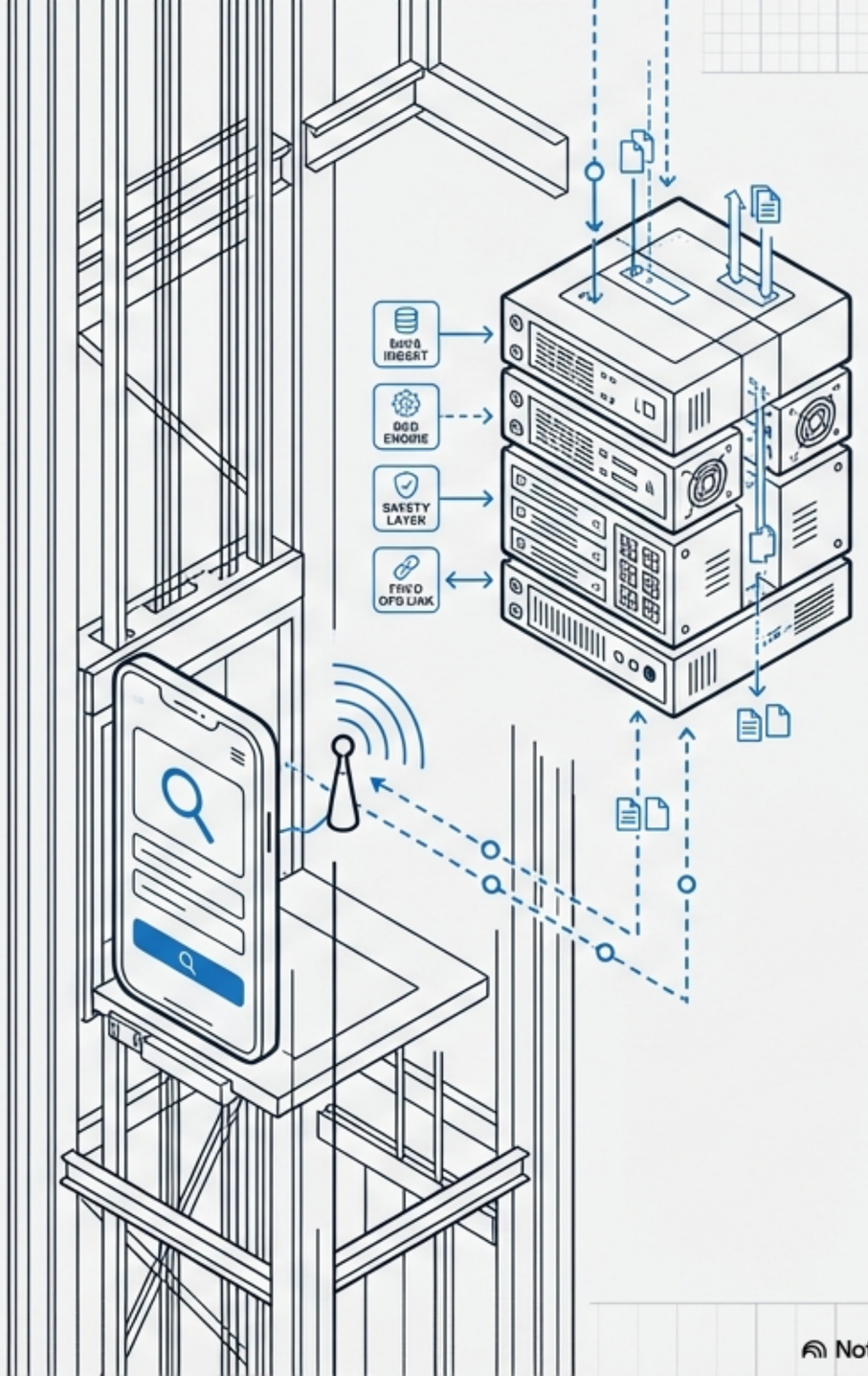
INTERNAL TECHNICAL REVIEW

Enterprise AI Architecture: Technical Implementation Framework

A Safety-First, RAG-Powered Assistant for Field Operations

Architecture Validation & Stack Selection

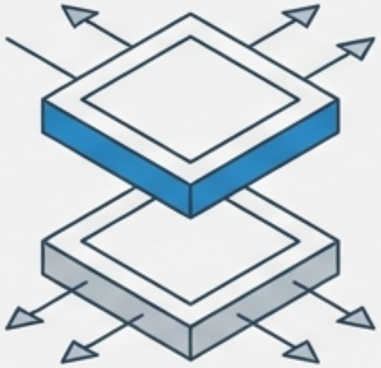


Operational Goals & Strategic Architecture

Core Objective

Reduce mean-time-to-repair while maintaining 100% safety compliance through a ‘Human-in-the-Loop’ design.

