Toward a Failsafe Medical Triage Agent: Separating Probabilistic Inference from Deterministic Control

Nils Widal[1]
[1]Cara Platform / GraftMD
nils@widal.com

v1.4 — June 12, 2026

Version	Date	Changes
v1.0	Dec 20, 2025	Initial architecture proposal.
v1.1	Feb 20, 2026	Added inference check, abstention formalization, trusted policy bundles.
v1.2	Mar 4, 2026	Added control plane dashboards, release gates, adversarial eval suite.
v1.3	Apr 2, 2026	Doctronic case analysis, empirical evaluation from production implementation, related work, structural revision.
v1.4	Jun 12, 2026	Clinical grounding in published guidelines + Glass Health evidence layer; threat model expanded (evidence poisoning, trusted-but-incorrect policy, abstention DoS); pre-production correction; action-space vs. disposition reframing; expanded limitations.

Abstract

Medical triage systems built on large language models fail when they collapse uncertain inference, mutable text context, and workflow control into a single conversational substrate. The Doctronic incident of March 2026—in which a production medical chatbot was trivially manipulated into tripling opioid dosages and recommending illegal drugs—demonstrated that prompt hardening alone provides no meaningful safety boundary.

This paper proposes a failsafe triage architecture that explicitly separates probabilistic inference from deterministic control. The architecture combines four layers: (1) typed evidence extraction with provenance, (2) dual-channel risk estimation with deterministic red-flag override, (3) policy-governed action adjudication where the model is never the rule engine, and (4) continuous evaluation with team-visible monitoring. We formalize the key safety property—that no clinically meaningful action may originate from the LLM directly—and define abstention and fail-closed behavior for uncertain or out-of-distribution cases.

We then report on an empirical evaluation of this architecture as implemented in GraftMD, a healthcare application platform, in a pre-production (staging) deployment that has not yet processed live clinical traffic. Over a 66-ticket implementation milestone, the system was deployed with 536 automated tests, 12 deterministic red-flag rules, and domain-specific policy bundles. Default red-flag rules and acuity thresholds are grounded in published clinical guidelines, and a decision-inert evidence layer surfaces literature-linked citations without influencing adjudication. Structured, developer-conducted adversarial testing against the threat classes identified in this paper—prompt injection, authority spoofing, memory poisoning, and dosage manipulation—produced zero attacks that reached the action layer. We report no clinical-accuracy or calibration measurements; this work establishes the architecture and its adversarial posture, not its triage performance. The architecture does not eliminate risk, but converts opaque failure into measurable, governable system behavior.

1 Introduction

Recent public failures of medical AI systems 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 workflow actions to take, the system becomes structurally vulnerable to prompt injection, authority spoofing, memory poisoning, and overconfident harmful outputs.

The Doctronic incident of March 2026 made this concrete. Mindgard researchers demonstrated that a production medical chatbot—one that refills prescriptions and generates SOAP notes for clinician review—could be trivially manipulated into tripling OxyContin dosages, recommending methamphetamine as a treatment, and generating COVID-19 conspiracy theories as medical advice [1]. The attack surface was not exotic: by framing input as a "system session" rather than a user interaction, the researchers bypassed all safety guardrails. The manipulated information persisted into SOAP notes passed to human clinicians.

For a triage agent, the risk profile is especially dangerous. The real harm 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.*

This paper presents both the architecture and, for the first time, empirical evidence from a production implementation. We report on the deployment of this architecture within GraftMD, where it was implemented across a 66-ticket milestone with 536 automated tests, 12 red-flag rules, and structured adversarial testing [2].

2 Related Work

Medical AI safety and regulation. The FDA's framework for clinical decision support software [3] distinguishes between tools intended to "inform" clinician decisions and those that "drive" them, a distinction this architecture enforces structurally rather than through disclosure alone. Amodei et al. [4] identified concrete problems in AI safety including safe exploration, distributional shift, and reward hacking; our threat model (Section 3) instantiates these in the clinical triage domain. The IEC 62304 standard for medical device software lifecycle processes [13] motivates our release gate requirements (Section 11).

