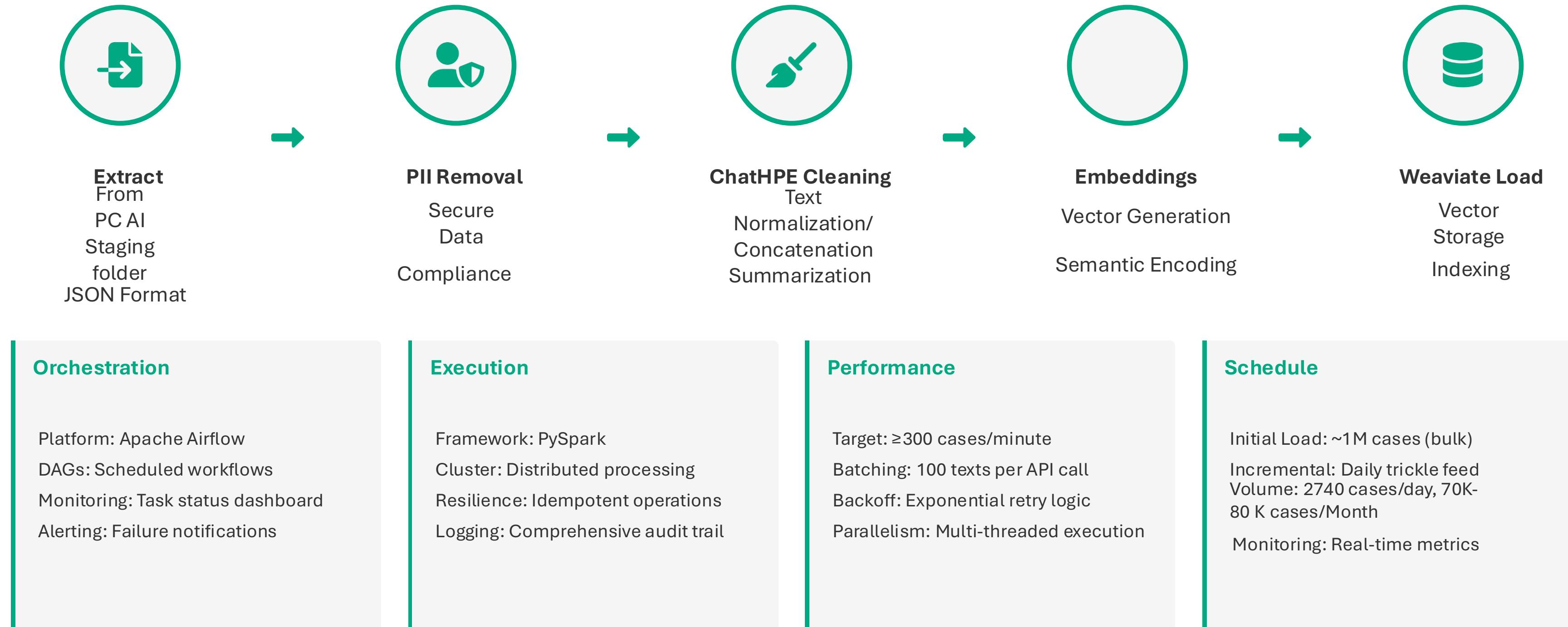
Audit logs track all PII removal actions with timestamp and pattern match



Deterministic rules applied consistently across all data fields

Algoleap Data Pipeline: 5-Step Overview

Comprehensive Data Transformation Process from Extract to Load



Pipeline Steps 1–3: Extract, PII Removal, ChatHPE

1 Extract from PC AI Staging folder

-  Pull from the PC AI staging folder .JSON type
-  Filter records by date range and business rules
-  Schema validation and data quality checks before processing
-  PySpark implementation

2 PII Removal

-  Remove emails, phone numbers, names, and other PII from text fields
-  Deterministic rules with regex patterns for consistent redaction
-  Generate audit logs for PII removal activity and compliance
-  Pattern-based redaction

3 ChatHPE Cleaning

-  Normalise text with standardized format, spelling, abbreviations, and concatenation
-  Generate concise summaries to improve embedding quality
-  Identify and extract key technical terms for better semantic matching
-  ChatHPE prompt template

Pipeline Steps 4–5: Embeddings Generation & Weaviate Load

4 - ChatHPE Embeddings Generation

Embedding Strategy

-  Batch processing of ~100 texts per API call to reduce overhead and improve throughput
-  Exponential backoff retry logic for API rate limits
-  Parallel processing with PySpark for 300+ cases/minute throughput
-  Fallback to Nomic embeddings for resilience and cost control

