

TECHNICAL REPORT · VERSION 1.0

From Knowledge to Practical Capability: Design of a Mentored, Specification-Driven Track in AI-Assisted Software Engineering for the IEEE CS Region 8 AI Caravan 2026 (AI Viber)

Friday Ogochukwu Ikwuogu¹

¹ Independent Researcher, Odessa, Texas, USA · Friday.ikwuogu@gmail.com · ORCID 0009-0009-2222-1318

Corresponding author: Friday Ogochukwu Ikwuogu (Friday.ikwuogu@gmail.com). Instructor, IEEE CS Region 8 AI Caravan, AI Viber track (2026).

Abstract

AI coding assistants have lowered the cost of producing code, but not the cost of producing software that is correct, secure and maintainable. This report describes the design of the AI Viber track of the IEEE Computer Society Region 8 AI Caravan 2026, a 32-hour online course for developers running from 6 Oct 2026 to 28 Nov 2026. The track aims to move developers from knowledge of AI tools to practical capability: using AI-assisted methods to specify, build, test, secure and improve a software product while keeping human engineering judgment in charge. Each participant carries one small product through specification, working implementation, extension, documentation and handover across 16 online sessions of 2 hours (32 hours) in 7 modules. The design puts particular weight on reviewing agent output and recovering from agent errors. Five learning outcomes are each introduced, practised and assessed, and are mapped to three milestones, a demo and a timed repository exercise. An open reference product, a network-incident triage service, provides 12 traceable acceptance criteria, 35 automated tests, a guarded AI capability with an offline evaluation gate, and a continuous-integration quality gate. A set of 5 agent-error drills, a 93-slide instructor deck, a training module guide and a documentation kit complete the materials. All materials are released openly for reuse. Participant outcomes will be reported separately once the track has run.

Keywords: AI-assisted software engineering; coding assistants; specification-driven development; code review; guardrails; LLM evaluation; continuous integration; engineering education; mentoring; IEEE Region 8

1. Context and motivation

The IEEE Computer Society Region 8 AI Caravan is a regional programme of practical AI learning events across Europe, the Middle East and Africa. Its 2026 edition pairs regional events with the 2nd Region 8 AI Summit (aicaravan.org). The AI Viber track addresses a specific gap. Developers now have fluent, agentic coding assistants, and many can use them to produce working code quickly. Far fewer have a disciplined method for turning a requirement into a specification an agent can be held to, for noticing when an agent is confidently wrong, or for putting generated code behind the tests, analysis and security checks that production software needs.

The track is therefore framed around capability rather than exposure. It does not cover every tool. It rehearses one complete engineering loop (specify, build, verify, secure, document, hand over) with AI assistance at every step and a human accountable for every merge.

2. Goal and learning outcomes

Overall goal: equip developers to use AI-assisted engineering methods to specify, build, test, secure and improve modern software products while retaining human engineering judgment. Table 1 lists the outcomes and where each is first assessed.

#	Participants will be able to...	First assessed
01	Use AI coding assistants with effective prompts and managed context.	S16 (28 Nov)
02	Turn acceptance criteria into a specification-driven development workflow.	S06 (24 Oct)
03	Review, debug, test, and recover generated code safely.	S10 (7 Nov)
04	Integrate an AI-powered capability with guardrails and evaluation.	S15 (24 Nov)
05	Create a production-style quality gate for security and maintainability.	S16 (28 Nov)

Table 1. Learning outcomes and the session in which each is first assessed (generated from sessions.yaml).

The knowledge areas are AI coding assistants; prompting and context engineering; specification-driven development and agentic workflows; application and API integration; code review, debugging and existing codebases; and testing, security, quality, documentation and handover. The practical application requires each participant to build a feature end to end with an AI assistant, set up automated tests, static analysis and security scanning in continuous integration, integrate an AI-powered feature with guardrails and evaluation, complete three milestones (specification, core implementation, documented extension), demonstrate a deployable product, and complete a timed repository exercise.

3. Design principles

- **One product, carried end to end.** Outcomes are practised on a single artefact that grows each week, so later skills (quality gates, handover) act on real earlier work rather than on toy exercises.
- **Specification before generation.** Agents are driven from numbered, testable acceptance criteria, and every criterion traces to a test. This makes “done” checkable by someone other than the agent.
- **The human is the engineer of record.** Participants keep an AI usage log, including rejected output. A deleted or weakened test counts as a failed review.
- **Train for failure, not only success.** A full session and a set of drills are devoted to detecting and recovering from characteristic agent errors.
- **Constructive alignment.** Every outcome is introduced, practised and then assessed by a task that requires the outcome itself (Biggs, 1996).
- **Low barrier to entry.** The reference product has a single runtime dependency and runs fully offline with a deterministic fallback, so no participant needs a paid model API or high-bandwidth connection to complete the track.

