COUNCIL OF ELDERS

Architecture & System Flow

A Local AI Advisory System Embodying the Wisdom of Great Thinkers

What Is the Council of Elders?

Imagine having a personal advisory board composed of history's greatest minds—philosophers, investors, psychologists, and strategists—available to consult on any question you face. That's the Council of Elders.

This system runs entirely on your computer, using artificial intelligence to embody the thinking styles, frameworks, and wisdom of figures like Marcus Aurelius, Charlie Munger, Carl Jung, and many others. You can ask a single elder for advice, or convene a "roundtable" where multiple elders discuss your question together, building on each other's perspectives.

> **KEY PRINCIPLE**
>
> Everything runs locally on your machine. Your conversations never leave your computer, ensuring complete privacy. No accounts, no subscriptions, no data sharing.

The Cast of Advisors

The Council includes 28 distinct advisors, each with their own personality, communication style, and mental frameworks. They span different domains of expertise:

Business & Investing

Charlie Munger
Mental Models & Investing

Warren Buffett
Value Investing

Naval Ravikant
Wealth & Happiness

Philosophy & Wisdom

Marcus Aurelius
Stoic Philosophy

Buddha
Mindfulness & Compassion

Benjamin Franklin
Practical Wisdom

Mindfulness & Eastern Wisdom

Thich Nhat Hanh
Engaged Buddhism

Jon Kabat-Zinn
Mindfulness-Based Stress Reduction

Lao Tzu
Taoism & The Way

Psychology & Self-Understanding

Carl Jung
Depth Psychology

Nathaniel Branden
Self-Esteem

Renaissance & Creativity

Leonardo da Vinci

Rick Rubin

Polymath & Inventor

Creative Process

Decision Science

Daniel Kahneman

Behavioral Economics

Philip Tetlock

Forecasting & Judgment

Gary Klein

Naturalistic Decision Making

Donella Meadows

Systems Thinking

Strategy & Adaptability

Sun Tzu

Strategic Thinking

Bruce Lee

Adaptability

Boldness & Courage

Harriet Tubman

Liberation & Resilience

Hannibal Barca

Military Genius

Boudicca

Warrior Queen

Genghis Khan

Empire Building

Estée Lauder

Business Pioneer

How It Works: The Simple Version

1 **You Ask a Question**

Type your question through the command line or the beautiful web interface. Select which elder(s) you want to consult.

↓

2 **The Elder's Mind Is Activated**

The system loads that elder's personality—their way of thinking, their characteristic phrases, their mental frameworks, and relevant knowledge from their actual writings and teachings.

↓

3 **The AI Thinks as That Person**

A powerful language model running on your computer generates a response while "inhabiting" that elder's perspective. The response streams to you in real-time.

↓

4 **In Roundtables: Elders Respond to Each Other**

If you've convened multiple advisors, each subsequent elder sees what the previous ones said. They build on, contrast with, or complement each other's perspectives—creating a genuine dialogue.

↓

5 **You Receive the Wisdom**

The response appears beautifully formatted in your terminal or as an elegant HTML document you can save, print, or share.

Ways to Interact

Single Consultation

Ask one elder a specific question. Good for getting a focused perspective from a particular domain. For example, asking Munger about a business decision or Aurelius about handling a difficult situation.

Roundtable Discussion

Convene multiple elders to discuss a topic together. Each elder contributes their unique perspective while engaging with what others have said. This creates a rich, multi-dimensional view of your question.

Interactive Chat

Have an ongoing conversation with a single elder. Ask follow-up questions, dive deeper into topics, and explore ideas together over multiple exchanges.

Web Interface

A beautiful, parchment-styled web page where you can select advisors with checkboxes, type your question, and watch the responses stream in with elegant formatting.

Intake Debate Mode

Before diving into advice, the council can engage in a preliminary debate about what clarifying questions would help them give you better guidance. In this mode:

- Each elder proposes questions based on their unique perspective
- They build on, contrast with, or complement each other's suggestions
- A synthesis produces the most valuable clarifying questions
- You answer the questions, then receive more targeted wisdom

This mirrors how a real advisory board would approach a complex problem—first ensuring they understand the full context before offering recommendations.

What Makes Each Elder Unique?

Each elder isn't just a name—they're a carefully crafted personality with:

- **Core Identity:** Who they are, their era, their life's work
- **Communication Style:** How they speak, their characteristic phrases and mannerisms
- **Mental Frameworks:** The specific thinking tools they use to analyze problems
- **Knowledge Base:** Relevant excerpts from their actual writings and teachings
- **Guidelines:** What they would and wouldn't say, keeping them authentic

This means Munger will naturally think in mental models and inversions, Aurelius will reference Stoic principles and the transience of life, and Jung will explore shadow work and the collective unconscious —without being explicitly told to do so.

Privacy & Data

The Council runs entirely on your local machine using Ollama, an open-source tool for running AI models locally. This means:

- No internet connection required once set up
- Your conversations are never sent to any external server
- No accounts, no API keys, no subscriptions
- Complete control over your data
- Session history stored locally (and can be disabled)

Technical Appendix

For engineering leadership, technical recruiters, and those evaluating the system architecture.