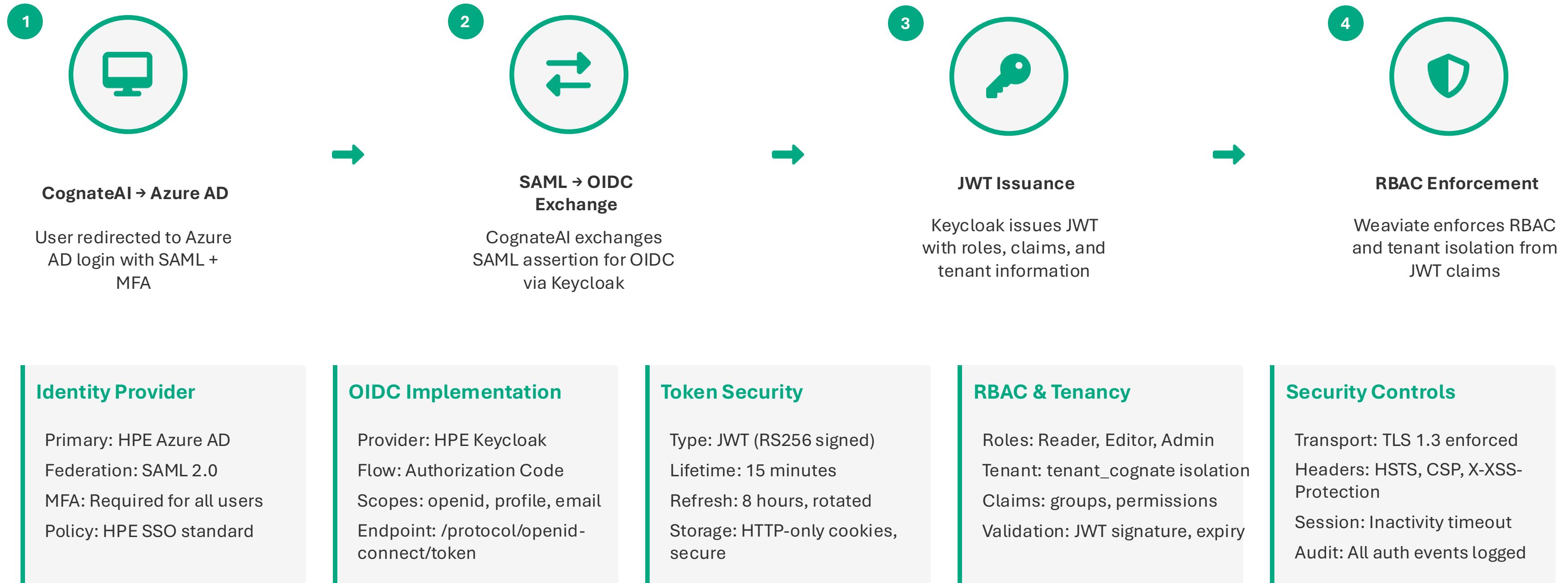
5 - Weaviate Load

Loading Strategy

-  Upserts with versioning to handle reprocessing and updates
-  Hybrid filters for combining vector similarity with metadata filtering
-  Real-time monitoring of throughput, errors, and index health
-  Idempotent batch operations with checksums and tracking

Security: OIDC/Keycloak Authentication Flow

Enterprise-Grade Authentication with OpenID Connect and Role-Based Access Control



Security: RBAC & Compliance Controls

Access Controls

Authentication & Authorization

-  Role-based access via JWT claims from Keycloak
-  tenant_cognate multi-tenant isolation for strict data separation
-  Least privilege access model with granular permissions
-  Secrets managed in secure vault with rotation policies

Compliance Measures

Data Governance

-  Zero PII post-transformation with deterministic removal
-  Structured data retention policies with automated enforcement
-  NetworkPolicies enforcing strict ingress/egress controls
-  Comprehensive audit trails with separation of duties

Embeddings Strategy: ChatHPE vs Nomic

Primary Model

ChatHPE text-embedding-3-large

-  3,072 dimensions — high-resolution vector representation
-  High semantic accuracy with multilingual support (100+ languages)
-  Target: >85% precision@5 (validated by GRS team)
-  Context window: 8,192 tokens (~6,000 words) for long-text support
-  Access via HPE ChatHPE API service with enterprise SLAs

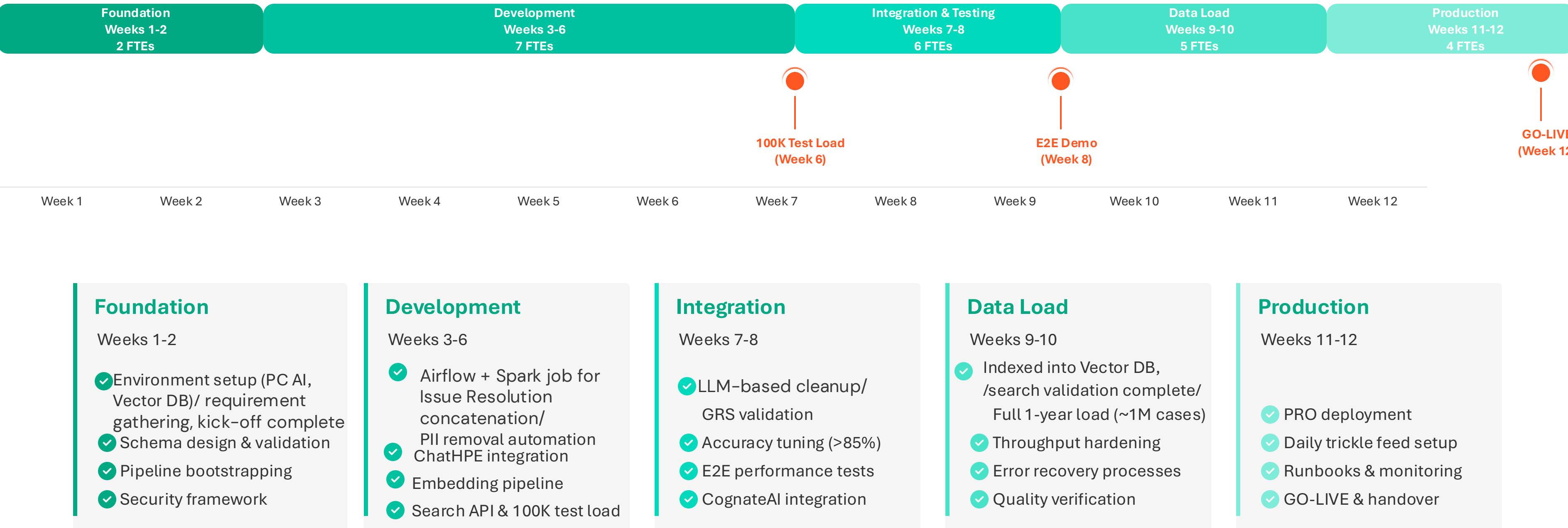
Fallback Strategy

Nomic Embeddings

-  Resilience for ChatHPE API downtime or rate limits
-  Cost control during bulk processing operations
-  Local deployment capability for higher throughput
-  Seamless switching with compatibility layer
-  Acceptable precision/recall trade-off (~80% precision@5)