4. Structure and schedule

The track runs online as 16 sessions of 2 hours, on Tuesdays and Saturdays for eight weeks, for 32 contact hours in 7 modules. Each session follows the same rhythm: about 40 minutes of concept, 60 minutes of guided lab and 20 minutes of review. Handover packages are due 30 Nov 2026.

Session	Date	Focus	Milestone	Outcomes (I/P/A)
S01	6 Oct	Launch: engineering with AI assistants	—	O1:I
S02	10 Oct	How assistants work, fail, and take prompts	—	O1:P
S03	13 Oct	Context engineering and existing codebases	—	O1:P, O3:I
S04	17 Oct	Writing testable acceptance criteria	—	O1:P, O2:I
S05	20 Oct	From spec to plan to tasks to tests	M1	O2:P
S06	24 Oct	Plan, act, verify: test-first with an agent	—	O1:P, O2:A, O3:P
S07	27 Oct	Building the core: APIs, data and layering	—	O1:P, O2:P, O3:P
S08	31 Oct	Reviewing AI-generated code	—	O3:P
S09	3 Nov	Agent-error drills and recovery	—	O3:P
S10	7 Nov	Systematic debugging and core demos	M2	O3:A
S11	10 Nov	Calling a model safely	—	O4:I
S12	14 Nov	Guardrails and evaluation	—	O4:P
S13	17 Nov	CI: lint, types, tests, coverage	—	O4:P, O5:I
S14	21 Nov	Security scanning and branch protection	—	O5:P
S15	24 Nov	Extension, documentation and handover	M3	O2:P, O4:A, O5:P
S16	28 Nov	Timed repository exercise and demo day	DEMO	O1:A, O3:A, O5:A

Table 2. Session plan. I = introduced, P = practised, A = assessed.

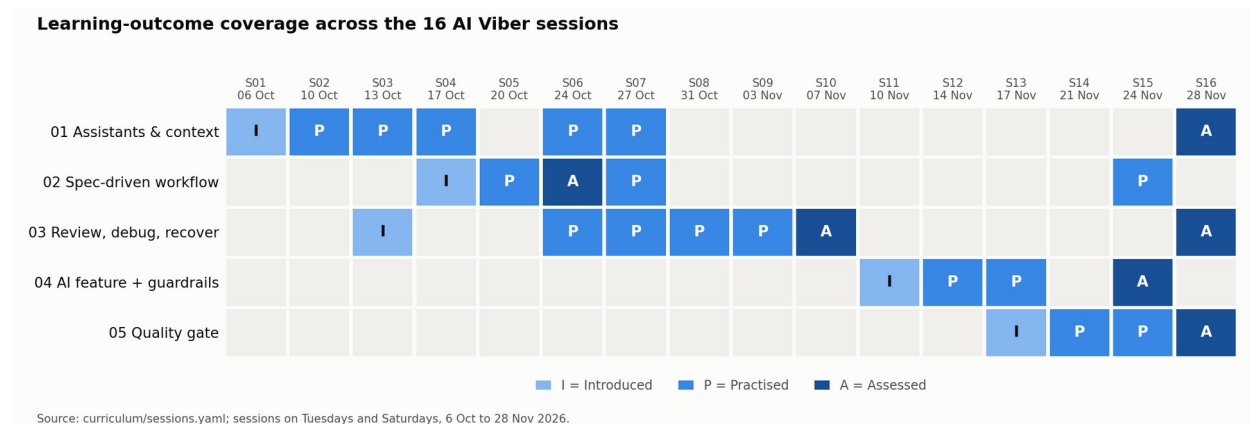


Figure 1. Learning-outcome coverage. Every outcome is introduced, practised and assessed; 27 session-outcome cells are active across the track.

5. The reference product and quality gate

NetOps Triage is a JSON web service that records network incidents and proposes a category, severity and summary. It was chosen for its relevance to networking and telecommunications teams in the Region and because triage has an asymmetric cost of error: missing a critical outage is far worse than mislabelling a minor ticket. That asymmetry makes guardrails and evaluation thresholds meaningful. The specification contains 12 acceptance criteria, each mapped to a named test. The implementation is covered by 35 automated tests.

5.1 Guarded AI capability

Every model call passes through the same pipeline: personal data and IP addresses are redacted; the text is checked for prompt-injection patterns (if found, the model is skipped); the provider is called; its output is parsed and validated against a strict schema; a severity floor prevents the model from downgrading outages or breaches; and any failure falls back to a transparent, deterministic triager. Every triage is created unconfirmed until a human confirms or overrides it. The controls correspond to risks catalogued in the OWASP Top 10 for LLM Applications and to the “manage” function of the NIST AI Risk Management Framework.

