C.A.L.M. System - Technical Architecture Report

Reply Student Tech Clash 2025 | Level 3: Multi-Agent Approach

1. Architecture Overview & Problem Statement

The C.A.L.M. (Critical Agent Logic Module) system is designed to address the systemic issue of 'Alarm Fatigue' in modern intensive care units. In current hospital environments, critical patient data is often trapped in departmental silos - pharmacy records do not automatically sync with bedside monitors, and patient history is often buried in complex EMR logs.

To solve this, C.A.L.M. moves beyond simple rule-based alerts by employing a Level 3 Multi-Agent Orchestration pattern. Instead of a single AI attempting to parse all data, the system delegates cognitive responsibilities to a 'Digital Team' of four specialized autonomous agents, each with distinct permissions and knowledge domains.

2. The Digital Team (Roles & Logic)

The system is composed of four distinct agents that operate in a collaborative loop. Each agent is implemented with specific prompts and decision trees to ensure specialized accuracy.

2.1 The Watchdog (The Observer)

The Watchdog acts as the system's sensory input layer. It is responsible for ingesting real-time telemetry from IoT devices and bedside monitors. Unlike standard alarms, the Watchdog is designed to be hypersensitive but low-context.

- Logic: It evaluates raw vital signs against strict medical thresholds defined in the codebase (e.g., Heart Rate < 50 bpm or > 120 bpm).
- Constraint: It deliberately ignores patient history to ensure no potential emergency is ever pre-filtered at the ingestion stage.
- Output: Flags events as 'NORMAL' or 'ANOMALY' with an associated severity score (Low, Medium, Critical).

2.2 The Historian (The Context Engine)

The Historian serves as the system's memory. Upon receiving an anomaly flag from the Watchdog, it queries the patient's longitudinal Electronic Medical Record (EMR). Its primary goal is to validate whether a flagged metric is truly abnormal for that specific patient.

- Logic: It checks for chronic conditions (e.g., 'Patient is a competitive athlete with low resting heart rate') or recent surgical notes (e.g., 'Post-op recovery').
- Output: Returns a verdict of 'CONFIRM' (if the alert is valid) or 'OBJECT' (if the patient's history explains the data).

2.3 The Pharmacist (The Safety Net)

The Pharmacist Agent is the toxicological safety layer. It analyzes the patient's active medication timeline to detect drug-induced physiological changes that mimic disease states.

- Logic: It cross-references the flagged vital sign against the side-effect profiles of administered drugs (e.g., detecting that a sudden drop in heart rate correlates with a recent dose of Atenolol/Beta-Blockers).
- Impact: This agent specifically prevents 'Medication Reconciliation Errors' and false sepsis alerts caused by drug interactions.

2.4 The Decision Maker (The Command Center)

The Decision Maker acts as the executive manager. It receives the conflicting or complementary reports from the Watchdog, Historian, and Pharmacist and applies a hierarchical decision tree to generate a final output.

- Protocol - Code Red: If the event is Life-Threatening and unexplained, it triggers an immediate pager alert.
- Protocol - Yellow Log: If the Pharmacist identifies a known side effect OR the Historian confirms the value is baseline for the patient, the alarm is suppressed and logged silently.
- Goal: To filter out noise and only escalate true emergencies, effectively reducing cognitive load on clinical staff.

3. Technical Communication Protocol

The agents communicate via a synchronous JSON-passing workflow orchestrated in the main execution loop. This ensures data integrity and traceable decision paths.

- Step 1 (Trigger): The system iterates through the 'events_stream', simulating live MQTT inputs from ICU devices.
- Step 2 (Parallel Broadcast): If the Watchdog detects an anomaly, the specific event object and patient_data are passed simultaneously to the Historian and Pharmacist agents for independent analysis.
- Step 3 (Negotiation): The specialized agents return structured JSON payloads. The Pharmacist injects drug knowledge (e.g., 'Interaction: True'), and the Historian injects context (e.g., 'Verdict: Object').
- Step 4 (Synthesis): The Decision Maker synthesizes these inputs into a final 'message_to_doctor', ensuring the final alert contains both the warning and the 'Why'.

4. Evaluation of Multi-Agent Benefit

Transitioning from a single-agent LLM to this Level 3 Multi-Agent architecture provided three measurable technical benefits:

C.A.L.M. System - Technical Architecture Report

Reply Student Tech Clash 2025 | Level 3: Multi-Agent Approach

A. Prevention of Hallucinations (Accuracy)

In a single-agent setup, AI models often attempt to 'justify' abnormal data even when it is contradictory. By separating the 'Detector' (Watchdog) from the 'Context Provider' (Historian), we force the system to cross-reference facts before generating a conclusion. For example, in Scenario P-808, the system correctly identified a Heart Rate of 42 as normal for an athlete, whereas a standard algorithm would have triggered a Code Blue.

B. Domain Specialization & Safety

General-purpose LLMs often miss specific drug-vital interactions. By creating a dedicated Pharmacist Agent with a system prompt focused solely on toxicology, the system successfully filters out physiological responses to treatment. This directly addresses the safety concerns regarding 'Alert Desensitization' found in the case study.

C. Scalability & Modularity

The modular class-based design allows for rapid scalability. New agents (e.g., a 'Lab Results Analyzer' or 'Insurance Authorization Checker') can be instantiated in 'agents.py' and added to the Decision Maker negotiation loop without requiring a rewrite of the core logic engine.