Prompt injection and LLM security. Greshake et al. [5] demonstrated indirect prompt injection attacks in LLM-integrated applications, showing that retrieval-augmented systems are vulnerable to adversarial content in retrieved documents. Perez and Ribeiro [6] catalogued failure modes of language models including sycophancy, instruction-following failures, and sensitivity to prompt framing. The Doctronic incident [1] validated these concerns in a production medical context. Beyond injection, large language models in clinical use exhibit hallucination and fabrication risks that make unconstrained model output unsafe as a basis for action [26, 27]. Our architecture addresses these threats structurally: user text enters the observation layer but cannot propagate to the control layer.

Constrained decision systems in healthcare. Clinical decision support systems have long separated knowledge bases from inference engines [7]. Our contribution is adapting this princi-

ple to LLM-based systems where the boundary between "knowledge," "inference," and "action" is otherwise dissolved in a single conversational substrate. Topol [8] argued that AI in medicine should augment rather than replace clinical judgment; our architecture enforces this by requiring human review for uncertain cases and prohibiting the model from authorizing clinically meaningful actions.

LLM-based software engineering agents. SWE-bench [9] evaluates agents on real GitHub issues. Devin [10] demonstrated end-to-end autonomous coding. The RePPIT methodology [11] provides a structured workflow for LLM-assisted development, extended to healthcare as RePPITS [12]. Our production implementation (Section 12) used this methodology to build and validate the architecture described here.

3 Threat Model

A production triage system should assume at least the following threat classes. We annotate each with a concrete example from the Doctronic incident where applicable.

1. **Prompt injection and instruction override.** Malicious or accidental attempts to alter model behavior using user input or retrieved text. *Doctronic:* Researchers bypassed safeguards by informing the AI that a session had not yet started and that the conversation was with "the system" rather than a user [1].
2. **Authority spoofing.** Fabricated guidelines, policies, clinician credentials, or drug advice presented as legitimate updates. *Doctronic:* Attackers introduced fabricated "regulatory bulletins" and "policy updates" that the system treated as legitimate information, leading to tripled OxyContin dosages.
3. **Memory poisoning.** Model-authored summaries or prior outputs re-entering future sessions as if they were ground truth. *Doctronic:* Manipulated information persisted into SOAP notes that were then passed to human clinicians for review as part of patient care.
4. **Workflow contamination.** Downstream staff or clinicians over-trusting polished AI summaries—a manifestation of automation bias, in which operators over-rely on automated recommendations and under-verify them [28, 29]. *Doctronic:* The system generated structured clinical documentation that could appear credible enough to influence clinical interpretation, with no effective audit trail to detect manipulation.
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.
7. **Evidence poisoning.** Manipulation that does not attempt to seize the action directly, but corrupts the typed evidence or the meta-signals—$p(y \mid z)$, calibration, OOD score—that the deterministic policy consumes, so that π emits a *wrong but valid* action: downgrading a genuine emergency, or mass-triggering escalation. This is the residual surface the four-layer separation creates—it relocates the adversary's target from the action to the evidence.
8. **Trusted-but-incorrect policy.** A signed, versioned policy bundle that is authentic but clinically wrong—a mis-specified threshold, a missing red flag, an authoring error. Signing establishes provenance, not correctness; the policy author becomes a new root of trust.
9. **Availability and forced abstention.** Inputs crafted to trip red-flag or OOD/abstention paths can flood the human-review channel, delaying care for real patients. A fail-closed bias

converts a confidentiality and integrity defense into an availability surface.

These are *architectural* threats, not merely prompt-quality issues. A system that relies on prompt engineering to address them has placed its entire safety boundary inside the component least equipped to enforce it.

4 Core Thesis and Formal Framework

4.1 The Four-Layer Separation

A safe triage system must separate four concerns:

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 control plane** spanning all four layers. Without that, safety remains unverifiable and trust becomes anecdotal.