- Target: **≥90%** Confidence for Automation
- **<1s** Latency



Decoupled Architecture

Frontend/Backend separation allows independent scaling and future mobile migration without refactoring the core logic.

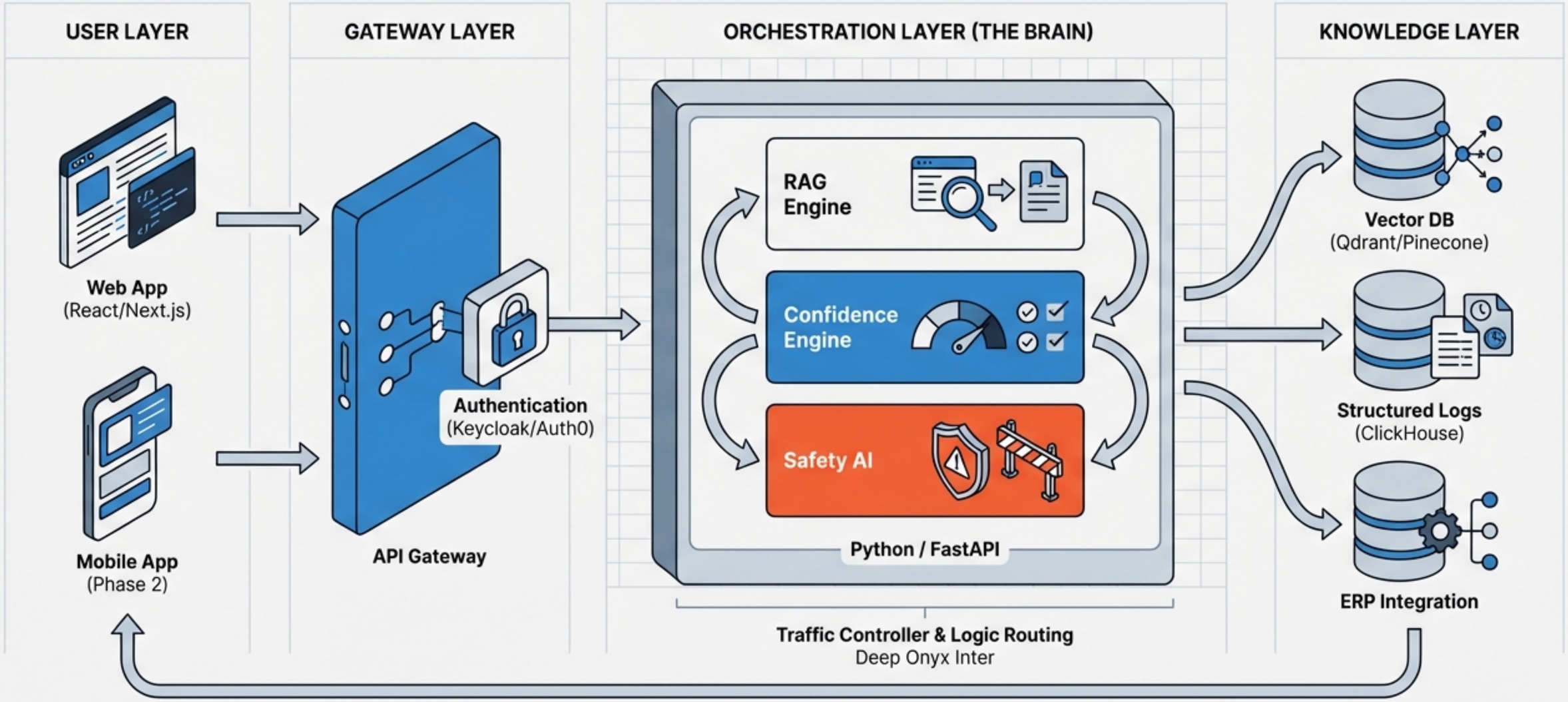
Verifiable Intelligence

A RAG (Retrieval-Augmented Generation) engine grounded strictly in technical manuals, governed by a multi-vector Confidence Engine.

Non-Negotiable Safety

An always-on 'Sentinel' layer that bypasses AI entirely for emergency keywords (e.g., 'Entrapment', 'Fire') to force human escalation.

