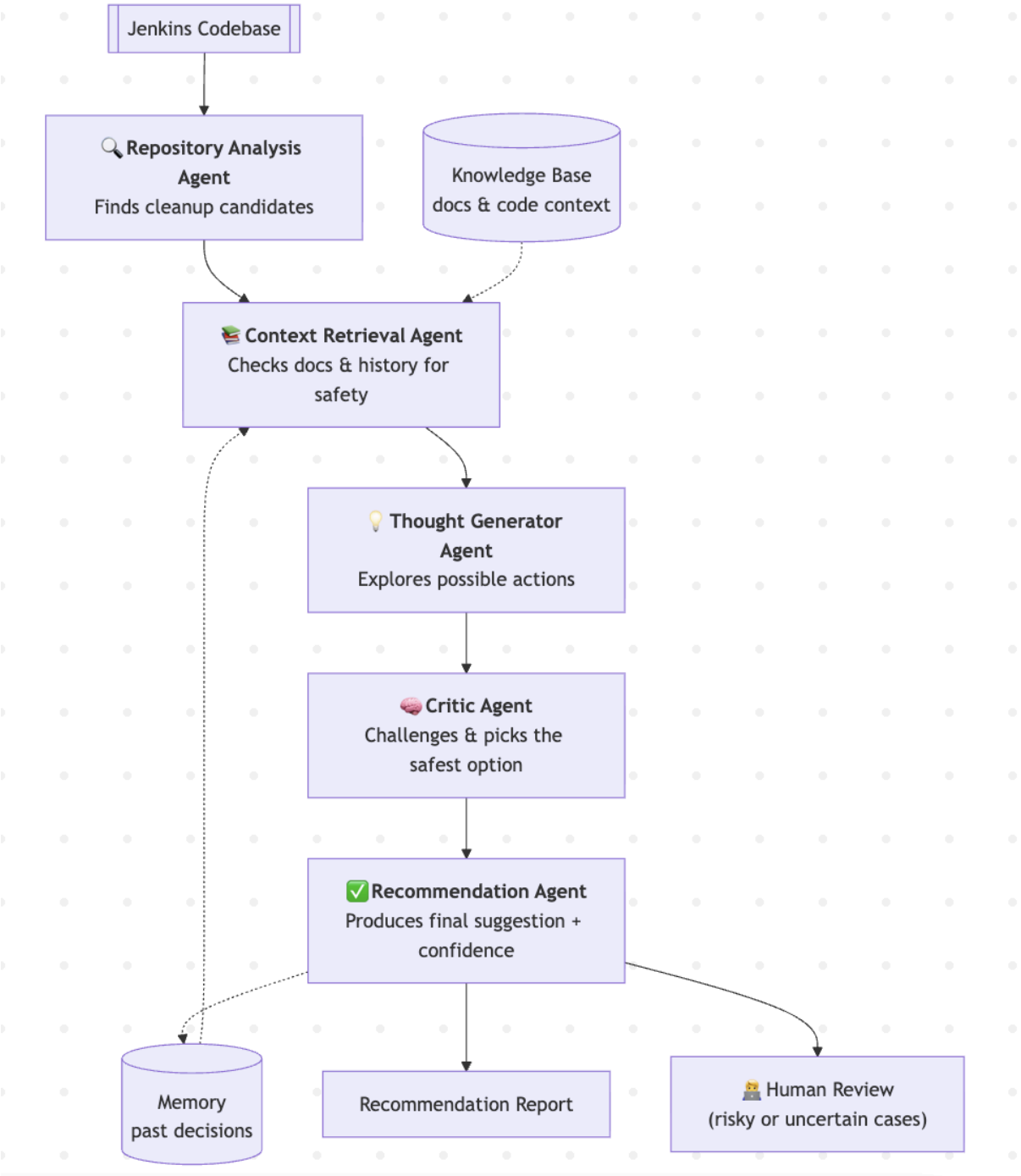
Enterprise Code Cleanup and Modernization Agent

I recently worked on building an enterprise code cleanup and modernization agent for safe technical debt reduction in large Java codebases. The problem is interesting and has weight because as teams grow multiple developers work on the same codebase which start collecting debt. This debt can be as simple as unused imports or variables to unused or deprecated methods to risky dependencies or rule violations, which start creating code quality issues during CI pipelines.

