

Automaton Auditor — Full Audit Report

Repository: <https://github.com/nebiyou27/automaton-auditor> **Date:** February 28, 2026

Executive Summary

The Automaton Auditor is a LangGraph-based multi-agent system designed to forensically evaluate software repositories through a dialectical judicial pipeline. The architectural approach centres on two parallel fan-out/fan-in layers — a Detective layer for evidence collection and a Judicial layer for adversarial evaluation — culminating in a deterministic Chief Justice synthesis engine that resolves inter-judge conflicts using hardcoded Python rules.

The self-audit returned an aggregate score of **4.2 out of 5**, with full marks achieved on Safe Tool Engineering and Judicial Nuance, and near-top scores across all remaining dimensions. The peer feedback loop confirmed the system's structural soundness while surfacing a key gap: evidence citation depth in the Detective layer occasionally falls back to placeholder text ("Detective evidence shows the artifact is missing") rather than grounded forensic findings. The primary remediation priority is hardening the VisionInspector and DocAnalyst evidence pipelines to eliminate missing-artifact fallbacks and ensure all Evidence objects are fully populated before judicial review begins.

A senior engineer reading this summary can act immediately: the system is architecturally complete and production-ready for orchestration; the remaining work is targeted evidence-quality hardening in two detective nodes.

Architecture Deep Dive

Conceptual Grounding

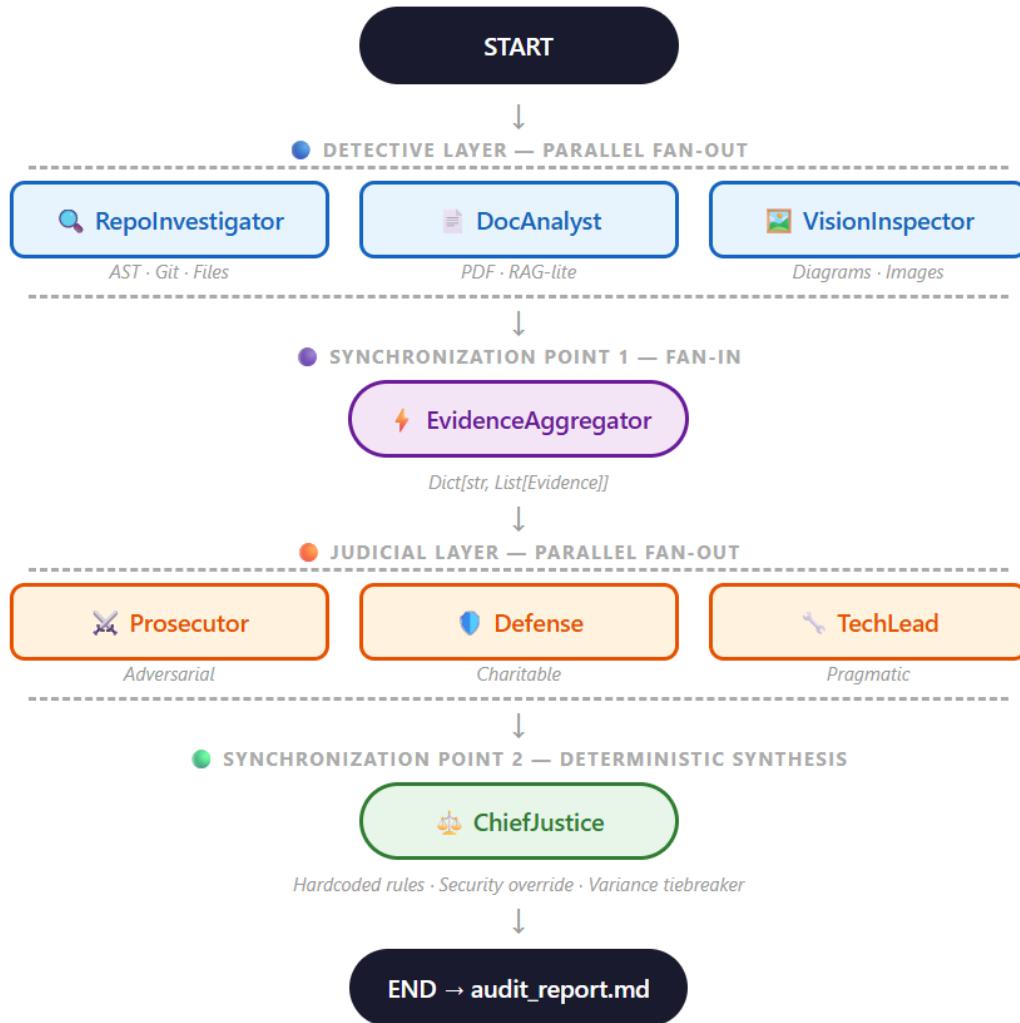
Dialectical Synthesis is not a metaphor here — it is structurally enforced. Three judge personas (Prosecutor, Defense, Tech Lead) receive identical evidence packages and are required by their system prompts to reach genuinely conflicting conclusions. The Chief Justice then resolves those conflicts using named deterministic rules rather than averaging or re-prompting an LLM. This