High-Level System Architecture



Frontend & User Experience Layer

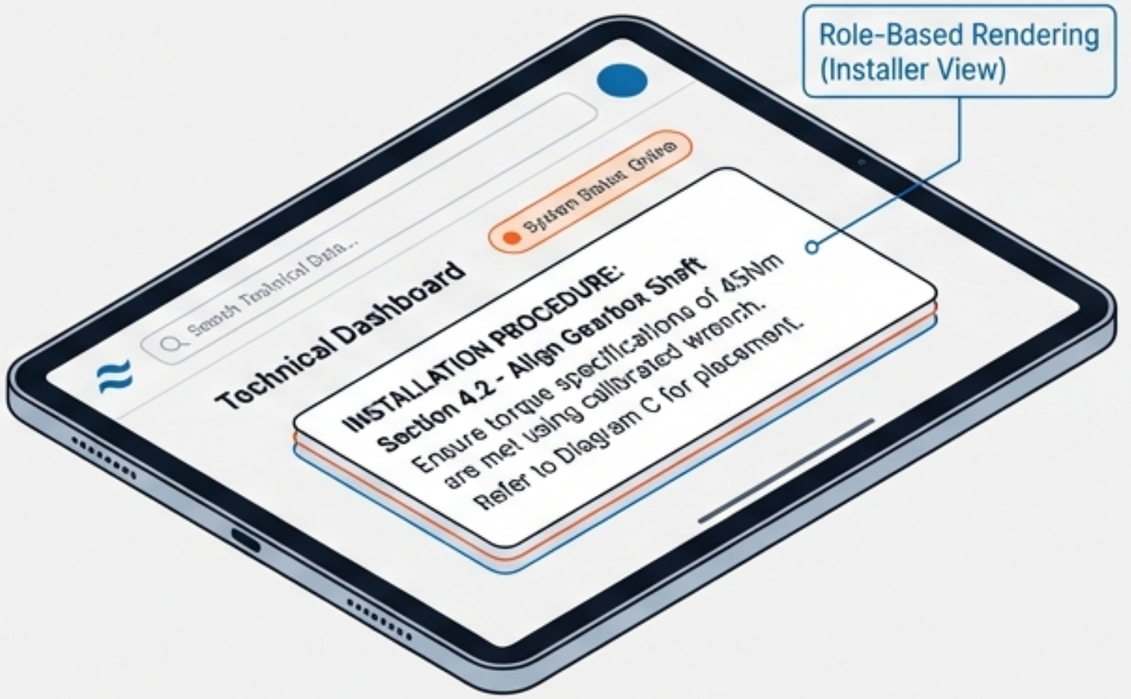
Focus: Performance, SEO, and Field Usability

Primary Stack (Web App)

- Framework: React.js + Next.js (Selected for SSR performance)
- UI Library: Tailwind CSS (Fast, consistent enterprise UI)
- Localization: i18n (react-i18next) for English, German, Greek

Secondary Stack (Mobile App - Phase 2)

- Framework: React Native
- Key Capability: Offline Caching (Critical for shaft work)
- Key Capability: Camera Access (Image Recognition)



Authentication & Access Control Strategy

Technology:
Keycloak (Self-hosted for sovereignty) or Auth0.

Protocol:
OAuth2 / OpenID Connect.

Compliance:
Full GDPR audit logging.

RBAC Matrix

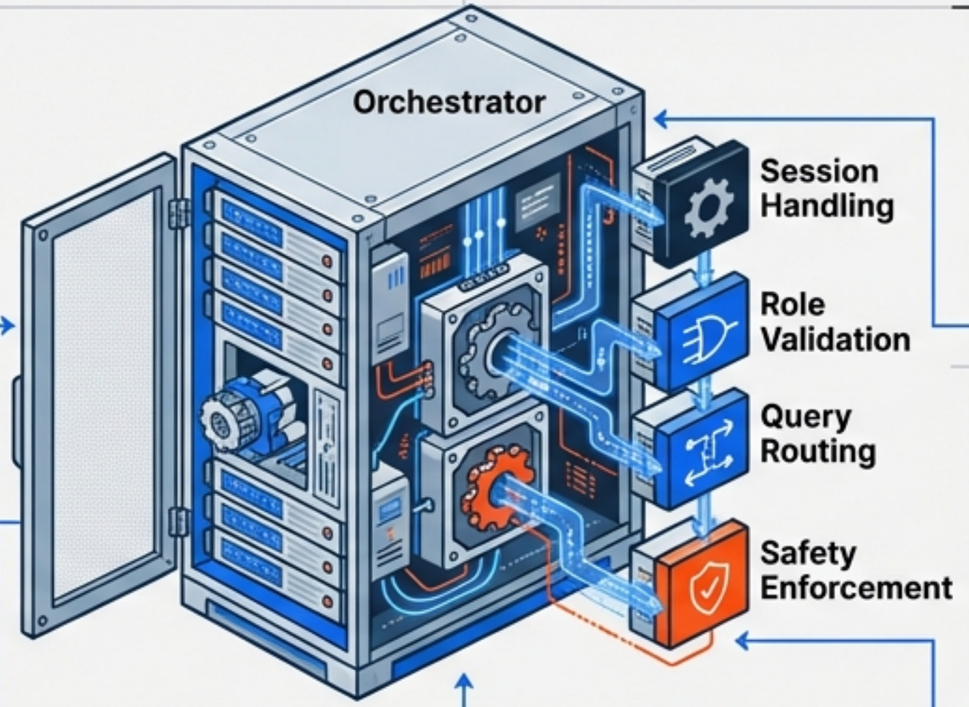
Role Name	System Code	Permissions Scope
Installer	ROLE_FIELD	Read-only access to Manuals & RAG Interface.
Employee	ROLE_INTERNAL	Internal Data Access.
Senior Engineer	ROLE_POWER	Advanced Diagnostics & Override Capabilities.
Admin	ROLE_ADMIN	System Config & User Management.
Management	ROLE_AUDIT	Read-only Access to KPIs & Logs.