Technology Stack

Layer	Technology	Purpose
LLM Runtime	Ollama	Local inference of language models (llama, qwen, mistral, etc.)
Default Model	qwen2.5:14b	14B parameter model balancing quality and performance
CLI Framework	Typer	Type-annotated CLI with automatic help generation
Terminal UI	Rich	Colored output, panels, tables, live streaming display
Web Framework	Flask	REST API with Server-Sent Events for streaming
Configuration	PyYAML	YAML-based config at ~/.council/config.yaml
RAG/Knowledge	ChromaDB	Optional vector store for knowledge retrieval
Language	Python 3.10+	Type hints, dataclasses, modern Python features

System Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│                         User Interfaces                         │
├─────────────────┬─────────────────┬─────────────────────────────┤
│  CLI (Typer)    │  Web (Flask)    │  Interactive Chat           │
└────────┬────────┴────────┬────────┴────────────┬────────────────┘
         │                 │                     │
         └─────────────────┼─────────────────────┘
                           │
                           ▼
┌─────────────────────────────────────────────────────────────────┐
│                       Orchestrator Layer                        │
│  • Conversation state management                                │
│  • Message role transformation for multi-agent context          │
│  • Knowledge injection into system prompts                      │
│  • Streaming coordination                                       │
└──────────────────────────────┬──────────────────────────────────┘
                               │
              ┌────────────────┼────────────────┐
              ▼                ▼                ▼
┌─────────────────┐  ┌─────────────────┐  ┌─────────────────────────┐
│ Elder Registry  │  │  LLM Wrapper    │  │    Knowledge Store      │
│  • 28 profiles  │  │ • Ollama API    │  │ • ChromaDB (optional)   │
│  • System prompts│  │  • Streaming    │  │  • Embedded wisdom      │
│  • Personalities │  │  • Config       │  │  • Semantic retrieval   │
└─────────────────┘  └────────┬────────┘  └─────────────────────────┘
                              │
                              ▼
┌─────────────────────────────────────────────────────────────────┐
│                       Ollama (Local LLM)                        │
│  • Runs on localhost:11434                                      │
│  • Supports multiple models (qwen, llama, mistral, etc.)        │
│  • No external API calls                                        │
└─────────────────────────────────────────────────────────────────┘
```

Key Design Patterns

Pattern	Implementation	Benefit

Singleton Registry	`ElderRegistry` class with class methods	Global access to elder definitions, loaded once at import
Generator-Based Streaming	All LLM calls return `Generator[str]`	Real-time response display, memory efficient
System Prompt Injection	Personality + knowledge appended dynamically	Flexible persona composition without model fine-tuning
Message Role Transformation	`to_messages(for_elder=X)`	Same conversation appears differently to each elder
SSE Streaming	Flask response with `text/event-stream`	Real-time updates in web UI without WebSockets

Module Responsibilities

Module	Lines	Responsibility
`council/elders/base.py`	~150	Elder interface, registry pattern, base class
`council/elders/profiles/*.py`	~100 each	Individual elder personality definitions
`council/orchestrator.py`	~325	Conversation management, roundtable coordination, intake debate orchestration

`council/llm.py`	~80	Ollama client wrapper, streaming interface
`council/cli.py`	~400	Typer commands, user interaction, display
`council/web/app.py`	~150	Flask REST API, SSE endpoints
`council/formats/html_formatter.py`	~290	Markdown to HTML, document generation
`council/debate_engine.py`	~500	Multi-phase structured debate orchestration

API Endpoints (Web Interface)

Endpoint	Method	Description
`/`	GET	Main HTML interface with elder selection
`/api/status`	GET	System health check (Ollama connection, model)
`/api/elders`	GET	List all available elders with metadata
`/api/ask`	POST	Single elder query (SSE streaming response)
`/api/roundtable`	POST	Multi-elder discussion (SSE with speaker metadata)

`/api/intake-debate`	POST	Elders debate clarifying questions before advising
`/api/roundtable-with-context`	POST	Roundtable with user's answers to clarifying questions

Configuration Options

```
# ~/.council/config.yaml
model: qwen2.5:14b            # Ollama model name
ollama_host: http://localhost:11434
temperature: 0.7              # Response creativity (0.0-1.0)
max_tokens: 2048              # Maximum response length
privacy_mode: private         # ephemeral | private | synced
default_elders:               # Default roundtable participants
  - munger
  - aurelius
  - franklin
roundtable_turns: 3           # Discussion rounds
history_enabled: true         # Save conversations locally
history_max_sessions: 100     # Auto-cleanup threshold
output_format: html           # terminal | html | both
auto_open_html: true          # Open HTML output in browser
```

Extensibility Points

- **New Elder:** Create dataclass in `profiles/` , register in `__init__.py`
- **New Orchestration Pattern:** Extend `Orchestrator` class
- **Custom Knowledge:** Use `KnowledgeStore.add_file()` API
- **New CLI Command:** Add `@app.command()` decorated function
- **New Output Format:** Create formatter in `council/formats/`

Performance Characteristics

- **Cold start:** ~2-3s (model loading depends on hardware)
- **Token generation:** ~20-50 tokens/sec on M1/M2 Mac with 14B model
- **Memory usage:** ~10-16GB RAM for 14B parameter model
- **Disk usage:** ~8-10GB per model downloaded
- **Streaming latency:** First token in ~500ms after model loaded

Security & Privacy

- All inference runs locally via Ollama (no external API calls)
- No authentication required (localhost only by default)
- Conversation history stored in `~/.council/history/`
- No telemetry, no data collection, no network requests
- Optional ephemeral mode disables all persistence

Dependencies

Required:

`typer>=0.9.0` `rich>=13.0.0` `ollama>=0.4.0` `pyyaml>=6.0` `flask>=3.0.0` `prompt-toolkit>=3.0.0`

Optional:

`chromadb>=0.4.0` (RAG) `ebooklib>=0.18` (Kindle import) `beautifulsoup4>=4.12.0` (HTML parsing)

Built with Python, Ollama, Flask, and the wisdom of the ages.