AI-Powered Engineering Workshop Curriculum

Part 1: Harnesses, Multiplexers & Environment Setup

We will go over configuring your dev environment: pick a harness TUI or GUI, pick and setup multiplexers, learn about non interactive mode and when to go for local models. *Why*? Because tools can make or break next level engineering.

1. **TUI harness fundamentals:** Understanding what a harness is, the difference between GUI and TUI harnesses, and configuring `cmux` or `tmux` for deterministic workspace control and pane management.
2. **Multiplexer deep-dive:** Comparing `cmux`, `zellij`, and `tmux` — when to use each and how to chain them together.
3. **Programmatic harness control:** Leveraging `-p` modes with `--json` output for machine-readable interaction, scripting harness commands, and building tooling on top of TUI harnesses.
4. **Harness-native features:** Exploring Ralph loops, modern `/loops`, `/goal`, and other unique capabilities that distinguish one harness from another.
5. **Worktrees and sandboxes:** Testing multiple branches locally using `portless`, understanding the trade-offs between local and remote development environments.
6. **Local vs remote models:** Surveying the open-source AI model landscape, when and why to self-host or go on-prem, and matching model choices to workload.

Part 2: Agent Anatomy, Context & Memory

Deconstruct any agent: system prompts, tools, MCP, subagents, context lifecycle, compaction, memory strategies, and personal agent use cases. *Why*? Because understanding the fundamentals of an agent, helps building workflows, saving on cost and reducing hallucinations

1. **Anatomy of an agent:** Deconstructing the system prompt, tools and MCP integration, skills, and custom commands — the building blocks of any agent regardless of harness.
2. **Subagents — pros, cons, and harness-native support:** When to delegate to subagents, the trade-offs of nested agents, and how harnesses handle subagent execution natively.
3. **Context management:** Understanding how context fills up, when to reset it, and the role of context compaction in maintaining long coding sessions without losing state.
4. **Memory architecture:** Distinguishing `AGENTS.md` from the system prompt, choosing between skills and `AGENTS.md` for persistent knowledge, and strategies for long-term memory.
5. **Personal agents in practice:** Real-world use cases for personal agents like OpenClaw and Hermes — email management, Slack monitoring, calendar automation, and beyond.

Part 3: Workflows — Coding, Admin & Business

Build deterministic workflows beyond agent loops: predefined steps, replay, artifacts, scorers, and end-to-end automation for coding, admin, and business. *Why*? Because workflows and automate repetitive predefined tasks

1. **What is a workflow (vs agent loops and custom commands):** Understanding predefined steps, deterministic and repeatable output, non-LLM sub-steps, loops, branching, replay and teleportation, artifacts, prompt diffs, and scorers/evals.
2. **Coding workflows:** Building automated pipelines for bug fixing, feature scaffolding, PR review, and security investigation — from trigger to verified result.
3. **Admin workflows:** Automating conversation summaries, meeting minutes, second-brain management, and bridging Linear tickets to PRs.
4. **Business workflows:** Designing end-to-end client request processing, from intake through delivery — integrating data pipelines, reporting, and CRM touchpoints.
5. **Security workflows:** Automated vulnerability scanning, dependency audits, compliance checklists, and audit-trail logging.
6. **DevOps workflows:** CI/CD pipeline generation, IaC review, environment provisioning, and rollback automation.
7. **Testing workflows:** E2E test generation, regression suites, performance benchmarking with scorers, and flaky-test detection.
8. **Data pipeline workflows:** ETL orchestration, data-quality validation, report generation, and dashboard refresh automation.
9. **Documentation workflows:** API doc generation from code, changelog drafting, knowledge-base indexing, and stale-content detection.
10. **Email automation workflows:** Inbox triage, smart replies, newsletter filtering, and priority inbox management.
11. **Personal assistant workflows:** Calendar scheduling, reminder management, task prioritization, and habit tracking.

12. **Social media workflows:** Content scheduling, engagement monitoring, analytics reporting, and cross-posting automation.
13. **Scheduled workflows:** Running workflows on cron-like schedules, recurring automation triggers, time-based execution, and managing scheduled job lifecycles.

Part 4: Full-Stack Agentic Project

Ship a full-stack app using harness, multiplexer, workflows, and agents — compare prompt strategies and build your personal playbook.

1. **Deliver a working application:** Build a complete, deployable full-stack project using the full stack of agentic tools — harness, multiplexer, workflows, and agents.
2. **Prompt and implementation comparison:** Evaluate how different prompts and implementation strategies produce results — emphasizing that *how* you get something done matters more than *what* you build, since outputs are now cheap and accessible.
3. **Reusable prompt library:** Curate a personal collection of battle-tested prompts for planning, building, debugging, and reviewing, ready to copy-paste for future projects.
4. **Workflow playbook and cost-benefit analysis:** Document before/after workflow comparisons, calculated savings, and a personal step-by-step guide for reproducing the project independently.