**SmalBlu LLM-Powered Infrastructure Optimization Agent**  
**System Design & Architecture Document**

**1. LLM Selection & Architecture**

**LLM Choice:**

We propose a **hybrid multi-model architecture** using:

* **GPT-4-turbo (via OpenAI API)** – for advanced reasoning, summarization, and generation.
* **Fine-tuned LLaMA (Meta AI)** – for on-prem/private inference with domain-specific training.
* **Claude 3 (Anthropic)** – for safety-critical reasoning and long-context analysis.

**Architecture Design:**

* **Multi-Agent System:** Specialized agents for:
  + **Compute Optimization Agent**
  + **Storage Optimization Agent**
  + **Database Query Agent**
  + **Carbon & Sustainability Agent**
  + **Cost Governance Agent**

Each agent processes its domain and forwards insights to an **Orchestration Layer** (meta-controller agent) for cross-layer tradeoff analysis.

**Why This Architecture:**

* **Accuracy:** Specialized agents lead to deeper domain-specific insights.
* **Latency:** Agents can operate in parallel.
* **Cost-efficiency:** Smaller fine-tuned models can handle common cases; GPT-4 is reserved for high-stakes scenarios.
* **Reliability:** Redundant reasoning across agents with a voting or confidence-based arbitration layer.

**Handling Model Limitations:**

* **Context Length:** Chunking data by layer and summarizing with windowed attention before passing to LLMs.
* **Hallucinations:**
  + Use rule-based validators.
  + Require every recommendation to cite metrics or logs.
* **Consistency:**
  + Use embedding-based memory to enforce similar outputs.
  + Prompt versioning and caching.

**2. Conflict Resolution & Human-in-Loop**

**Conflict Handling Logic:**

* Build a **Tradeoff Reasoning Module**:
  + Detect conflicting metrics (e.g., low CPU, high memory).
  + Use a **weighted optimization matrix**: (e.g., 40% cost, 30% performance, 20% sustainability, 10% risk).
  + Let the orchestration agent mediate with scenario-based reasoning.

**Confidence Scoring:**

* For each recommendation, assign confidence using:
  + **Model output logprobs**
  + **Number of agreeing agents**
  + **Completeness of input data**
  + **Historical similarity retrieval** (via vector database)

**Escalation Triggers:**

* Low confidence (<60%)
* Conflicting agent outputs
* Mission-critical systems or Tier-0 workloads
* New/unseen infrastructure configuration

**Quality Assurance:**

* All outputs pass through a **QA filter layer**:
  + Schema validation
  + Cross-checked with business rules
  + Outlier detection (cost, risk, performance)

**3. Prompt Engineering Strategy**

**Prompt Structure:**

* Use **modular templates**:

"You are a [Domain] Optimization Agent. You are analyzing the following metrics: [Insert preprocessed summary]..."

* Prompts include:
  + **System role**
  + **Structured input (metrics, logs, thresholds)**
  + **Explicit optimization goal** (e.g., minimize cost, preserve latency)

**Reasoning Approach:**

* Adopt **Chain-of-Thought (CoT)** prompting:
  + First, analyze component health
  + Then, evaluate optimization opportunity
  + Finally, generate recommendation with justification

**Output Standardization:**

* JSON schema enforced:

{

"component": "RDS instance db-prod-1",

"recommendation": "Downsize to db.m5.large",

"reason": "CPU under 15% for 7 days",

"confidence": 92,

"estimated\_savings": "$430/month"

}

* Central schema validator rejects non-conforming outputs

**Why CoT over direct prompting?**

* More reliable in ambiguous or conflicting input
* Improves explainability and trust

**4. Human-in-Loop: Escalation Design**

**When to Escalate:**

* Confidence < 60%
* Novelty score > threshold (e.g., <10% similarity to past cases)
* Agents disagree on action
* Data missing or incomplete

**How to Present to Humans:**

* UI panel shows:
  + Conflicting inputs
  + Agent chain-of-thought traces
  + Risk vs. reward estimate
  + Call-to-action buttons (Accept / Reject / Modify)

