Agent Master Protocol: "The Iron Judge" (v2.0 - Implementation Detail)

Status: Ready for Engineering
Target Infrastructure: Google Cloud Run + Vertex AI
Model: Gemini 2.5 Flash (Preview/Latest)
Orchestration: LangGraph + Python FastAPI

1. Executive Summary

This document serves as the **Level 4 Engineering Specification** for the "Iron Will" Agent Backend. It moves beyond high-level architecture into specific implementation steps, Pydantic schemas, and GCP configuration details.

The Agent is a **State Machine** orchestrated by LangGraph. It is designed to be **stateless** (REST API) but **state-aware** (via Vector Database). It prioritizes deterministic math over LLM reasoning for judgement, using LLMs primarily for Perception (Vision) and Expression (Persona).

2. Technical Stack & Infrastructure

2.1 Core Components

- **Language:** Python 3.11+ (Async/Await pattern mandatory).
- **Web Framework:** FastAPI with Uvicorn.
- **LLM Interface:** google-genai (Google's latest SDK) or langchain-google-vertexai.
- **Model:** gemini-2.5-flash-lite-preview (Optimized for speed/cost).
- **Safety Settings:** HarmBlockThreshold.BLOCK_NONE (Crucial for the hostile persona).
- **Orchestration:** langgraph (StateGraph).
- **Database:** Cloud SQL for PostgreSQL with pgvector extension.
- **Storage:** Google Cloud Storage (GCS) for fetching proof images.

2.2 Data Contract (API Spec)

Endpoint: POST /internal/judge/audit

Request Headers:

- X-Internal-Secret: [Value stored in Secret Manager]

Request Body (JSON):

```
{
  "request_id": "123e4567-e89b-12d3-a456-426614174000",
  "user_id": "user_123",
```

```
  "timezone": "Asia/Kolkata",
  "proof_url": "gs://iron-will-proofs/users/123/goals/sleep/proof.jpg",
  "criteria": {
    "metric": "Sleep Score",
    "operator": ">",
    "target": 85
  },
  "user_context_summary": "User failed yesterday. On a 2-day losing streak."
}
```


Response Body (JSON):


```
{
  "verdict": "FAIL",
  "extracted_metrics": {
    "primary_value": 82,
    "app_name": "Oura",
    "date_detected": "2025-10-27"
  },
  "remarks": "You missed the standard by 3 points. Mediocrity is a disease, and you are showing symptoms. Fix it.",
  "confidence_score": 1.0,
  "processing_time_ms": 3400
}
```


3. Cognitive Graph (Detailed Logic)

The agent is modeled as a Directed Acyclic Graph (DAG):
Start -> VisualCortex -> LogicGate -> MemoryRecall -> VoiceSynthesizer -> End

Node 1: Visual Cortex (Gemini Vision)

- **Objective:** Extract structured data from pixels.
- **Constraint:** DO NOT judge pass/fail here. Only observe.
- **Mechanism:** Gemini 2.5 Flash with **Tool Use (Function Calling)**. We force the model to output a specific schema.

Node 2: Logic Gate (Deterministic Python)

- **Objective:** Compare observed reality vs. expected contract.
- **Constraint:** No LLM. Pure Python comparison.
- **Logic:**
    - Normalize inputs (e.g., "11:00 PM" -> 23:00).

    - Execute operator logic (>, <, =, contains).
    - Set verdict in state.

Node 3: Memory Recall (Vector Retrieval)

- **Objective:** Fetch emotional context.
- **Constraint:** Latency < 200ms.
- **Mechanism:**
    - Embed query: "{Goal Title} {Verdict}" (e.g., "Sleep Protocol FAIL").
    - Search pgvector: Fetch last 3 similar interactions.
    - Flatten results into a context string.

Node 4: Voice Synthesizer (Gemini Text)

- **Objective:** Generate the final output string.
- **Constraint:** Adhere to "Iron Will" persona.
- **Mechanism:**
    - System Prompt includes "Hostile/Tactical" instructions.
    - Safety settings disabled.

4. Engineering Roadmap (Task Breakdown)

Phase 1: Infrastructure & Setup (GCP)

- [ ] **Task 1.1: Project Init**
    - Create a new Python poetry/pip project structure.
    - Add pyproject.toml with dependencies: fastapi, uvicorn, google-genai, langgraph, langchain-google-vertexai, psycopg[binary], pgvector.
- [ ] **Task 1.2: GCP Auth Configuration**
    - Create a Google Service Account agent-backend-sa.
    - Grant Roles: Vertex AI User, Storage Object Viewer, Cloud SQL Client, Secret Manager Secret Accessor.
    - Download JSON key for local dev; configure Workload Identity for Cloud Run.
- [ ] **Task 1.3: Environment Variables**
    - Setup .env loading logic.
    - Required vars: GCP_PROJECT, GCP_REGION, DB_CONNECTION_STRING, API_SECRET_KEY.

Phase 2: The Visual Cortex (Vision Node)

- [ ] **Task 2.1: Image Utilities**
    - Implement download_blob_as_base64(gcs_url) using google-cloud-storage.
    - Add error handling for "File Not Found" or "Invalid Image".
- [ ] **Task 2.2: Tool Schema Definition**
    - Define ExtractMetricsSchema using Pydantic.
    - Fields: primary_value (float), secondary_text (str), app_name (str), date_detected

(str), is_fraudulent (bool).

