

Automaton Auditor Week 2 - Interim Report

1. Executive Summary

This interim submission implements a production-oriented detective layer for the Automaton Auditor.

The current system can clone a target repository in an isolated temporary directory, extract git history, parse project structure with AST-based checks, and ingest a PDF report for chunked analysis.

The LangGraph topology is implemented with detective fan-out and evidence fan-in, ready for judicial layer integration.

2. Architecture Decisions

2.1 Why Pydantic + TypedDict over plain dict

- We use 'Pydantic' models ('Evidence', 'JudicialOpinion', 'AuditReport') to enforce schema-level guarantees and reduce downstream parsing errors.
- We use 'TypedDict' for 'AgentState' to keep LangGraph state explicit and compatible with reducers.
- We use reducers in state:
 - 'operator.ior' for evidence maps so parallel nodes can merge dimension-keyed evidence.
 - 'operator.add' for opinion lists in the upcoming judicial layer.
- Rationale: plain nested dicts become brittle in parallel graph execution and are harder to validate and audit.

2.2 AST-first forensic strategy

- We explicitly avoid regex-only checks for structural verification.
- 'src/tools/repo_tools.py' uses Python 'ast' to detect:
 - 'StateGraph' builder assignment.
 - 'add_edge' and 'add_conditional_edges' call patterns.
 - fan-out indicators through out-degree analysis.
- We also parse 'src/state.py' with AST to verify presence of 'BaseModel' and 'TypedDict' definitions.
- Rationale: AST checks are more resistant to formatting variations and produce stronger evidence quality.

2.3 Sandboxing strategy for repository inspection

- Repository cloning runs in 'tempfile.TemporaryDirectory()' to prevent unknown code from polluting the live workspace.
- 'subprocess.run(..., capture_output=True, check=False)' is used for command execution with explicit return-code checks.
- Failures raise structured exceptions (e.g., clone/authentication failures).
- Rationale: this is safer and easier to reason about than 'os.system'-based shell execution.

3. Current StateGraph (Interim)

- Implemented graph nodes:
 - 'RepoInvestigator'
 - 'DocAnalyst'
 - 'VisionInspector' (scaffolded; execution optional this phase)
 - 'EvidenceAggregator'
- Implemented topology:

- Parallel detective fan-out from 'START'.
- Fan-in aggregation before 'END'.

3.1 Diagram - Planned and Current Detective Flow

START

```
|---> RepoInvestigator ----\
|---> DocAnalyst -----> EvidenceAggregator ---> END
|---> VisionInspector -----/
```

3.2 Planned Final Flow (for Saturday milestone)

START

```
|---> RepoInvestigator ----\
|---> DocAnalyst -----> EvidenceAggregator ---> +--> Prosecutor --\
|---> VisionInspector -----/          +--> Defense -----> ChiefJustice -->
END
                                     +--> TechLead -----/
```

4. Known Gaps and Concrete Plan

4.1 Missing judicial layer

Gap:

- 'src/nodes/judges.py' is not implemented yet.

Plan:

1. Add three persona-specific judge nodes (Prosecutor, Defense, TechLead).
2. Enforce structured output via 'with_structured_output(JudicialOpinion)'.
3. Add parser-retry logic for malformed outputs.
4. Run judges in parallel on the same evidence per criterion.

4.2 Missing deterministic synthesis engine

Gap:

- 'src/nodes/justice.py' is not implemented yet.

Plan:

1. Implement 'ChiefJusticeNode' with deterministic Python rules:
 - security override
 - fact supremacy
 - functionality weighting
 - dissent requirement
2. Add variance-based re-evaluation ('score variance > 2').
3. Serialize final 'AuditReport' to markdown output.

4.3 Missing end-to-end courtroom graph

Gap:

- Current 'src/graph.py' contains only the interim detective pipeline.

Plan:

1. Add judicial fan-out/fan-in after 'EvidenceAggregator'.
2. Add conditional edges for evidence-missing and node-failure routes.
3. Add end-to-end run path from input artifacts to generated markdown report.

5. Validation Status

- 'src/state.py' contains typed models and reducers.
- 'src/tools/repo_tools.py' includes sandboxed clone, git log extraction, and AST graph analysis.
- 'src/tools/doc_tools.py' includes PDF ingestion + chunked query helper.
- 'src/nodes/detectives.py' outputs structured 'Evidence' objects.
- 'src/graph.py' compiles as a partial parallel detective graph with evidence fan-in.

6. Submission Note

This interim report reflects the state of the repository at the interim checkpoint.

The final submission will extend this architecture into full dialectical judicial orchestration and deterministic synthesis.