12-Week Timeline: Phase Overview

Accelerated Delivery Plan with Strategic Team Ramp



Team Structure: Roles & Responsibilities (6–7 FTEs)

Key Roles & Responsibilities



Tech Lead / ML Engineer

- ✓ Architecture design (pipeline, schema, embedding strategy)
- ✓ Performance optimization & ChatHPE integration
- ✓ GRS validation coordination (Week 7-8)
LLM integration, embedding generation, prompt optimization



Python Developer

- ✓ ETL pipeline development & PII removal
- ✓ ChatHPE API integration & embedding generation
- ✓ Weaviate client/loader implementation



Data Engineer(2)

- ✓ UDP interfaces & staging pipeline
- ✓ Data quality checks & schema validation
- ✓ Performance testing & monitoring

Spark job development, optimization
DAG orchestration, data flow



DevOps/SRE

- ✓ Kubernetes, CI/CD pipeline setup
- ✓ IaC, observability & security integration
- ✓ Deployment automation & scaling



QA Engineer

- ✓ Functional, performance & data quality testing
- ✓ Test & validation framework
- ✓ PII detection & security verification



Program Manager

- ✓ Project planning & risk management
- ✓ Stakeholder coordination & reporting
- ✓ Resource management & timeline tracking

Resource Utilization

Resource Loading

Role	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Engagement	Notes
Tech Lead /ML Engineer	Full	Full	Full	Full	Full	Full	Full	Full	Full	Full	Full	Full	Full	
Data Engineer #1			Full											
Data Engineer #2			Full											
Python Developer			Full											
DevOps Engineer			Partial											
QA Engineer						Partial								
Project Manager	Full	Full	Full	Full	Full	Full	Full	Full	Full	Full	Full	Full	Full	
Legend:														
Full Load (100%)														
Partial Load (50%)														
No Engagement														

Hours Settings

Hours Settings														
Full	40													
Partial	20													
Role	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Total Hours	Engagement
Tech Lead /ML Engineer	40	40	40	40	40	40	40	40	40	40	40	40	480	Full
Data Engineer #1	0	0	40	40	40	40	40	40	40	40	0	360	Full	75.00%
Data Engineer #2	0	0	40	40	40	40	40	40	40	40	0	360	Full	75.00%
Python Developer	0	0	40	40	40	40	40	40	40	40	0	360	Full	75.00%
DevOps Engineer	0	0	20	20	20	20	20	20	20	20	20	200	Partial	41.67%
QA Engineer	0	0	0	0	0	20	20	20	20	20	20	140	Partial	29.17%
Project Manager	40	40	40	40	40	40	40	40	40	40	40	480	Full	100.00%
	0	0	0	0	0	0	0	0	0	0	0	0		
	0	0	0	0	0	0	0	0	0	0	0	0		
Legend:	0	0	0	0	0	0	0	0	0	0	0	0		
Full Load (100%)	0	0	0	0	0	0	0	0	0	0	0	0		
Partial Load (50%)	0	0	0	0	0	0	0	0	0	0	0	0		
No Engagement	0	0	0	0	0	0	0	0	0	0	0	0		
Total per Week	80	80	220	220	220	240	240	240	240	240	240	120		
Legend:														
Full Hours/Week	40													
Partial Hours/Week	20													

Commercial

Role	# Resources	Cost Per Hr. / Per Resource	Cost Calculation	Cost
Program Manager	1	\$35	$1 * \$35 * 480\text{hrs}$	\$16,800
Data Engineers	2	\$35	$2 * \$35 * 360\text{hrs}$	\$25,200
Tech Lead /ML Engineer	1	\$40	$1 * \$40 * 480\text{hrs}$	\$19,200
Python Developer	1	\$30	$1 * \$30 * 360\text{hrs}$	\$10,800
DevOps / SRE Engineer	1	\$30	$1 * \$30 * 200\text{hrs}$	\$6000
QA Engineer	1	\$25	$1 * \$25 * 140\text{hrs}$	\$3500

Total Base Cost: \$81,500

Final Cost with Overheads

- ❖ Base Cost: **\$81,500**
- ❖ PM/Admin Overhead (5%): **\$4075**
- ❖ Contingency (10%): **\$8150**
- ❖ *******Total Estimated Project Cost: ≈ \$93,725 USD*******

Key Assumptions

- Team Size: 7 members | Duration: 12 weeks (480 hrs/FTE).
- 40 hrs/week assumed; no overtime
- USD rates, exclude currency fluctuations
- Full-time dedicated roles
- 10% contingency + 5% PM overhead
- Cloud/tool costs excluded
- Client inputs/approvals are timely

Dependencies/Disclaimer/Delay Risks

Dependencies	Standard Disclaimers	Common Delay Risks
<ul style="list-style-type: none">1. Environment readiness before start1. Data & API availability2. Access & credentials provisioned1. Stakeholder availability2. Tool licenses funded by the client3. Security & compliance approvals4. Change management adherence5. Infrastructure support6. Third-party API integrations	<ul style="list-style-type: none">1. Indicative estimate; actuals may vary2. Out-of-scope items excluded3. Market-dependent rate variability4. Fixed 40-hour week5. Infra/tools excluded6. Timely client dependencies7. Contingency covers minor variation8. Confidential use only9. Validity: 30–60 days10. Email attachment Data is out of scope	<ul style="list-style-type: none">1. Access delays (idle time)2. Data readiness issues3. Requirement clarity delays4. Security review lag5. Frequent change requests6. Resource turnover7. Approval delays8. External API/vendor outages

