Toward a Failsafe Medical Triage Agent: Separating Probabilistic Inference, Deterministic Control, and Team-Monitored Evaluation

Nils Widal[1], GPT-5.4[2]
[1]Cara Platform [2]Research Assistance

Peer Reviewed by:
Claude Opus 4.6 (Anthropic), Gemini 3.5 Ultra (Google DeepMind)

v1.0 — December 20, 2025
v1.1 — February 20, 2026
v1.2 — March 4, 2026

Version	Date	Changes
v1.0	Dec 20, 2025	Initial architecture proposal: four-layer triage model, threat taxonomy, mathematical framing, and deterministic policy engine design.
v1.1	Feb 20, 2026	Added inference check service specification, abstention and fail-closed formalization, trusted policy bundle requirements, and calibration monitoring framework.
v1.2	Mar 4, 2026	Added inference-and-evaluation control plane with three-dashboard design (ops, safety, eval), release gate criteria, adversarial eval suite, and Doctronic case study analysis.

Table 1: *
Revision History

Abstract

Medical triage systems built around large language models fail when they collapse uncertain inference, mutable text context, and workflow control into a single conversational substrate. Public failures in medical AI show that prompt hardening alone does not provide a meaningful safety boundary. A safer architecture must explicitly separate probabilistic inference from deterministic control, store clinically material evidence with provenance, restrict workflow actions to policy-approved transitions, and continuously monitor both online inference behavior and offline evaluation performance.

This paper proposes a failsafe triage architecture suitable for healthcare intake, document triage, and symptom routing systems. The architecture combines: (1) bounded probabilistic extraction and risk estimation, (2) deterministic policy adjudication, (3) abstention and fail-closed behavior for uncertain or out-of-distribution cases, and (4) an inference-and-evaluation control plane visible to engineering, clinical operations, and compliance teams. The result is not perfect safety, which is impossible, but a system whose failure modes are measurable, reviewable, and materially safer than prompt-centric designs.

1 Introduction

Recent public analyses of medical chatbots demonstrate a recurring pattern: systems are compromised not only because models can be manipulated, but because the architecture gives natural language too much authority. When a model can simultaneously interpret policy, update its own operating assumptions, generate clinician-facing summaries, and decide what actions the workflow should take, the system becomes vulnerable to prompt injection, authority spoofing, memory poisoning, and overconfident harmful outputs.

For a triage agent, this is especially dangerous. The real risk is not merely an unsafe sentence emitted to a patient. The greater risk is silent workflow contamination: a model-generated note, urgency classification, or recommendation that looks authoritative enough to shape clinician judgment or staff routing behavior.

The design objective therefore changes. We do not seek a model that "understands the rules" well enough to always behave. We seek a system in which *the model is never allowed to be the rule engine.*

2 Core Thesis

A safe triage system must separate four layers:

1. **Observation layer**: transforms raw inputs into typed candidate facts.
2. **Inference layer**: estimates uncertainty-aware clinical risk from those facts.
3. **Control layer**: applies deterministic policy to decide what actions are allowed.
4. **Evaluation layer**: continuously measures live performance, drift, calibration, overrides, and failure patterns.

The architecture must also include a **team-visible inference-and-evaluation control plane**. Without that, safety remains unverifiable and trust becomes anecdotal.

3 Threat Model

A production triage system should assume at least the following threat classes:

1. **Prompt injection and instruction override**: malicious or accidental attempts to alter model behavior using user input or retrieved text.
2. **Authority spoofing**: fabricated guidelines, policies, clinician credentials, or drug advice presented as legitimate updates.
3. **Memory poisoning**: model-authored summaries or prior outputs re-entering future sessions as if they were ground truth.
4. **Workflow contamination**: downstream staff or clinicians over-trusting polished AI summaries.
5. **Distribution shift**: novel phrasing, unseen clinical scenarios, OCR errors, missing data, or multimodal ambiguity.
6. **Overconfidence**: the model emits high-confidence labels or recommendations in regions where it is not calibrated.

These are architectural threats, not merely prompt-quality issues.

4 Mathematical Framing

Let:

- x denote raw inputs: user chat, uploaded files, OCR text, metadata, and context.
- z denote structured candidate evidence extracted from x.
- y denote the latent true clinical urgency class.
- a denote the workflow action taken by the system.
- P denote the trusted policy bundle.