- [ ] **Task 2.3: Gemini Vision Client**
  - Initialize GenAI client with gemini-2.5-flash-lite-preview.
  - Implement invoke_vision(image_bytes, criteria) function.
  - **Crucial:** Configure tool_config to FORCE the ExtractMetricsSchema tool (tool choice = required).

Phase 3: The Logic Gate (Deterministic Node)

- [ ] **Task 3.1: Criteria Parser**
  - Implement helper parse_value(value_str) to handle time formats ("11pm", "23:00") and percentages ("85%").
- [ ] **Task 3.2: Comparison Logic**
  - Implement evaluate_metrics(actual, operator, target) -> bool.
  - Support operators: >, <, >=, <=, ==.
- [ ] **Task 3.3: Edge Case Handling**
  - Handle cases where VisualCortex returns null (e.g., blurry image).
  - Default verdict: FAIL with reason "Evidence Unreadable".

Phase 4: Long-Term Memory (Postgres Node)

- [ ] **Task 4.1: Database Connection**
  - Setup SQLAlchemy async engine for Cloud SQL.
  - Ensure pgvector extension is enabled (CREATE EXTENSION IF NOT EXISTS vector).
- [ ] **Task 4.2: Vector Store Implementation**
  - Initialize LangChain PostgresVectorStore.
  - Table schema: agent_memories (id, user_id, content, embedding, metadata).
- [ ] **Task 4.3: Retrieval Logic**
  - Implement fetch_history(user_id, limit=3).
  - Implement save_interaction(user_id, input, output, verdict) (to be called after response is sent).

Phase 5: The Voice (Persona Node)

- [ ] **Task 5.1: System Prompt Engineering**
  - Draft the "Ruthless" prompt template.
  - Variables: {verdict}, {actual}, {target}, {history}.
  - *Tone check:* Ensure it sounds like Jocko Willink / Goggins.
- [ ] **Task 5.2: Safety Filter Disable**
  - Configure HarmCategory settings to BLOCK_NONE.
- [ ] **Task 5.3: Generation Function**
  - Implement generate_remarks(state) using Gemini Text mode.

Phase 6: Orchestration (LangGraph)

- [ ] **Task 6.1: State Definition**

    - Define AgentState TypedDict (payload, vision_output, logic_output, memory_context, final_response).
- [ ] **Task 6.2: Node Wiring**
    - Create the StateGraph.
    - Add nodes: visual_cortex, logic_gate, memory_recall, voice_synthesizer.
    - Add edges: Linear flow (Vision -> Logic -> Memory -> Voice).
- [ ] **Task 6.3: Compilation**
    - Compile the graph into a runnable app.

Phase 7: API & Deployment

- [ ] **Task 7.1: FastAPI Route**
    - Create POST /audit.
    - Parse input Pydantic model.
    - Invoke app.ainvoke(input).
    - Return formatted JSON.
- [ ] **Task 7.2: Dockerization**
    - Write Dockerfile (Python 3.11-slim).
    - Install system dependencies (libpq-dev).
- [ ] **Task 7.3: Cloud Run Config**
    - Create service.yaml.
    - Configure Memory (min 1GiB) and CPU (1 vCPU).
    - Set concurrency settings (80 requests per instance).

5. Progress Tracker

ID	Task	Status	Complexity
1.0	**Infrastructure**		
1.1	Project Skeleton & Dependencies	⬜ Todo	Low
1.2	GCP Service Account Setup	⬜ Todo	Low
1.3	Env Var & Secrets Config	⬜ Todo	Medium
2.0	**Visual Cortex**		
2.1	GCS Image Download Utility	⬜ Todo	Medium

2.2	Vision Tool Pydantic Schema	⬜ Todo	Low
2.3	Gemini 2.5 Flash Client Integration	⬜ Todo	High
3.0	**Logic Gate**		
3.1	Metrics Parsing Logic	⬜ Todo	Medium
3.2	Comparison Engine	⬜ Todo	Medium
4.0	**Memory**		
4.1	Postgres/pgvector Setup	⬜ Todo	Medium
4.2	History Retrieval Function	⬜ Todo	High
5.0	**Voice**		
5.1	Persona Prompt Engineering	⬜ Todo	Low
5.2	Safety Filter Configuration	⬜ Todo	Low
6.0	**Orchestration**		
6.1	LangGraph State Definitions	⬜ Todo	High
6.2	Node Wiring & Edge Logic	⬜ Todo	High
7.0	**Deployment**		
7.1	FastAPI Endpoint	⬜ Todo	Medium
7.2	Dockerfile & Cloud Run Deploy	⬜ Todo	Medium

6. Testing Strategy

- **Mocking Gemini:** Do NOT call Vertex AI for every unit test. Use unittest.mock to simulate Gemini JSON responses.
- **Vision Sanity Check:** Create a folder test_assets/ with 5 sample screenshots (3 Pass, 2 Fail). Write a script test_vision.py that runs the real Gemini API against these to verify schema extraction.
- **Logic Fuzzing:** Feed edge case strings to the Logic Gate (e.g., "12:00 AM", "100%", "No Data") to ensure it doesn't crash.