Risk Mitigation & Q&A

1 ChatHPE API Issues

High latency, rate limits, or API downtime during critical data load

Mitigation Strategy

Batch 100 texts per API call (reduces API overhead 100x)

Implement retry logic with exponential backoff (2s, 4s, 8s)

Fallback to Nomic embeddings when ChatHPE unavailable

2 Stakeholder Alignment

Conflicting priorities or shifting requirements impact timeline

Mitigation Strategy

Weekly steering committee with key stakeholders

Clear acceptance criteria for each phase gate

Regular demos to validate functionality meets expectations

Documented scope management process

3 PII Leakage

Sensitive customer/engineer data exposed in vector database

Mitigation Strategy

Deterministic PII scrubbing with pattern matching

Regular audit logs and sampling verification

Red-team testing for edge case PII detection

4 Performance & Scale Issues

Search latency increases or throughput bottlenecks at scale

Mitigation Strategy

HNSW parameter tuning (ef, efConstruction) for optimal performance

Implement sharding and replication for horizontal scaling

K8s autoscaling based on CPU/memory metrics

Regular load testing with production-level volumes

5 Data Quality Variance

Inconsistent case formats, missing fields, schema changes

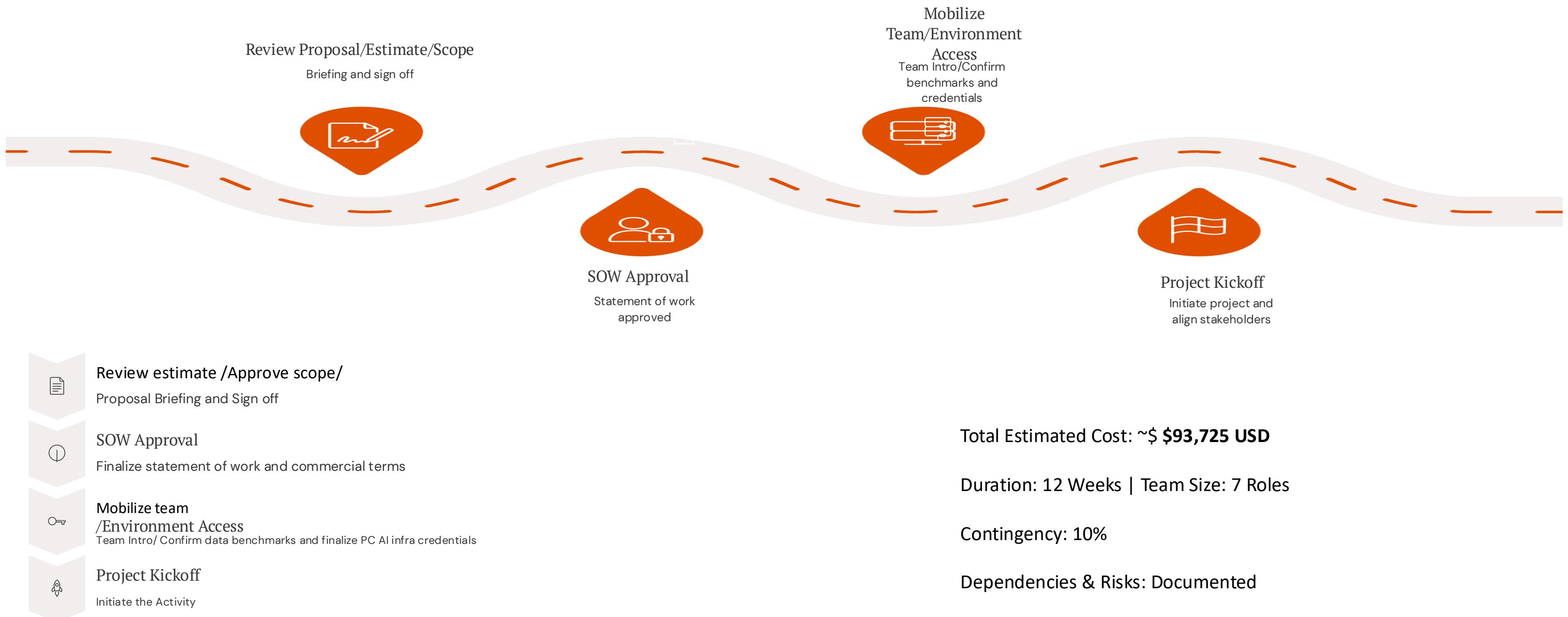
Mitigation Strategy

DQ checks in UDP staging with alerting

Schema validation pre-transform with clear error handling

Field mapping documentation with validation rules

Executive summary & Next steps



Total Duration: 12 weeks from project kick-off to production validation, with optional ongoing managed support

Multi-Table Data Architecture

					
Case 21 fields Subject, Description, Error Codes, Status, Priority, Resolution	Task 2 fields Troubleshooting steps, Plan of Action	WorkOrder 3 fields Field engineer notes, Onsite actions, Parts replaced	CaseComments 1 field Engineer comments thread	WorkOrderFeed 2 fields Feed updates, Service notes	EmailMessage 2 fields Email subject, Email body

Processing Pipeline: 44 Fields → 1 Vector



HTML Cleanup

Convert rich HTML to structured plain text



PII Removal

6-stage context-aware redaction process



Text Concatenation

Build structured sections from all 44 fields



Smart Truncation

≤30k chars while preserving key context

One Vector Per Case (3,072 dimensions)

