ChatGPT

ISA-CoreDev-AI – Autonomous Evolution Planning (Prefilled Prompt + Evaluation Output)

Repository Overview

The Intelligent Standards Assistant (ISA) is a Next.js 15.2 project that orchestrates Genkit AI flows to help GS1 stakeholders analyze and manage standards. The roadmap identifies ISA as a key strategic asset for handling complex GS1 standards and aiding compliance. Core implementation uses React 18 and Tailwind CSS on Firebase App Hosting, with Genkit and Google's Gemini models for AI logic.

Setup runs through `scripts/setup.sh`, which installs dependencies and builds the project. `.env.example` lists required variables: `GOOGLE_API_KEY`, `JIRA_WEBHOOK_SECRET`, `GCP_SA_KEY`, `GROUP_MAPPING`.

Structured logging is handled via the Pino library. The CI workflow runs on push events, checking lint, type correctness, and successful builds. Firebase emulators are defined in `firebase.json`. IAM automation is managed via Terraform modules and `iam_ci_cd.yml`, supporting `terraform plan`/`apply` with approval gates.

Project Status

- **Phase 1** (Core functionality, CI, IAM, and logging) is complete.
- **Phase 2** is underway, focusing on:
- Implementing vector search
- Building a knowledge graph in TypeDB
- Developing advanced ETL and flow logic
- Expanding test/evaluation coverage

Code quality is solid; type and lint checks run automatically. Logging and secrets handling are structured but need production hardening. Existing AI flows are conceptual but structured.

ISA-CoreDev-AI: Evaluation & Self-Evolving System Design Prompt

Improvement 1 – Three-Tiered Evaluation Strategy

A. Autonomous Reinterpretation - **Function**: Gatekeeper of quality across deterministic and generative code. - **Self-Functionality**: Automatically detects, scores, and repairs model behavior regressions.

B. Subsystem Blueprint - `Vitest Runner Agent`: Executes CI-bound unit and integration tests. - `Golden Dataset Validator`: Matches known-good Q&A pairs against Genkit outputs. - `LLM Judge`: Uses Gemini 1.5 Pro to evaluate factuality, structure, and trace integrity. - `Fix Suggestor Agent`: Proposes edits to source prompts, flows, or test data.

C. Feedback & Growth Loops - Failed tests are logged to Firestore with metadata. - The `Fix Suggestor` observes test trends, generates patches, and triggers PRs. - New test cases are automatically added to regression sets.

D. Implementation Roadmap - CI: Extend GitHub Actions to run all three tiers. - Data: Populate Firestore with golden test input/output pairs. - Evaluation: Use Vertex AI Evaluation API with Gemini Judge for generative output scoring. - Automation: Auto-prune flaky tests, reweight scores based on failure type.

E. Critical Appraisal - Cost and token budgeting for LLM evaluation must be controlled. - Mitigate overfitting to golden dataset by incorporating adversarial augmentation (see Red Teaming).

Improvement 2 – Production-Grade Observability and Security

A. Autonomous Reinterpretation - **Function**: Central nervous system for reflexive behavior and remediation. - **Self-Functionality**: Reacts in real-time to metrics degradation, threat signatures, and telemetry spikes.

B. Subsystem Blueprint - `Telemetry Ingestor`: Wraps OpenTelemetry spans, traces, logs. - `Anomaly Detector`: Correlates signal clusters and initiates remediation. - `Secret Watcher`: Audits required secrets and auto-rotates when flagged. - `Remediation Engine`: Restarts flows, reroutes traffic, or reverts deployments.

C. Feedback & Growth Loops - Persistent anomalies trigger model revalidation or flow refactor. - Playbooks evolve from observed edge case handling. - Secret lifetimes are tracked and optimized by ML.

D. Implementation Roadmap - Use `enableFirebaseTelemetry()` + Cloud Logging/Trace. - Add App Check enforcement to all Genkit endpoints. - Connect Cloud Monitoring alerts to Cloud Functions that patch faults. - Audit Terraform-deployed IAM for drift weekly.

E. Critical Appraisal - Avoid alert fatigue by layering intelligent filtering. - Add model interpretability for anomaly root-cause explanations.