The Orchestration Layer: System Core

Technology: Python + FastAPI

Architectural Justification:
Performance

Ultra-fast (<1s) response time essential for field diagnostics.



Architectural Justification:
Concurrency

Native Async support handles multiple RAG queries and ERP calls simultaneously.

Architectural Justification:
Ecosystem

Native integration with AI/ML pipelines (PyTorch, LangChain).

RAG Engine & Knowledge Retrieval Pipeline

Ingestion

Unstructured.io Processing

Chunking

Page-level Chunking & Versioning

Vector Storage

Cloud: Pinecone

On-Prem: Qdrant (GDPR)

Generation

LLM: GPT-4.1 (Cloud)
LLaMA 3.1 (On-Prem)

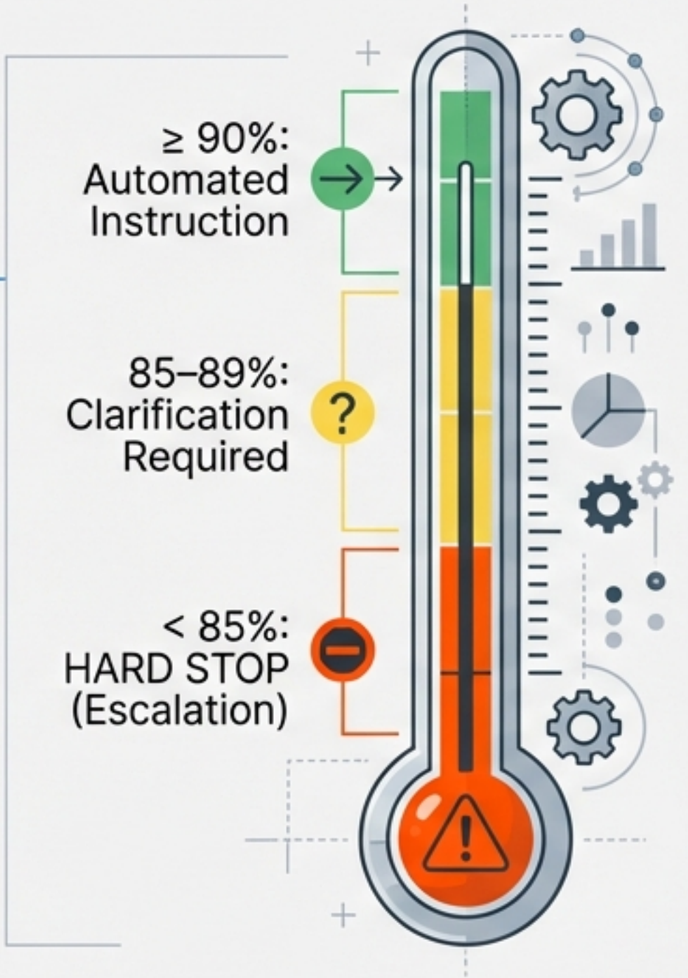
Strategy: Prioritize Ground Truth Manuals over Generic Data