ChatHPE text-embedding-3-large model



Full case narrative for complete context search



98% cost savings vs. per-field embeddings

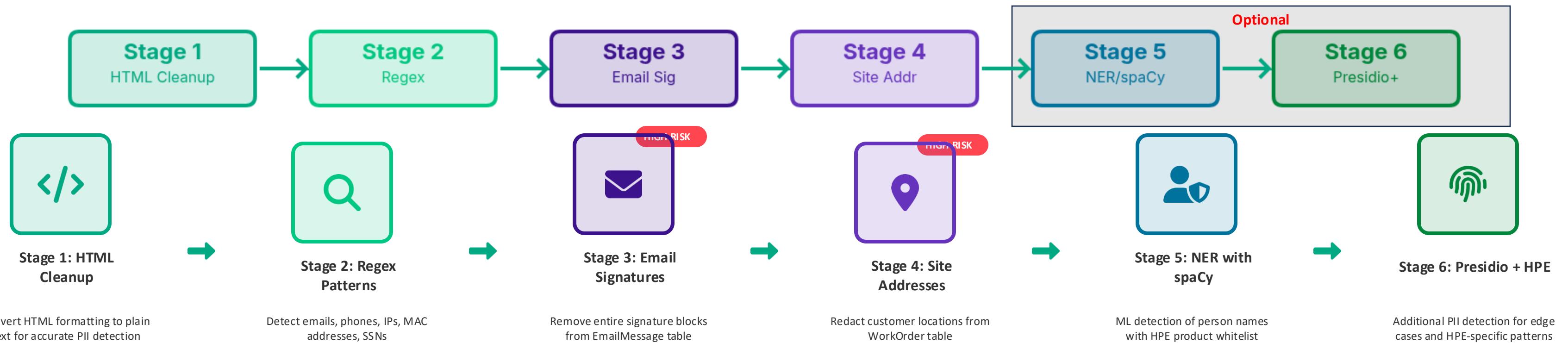


Faster queries with single vector lookup



Resilient to missing child tables/records

PII Removal Deep Dive



Before/After PII Redaction Examples

HTML Cleanup

Before

```
<div>Customer email:<a href="mailto:john.smith@acme.com">john.smith@acme.com</a></div>
```

After

```
Customer email: john.smith@acme.com
```

Regex Patterns

Before

```
Please call me at +1-555-123-4567 or email john.smith@acme.com
```

After

```
Please call me at [PHONE_REDACTED] or email [EMAIL_REDACTED]
```

Email Signatures

Before

```
I'll check on this.  
--  
John Smith  
Acme Corp  
+1-555-123-4567
```

After

```
I'll check on this.  
[SIGNATURE_BLOCK_REMOVED]
```

Site Addresses

Before

```
Server installation at 123 Main St, Floor 5, New York, NY 10001
```

After

```
Server installation at [LOCATION_REDACTED]
```

Hybrid Search Strategy (Semantic + Keyword + Filters)

Components



Vector (semantic): Understands meaning, not just words

Example: "DIMM error" matches "memory failure" even though words differ



BM25 (keyword): Finds exact term matches

Example: Error codes "218004", product numbers "867055-B21"



Metadata Filters: Narrows results by structured data

Examples: Product family, priority, date range, status

Adaptive Fallback Strategy

Stage 1: Product-Specific Search

Exact product match + semantic relevance ($\alpha=0.75$)

Stage 2: Product Family Search

If <5 results: Broaden to product family

Stage 3: Open Search

If still <5 results: Remove product filter, adjust $\alpha=0.6$

Real-World Example

Query: "server memory error causing crash during boot"