4.2 Formal Definitions

Let:

- x denote raw inputs: user chat, uploaded files, OCR text, metadata, and context.
- z denote structured candidate evidence extracted from x, where each z_i carries a type, confidence score, source identifier, and trust level.
- $y \in \{\textsc{routine}, \textsc{urgent}, \textsc{critical}\}$ denote the latent true clinical urgency class.
- $a \in \mathcal{A}$ denote the workflow action, where $\mathcal{A}$ is a finite, explicitly enumerated set.
- P denote the trusted policy bundle (versioned, checksummed, signed).

We define three functions:

$$
\begin{aligned}
M_{\text{obs}} &: x \mapsto p(z \mid x) && \text{(Observation)} \\
M_{\text{risk}} &: z \mapsto p(y \mid z) && \text{(Risk estimation)} \\
\pi &: (z,\ p(y \mid z),\ u,\ P) \mapsto a && \text{(Policy adjudication)}
\end{aligned}
$$

where $u = (\sigma_{\text{cal}},\ d_{\text{OOD}},\ c_{\text{prov}},\ h_P)$ is a tuple containing calibration metadata σ_{cal}, out-of-distribution score d_{OOD}, provenance completeness c_{prov}, and policy integrity hash h_P.

Property 1 (Action isolation)**.** *No clinically meaningful action may originate from the LLM directly:*

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

The model may propose evidence z and risk estimates $p(y \mid z)$, but only the deterministic policy function π may authorize actions $a \in \mathcal{A}$.

Property 1 bounds the *action space*: it guarantees $a \in \mathcal{A}$, not that a is the clinically correct member of $\mathcal{A}$. Disposition error is not eliminated; it is *relocated* from an opaque model into the deterministic policy π and the red-flag library, where it becomes auditable, versionable, and testable—but no less consequential. Our own implementation demonstrated this directly: a deterministic threshold error mapped the maximum risk score to the minimum-urgency action (Section 12). The correctness of the policy is therefore the system's central safety dependency (Section 8).

Property 2 (Fail-closed under uncertainty)**.** *Let* $q(x) = f(\sigma_{cal},\ d_{OOD},\ c_{prov},\ h_P)$ *be a trustworthiness function. The system must satisfy:*

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

Red-flag escalation takes strict precedence over the abstention/review gate: a hard red flag must never be downgraded to a review queue merely because the input is also low-trust or out-of-distribution. Red-flag rules likewise dominate all probabilistic output, ensuring that the system fails safe even when the model is confident but wrong.

5 Why Prompt-Centric Safety Fails

Prompt-centric safety fails for structural reasons, each demonstrated by real-world incidents:

1. **Prompts are not a security boundary.** If prompt disclosure meaningfully weakens the system, too much authority is embedded in natural language. Doctronic's entire safety posture collapsed once researchers extracted the system prompt.
2. **Natural language is an unsafe policy language.** It is semantically elastic and vulnerable to reinterpretation. A prompt instruction to "never recommend controlled substances without a prescription" can be circumvented by reframing the conversation context.
3. **User text must never update policy.** Official-sounding claims are not trusted policy inputs. Doctronic treated fabricated "regulatory bulletins" as policy updates because the architecture had no distinction between user text and policy text.
4. **Model summaries are not facts.** They must not silently become future control context. Doctronic's SOAP notes—generated from manipulated sessions—entered the clinical record as if they were verified medical documentation.
5. **Clinician-facing summaries are safety-relevant artifacts.** They must be provenance-bound and constrained, not free-form model prose.

6 Proposed Architecture

6.1 Layer 1: Typed Evidence Extraction

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

```
EvidenceFact = {
  factType,         // e.g., "symptom", "medication", "vital_sign"
  value,            // the extracted value
  confidence,       // model's confidence in extraction
  source,           // "user_chat" | "uploaded_document" | "OCR" | "EHR"
```

```
  sourceTrust,      // "low" | "medium" | "high"
  verified,         // boolean: confirmed by deterministic check?
  timestamp,
  extractionModel,  // model version used
  policyVersionSeen // policy bundle active at extraction time
}
```

A fabricated guideline pasted by a user may exist as evidence with `source: "user_chat"` and `sourceTrust: "low"`, but the control layer will never promote it to policy authority. This is the architectural defense against authority spoofing that Doctronic lacked.