The Confidence Scoring Engine

Mathematical Guardrails Against Hallucination

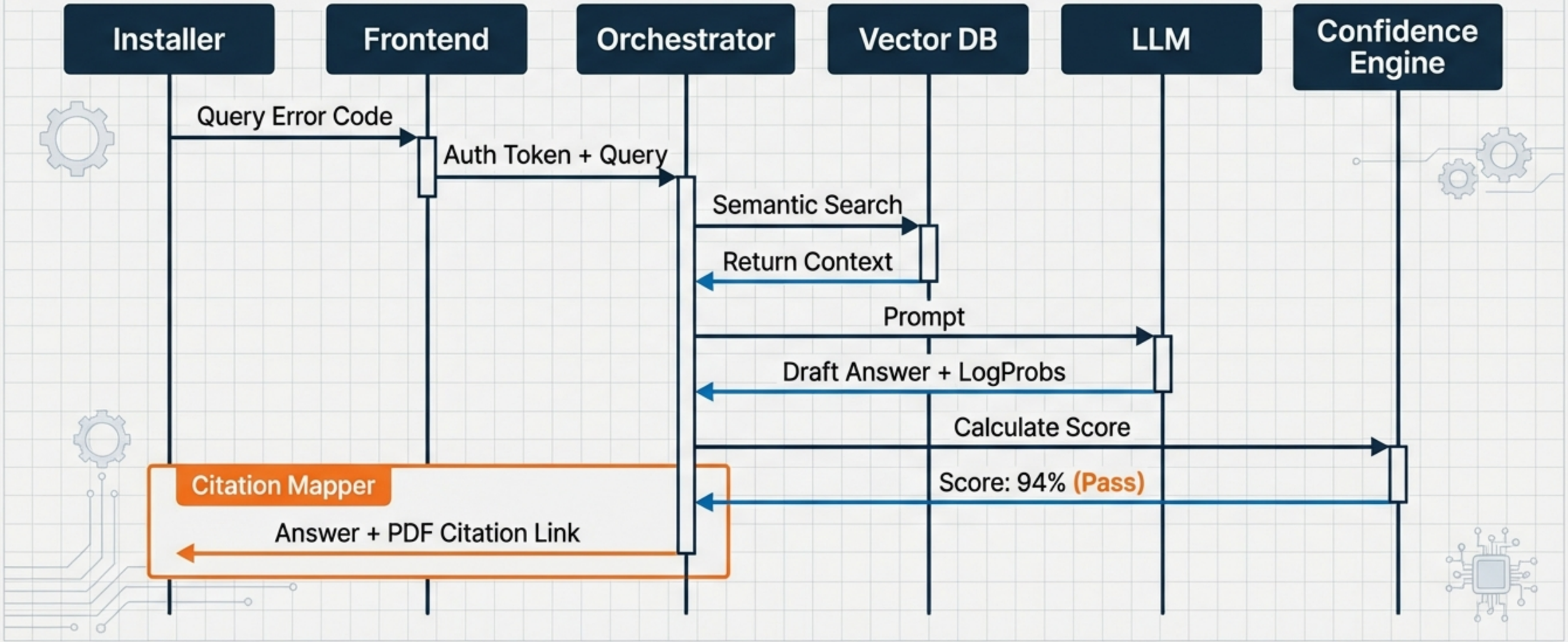
$$\text{Confidence} = (0.4 \times \text{Semantic Similarity}) + (0.4 \times \text{Source Reliability}) + (0.2 \times \text{Model Probability})$$

Semantic Similarity:	Cosine score from Vector DB
Source Reliability:	Metadata weight (Gold Standard PDF > Generic FAQ)
Model Probability:	Token entropy / logprobs