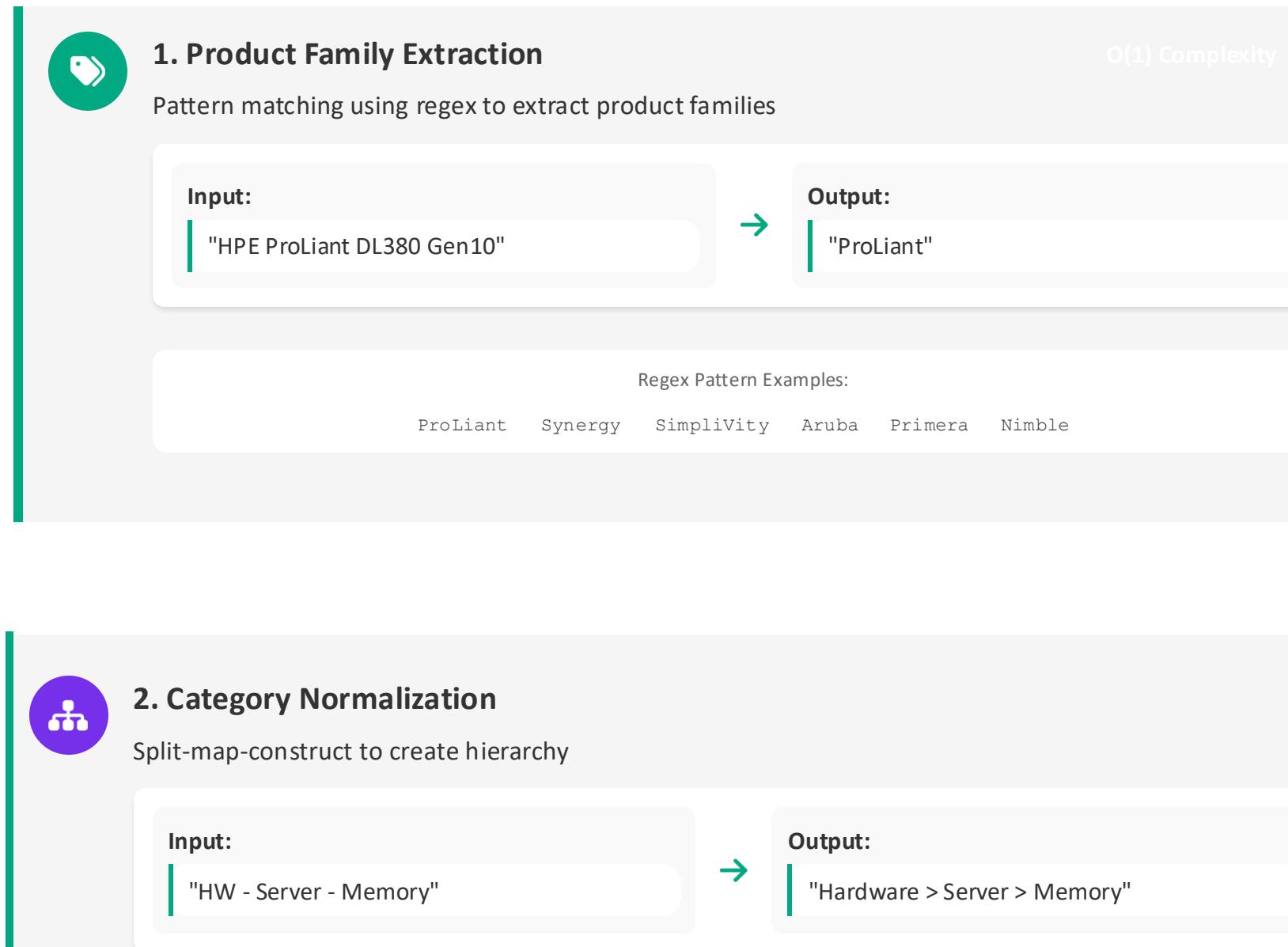
Strategy	Top Result	Semantic/Keyword Match
Discovery	Case #4592: "System crash after DIMM replacement - POST error 201"	High semantic relevance (concept match) despite different words
Balanced	Case #3201: "Memory error code 3020 causing server boot failure"	Good balance of semantic meaning and keyword matches
Precise	Case #5510: "Server boot error memory crash log attached"	More exact keyword matches but potentially less conceptual relevance

Scoring Impact: As alpha decreases, results favor exact keyword matches over conceptual similarities