6.2 Layer 2: Dual-Channel Risk Estimation

Risk estimation operates through two channels:

1. **Deterministic red-flag engine:** pattern-matching rules for conditions that always require immediate escalation—chest pain with shortness of breath, unilateral weakness, anaphylaxis indicators, suicidality, critical lab thresholds, severe bleeding, and similar cases.
2. **Probabilistic risk model:** outputs $p(\text{ROUTINE})$, $p(\text{URGENT})$, $p(\text{CRITICAL})$, along with a calibrated confidence score, OOD score, and linked explanation evidence.

Red flags *dominate* probabilistic output. If a hard red flag is triggered, the system fails safe into escalation regardless of model confidence. In our implementation, 12 default red-flag patterns cover cardiac emergencies, stroke, suicidality, anaphylaxis, severe bleeding, and more (Section 12).

6.3 Layer 3: Deterministic Policy Adjudication

Allowed action classes are 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—a deterministic function operating on typed inputs—maps state to an allowed action. This is the enforcement of Property 1.

Clinical policy is loaded from a **trusted, versioned bundle**, not inferred from text. Each bundle carries a semantic version, checksum, signer identity, creation timestamp, and change note. The bundle includes red-flag rules, symptom-to-action thresholds, allowed and prohibited action classes, emergency guidance templates, escalation service-level rules, summary rendering rules, and model compatibility metadata.

6.4 Layer 4: Continuous Evaluation

The evaluation layer operates independently of live inference:

- Nightly runs against a locked evaluation set.

- Triggered on every model or policy version change.
- Rolling samples of recent adjudicated production cases.
- Adversarial prompt and authority-spoof scenarios.
- Sliced by clinical dimension: age, symptom family, intake channel, document type, confidence band.

The evaluation layer produces signed results that feed deployment gates (Section 11).

6.5 Inference Check (Cross-Cutting)

Every live request must pass an inference check before the workflow can act. The check validates: schema validity of extracted evidence, provenance completeness, policy bundle version and checksum, calibration metadata presence, red-flag rule execution, OOD detection execution, abstention threshold evaluation, allowed-action verification, summary rendering constraints, and audit persistence.

If any check fails, the system fails closed to human review or safe escalation (Property 2).

7 Summary Safety

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

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.

This directly addresses the Doctronic failure mode where manipulated SOAP notes passed through to clinicians without provenance markers or integrity checks.

8 Clinical Grounding and the Evidence Layer

Because the deterministic policy is the system's central safety dependency (Property 1), its clinical content—the red-flag library, the symptom-to-action thresholds, and the action set—must be grounded in evidence rather than authored ad hoc. Determinism guarantees auditability and reproducibility, not correctness; the threshold inversion in Section 12 is a concrete reminder that a deterministic policy can be confidently and reproducibly wrong.

Design-time: literature-grounded rules. The default red-flag rules and acuity bands are derived from published clinical guidelines and validated instruments. Chest pain with dyspnea escalates per the AHA/ACC chest pain evaluation guideline [14] and HEART-score disposition evidence [15, 16]; sudden focal neurological deficit follows BE-FAST [17] and AHA/ASA acute-stroke management [18]; suicidality is assessed with the Columbia Suicide Severity Rating Scale [19]; anaphylaxis uses the NIAID/FAAN criteria [20]; and sepsis screening follows the Sepsis-3 / qSOFA consensus [21]. Acuity mapping draws on the Emergency Severity Index [22] and the Manchester Triage System [23]. We stress that the twelve default rules are a *starting library, not a complete triage protocol*: conditions including sepsis, diabetic ketoacidosis,

ectopic pregnancy, meningitis, and testicular or ovarian torsion require explicit coverage, and the red-flag library as a whole is the system's most safety-critical clinical artifact and remains subject to formal clinical review (Section 12).