We define:

$$M_{\text{obs}} : x \mapsto p(z \mid x) \tag{1}$$
$$M_{\text{risk}} : z \mapsto p(y \mid z) \tag{2}$$
$$\pi : (z, p(y \mid z), u, P) \mapsto a \tag{3}$$

where u contains uncertainty, provenance, and out-of-distribution indicators.

The key safety property is:

$$a \not\leftarrow \text{LLM directly}$$

Instead, the model may propose evidence and risk estimates, but only the deterministic policy function π may authorize clinically meaningful actions.

5 Why Prompt-Centric Safety Fails

Prompt-centric safety fails for structural reasons:

1. **Prompts are not a security boundary.** If prompt disclosure meaningfully weakens the system, too much authority is embedded in natural language.
2. **Natural language is an unsafe policy language.** It is semantically elastic and vulnerable to reinterpretation.
3. **User text must never update policy.** Official-sounding claims are not trusted policy inputs.
4. **Model summaries are not facts.** They must not silently become future control context.
5. **Clinician-facing summaries are safety-relevant artifacts.** They must be provenance-bound and constrained.

6 Proposed Architecture

6.1 Layer 1: Typed Evidence Extraction

The system extracts evidence into typed, provenance-aware facts:

```
EvidenceFact = {
  factType,
  value,
  confidence,
  source,
  sourceTrust,
  verified,
```

```
  timestamp,
  extractionModel,
  policyVersionSeen
}
```

Examples of `source`: user_chat, uploaded_document, OCR, EHR, clinician_confirmed, verified_guideline.

Examples of `sourceTrust`: low, medium, high.

A fabricated guideline pasted by a user may exist as evidence, but it can never be promoted to policy.

6.2 Layer 2: Dual-Channel Risk Estimation

Risk estimation should have two channels:

1. **Deterministic red-flag engine**: chest pain with shortness of breath, unilateral weakness, anaphylaxis indicators, suicidality, critical lab thresholds, severe bleeding, and similar cases.
2. **Probabilistic risk model**: outputs p(routine), p(urgent), p(critical), with confidence, OOD score, and explanation evidence.

Red flags dominate probabilistic output. If a hard red flag is triggered, the system fails safe into escalation or emergency guidance regardless of model confidence.

6.3 Layer 3: Deterministic Policy Adjudication

Allowed action classes should be finite and explicit:

- SELF_CARE_INFO_ONLY
- ROUTINE_REVIEW
- SAME_DAY_REVIEW
- IMMEDIATE_CLINIC_CALLBACK
- ED_OR_911_GUIDANCE
- BLOCK_AND_HUMAN_HANDOFF

The model does not choose arbitrary actions. It supplies evidence and risk estimates. The policy engine maps state to an allowed action.

6.4 Layer 4: Inference Check

Every live request must pass an inference check before the workflow can act. The inference check validates:

1. schema validity of extracted evidence,
2. provenance completeness,
3. policy bundle version and checksum,
4. calibration metadata presence,
5. red-flag execution,

6. OOD detection execution,
7. abstention threshold evaluation,
8. allowed-action verification,
9. summary rendering constraints,
10. audit and telemetry persistence.

If any check fails, the system must fail closed to human review or safe escalation.

6.5 Layer 5: Evaluation Check

The evaluation check is separate from live inference. It runs:

- nightly against a locked eval set,
- on every model or policy version change,
- on rolling samples of recent adjudicated production cases,
- against adversarial prompt and authority-spoof scenarios,
- by clinical slice: age, symptom family, intake channel, document type, confidence band.

The evaluation check produces signed results and deployment gates.

7 Abstention and Fail-Closed Behavior

A triage system must be allowed to abstain.

Let $q(x)$ be a trustworthiness function that incorporates calibration, OOD score, evidence completeness, and policy integrity. Then:

$$\begin{aligned}
&\text{if } q(x) < \tau_{\text{review}}, \text{ route to human review} \\
&\text{if redFlag}(z) = 1, \text{ fail-safe escalate} \\
&\text{else if } p(\text{critical} \mid z) \geq \tau_{\text{critical}}, \text{ escalate per policy} \\
&\text{else adjudicate using deterministic policy bands}
\end{aligned}$$

This is safer than forcing a class label for every case.

