AI-Assisted Software Development — 5-Day Workshop Outline

Course Overview

- **Audience:** Software developers, architects, QA, and tech leads familiar with Git and at least one programming language.
- **Format:** 5 days, instructor-led, lectures + demos + hands-on labs, pair programming encouraged.
- **Learning Goals:** Safely and effectively use GitHub Copilot and LLMs to accelerate development, improve tests and documentation, and make low-risk changes to legacy code.
- **Materials:** CODE/AIASD-slides.md, starter repos (calculator), VS Code with Copilot, sample legacy repo forks.

Daily Structure (typical day)

- Morning lecture (60–90m): concepts + demos
- Midday hands-on lab (90m): guided exercises
- Afternoon project/practice (120m): deeper tasks, pair-review
- End-of-day checkpoint (30m): share progress, blockers, deliverables

Day 1 — Foundations & Hands-On Intro

Objectives

- Understand LLM basics, limitations, and safe usage.
- Configure Copilot and use Ask/Edit modes.
- Complete a small coding exercise to experience assisted development.

Morning — Foundations

- Topics: LLM conceptual architecture, tokens & attention, capabilities vs limitations, software engineering vs "vibe coding".
- Deliverable: short discussion/quiz listing 3 risks and 3 mitigations when using LLMs.

Midday — Copilot Overview

- Topics: Ask/Edit/Agent/Custom Chat Modes; instruction files vs prompt files vs chat

Day 2 — Prompting, Testing, and TDD with AI

Objectives

- Master prompt engineering patterns and workspace-level instruction files.
- Use TDD with AI: write tests first, then implement.

Morning — Effective Prompting & Instruction Files

- Topics: crafting multi-step prompts, constraining outputs, examples of `.copilot.json` & prompt files, prompt templates.
- Exercise: convert a vague prompt into a structured multi-step prompt with expected outputs.

Midday — TDD with AI

- Exercise: Write tests for calculator memory/logging features first (tests intentionally fail)

Day 3 — Working with Legacy Code & Risk Management

Objectives

- Learn legacy-code analysis techniques and how to minimize change risk.
- Produce an initial test plan and component summary for a real repo.

Morning — Legacy Code Theory

- Topics: what is legacy code, evergreen code characteristics, safety nets for changes.
- Reading: selected excerpt from *Working Effectively with Legacy Code*.

Midday — Exploration Techniques (Read-only)

- Exercise: Load your forked repo, use Copilot/Chat to summarize modules, surface likely risks, and propose tests to add.
- Deliverable: component summary document and prioritized test plan.

Day 4 — Advanced Copilot Modes, Governance & IP

Objectives

- Use custom chat modes and agents for team workflows.
- Understand IP, privacy, and governance for Copilot in organizations.

Morning — Chat Modes & Agents

- Topics: authoring a Chat Mode persona (code reviewer persona), agent-based workflows for multi-step automation.
- Exercise: create a Chat Mode that enforces tests-first feedback and suggested edits.

Midday — Governance, Security, Legal

- Topics: Copilot for Business/Enterprise protections, public-code filter, secrets policy, licensing risks

Day 5 — Capstone: Legacy Refactor Mini-Project & Review

Objectives

- Deliver an end-to-end vertical slice on a real forked repo using AI assistance.
- Produce evergreen artifacts: tests, docs, CI rules, and instruction files.

Morning — Planning & Triage

- Activity: Teams pick a mini-project from their forks and break it into vertical slices with acceptance criteria.

Midday & Afternoon — Implement Slices

- Activity: Implement at least one full vertical slice: code + tests + docs + PR.
- Deliverable: review-ready PR, test coverage notes, demonstration of behavior and risk mitigation.

Prompts Bank (selected)

- Implement CLI calculator in Golang
- "Write tests for a memory feature that stores previous calculations (TDD)"
- "Explain this module and list likely edge cases and tests to add"
- "Suggest minimal safe refactor to add dependency injection for testability"

Instructor Resources & Templates

- Lecture slides: CODE/AIASD-slides.md
- Lab starter: `calculator` sample repo (include README with run/test commands)
- PR template: summary, risk, test plan, rollback steps
- Team policy checklist: prompt hygiene, secrets policy, public-code filter settings

Evaluation Rubric (brief)

Next Steps

1. Instructor: review the outline and request any customizations (language choices, timing adjustments, extra labs).
2. I can split this into per-day markdown lesson files or produce a printable PDF/slide-ready version — tell me which you want.