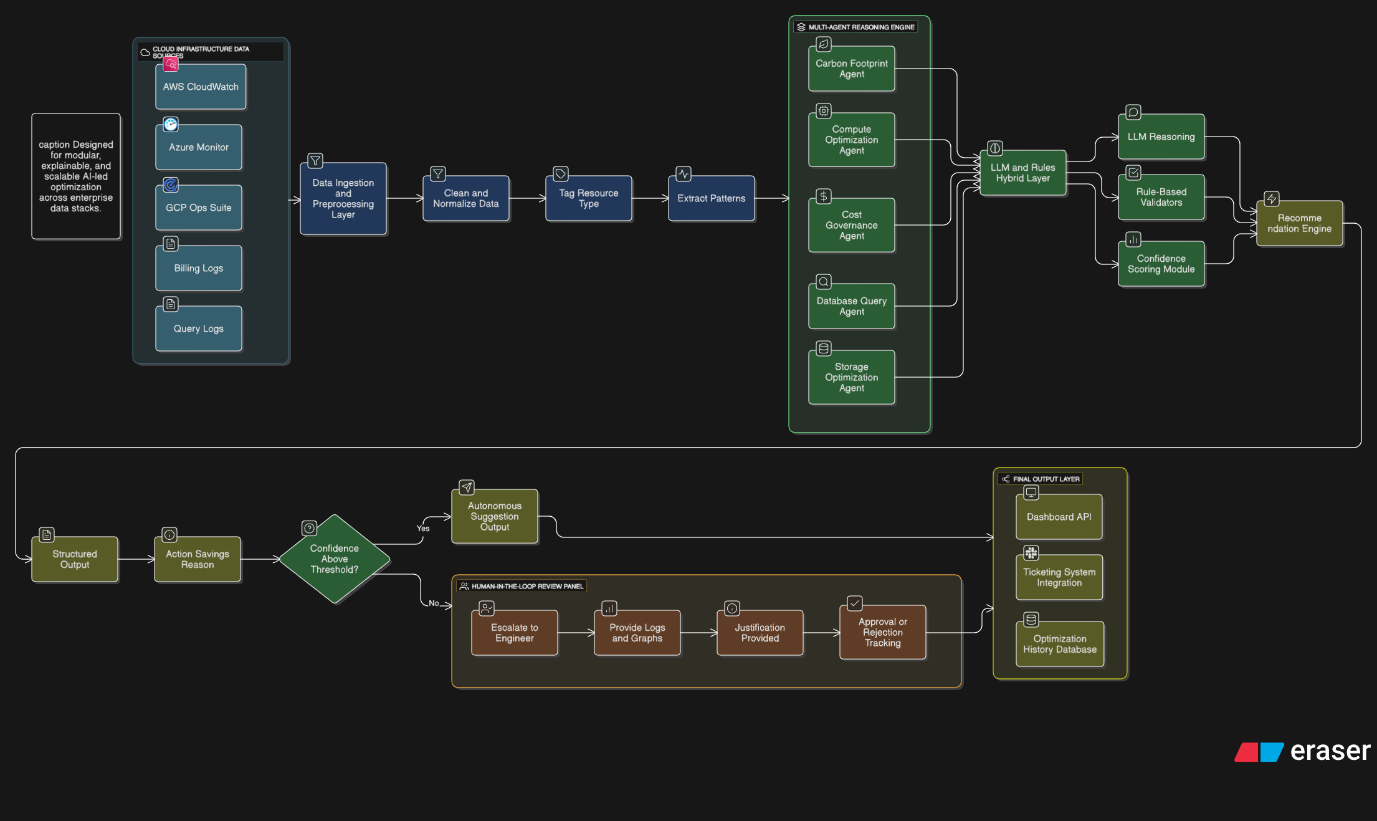
**Feedback Loop Design:**

* Store human decisions
* Use decisions to:
  + Fine-tune domain-specific agents
  + Update vector DB for similarity-based context injection
  + Improve future confidence scoring

**Error Handling:**

* Missing data → fallback to baseline thresholds or human-only alerts
* Bad format → auto-correction or reroute to admin review
* Logging & telemetry for all errors

**Simple Architecture Diagram**



**Conclusion**

This architecture ensures:

* **Accurate**, domain-specific analysis via specialized agents
* **Safe**, trustworthy recommendations through hybrid validation
* **Scalable** performance for enterprise-scale cloud environments
* **Continuous learning** through structured human feedback

This system is both **intelligent and pragmatic**, ready for enterprise deployment with trust at the center of every decision.