Improvement 3 – Knowledge Graph as Semantic Core

A. Autonomous Reinterpretation - **Function**: Evolving brain of ISA. - **Self-Functionality**: Discovers, expands, and validates its own semantic knowledge model.

B. Subsystem Blueprint - `Ontology Synthesizer`: Converts new standards into TypeQL schema proposals. - `KG Population Agent`: Uses LLMs + regex + doc parsers to ingest facts. - `GraphRAG Engine`: Combines KG traversals and semantic chunk retrieval. - `Schema Validator`: Detects redundant or conflicting relationships.

C. Feedback & Growth Loops - Gaps in response confidence or failure cases trigger auto-injection of new rules. - Low KG coverage alerts are linked to ingestion queues.

D. Implementation Roadmap - Build first KG schema: `trade-item`, `standard`, `document`, `section`, `conflicts-with`, etc. - Use Document AI + LangChain parser → structured ingestion pipeline. - Deploy Genkit tool wrapping TypeDB Python driver. - Schedule weekly schema sync and drift detection.

E. Critical Appraisal - TypeDB's operational footprint is significant. - Consider fallback to Neo4j Aura for early adopters.

Improvement 4 – Explainable Reasoning via Chain-of-Thought + Zod

A. Autonomous Reinterpretation - **Function**: Mirror for self-introspection. - **Self-Functionality**: Diagnoses logical missteps and rewrites flawed reasoning autonomously.

B. Subsystem Blueprint - `CoT Trace Generator`: Prompt templates with structured explanation steps. - `Trace Scorer`: Vertex AI-based evaluation for logic, completeness, grounding. - `Critique Agent`: Analyzes trace structure and flags incoherence or hallucination. - `Prompt Refiner`: Updates prompts based on recurring trace failures.

C. Feedback & Growth Loops - Weak reasoning traces generate training data for future CoT optimizers. - Rejected responses retrigger generation with refined critique context.

D. Implementation Roadmap - Add `reasoningTrace` to all answer flows via Zod. - Enable CoT-specific evaluation jobs (pairwise or absolute scoring). - Store traces in Firestore with score/version metadata. - Add UI toggle in React frontend for expandable audit display.

E. Critical Appraisal - CoT verbosity must be tuned to minimize token waste. - Consider multi-hop trace merging for complex answers.

Improvement 5 – Agentic Workflow Orchestration via LangGraph

A. Autonomous Reinterpretation - **Function**: Self-planning mind of ISA. - **Self-Functionality**: Constructs and adapts workflows based on goals, tools, and partial results.

B. Subsystem Blueprint - `LangGraph Supervisor`: Master planner that routes via conditions. - `Workflow Registry`: Stores available tools and invocation signatures. - `Human Feedback Gate`: Optionally pauses flow for reviewer input. - `Plan Mutator`: Refines node/edge configs based on prior success/failure.

C. Feedback & Growth Loops - Incomplete workflows are restructured using reroute policies. - Failing agents are flagged and retrained or replaced.

D. Implementation Roadmap - Wrap Genkit flows as LangGraph tools. - Define task-oriented workflows: `standardProposalReview`, `impactAssessment`, etc. - Use Firestore to persist state + transitions. - Build error gates and human feedback nodes.

E. Critical Appraisal - Ensure logs + traces are available per node. - Visualize graph structure using LangSmith for clarity.

Strategic Autonomy Synthesis

Improvement	Autonomy Leverage
Agentic Workflow Orchestration	Very High
Knowledge Graph as Semantic Core	Very High
Explainable Reasoning with CoT	Medium
Three-Tiered Evaluation System	Medium
Production Observability & Security	⚙ Essential

Recommended Phases

Phase 1 – Build Reflexes - Implement full CI + three-tier test pipeline - Enable OpenTelemetry, log-based alerts, secret validation agents

Phase 2 – Build Reasoning - Launch KG schema with GraphRAG-enabled QA - CoT explanations stored, evaluated, and optimized over time

Phase 3 – Build Governance - Deploy LangGraph agent supervisor with multiple tool nodes - Connect research agents and document readers to enable workflow self-assembly - Explore decentralized governance for prompt policy management

By following these phases, ISA becomes not just smarter—but **alive in its own maintenance, learning, and strategy.**