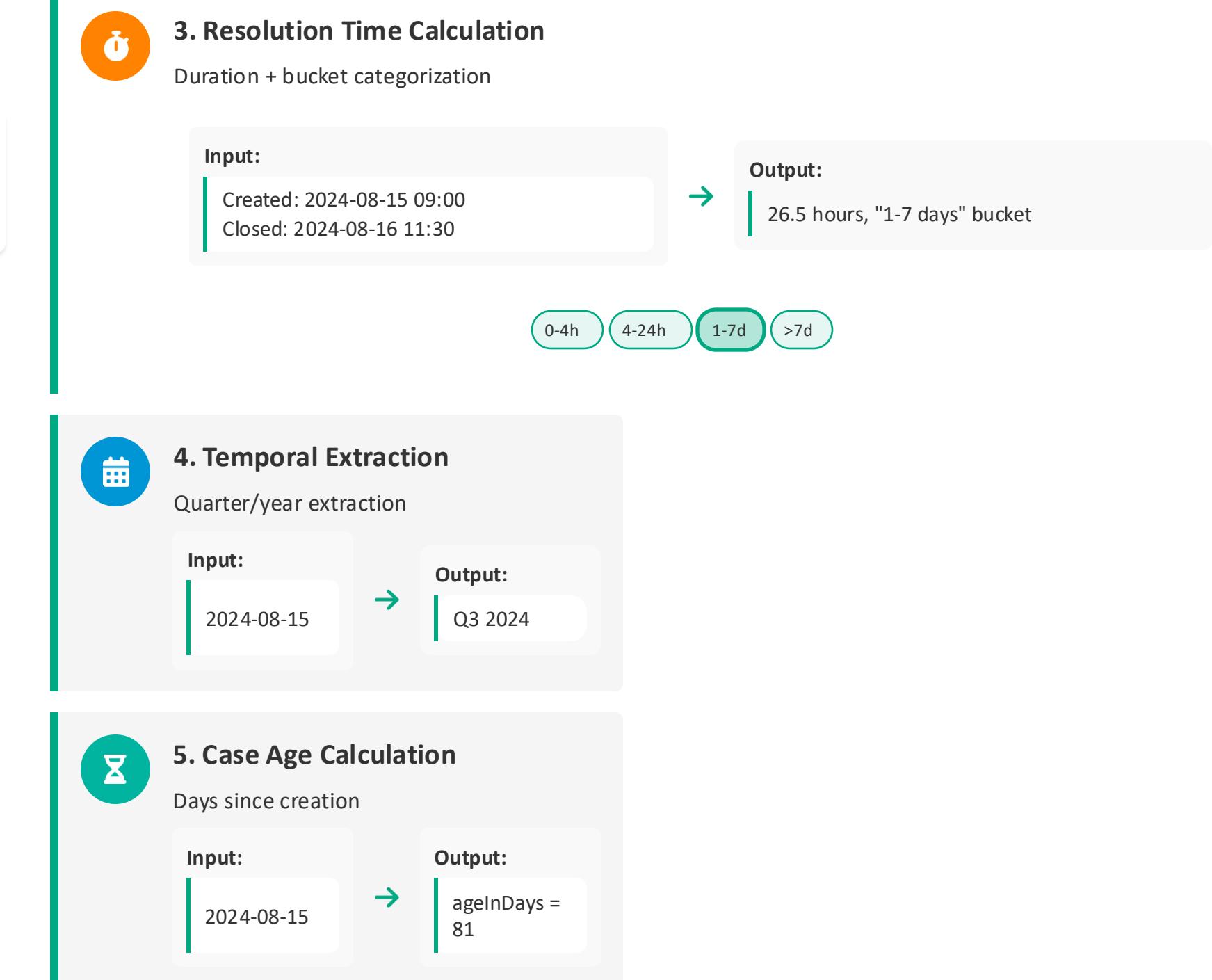
Metadata Creation

Specific approaches needed to create the metadata filters from raw SFDC data

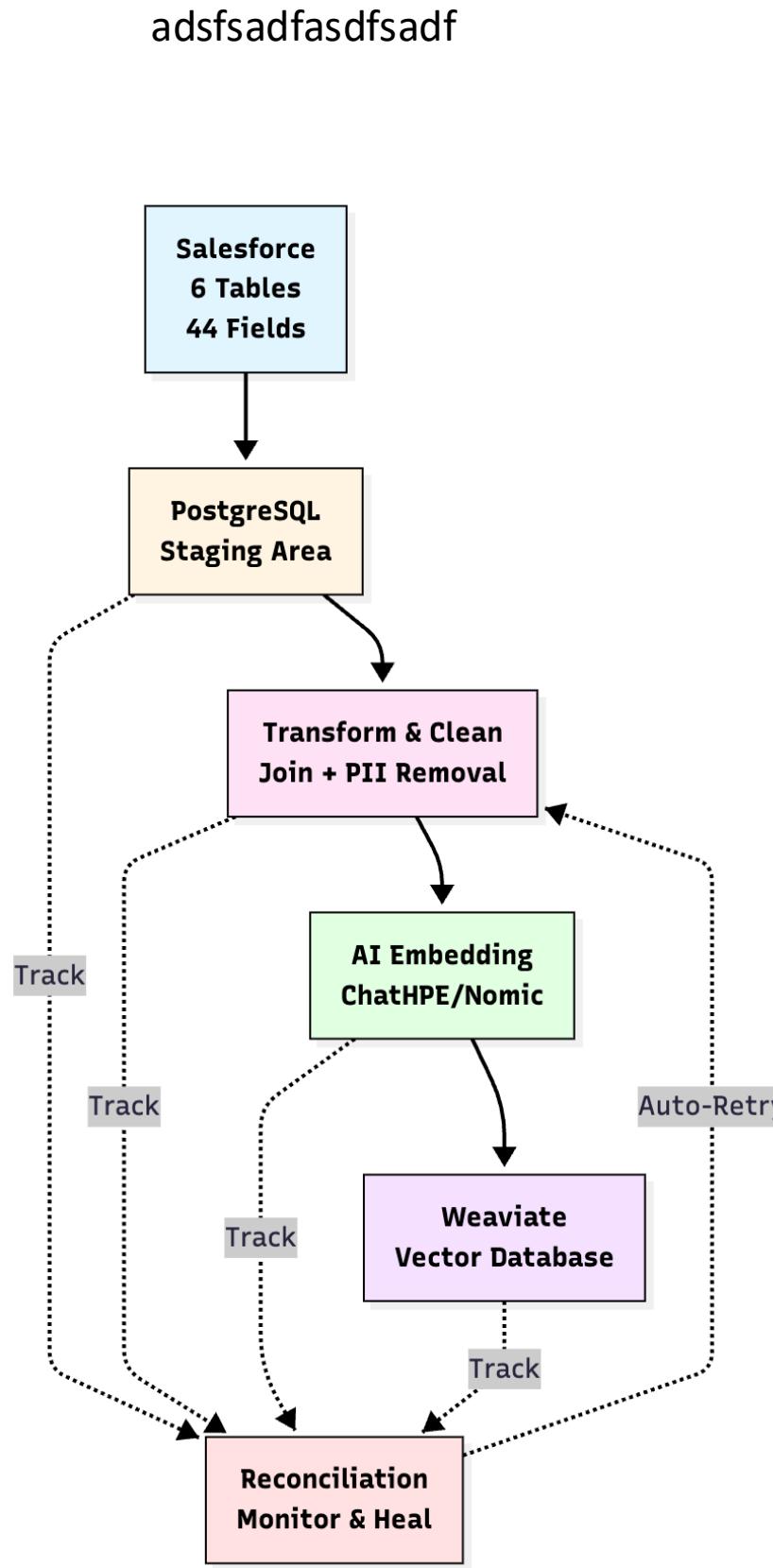
Extraction and Normalization