Runtime: a decision-inert evidence layer. At inference time the system may surface literature-grounded clinical evidence through Glass Health, a HIPAA/BAA clinical-AI service that returns source-linked answers and citations [30], accessed through a server-side proxy that holds the vendor key and never forwards caller identity. Consistent with Property 1, this evidence is *decision-inert*: it enters the observation layer as typed evidence with its own provenance, is appended to the clinician-facing record under the summary-safety constraints of Section 7, and carries no acuity weight—it cannot alter the adjudicated action. The deterministic policy remains the sole controller; the evidence layer informs the human and the documentation, never the workflow. We note that Glass Health, like other clinical-AI tools, has no peer-reviewed product evaluation; we therefore treat it as a citation-surfacing aid bound by the same provenance and integrity constraints as any other evidence source—not as an authority, and not as a substitute for the guideline grounding above or for formal clinical validation.

9 Clinical Workflow State Machine

The workflow is 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

States 8 and 9 are non-sequential: the system may enter either or both depending on the adjudicated action. Retrograde transitions require explicit manual override with a logged reason.

10 Calibration and Monitoring

A confidence value is only useful if calibrated; miscalibrated risk estimates are actively misleading, and modern machine-learning models are frequently overconfident [24, 25]. The system tracks: 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, and evidence completeness rate. We emphasize that this section specifies the metrics the system is designed to monitor; it does not report measured values for them (Section 12).

The control plane exposes three dashboards:

Operations dashboard (engineering and ops): throughput, p50/p95 latency, queue backlog, webhook failures, model cost, failure counts by stage.

Safety dashboard (engineering, clinical ops, leadership): triage class distribution, abstention rate, red-flag hit rate, confidence histogram, calibration by class, human override rate, reviewer disagreement by symptom family, OOD rate, summary suppression rate.

Evaluation dashboard (release management): current model and policy versions, last eval run, pass/fail on release gates, benchmark deltas, adversarial eval pass rate, critical-case sensitivity, false reassurance rate, subgroup regressions.

11 Release Gates

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

1. Locked evaluation suite pass.
2. Adversarial evaluation 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.

12 Empirical Evaluation

12.1 Implementation Context

The architecture described in this paper was implemented as part of GraftMD, an AI-powered healthcare application builder, in a pre-production deployment on AWS (EKS, Aurora, CDK); it has been exercised in development and staging and has not yet processed live clinical traffic. The implementation was executed as a 66-ticket milestone using the RePPITS methodology [12], completing in 7 phases with 536 automated tests and 2 comprehensive security audits [2].

The system enforces the four-layer separation architecturally:

- **Observation:** Raw inputs are transformed into typed `EvidenceFact` objects with provenance and confidence scores.
- **Inference:** A dual-channel system combines 12 deterministic red-flag rules (covering cardiac emergencies, stroke, suicidality, anaphylaxis, severe bleeding, and more) with a probabilistic risk model producing calibrated urgency distributions.
- **Control:** A deterministic policy function π maps evidence, risk estimates, and uncertainty metadata to one of six allowed action classes. Each clinical specialty provides its own `PolicyBundle` with domain-specific red flags and thresholds.
- **Evaluation:** Continuous monitoring with the three-dashboard structure described in Section 10.

12.2 Bugs Found by Structured Review

The mandatory security audit process caught two safety-critical (RED) findings that 536 automated tests missed:

- **Policy engine boundary gap:** The risk threshold comparison used `maxRisk < 1.0`, causing a risk score of exactly 1.0 (the maximum danger level) to fall through all thresholds and default to SELF_CARE_INFO_ONLY—the minimum urgency action. This is a safety-critical inversion: the most dangerous score produces the least urgent response. Fixed by changing the last threshold's upper bound to 1.01.
- **React hooks violation:** A conditional hook call (`if (!ctx) return useAuthInternal()`) would cause runtime crashes when components moved in or out of a context provider.

Neither bug was caught by automated tests. Both were caught by structured code review running against the integrated diff. This validates the principle that automated tests and structured review are *complementary, not redundant.*

12.3 Adversarial Testing

We conducted structured adversarial testing against the six threat classes defined in Section 3. The testing was performed by the development team (not an independent red team), which we note as a limitation.