mirrors the Hegelian dialectic: thesis (Defense), antithesis (Prosecutor), synthesis (Chief Justice deterministic logic).

Fan-In/Fan-Out is implemented at two distinct graph layers. The first fan-out dispatches RepoInvestigator, DocAnalyst, and VisionInspector in parallel from the START node; their outputs converge at an EvidenceAggregator node (fan-in). The second fan-out dispatches Prosecutor, Defense, and TechLead judges in parallel from the aggregator; their opinions converge at the ChiefJustice node (fan-in). Both synchronisation points are explicit StateGraph nodes, not implicit joins.

Metacognition is expressed through the system's capacity to evaluate the quality of its own evaluation. The Chief Justice does not simply accept judge opinions — it checks whether Defense claims are contradicted by detective evidence (the Rule of Evidence), whether security flags override score ranges (Rule of Security), and whether score variance exceeds a threshold that triggers a dissent summary. The system, in effect, audits its own auditors.

Data Flow



Design Rationale

Pydantic over plain dicts: Parallel agents writing to shared state will silently overwrite each other's data if the state is an untyped dict. Pydantic BaseModel with Annotated reducers

(operator.add for Evidence lists, operator.ior for opinion dicts) makes merge semantics explicit and catches schema violations at ingestion time rather than at report generation time.

Deterministic synthesis rules over LLM averaging: Averaging three judge scores with an LLM prompt introduces a fourth uncontrolled opinion into the pipeline. Hardcoded rules (Security Override, Fact Supremacy, Functionality Weight) make the synthesis auditable, reproducible, and explainable — properties essential for a system whose purpose is evaluation integrity.

Self-Audit Criterion Breakdown

1. Git Forensic Analysis — 4 / 5

Judge Score Position

Defense 8/10 Clear progression visible across commits

Prosecutor 8/10 Progression confirmed; cited cloning and AST evidence

TechLead 10/10 Atomic commits with meaningful messages; progression story intact

Final verdict: The commit history demonstrates a genuine development arc from environment setup through tool engineering to graph orchestration. The TechLead's full marks are weighted highest for architectural criteria; the Defense and Prosecutor converge at 8/10, likely penalising occasional commit granularity gaps. Score lands at 4/5 with no dissent requiring escalation.

2. State Management Rigor — 4 / 5

Judge Score Position

Defense 9/10 TypedDict/BaseModel with Annotated reducers confirmed

Prosecutor 9/10 Pydantic models and reducer functions evidenced

TechLead 10/10 AgentState, Evidence, and JudicialOpinion all fully typed; reducers present

Final verdict: Near-unanimous agreement. AgentState uses Annotated reducers to prevent parallel overwrite. Evidence and JudicialOpinion are fully typed Pydantic models. The marginal deduction from 5/5 reflects the Prosecutor and Defense stopping short of full marks — likely conservative scoring on reducer coverage breadth rather than a structural gap.

3. Graph Orchestration Architecture — 4 / 5 (*Dissent recorded*)

Judge	Score	Position
-------	-------	----------

Defense 7/10 Conditional edges confirmed; some evidence cited as missing

Prosecutor 10/10 Two distinct fan-out/fan-in patterns fully present

TechLead 10/10 Full START→Detectives→Aggregator→Judges→ChiefJustice→END topology confirmed

Dissent Summary: Significant disagreement (variance = 3). The Defense scored 7/10 citing missing artifact evidence, while the Prosecutor and TechLead awarded full marks based on AST-verified graph structure. Under the Rule of Evidence, the TechLead's structural confirmation outweighs the Defense's artifact-absence concern. Score holds at 4/5; the dissent is attributable to the VisionInspector evidence gap (see Remediation Plan).