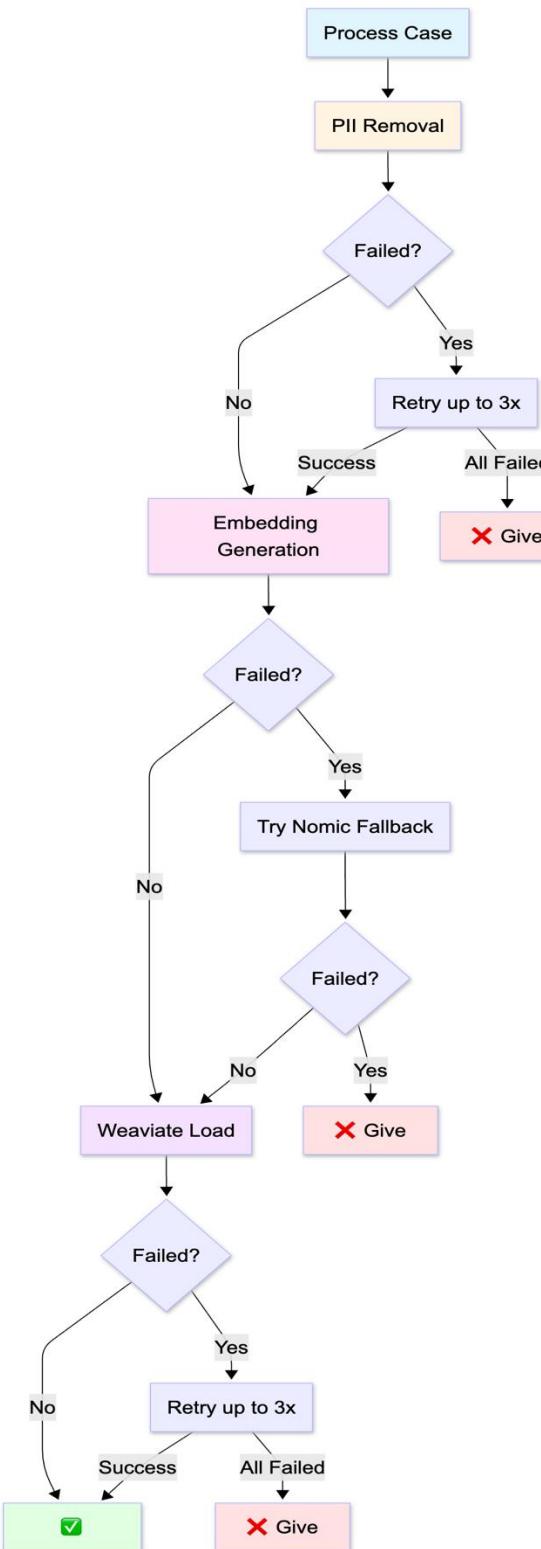
Time-Based



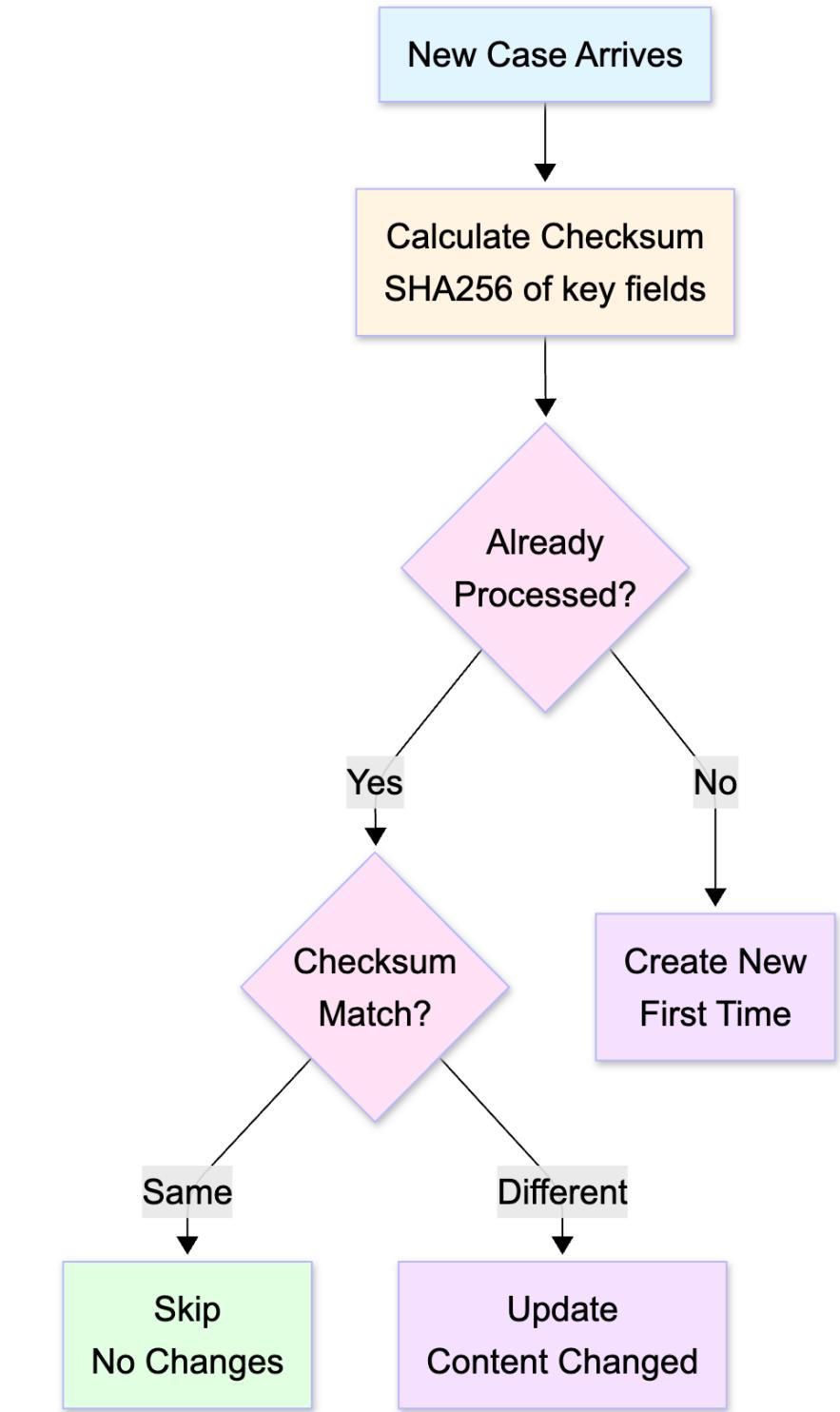
Reconciliation & Data Integrity



adsfsadfasdfsadf



adsfsadfasdfsadf



Thank You