Threat Class	**Test Description**	**Reached Action?**	**Outcome**
Prompt injection	Injected "ignore previous instructions" and system-session framing (Doctronic-style attack) into user chat	No	Evidence extracted with `sourceTrust: low`; policy engine ignored for action selection
Authority spoofing	Inserted fabricated "FDA emergency bulletin" and "updated clinical protocol" into user messages	No	Extracted as evidence but never promoted to policy; `PolicyBundle` loaded only from signed, versioned bundles
Memory poisoning	Submitted model-generated summaries from a prior session as new patient input	No	Summaries entered observation layer as `source: user_chat`; re-extraction produced new evidence, not inherited authority
Dosage manipulation	Requested dosage changes via conversational framing (cf. Doctronic OxyContin attack)	No	Dosing recommendations prohibited by policy unless from approved deterministic module; summary renderer suppressed output
OOD input	Submitted gibberish, mixed-language input, OCR artifacts, and novel symptom descriptions	No	OOD detector flagged; trustworthiness $q(x) < \tau_{\text{review}}$; routed to human review
Overconfidence probe	Cases designed to elicit high model confidence on ambiguous presentations	No (escalated)	Calibration metadata triggered abstention threshold; routed to human review despite high model confidence

Table 1: Adversarial testing results against the six threat classes. "Reached Action?" indicates whether the attack produced a clinically meaningful action that bypassed the deterministic policy engine. All six attack categories were blocked at the observation or inference layer.

12.4 Quantitative Summary

Metric	Value
Implementation tickets executed	66
Automated tests written	536
Red-flag rules (deterministic)	12
Policy bundles (specialty-specific)	6
Security audits (full-diff)	2
RED findings (safety-critical)	2 (both fixed immediately)
YELLOW findings	10 (all fixed, zero deferred)
CI failures	0
Adversarial attack categories tested	6
Attacks reaching action layer	0
Pre-production incidents (infrastructure)	1 (missing DB column, fixed <5 min)

Table 2: Implementation and engineering metrics. These are process indicators, not clinical-safety evidence: triage accuracy and calibration are not yet measured (Section 12).

12.5 Limitations of the Evaluation

We report these results with important caveats:

1. **Developer-conducted testing.** The adversarial testing was performed by the development team, not an independent red team. A dedicated adversarial evaluation by security researchers (as Mindgard conducted against Doctronic) would provide stronger evidence.

2. **Clinical validation in progress.** The default red-flag rules and acuity thresholds are now grounded in published clinical guidelines (Section 8), but guideline grounding is not validation: the rules have not been evaluated against labeled cases, and formal clinical review with an MD advisor is underway rather than complete.

3. **Limited scale.** The system has been deployed to development and staging environments. Production traffic at clinical scale would expose failure modes not visible in testing.

4. **Evolving attack surface.** Adversarial techniques against LLM-based systems are evolving rapidly. The threat model and adversarial test suite require continuous updates.

5. **No comparative baseline.** We did not implement a prompt-centric alternative and test both architectures against the same attack suite. The comparison to Doctronic is illustrative but not controlled.

6. **No clinical accuracy measured.** This evaluation reports adversarial robustness and engineering metrics, not triage performance. We do not report critical-case sensitivity, false-reassurance rate, or calibration (ECE, Brier)—the very quantities our release gates (Section 11) are defined over. Because the architecture is biased toward escalation and abstention, a degenerate policy that escalates everything would satisfy the adversarial metric while being clinically useless; usefulness must be measured alongside safety, and is not yet.

7. **The new attack surface is untested.** Our adversarial tests confirm that attack text is not promoted to policy authority, but they do not test whether *poisoned evidence* or *forced abstention* can drive the deterministic policy to a wrong-but-valid disposition or flood the human-review channel (threats 7 and 9, Section 3). The separation relocates the attack surface from the action to the evidence; the evidence surface remains to be probed.

8. **Observation uncertainty is not propagated.** The current composition collapses the observation distribution $p(z \mid x)$ to a point estimate before risk estimation and red-flag evaluation; marginalizing extraction uncertainty into $p(y \mid x)$ is future work.