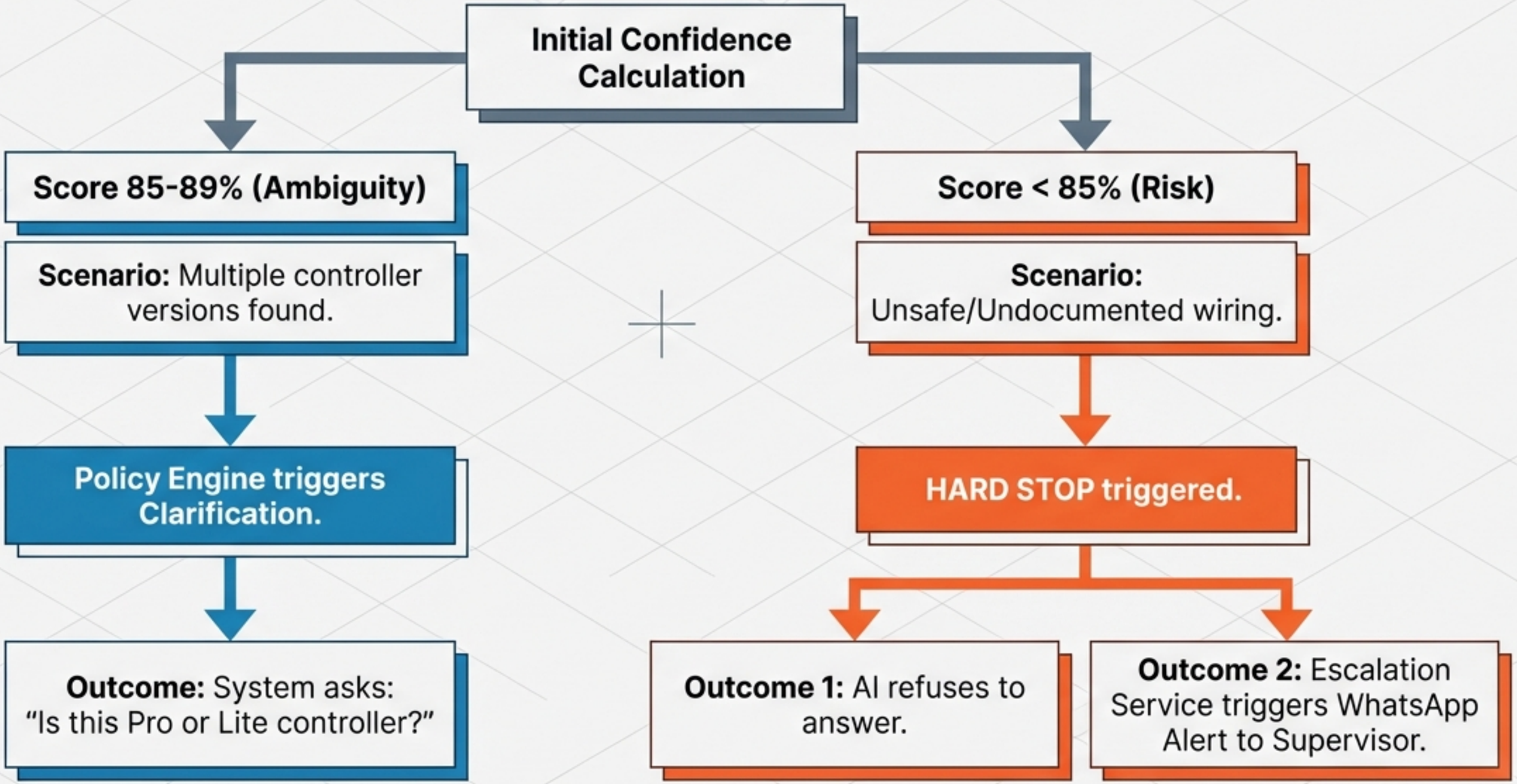


Sequence I: High Confidence Query ('The Happy Path')

Scenario: Installer asks about error code; System matches with ≥90% confidence.