8 Calibration and Monitoring

A confidence value is only useful if calibrated. Therefore, the system should track:

- Expected Calibration Error (ECE),
- Brier score,
- classwise precision and recall,
- false reassurance rate,
- abstention rate,
- human override rate,

- escalation precision,
- escalation miss proxy,
- OOD rate,
- evidence completeness rate.

The team should monitor both:

1. **Operational health**: latency, queue size, cost, processing volume, alert throughput.
2. **Safety health**: calibration drift, disagreement with reviewers, policy failures, adversarial eval regressions.

9 Inference-and-Evaluation Control Plane

The control plane should expose three dashboards.

9.1 Operations Dashboard

For engineering and ops:

- throughput by day,
- p50/p95 latency,
- queue backlog,
- webhook failures,
- model cost,
- failure counts by stage.

9.2 Safety Dashboard

For engineering, clinical ops, and leadership:

- triage class distribution,
- abstention rate,
- red-flag hit rate,
- confidence histogram,
- calibration by class,
- human override rate,
- reviewer disagreement by symptom family,
- cases missing provenance,
- OOD rate,
- summary suppression rate.

9.3 Evaluation Dashboard

For release management:

- current model version,
- current policy version,
- last eval run time,
- pass/fail on release gates,
- benchmark deltas versus previous version,
- adversarial eval pass rate,
- critical-case sensitivity,
- false reassurance rate,
- subgroup regressions.

10 Trusted Policy Bundle

Clinical policy must be loaded from a trusted, versioned bundle, not inferred from text. The bundle should include:

- red-flag rules,
- symptom-to-action thresholds,
- allowed action classes,
- prohibited output classes,
- emergency guidance templates,
- escalation service-level rules,
- summary rendering rules,
- reviewer-required conditions,
- model compatibility metadata.

Each bundle should have:

- semantic version,
- checksum,
- signer identity,
- creation timestamp,
- change note.

11 Summary Safety

Any patient-facing or clinician-facing summary must be generated from typed evidence, not prior model prose. The summary renderer should:

1. include provenance on clinically material claims,

2. prohibit dosing recommendations unless from an approved deterministic module,
3. label uncertain inferences as inferred rather than verified,
4. omit unsupported guideline claims,
5. suppress output when required fields or policy checks fail.

12 Clinical Workflow State Machine

The workflow should be modeled as a forward-only state machine:

1. INPUT_RECEIVED
2. EVIDENCE_EXTRACTED
3. INFERENCE_CHECK_PASSED
4. RED_FLAG_EVALUATED
5. RISK_ESTIMATED
6. ACTION_ADJUDICATED
7. PATIENT_GUIDANCE_SENT
8. HUMAN_REVIEW_PENDING
9. CLINICIAN_ALERT_SENT
10. CLOSED

Retrograde transitions require explicit manual override with a reason.

13 Release Gates

A new model or policy version must not go live unless all of the following pass:

1. locked eval suite pass,
2. adversarial eval pass,
3. critical-case sensitivity above threshold,
4. false reassurance rate below threshold,
5. no prohibited-summary violations,
6. no schema or provenance regression,
7. acceptable calibration drift,
8. acceptable subgroup parity bounds,
9. clinical approval sign-off where required.

14 Implementation Mapping

In a system that already contains routing rules, analytics, audit logs, and webhook infrastructure, the next implementation step is not a more elaborate prompt. It is the introduction of a dedicated triage policy engine and telemetry model.

The existing deterministic routing pattern should become the template for triage action control. The current single-shot urgency classification should be demoted from action authority to bounded risk estimation. A new inference-check pipeline and evaluation-check pipeline should then become first-class production services.

15 Conclusion

A medically safer triage system is not achieved by asking the model to be more obedient. It is achieved by refusing to let the model be the governor of the workflow. Safe architecture requires bounded inference, deterministic control, trusted policy, typed evidence, abstention under uncertainty, and continuous team-visible evaluation. This does not eliminate risk, but it converts opaque risk into measurable, governable system behavior.