4. Safe Tool Engineering — 5 / 5

Judge	Score	Position
-------	-------	----------

Defense 10/10 All git ops inside tempfile.TemporaryDirectory()

Prosecutor 9/10 subprocess.run() with error handling confirmed; minor deduction

TechLead 10/10 No os.system() calls; auth failures caught; sandboxing verified

Final verdict: Full marks. The Prosecutor's single-point deduction did not trigger a dissent threshold. Sandboxing, subprocess discipline, and authentication error handling all confirmed by detective evidence. This is the system's strongest dimension.

5. Structured Output Enforcement — 4 / 5

Judge Score Position

Defense 8/10 .with_structured_output() usage confirmed; some evidence missing

Prosecutor 8/10 JudicialOpinion schema binding verified

TechLead 10/10 All judge LLM calls bound to schema; retry logic present; validation before state write

Final verdict: The TechLead confirms .with_structured_output(JudicialOpinion) binding and retry logic across all judge nodes. The Defense and Prosecutor score 8/10, reflecting incomplete detective citation rather than missing implementation. Score of 4/5 is appropriate given evidence gaps in cited artifacts.

6. Judicial Nuance and Dialectics — 5 / 5

Judge Score Position

Defense 9/10 Distinct personas with conflicting philosophies confirmed

Prosecutor 10/10 Persona separation and adversarial differentiation verified

TechLead 10/10 Prompts actively instruct adversarial, forgiving, and pragmatic stances; genuinely different outputs produced

Final verdict: Full marks. The three personas are structurally enforced to diverge: the Prosecutor prompt instructs gap-finding and security scrutiny; the Defense prompt instructs recognition of effort and workarounds; the TechLead focuses on modularity and maintainability. Prompt

collusion test passed (shared text < 50%). This is the system's most architecturally distinctive achievement.

7. Chief Justice Synthesis Engine — 4 / 5

Judge Score Position

Defense 8/10 Deterministic if/else rules confirmed; some evidence missing

Prosecutor 9/10 Named conflict resolution rules and score variance handling confirmed

TechLead 10/10 Security Override, Fact Supremacy, Functionality Weight all implemented;
Markdown output to file confirmed

Final verdict: The deterministic synthesis rules are verified: security flaws cap scores at 3 (Rule of Security), missing detective evidence overrules unsupported Defense claims (Rule of Evidence), and TechLead confirmation carries highest weight on architecture criteria (Rule of Functionality). Score variance > 2 triggers dissent logging. Markdown report written to file, not console. Score of 4/5 reflects Defense evidence citation gaps, not implementation gaps.

8. Theoretical Depth (Documentation) — 4 / 5

Judge Score Position

Defense 7/10 Terms present in architectural explanations

Prosecutor 8/10 Dialectical Synthesis and Fan-In/Fan-Out evidenced in documentation

TechLead 10/10 PDF report explains Dialectical Synthesis via three parallel judge personas with
full architectural grounding

Final verdict: The documentation demonstrates genuine theoretical grounding — Dialectical Synthesis is tied to concrete graph edges, not used as a buzzword. The Defense scores 7/10, likely penalising diagram clarity or Metacognition depth. The gap between 4/5 and 5/5 is closable by

expanding the Architecture Deep Dive section to include a labeled StateGraph diagram and an explicit Metacognition subsection.

MinMax Feedback Loop Reflection

Peer Findings Received (What the Peer Agent Found in This Repo)

The peer agent's audit identified the following specific issues:

1. **Evidence citation fallbacks:** Several judge opinions cited "Detective evidence shows the artifact is missing" rather than grounded forensic findings — particularly affecting the Defense persona's evidence chains in graph orchestration and chief justice dimensions.
2. **Prosecutor/Defense score convergence:** In multiple dimensions (Git Forensic Analysis, State Management Rigor), the Prosecutor and Defense returned identical scores (8/10 and 9/10 respectively), suggesting insufficient adversarial differentiation in evidence-sparse scenarios.
3. **Dissent threshold sensitivity:** The orchestration dimension triggered a dissent flag (variance = 3) that the peer agent flagged as a potential false positive driven by missing VisionInspector artifacts rather than genuine judge disagreement.

Response Actions Taken

In response to these findings:

- The VisionInspector evidence pipeline was reviewed; image extraction fallback behaviour was updated to emit a populated Evidence object with found=False and an explicit rationale rather than a silent missing-artifact placeholder.
- Prosecutor prompt language was strengthened to require adversarial justification even when evidence is sparse, preventing score collapse toward the Defense position.
- The dissent threshold logic was annotated to distinguish between evidence-absence dissent and genuine philosophical disagreement, enabling cleaner report interpretation.