As part of my learning of multi-agent systems, I worked on this agentic design which is nothing but a set of agents orchestrated to come up with a code cleanup recommendation with a decision, confidence score and diff.



The codebase goes in, five agents each do one job in sequence — find issues, gather context, brainstorm options, critique them, and recommend — and the system checks its own memory of past decisions along the way so it doesn't repeat mistakes or re-flag things a developer already dismissed. Anything risky gets kicked to a human instead of being auto-recommended.

Agents in detail:

1. Repository Analysis Agent
Scans the target codebase (Jenkins, 500k+ lines) using static analysis tools — PMD, SpotBugs, and jdeps — to find cleanup candidates: dead code, unused imports, risky dependencies, and rule violations. It ranks these candidates by severity and confidence so the pipeline works through the highest-value issues first. However, this alone cannot be trusted in terms of cleanup as some classes are dynamically loaded and removal can break things.

2. Context Retrieval Agent
Before deciding anything, this agent gathers evidence. It queries a ChromaDB-backed knowledge base (indexed docs, Javadoc, config guides) using RAG to understand what a piece of code is actually for, and checks memory for whether this file or rule has been seen — and judged — before. It outputs a preliminary verdict: SAFE, RISKY, or UNCERTAIN and confidence: HIGH, MEDIUM or LOW.

3. Thought Generator Agent
Rather than jumping to one answer, this agent brainstorms — using a Tree-of-Thoughts approach, it generates several possible actions for a candidate (e.g., remove, refactor, keep, deprecate, flag for review), each with its own reasoning path. Each path is self-rated for the feasibility, safety, impact and overall score.

4. Critic Agent
Acts as a skeptic. It reviews the branches from the Thought Generator, challenges the weaker assumptions, and prunes down to the single safest, best-supported option — reducing false positives before anything reaches a human.

5. Recommendation Agent
Produces the final, human-readable output: the recommended action, a confidence score, the rationale behind it, and a sample diff showing what the change would look like — without ever touching the actual code.

Supporting systems:
- **Knowledge Base** — the RAG index the Context Retrieval Agent draws on for grounding.
- **Memory** — a persistent record of past decisions and developer overrides, so the system doesn't re-flag something a developer already dismissed, and gets smarter across runs.
- **Human Review** — anything flagged risky, low-confidence, or a removal gets escalated here instead of being auto-approved; humans retain final say.

LLM Usage:
All agents except Repository Analysis Agent are dependent on LLM usage. Repository Analysis Agent is purely deterministic and parses output from static analysis tools.
The model used for agents is gpt-4o except o3-mini for critic agent to get a different adversarial perspective.

How final confidence and accuracy is calculated:
Accuracy is not measured against real-world outcomes - it’s measured against a hand-labeled ground truth curated by me. So 95% accuracy means: 19 out of a fixed set of 20 candidates I hand-picked and hand-labeled got the action my labels say is right. It's not

accuracy against an independent test set or real developer decisions — it's a self-defined benchmark.
Confidence is a hand-tuned heuristic (LLM self-rating + rule adjustments + hard caps) rather than a model-calibrated probability. Confidence (per-item scores, avg 0.55).

- +/- adjustment from the RAG verdict: SAFE: +0.1, UNCERTAIN: -0.2, RISKY: -0.25.
- +/- adjustment from the RAG's own stated confidence: HIGH: +0.05, MEDIUM: 0, LOW: -0.1
- Hard cap by action type — a deliberate design choice, not learned: REMOVE capped at 0.65, REFACTOR at 0.75, KEEP at 0.60 (REMOVE is irreversible so it should never look "very confident").
- 0.6 threshold to decide human-review escalation.