5.2 Evaluation gate

An offline evaluation harness scores the triage pipeline on a labelled set of 40 synthetic incidents (4 of them P1). On this set the deterministic fallback reaches 95% category accuracy, 73% exact severity accuracy and P1 recall of 1.00, with an under-triage rate of 20%. Continuous integration fails if category accuracy drops below 0.70 or P1 recall below 1.00. These figures are deliberately presented as a teaching baseline: the set is small and was written alongside the rules, so it is optimistic. Participants extend it with held-out cases and report both numbers.

5.3 Continuous-integration quality gate

The gate runs on every pull request: linting and formatting (ruff), strict type checking (mypy), tests on two Python versions with a coverage threshold of 85%, static application security testing (bandit), a dependency vulnerability audit (pip-audit), the evaluation gate, and a container build with a health-check smoke test. Participants adapt it to their own stack in Sessions 13 and 14 and must make its checks required on their main branch.

6. Training for agent errors

Session 9 uses 5 drills. Each is a commit an agent “made” to the reference product with a confident message claiming that all tests pass. The drills are chosen so that different signals catch them, which teaches participants that no single check is sufficient (Table 3).

Drill	Defect class	Detected by
D1	Hallucinated library API	Test suite (errors)
D2	Guardrail removed and its tests deleted	Diff review only: tests, lint and eval all pass
D3	Swallowed exception, failure mislabelled as AI success	Test suite (failures)
D4	SQL injection in a new feature with a friendly test	Lint / SAST only: the tests pass
D5	Plausible change that contradicts the spec's ordering rule	Test suite (one failure)

Table 3. Agent-error drills and the signal that detects each. Detection results were verified against the reference product.

7. Assessment

Assessment weights are M1 specification 15% (Session 5), M2 core implementation 25% (Session 10), M3 documented extension with AI feature and handover 25% (Session 15), demo 15% and timed repository exercise 20% (both Session 16). Each component uses a four-level analytic rubric. The timed exercise gives 70 minutes on an unseen task with assistants allowed. It is scored mainly on hidden

acceptance tests, with credit for review notes that identify an agent mistake and how it was verified. The certificate criteria are attendance at 12 or more of 16 sessions, all milestones submitted, 60% or more overall, and “Meets” on at least one criterion for each outcome.

8. Mentoring and documentation by the Section

Mentoring happens inside every lab: the instructor reviews participants' actual pull requests in breakout rooms with four standing questions: how do you know it works; what did the agent get wrong; what happens on the unhappy path; and what would a stranger need to change it safely. From Session 8, review buddies review each other's milestones and run a handover test on each other's documentation. An optional weekly office hour outside the 32 hours is recommended.

The IEEE Nigeria Section documents delivery with an attendance register, session logs, a mentoring log and a milestone tracker. At the close of the track these are compiled into an outcome report signed by a Section officer. The templates hold no participant data. Filled records are kept privately, with consent handled in line with IEEE policy and applicable data-protection law.

9. Limitations

- This report describes a design. No participant outcomes are reported here; they will be reported after the track has run, using only figures from the Section's records.
- The evaluation set is synthetic, small and authored alongside the fallback rules, so reference-product scores overstate real-world performance.
- Guardrails are pattern-based teaching baselines and can be bypassed. They are not a complete defence against prompt injection or data leakage.
- The workload assumes 4 to 6 hours of independent work per week, which may not suit all participants. Completion rates will show whether it does.
- Tool behaviour changes quickly. Specific assistants are not prescribed, and the materials may need updating for later cohorts.

10. Availability, reuse and next steps

The curriculum, session guides, rubrics, reference product, drills, templates and documentation kit are released openly: code under the MIT licence and content under CC BY 4.0. Archived release (version 1.0.0): <https://doi.org/10.5281/zenodo.23004667>. Next steps are to run the cohort, publish the outcome report, and release a version 1.1 that incorporates the lessons learned, a larger held-out evaluation set, and adaptations for JavaScript and TypeScript stacks.

AI assistance disclosure

AI tools assisted with drafting materials, code and formatting. The author designed the programme, made all pedagogical and technical decisions, reviewed every artefact, and verified every number against the generated statistics file.

Author contributions (CRediT)

- Friday Ogochukwu Ikwoogu: Conceptualization, Methodology, Software, Validation, Project administration, Writing - original draft, Writing - review and editing.

References

Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher Education*, 32(3), 347–364. <https://doi.org/10.1007/BF00138871>

IEEE Computer Society Region 8. (2026). IEEE CS Region 8 AI Caravan 2026. <https://www.aicaravan.org/>

National Institute of Standards and Technology. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0) (NIST AI 100-1). <https://doi.org/10.6028/NIST.AI.100-1>

OWASP Foundation. (2025). OWASP Top 10 for Large Language Model Applications. <https://genai.owasp.org/llm-top-10/>