COURSE SYLLABUS · COURSE 08 — HERMES AGENT MASTERY

Course 08 — Hermes Agent Mastery

A 6-modules CyberG7 system — research-backed, build-as-you-go.

THE COURSE AT A GLANCE

The only build-it-yourself course that takes a working engineer from stateless prompting to a self-evolving, multi-agent Hermes workforce running on their own infrastructure — durable systems engineering, not model training, with safety rails (fixed SOUL constraints, Curator rollback, GEPA PR gates) baked in.

Who This Is For

The only build-it-yourself course that takes a working engineer from stateless prompting to a self-evolving, multi-agent Hermes workforce running on their own infrastructure — durable systems engineering, not model training, with safety rails (fixed SOUL constraints, Curator rollback, GEPA PR gates) baked in.

This course is built for:

- Working software engineers (junior to staff) who use AI assistants daily and want to graduate to building their own agent
- Mid-to-senior developers who have hit the assistant ceiling and want compounding, persistent leverage
- Staff and lead engineers who need defensible multi-agent architecture they can run for a whole team
- Indie hackers and technical founders who want an autonomous, self-improving agent on their own infrastructure

You'll feel right at home if any of these sound familiar:

- Re-teaching the codebase and conventions to a stateless chat window every single session
- Solving the same gnarly problem from scratch because the assistant retains nothing across sessions
- Juggling multiple chat tabs because no single session holds the whole task
- Watching peers compound with self-evolving agents while their own workflow stays flat

What You'll Build

By the end, you won't just understand the ideas — you'll have assembled a working system, module by module. Across the course you'll develop:

- The ReAct Runtime Loop
- Operational Constraints: The 90-Turn Cap
- Model and Execution Layers
- The Directory That Holds Everything
- Glossary
- Why Identity Comes First
- The Fixed-Frame Principle
- What Goes in the Four Sections
- Identity and Specialization

Course Outline

Module 1 · Hermes Agent Architecture: Foundations of a Self-Improving Agent

Hermes is not a chatbot with memory bolted on; it is a single AIAgent class that unifies execution, routing, and learning into one coherent runtime loop. Unlike standard AI chatbots that operate with amnesia between sessions, Hermes is built around a core, self-improving learning loop designed to run persistently on local hardware. The framework crossed 90,000 GitHub stars within two months of release, and its core differentiator is the ability to record successful task approaches as reusable skills that survive restarts via a local file system. This module maps the Hermes core — the ReAct loop, the tool interface, the model layer, and the runtime — and has you stand up a minimal working instance and trace one task end to end.

Key themes:

- The ReAct Runtime Loop
- Operational Constraints: The 90-Turn Cap
- Model and Execution Layers

Module 2 · SOUL.md: The Identity Layer That Steers Every Decision

Hermes addresses the generic-bot problem with a dedicated identity layer governed by a single static file: ~/.hermes/SOUL.md. This file occupies slot #1 in the system prompt, loaded before memory, before skills, before anything else. SOUL.md is the fixed frame; memory and skills are the moving parts inside it. Because every memory the agent writes and every skill it creates is filtered through this identity, defining SOUL.md well is the highest-leverage thing you do before building anything else. This module has you write a four-section SOUL.md for your primary agent.

Key themes:

- Why Identity Comes First
- The Fixed-Frame Principle
- What Goes in the Four Sections

Module 3 · Memory Systems That Compound: The Three-Tier Stack

Hermes balances the strict token limits of LLMs against the need for near-infinite historical recall by splitting memory into three distinct tiers. The result is an agent that keeps what must always be in context lean, searches the rest on demand, and optionally plugs in deeper external providers. This is what turns Hermes from a stateless assistant into an institutional memory system: every solved problem becomes a reusable precedent. This module has you design the three-tier stack, build retrieval that surfaces the right memory without flooding context, and run a memory audit.

Key themes:

- Tier 1: System-Prompt Snapshots
- Memory Consolidation
- Tier 2: Persistent Session Search

Module 4 · The Self-Evolving Skill Library: Procedural Memory That Compounds

Where memory handles facts, skills handle procedures. Skills are Markdown files with YAML frontmatter that encode how to do things, and Hermes lets the agent write, test, version, and reuse them autonomously. Through progressive disclosure and a 687-skill Hub spanning 18 categories, an agent gains a growing capability set without paying a token tax for skills it does not use. This module integrates architecture, memory, and identity into one skill-acquisition pipeline and has you install a Hub skill and have the agent author one from scratch.

Key themes:

- Skill Anatomy
- Progressive Disclosure
- The Learning Loop

Module 5 · The Self-Evolution Loop: GEPA, the Curator, and Production-Safe Improvement

The runtime loop captures experience; the evolution loop turns that experience into provably better capability. The Curator keeps the skill library clean, while GEPA makes the best skills demonstrably better using execution traces rather than self-assessment. Together they close the loop that makes Hermes compoundingly better over time. This module wires the reflection cycle — act, critique, extract the lesson, update the agent — installs the safety rails that let the agent evolve its capabilities but never its constraints, and prepares Hermes to run real work with approval gates and cost control.

Key themes:

- The Self-Congratulation Flaw
- GEPA: Genetic-Pareto Prompt Evolution
- The Reflection Cycle and Safety Rails

Module 6 · The Multi-Agent Workforce: From One Agent to a Coordinated Team

The final stage scales Hermes from a single agent to a managed multi-agent workforce of delegated specialists. Profiles give each agent its own isolated identity, memory, and skill set, so you move from writing repetitive prompts to managing a dedicated team with distinct roles. This module forks Hermes into specialists, wires the delegation protocol between them, and runs the management layer — task routing, shared memory, and workforce-level QA — with you as the manager.

Key themes:

- Profiles: Walled-Garden Isolation
- Three Specialist Patterns
- The Delegation and Management Layer

Outcomes

Complete the course and you'll be able to:

- Open run_agent.py and trace one full ReAct turn end to end before changing anything — add a print statement that outputs the turn number and last tool called.
- Write a four-section SOUL.md for your primary agent today: role/persona, communication style, hard limits, and values.
- Audit MEMORY.md and delete anything not needed every single session; keep it under ~1,800 characters to leave room before consolidation triggers.
- Install one skill from the Hub and have the agent author one from scratch to see both ends of the pipeline.
- When an agent consistently fails a task but claims success in its logs, bypass the internal loop and run GEPA to force a trace-based rewrite.
- Start with one agent and clone into specialists only when workflows genuinely demand conflicting skill sets or separate API configs.

> **ENROLL**
>
> Enroll me in H6 Start *Course 08 — Hermes Agent Mastery* today — the full module-by-module system lives at https://hermes-agent-mastery-edu.cyberg7.com.sg.