TAE Week 4 Coaching Guide

Agents, MCP, and the Incident Command Capstone

The AI Engineer Program

November 23, 2025

Abstract

Week 4 moves from “model reasoning” to production-grade agent systems. You will pair the **AI Agents & Automation book/slides** with **Part III of the Engineering book** and the new **MCP Capstone Note**. The capstone delivers a practical “Incident Command Agent” that runs on the Model Context Protocol (MCP), orchestrates real tools, captures telemetry, and documents memory. This guide distills what to study, how to pace yourself, and how to review MCP-specific concepts before building the capstone.

Contents

1 Purpose & Scope (Week 4)

This guide coaches you through Week 4, where the spotlight shifts to agentic systems, orchestration, and the MCP-based Incident Command capstone. Focus on the **AI Agents & Automation book/slides** [1, 2], **Engineering book Part III + slides** [3, 4], the **MCP Capstone Note** [5], and the course schedule [6].

> **Week 4 Outcomes**
>
> By the end of the week you should be able to: (1) explain the Observe → Plan → Act → Learn loop and memory touchpoints, (2) implement and instrument an MCP server that advertises tools/resources, (3) build a planner/orchestrator that respects budgets and telemetry, and (4) deliver the Incident Command capstone with logs, replay traces, and documentation.

2 Core Resources

This section lists the Week 4 documents you should keep open. Skim each before deep dives so you always know where to look for clarification while building the capstone.

- **AI Agents & Automation Book (age.html)** [1] — conceptual foundations, loop anatomy, and system patterns.
- **AI Agents Slides (age_slides.pdf)** [2] — high-level framing; use for quick refreshers before study blocks.

- **Engineering Book, Part III (eng.html)** [3] — retrieval, orchestration, scaling, and guardrails that ground the capstone.
- **Engineering Slides, Part III (eng_slides.pdf)** [4] — condensed checklists for observability, evaluation, and deployment.
- **MCP Capstone Note (mcp_agents.pdf)** [5] — detailed briefing on MCP fundamentals, capstone goals, and step-by-step implementation hints.
- **Program Schedule (schedule.html)** [6] — suggested pacing, submission reminders, and context for Week 4 milestones.

3 What You Will Learn

Glance through the bullets below whenever you need to re-anchor your study plan; they capture the essential skills Week 4 expects you to practice.

- **Loop discipline**: Observe → Plan → Act → Learn instrumentation + memory deltas [1].
- **MCP basics**: JSON-RPC handshake, resource listings, tool invocation schemas, telemetry fields [5].
- **Tool engineering**: adapters with validation, auth, sandboxing, metrics, and error propagation [3].
- **Planner/orchestrator design**: FSM vs. LLM planner, budget enforcement, replay logging, correlation IDs [1, 3, 5].
- **Applied case study**: map these ideas onto the Incident Command narrative so you can later design tools/resources with confidence [1, 5].

4 How to Study

Pick the track that fits your schedule and stick to short, focused loops. Each plan balances reading, coding, and engineering hygiene so the capstone never drifts.

Full-Time Track (3–4 h/day)

Use weekdays to absorb the texts and reserve the weekend for concentrated capstone sprints.

- **Mon–Tue (study)**: Agents book/slides on loop anatomy, memory, planner patterns; redraw the Observe → Plan → Act → Learn table in your own words [1, 2].
- **Wed–Thu (study)**: Engineering book/slides Part III (retrieval, orchestration, scaling); list the telemetry/budget metrics you will enforce [3, 4].
- **Fri (bridge)**: MCP note protocol + demo server/client; narrate each JSON-RPC exchange and outline your custom tools/resources [5].
- **Sat (build)**: Implement the MCP server, expose resources (alerts/runbooks/memory), and ship retrieval/diagnostic/summarizer tools.
- **Sun (integrate)**: Add planner/orchestrator logic, telemetry, replay traces, and documentation; run one full loop end-to-end.

Time-Constrained (60–90 min/day)

Even with limited time you can study during the week and save the heavy lifting for the weekend.

- **Mon**: Skim Agents book/slides for loop + memory; capture a one-paragraph summary [1, 2].
- **Tue**: Read Engineering book Part III overview; note the guardrails/telemetry you will log [3, 4].
- **Wed**: Work through the MCP note protocol overview and run the demo server/client once [5].
- **Thu**: Draft your Incident Command design (resources, tools, planner states) and prep configs/secrets.
- **Fri evening**: Set up the repo/venv and scaffold the MCP server entrypoint.
- **Weekend**: Saturday for implementing server + tools + resources; Sunday for planner/memory/logging, replay validation, and README polish.

5 Focus First (Priorities)

These study priorities help you master the Week 4 material before touching the weekend capstone build.

- **Loop & memory fundamentals**: Revisit the Observe → Plan → Act → Learn loop tables and memory notes in the Agents book/slides; be ready to explain each stage from memory [1, 2].
- **Engineering guardrails**: Read Engineering Part III sections on retrieval/orchestration/scaling with emphasis on telemetry, budgets, and evaluation; summarize the guardrails you will reuse [3, 4].
- **MCP protocol trace**: Study the MCP note's JSON-RPC handshake/resource/tool flow; manually trace one demo session and annotate every request/response [5].
- **Tool/resource design**: Draft schemas for alerts, runbooks, diagnostics, and summaries; ensure inputs/outputs/metrics are explicit before you code.
- **Telemetry & replay mindset**: Decide how you will log correlation IDs, budgets, and memory deltas; reread the relevant Engineering sections so instrumentation feels natural once you start building [3].

