Title: Ultimate Practical AI

Level: Intermediate

Already registered? If you signed up for the RAG, tool calling, or advanced optimizations content, don't worry - none of it is going away. I'm reshaping the class to fit the reality of summer 2026, so you get the most out of it and can put it to work on your job or side projects.

Description: AI is everywhere in our workflows now, and that raises the bar for us as Go developers. The tools are powerful - but we still have to ship code that works, and keep up with a field that moves fast. How do we do both?

This workshop is the story of how I figured that out. I started as a skeptic who thought these models were just glorified autocomplete. What changed my mind was learning to treat the model like a pair programmer: an engineer who writes Go incredibly fast, but who has zero feel for the business or the context. That's where you come in - the human in the loop.

For the agent work, we won't be working on toy code. We'll use the **ArdanLabs Service** example as our real application - a production-style codebase that gives the agent real architecture to navigate instead of a hello-world demo.

I'll show you the things I actually do day to day:

- **Context engineering** - getting the right information in front of the agent, and how I work with AGENTS.md/ CLAUDE.md.
- **Skills** - the ArdanLabs Go development skills I rely on, and how to figure out which ones are worth building for your own work.
- **Code reviews and walkthroughs** - putting the agent to work understanding and critiquing code.
- **Testing** - leaning on the agent for integration and black-box testing.
- **Features and refactoring** - adding to existing code and giving it a proper structure.

Structure is what makes or breaks a project in a world of agents. The Service example shows you what good structure looks like - but most real code isn't there yet. So we'll practice the fix: take a deliberately messy, unstructured project and work through restructuring it together, until the agent is noticeably better at working with it than when we started. You'll leave with a process you can take straight back to your own codebase.

For the rest, we'll work in standalone examples, small self-contained pieces of code you can lift out and reuse:

- **RAG** - connecting the service to a model and feeding it your own data with retrieval.

- **Tool calling and function execution** - letting the model call real Go functions to hit APIs, query databases, and trigger workflows.
- **MCP** - exposing those tools the standard way, with a simple MCP server in Go.
- **Optimizations** - speculative decoding, semantic caching, and model routing, the things that make it fast enough to ship.
- **Security** - how to reason about the attack vectors these systems open up, from prompt injection to data exfiltration, with a hands-on look at some practical uses of LLMs in security.

What a student is expected to learn

A repeatable way to put an agent to work on a Go codebase - and the judgment to know when to trust it, when to steer it, and how to set up your projects so it can do its best work.

Prerequisites

You'll bring access to a frontier model - Anthropic, OpenAI, or any other provider with a state-of-the-art model. If you'd rather run locally, we'll use Kronk with the Qwen 3.6 35B A3B model. None of this is mandatory, but having access to a frontier model gives you the best experience - part of the skill is learning to adapt to whatever model you're given.

- Access to a frontier model, or a local setup as described below.
- Direct access to hardware which can run the workshop will be provided during the class. All the code will be OpenAI-compatible and you can run this against any environment, from Kronk to AWS Bedrock or Vertex AI and anything in-between.
- It is expected that you will have been coding in Go for several months.
- Have a functioning **Go environment** installed with Go 1.26 or later.

Recommended Preparation

- Before the workshop, you'll be asked to clone a repository that will be shared with you ahead of time.
- Please read the README.md for installing all the tooling before class.
- The repository will contain some of the code that we'll work on during the class.
- It's recommended that you run **Docker** or any other container runtime as some of the dependencies will be downloaded in that format.
- To save on the bandwidth, and not rely on Internet access during the workshop, the repository will direct you on how to download and cache the models required to run the class.
- Please **email** the instructor, Florin Patan, for assistance.