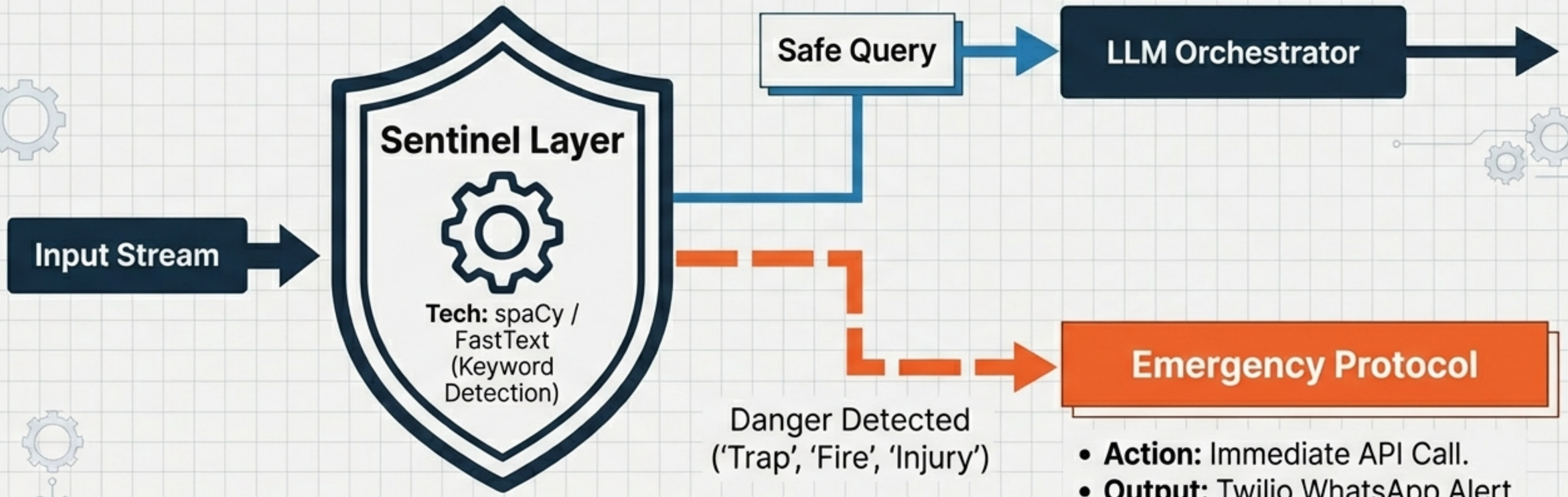


Sequence II: Handling Uncertainty & Hard Stops



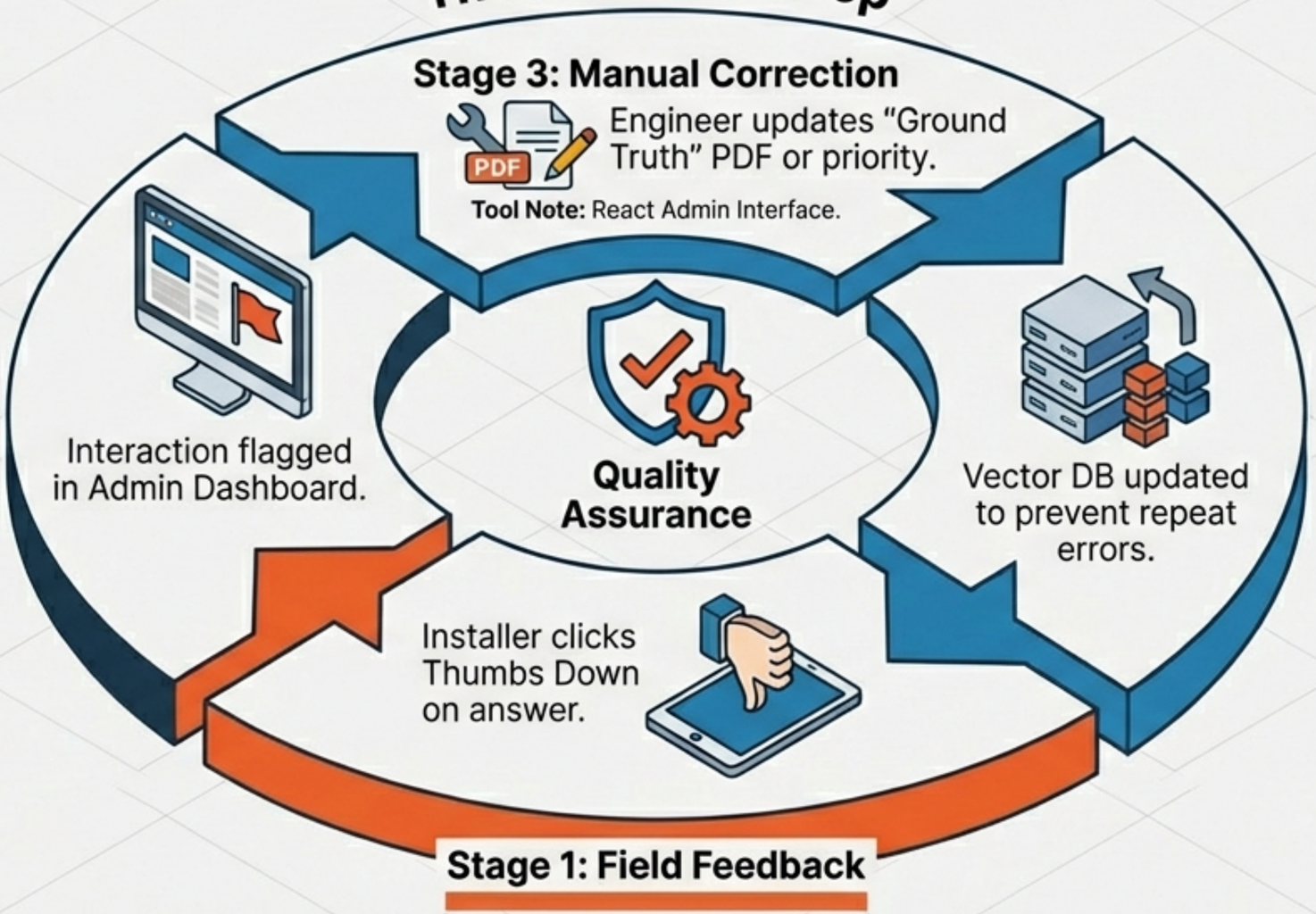
Safety & Emergency AI Layer

The ‘Always-On’ Sentinel (Runs Before LLM)