6 Key Concepts — Concise Recap

Use these mini-cheat-sheets to refresh the main ideas before debugging or refactoring.

Observe → Plan → Act → Learn

- Treat each stage as a contract: observation payload, planner decision, tool invocation, memory delta [1].
- Memory participates throughout (read before planning, write after acting); stray facts belong in structured deltas, not ad-hoc text [1].

MCP Fundamentals

- JSON-RPC 2.0 handshake (`initialize`), `getResource`, and `callTool`; reuse unique IDs for correlation [5].
- Capabilities advertise tool schemas + resource URIs; clients cache them before planning [5].
- Responses must include data + metrics (latency, cost, status) so budgets and dashboards stay accurate [3, 5].

Tooling & Telemetry

- Wrap every tool in an adapter that validates input, enforces timeouts, and records latency/result/side effects [3].
- Record budgets (tokens, ms, dollars) per loop; raise guardrails when thresholds trigger [3].

Engineering Systems View (Part III)

- Retrieval + Orchestration: pair local knowledge sources with LLM/tool calls; always plan for provenance and guardrails [3, 4].
- Scaling + Reliability: monitor latency/cost, add caching/batching, and define fallback paths before deployment [3].
- Evaluation + Governance: log inputs/outputs, tag runs, and create lightweight regression checks so agents remain auditable [3, 4].

7 Review Questions

Answer these prompts in writing or code; they double as lightweight retrospectives on your current progress.

- Outline the data included in each Observe → Plan → Act → Learn stage for an incident-response agent.
- Describe the JSON schema for one of your tools. Which fields carry telemetry and how are errors reported?
- How do you persist memory deltas? Show the structure of one entry and how the planner reads it back.
- Which budgets do you enforce (latency, cost, tokens, retries)? How do you surface budget breaches?
- How would you replay a recorded session without touching live tools? Which files/commands are involved?

8 Practice Checklist

Before dedicating the weekend to implementation, confirm these study exercises are complete.

- Recreate the Observe → Plan → Act → Learn table from memory and explain each stage aloud.

- Summarize (in your notebook) the telemetry/guardrail recommendations from Engineering Part III that you will reuse [3].
- Trace the MCP demo session request-by-request (initialize, getResource, callTool) and annotate what each field means [5].
- Run the demo MCP server/client end-to-end; capture one transcript so you can reference the JSON-RPC flow later [5].
- Write a half-page design brief describing your planned resources, tools, planner states, and logging strategy.

9 Daily Cadence (Example)

Here is a sample pacing plan for weekdays. Use the weekend for the capstone sprints outlined earlier.

Block	Focus	Resource
45–60 min	Loop + memory study	`age.html`, age slides
30–45 min	Engineering guardrails	`eng.html`/eng_slides (Part III)
20–30 min	MCP protocol + journaling	MCP note + personal run log

10 Capstone Project (Week 4)

The Incident Command Agent is the Week 4 deliverable. Keep these guardrails in mind while you design, build, and document your submission. Deliver an **Incident Command Agent** that speaks MCP. The server must expose resources (alerts, runbooks, memory) and tools (retrieval, diagnostics, summarizer). The client/orchestrator cycles through Observe → Plan → Act → Learn, enforces budgets, records telemetry, and produces human-readable summaries or escalations. Provide:

- **Code**: MCP server + orchestrator CLI (or notebook) runnable locally; configuration documented.
- **Artifacts**: sample logs/replays, memory snapshots, example transcripts, and instructions for reproducing the demo.
- **Write-up**: short README describing scenario, tooling, budgets, and how to extend the agent.

11 Key Coaching Prompts

Ask yourself the following questions to challenge your understanding and uncover any weak spots before building.

- Can I explain the last three memory deltas — what triggered them and how the planner uses them?
- For one chosen tool, can I state its JSON schema, telemetry fields, and the tests I would run in isolation?
- Can I narrate a full loop, mentioning every MCP request/response and where I log each event?
- If latency suddenly doubles, what metrics or dashboards would reveal the cause, and how would I respond?

12 Troubleshooting Guide

If something breaks, scan this table before diving into a long debugging session. Most Week 4 issues fall into these buckets.

- **Handshake fails**: confirm server is running, subprotocol set to `mcp`, and messages include `jsonrpc=2.0`.
- **Schema mismatch**: log incoming payloads; add pydantic/jsonschema validation before executing tools.
- **Planner loops forever**: enforce max steps; log budget counters and break on repetition.
- **Memory drift**: timestamp every write; include summary + raw data; add tests that load/replay memory entries.

13 Final Checks Before Submission

Run through this checklist before you package your capstone. It ensures reproducibility and professional polish.

- Run a fresh session end-to-end with logging enabled; archive the transcript.
- Verify README instructions on a clean environment (new venv or Codespace).
- Ensure secrets/configs are externalized (env vars, .env, or config files ignored by git).
- Provide evidence (screenshots or logs) showing budgets, telemetry, and human handoff text.

References

[1] AI Agents & Automation — Engineering Intelligent Workflows. age.html

[2] AI Agents & Automation (Summary) — Slide Deck. age_slides.pdf

[3] AI, ML & Software Engineering (Part III). eng.html

[4] AI, ML & Software Engineering (Summary, Part III) — Slide Deck. eng_slides.pdf

[5] Model Context Protocol Agents: Systems Note & Capstone Guide. mcp_agents.pdf

[6] Program Schedule. schedule.html

Contact

TAE Week 4 Coaching Guide

Agents, MCP, and the Incident Command Capstone

Get in touch:

https://linktr.ee/dyjh