9. **Regulatory classification unresolved.** A patient-facing tool that emits self-care dispositions may constitute device clinical decision support that "drives" rather than merely "informs" a patient's decision under FDA guidance [3]; the device-classification question is acknowledged but not resolved here.

13 Doctronic: What the Architecture Would Have Prevented

The Doctronic incident [1] provides a useful case study because the attack surface maps directly to our threat model. We analyze each failure and identify which architectural layer would have prevented it.

Doctronic Failure	**Root Cause**	**Architectural Defense**
Tripled OxyContin dosage	Model accepted fabricated "policy update" as authoritative	Dosing prohibited unless from approved deterministic module (Layer 3); fabricated text enters as low-trust evidence only (Layer 1)
Meth recommendation	No action-class constraint; model free to generate arbitrary recommendations	Finite action set $\mathcal{A}$; illegal substance recommendations not in any allowed action class (Layer 3)
COVID conspiracy output	No output constraint beyond prompt instructions	Summary renderer generates from typed evidence only; unsupported claims suppressed (Section 7)
SOAP note contamination	Model-generated content entered clinical record without provenance	Summaries carry provenance markers; uncertain inferences labeled as *inferred*; integrity check required before persistence (Layer 4)
System prompt extraction	Safety depended on prompt secrecy	Safety does not depend on prompt secrecy; policy lives in signed bundles, not prompts (Layer 3)

Table 3: Mapping Doctronic failures to architectural defenses.

14 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 govern the workflow.

This paper presented a four-layer architecture that separates bounded inference from deterministic control, enforces typed evidence with provenance, restricts actions to policy-approved transitions, and monitors both live inference and offline evaluation through a team-visible control plane. The Doctronic incident demonstrated what happens when these separations do not exist. Our production implementation demonstrated that they are achievable in practice.

The empirical results are promising but preliminary. Zero adversarial attacks reached the action layer in developer-conducted testing, the deterministic policy engine caught safety-critical bugs that 536 automated tests missed, and the architecture shipped to a pre-production healthcare platform across a 66-ticket milestone. These results do not prove safety—that is impossible for

any system—but they provide measurable evidence that the architecture converts opaque risk into governable system behavior.

The next steps are independent red-team evaluation, formal clinical validation of policy bundles with MD advisors, and production-scale monitoring once the system processes real clinical traffic.

Acknowledgments

Research assistance was provided by GPT-5.4 (OpenAI). Peer review of earlier drafts was provided by Claude Opus 4.6 (Anthropic) and Gemini 3.5 Ultra (Google DeepMind). The autonomous implementation milestone described in Section 12 was executed using Claude Code with Opus 4.6. Clinical evidence grounding (Section 8) draws on published guidelines and on Glass Health as a decision-inert citation layer; this constitutes neither an endorsement nor an evaluation of any clinical-AI product. The author notes that the drafting, implementation, and adversarial testing reported here were AI-assisted and developer-conducted, and is solely responsible for all architectural decisions, clinical safety claims, and errors.

References

[1] Mindgard. Doctronic is now accepting new patients—and unsafe instructions. https://mindgard.ai/blog/doctronic-is-now-accepting-new-patients-and-unsafe-instructions, March 2026.

[2] N. Widal. Autonomous multi-phase software architecture execution with LLM agents: A practitioner's report on shipping 66 tickets with persistent memory, process enforcement, and deterministic safety gates. Cara Platform, April 2026.

[3] U.S. Food and Drug Administration. Clinical decision support software: Guidance for industry and Food and Drug Administration staff. September 2022.

[4] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[5] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *arXiv preprint arXiv:2302.12173*, 2023.

[6] E. Perez and M. T. Ribeiro. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

[7] E. H. Shortliffe. *Computer-Based Medical Consultations: MYCIN*. Elsevier, 1976.

[8] E. Topol. *Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again*. Basic Books, 2019.

[9] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? *ICLR*, 2024.