Knowledge Management & Continuous Improvement

The Feedback Loop

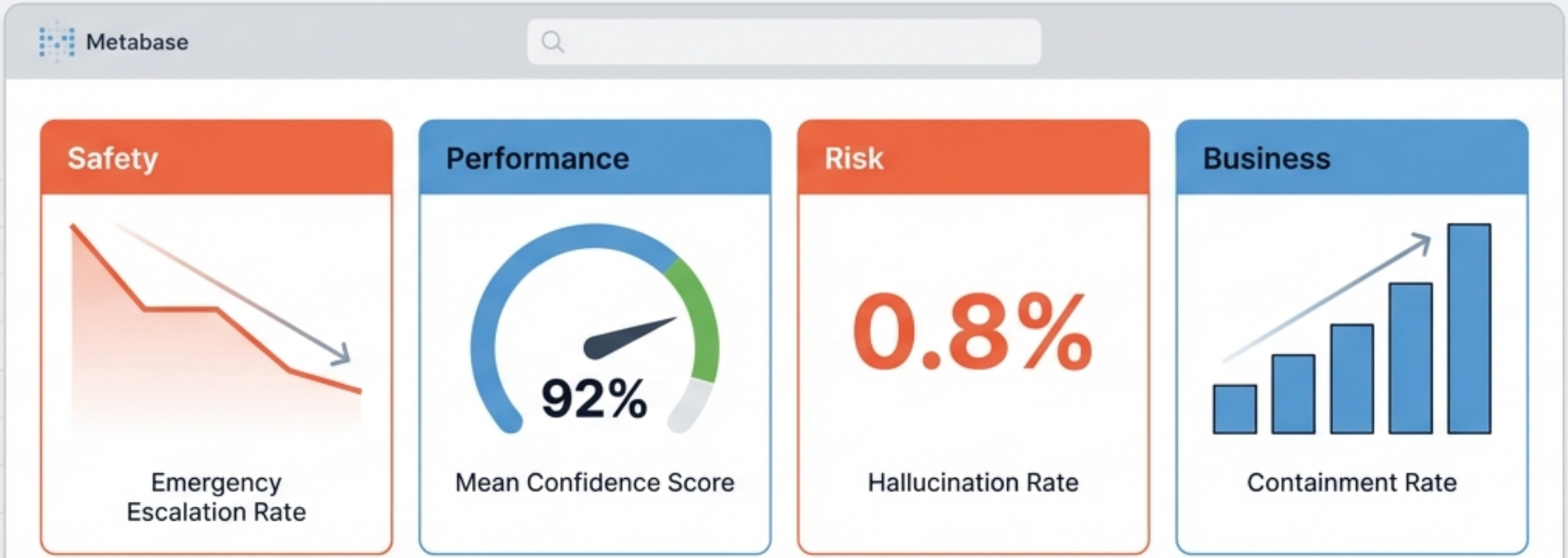


Analytics & KPI Engine

PostgreSQL
(Structured Data)

ClickHouse
(High-Volume Logs)

Redis
(Real-time Counters)

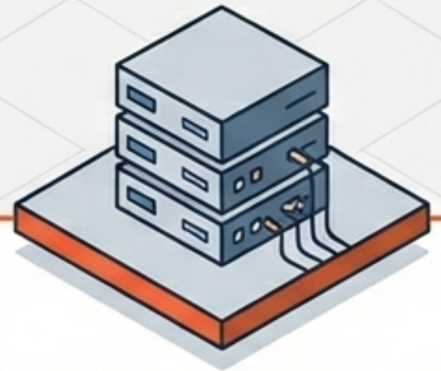


Implementation Roadmap: Cloud to On-Prem

Cloud Deployment (Immediate)

- AWS/Azure Kubernetes (EKS/AKS)
- OpenAI GPT-4.1
- Pinecone Vector DB

Goal: Quick ROI & Safety Validation.



On-Prem Migration (Data Sovereignty)

- Local Kubernetes (GPU Servers)
- LLaMA 3.1
- Local Qdrant

Goal: Full Control & GDPR Max-Compliance.

Visual Intelligence (Future)

- YOLOv8 / Detectron2 Integration

Goal: Automated Controller ID & Error Recognition.

Technical Stack Summary & Next Steps

Frontend	React + Next.js (Web), React Native (Mobile)
Backend	Python + FastAPI
Intelligence	GPT-4.1 (Cloud) / LLaMA 3 (On-Prem)
Vector DB	Qdrant / Pinecone
Security	Keycloak / Auth0
Data & Logs	PostgreSQL + Redis + ClickHouse
Safety	spaCy + Twilio WhatsApp
Hosting	AWS / Azure EKS

Recommendation: Proceed immediately with Phase 1 (Cloud) to capture safety data and validate the Confidence Engine before on-prem migration.