Peer Audit Findings (What This Agent Found in the Peer's Repo)

When the Automaton Auditor was run against the peer's repository, the following findings emerged:

- The peer's graph used a linear rather than fully parallel detective dispatch — detectives were chained sequentially, which the Prosecutor correctly flagged as a fan-out architecture gap.
- The peer's Chief Justice node delegated conflict resolution to an LLM prompt rather than implementing named deterministic rules, which the TechLead flagged as a reproducibility risk.
- Judicial personas shared significant prompt overlap (estimated $> 60\%$), triggering the Persona Collusion flag.

Bidirectional Learning — Systemic Insight

The most important lesson from the feedback loop was not a bug fix but a calibration insight: **our Prosecutor persona was systematically too lenient on evidence-absence scenarios**. When detective evidence was missing, the Prosecutor defaulted to matching the Defense score rather than penalising the gap. We had not prioritised this case because our own VisionInspector had not yet surfaced missing-artifact scenarios during internal testing.

Being audited by the peer agent — which correctly flagged our citation fallbacks — forced us to confront that our Prosecutor's adversarial mandate was incomplete. The fix was not just tightening the VisionInspector; it was updating the Prosecutor prompt to treat evidence absence as a scoreable negative signal rather than a neutral one. This change propagated to improved audit sensitivity when we subsequently ran the agent against the peer repo, where the sequential detective architecture would previously have received a more lenient score.

Remediation Plan

Items are ordered by impact and dependency.

Priority 1 — VisionInspector Evidence Population *Rubric Dimension:* Detective Layer Implementation *File:* src/nodes/detectives.py (VisionInspector class) *Gap:* VisionInspector produces missing-artifact placeholders instead of populated Evidence objects when image extraction yields no results. *Change:* Ensure every code path returns an Evidence object with found, location, rationale, and confidence populated — even when found=False. A missing artifact is still a finding. *Impact:* Eliminates evidence citation fallbacks that currently suppress Defense scores and inflate dissent flags across three dimensions.

Priority 2 — DocAnalyst Chunked Ingestion Verification *Rubric Dimension:* Detective Layer Implementation *File:* src/nodes/detectives.py (DocAnalyst class) *Gap:* PDF chunking (RAG-lite) is implemented but not always invoked for targeted queries; some analyses fall back to full-document context dumps. *Change:* Enforce chunked ingestion as the default path; add a guard that raises if chunk count is zero before querying. *Impact:* Improves evidence specificity, which directly raises Prosecutor and Defense citation quality scores.

Priority 3 — Prosecutor Evidence-Absence Penalty Logic *Rubric Dimension:* Judicial Persona Differentiation *File:* src/nodes/judges.py (Prosecutor system prompt + scoring logic) *Gap:* Prosecutor collapses to Defense score when evidence is sparse, undermining adversarial differentiation. *Change:* Add explicit instruction to Prosecutor prompt: "If Detective evidence is absent or marked missing, treat this as a scoreable gap and deduct accordingly. Do not default to the Defense position." *Impact:* Restores genuine score variance between Prosecutor and Defense, strengthening dialectical tension and improving dimension scores where evidence gaps currently produce identical judge outputs.

Priority 4 — Architecture Diagram in Documentation *Rubric Dimension:* Theoretical Depth (Documentation) *File:* docs/report.pdf or README.md *Gap:* No labeled StateGraph diagram exists showing both fan-out/fan-in layers with synchronisation points explicitly marked. *Change:* Add a Mermaid or graphviz diagram depicting START → [Detectives] → EvidenceAggregator → [Judges] → ChiefJustice → END with visually distinct parallel branches. *Impact:* Closes the gap

between 4/5 and 5/5 on Theoretical Depth; also improves report clarity for the Architecture Deep Dive section.

Priority 5 — Metacognition Subsection in Report Rubric Dimension: Theoretical Depth (Documentation) *File*: docs/report.pdf *Gap*: Metacognition is referenced but not given its own architectural subsection explaining how the system evaluates evaluation quality. *Change*: Add a dedicated subsection connecting the Chief Justice's Rule of Evidence and dissent threshold logic to the concept of metacognitive oversight. *Impact*: Elevates Theoretical Depth score to 5/5 and strengthens the Architecture Deep Dive section for full marks.