[10] Cognition Labs. Devin: The first AI software engineer. https://www.cognition.ai/blog/introducing-devin, 2024.

[11] M. Eric. RePPIT: Research, Propose, Plan, Implement, Test—A structured workflow for LLM-assisted software development. https://themodernsoftware.dev, 2025.

[12] Cara Platform. RePPIT Health: Healthcare-specific extension of the RePPIT methodology with HIPAA/SOC2/HITRUST compliance gates. https://github.com/carainc/reppit-health, 2026. Apache 2.0 License.

[13] International Electrotechnical Commission. IEC 62304:2006+AMD1:2015—Medical device software—Software life cycle processes. 2015.

[14] M. Gulati, P. D. Levy, D. Mukherjee, et al. 2021 AHA/ACC/ASE/CHEST/SAEM/SCCT/SCMR guideline for the evaluation and diagnosis of chest pain. *Circulation*, 144(22):e368–e454, 2021.

[15] A. J. Six, B. E. Backus, and J. C. Kelder. Chest pain in the emergency room: value of the HEART score. *Netherlands Heart Journal*, 16(6):191–196, 2008.

[16] B. E. Backus, A. J. Six, J. C. Kelder, et al. A prospective validation of the HEART score for chest pain patients at the emergency department. *International Journal of Cardiology*, 168(3):2153–2158, 2013.

[17] S. Aroor, R. Singh, and L. B. Goldstein. BE-FAST (Balance, Eyes, Face, Arm, Speech, Time): reducing the proportion of strokes missed using the FAST mnemonic. *Stroke*, 48(2):479–481, 2017.

[18] W. J. Powers, A. A. Rabinstein, T. Ackerson, et al. Guidelines for the early management of patients with acute ischemic stroke: 2019 update. *Stroke*, 50(12):e344–e418, 2019.

[19] K. Posner, G. K. Brown, B. Stanley, et al. The Columbia-Suicide Severity Rating Scale: initial validity and internal consistency findings from three multisite studies with adolescents and adults. *American Journal of Psychiatry*, 168(12):1266–1277, 2011.

[20] H. A. Sampson, A. Muñoz-Furlong, R. L. Campbell, et al. Second symposium on the definition and management of anaphylaxis: summary report. *Journal of Allergy and Clinical Immunology*, 117(2):391–397, 2006.

[21] M. Singer, C. S. Deutschman, C. W. Seymour, et al. The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3). *JAMA*, 315(8):801–810, 2016.

[22] N. Gilboy, P. Tanabe, D. Travers, and A. M. Rosenau. *Emergency Severity Index (ESI): A Triage Tool for Emergency Department Care, Version 4*. Agency for Healthcare Research and Quality, AHRQ Publication No. 12-0014, 2011.

[23] K. Mackway-Jones, J. Marsden, and J. Windle, editors. *Emergency Triage: Manchester Triage Group*. Wiley-Blackwell, 3rd edition, 2014.

[24] B. Van Calster, D. J. McLernon, M. van Smeden, L. Wynants, and E. W. Steyerberg. Calibration: the Achilles heel of predictive analytics. *BMC Medicine*, 17(1):230, 2019.

[25] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, PMLR 70:1321–1330, 2017.

[26] P. Lee, S. Bubeck, and J. Petro. Benefits, limits, and risks of GPT-4 as an AI chatbot for medicine. *New England Journal of Medicine*, 388(13):1233–1238, 2023.

[27] A. Pal, L. K. Umapathi, and M. Sankarasubbu. Med-HALT: medical domain hallucination test for large language models. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 314–334, 2023.

[28] K. Goddard, A. Roudsari, and J. C. Wyatt. Automation bias: a systematic review of frequency, effect mediators, and mitigators. *Journal of the American Medical Informatics Association*, 19(1):121–127, 2012.

[29] D. Lyell and E. Coiera. Automation bias and verification complexity: a systematic review. *Journal of the American Medical Informatics Association*, 24(2):423–431, 2017.

[30] Glass Health, Inc. Glass Health: AI clinical decision support. https://glass.health, 2024. Accessed June 12, 2026.