LLM Council Technical Report

Project Information

- Project: LLM Council (local + distributed)
- Requirement reference: LLM Council Local Deployment (offline, multi-machine)
- Team: Yunhao ZHOU, Yesmine BETTAIEB, Sebastien LEVESQUE
- Date: 2026-01-18

Executive Summary

This project implements a complete 3-stage LLM Council workflow that runs fully offline
with local inference and supports distributed deployment across multiple machines via REST.
Stage 1 gathers independent answers, Stage 2 performs peer review with accuracy and insight
scoring (excluding self review), and Stage 3 synthesizes a final response from a dedicated
Chairman service. The system includes a web UI, health monitoring, and robust parsing and
aggregation to keep results consistent even when model outputs vary.

System Architecture

Core components:
- Orchestrator (FastAPI, port 8001): coordinates all stages and aggregates results
- Council workers (FastAPI, /api/chat): handle Stage 1 and Stage 2
- Chairman worker (FastAPI, /api/synthesize): handles Stage 3 only
- Frontend (React + Vite, port 5173): user interface and visualization
- Storage: JSON conversations in data/conversations
- Local inference: Ollama on each worker machine (default port 11434)

Data flow (simplified):
User -> Frontend -> Orchestrator
-> Council workers (Stage 1 and Stage 2) -> local Ollama
-> Chairman worker (Stage 3) -> local Ollama
-> JSON storage + UI rendering

Workflow Details

Stage 1: First opinions
- The orchestrator sends the user query to all council workers in parallel.
- Each worker returns a response along with latency and model info.

Stage 2: Review and scoring
- Responses are anonymized as Response A/B/C...
- Each council model reviews all other responses (exclude self).
- Output format contains three sections:
- EVALUATION: one bullet per response
- SCORES: accuracy (0-10) and insight (0-10), with total = accuracy + insight
- FINAL RANKING: based on total, tie break by accuracy then insight
- The backend parses scores and rankings, then computes aggregate scores and aggregate rankings across reviewers.

Stage 3: Chairman synthesis
- The chairman receives Stage 1 responses and Stage 2 reviews.
- A separate chairman service synthesizes a final answer.

Distributed Deployment (3 PCs)

Recommended role split:
- PC1: Chairman worker
- PC2: Orchestrator + Frontend + Council worker A/B
- PC3: Council worker C/D

Ports:
- 8001 Orchestrator, 5173 Frontend, 8002/8004/8005/8006 Council workers, 8003 Chairman
- 11434 Ollama (local only)

Configuration:
- council_config.json defines worker names and URLs
- worker.env.example and orchestrator.env.example document environment variables

Local Demo (single PC)

One command starts all services (4 council workers + chairman + orchestrator + UI):
- demo_start_local.ps1 -OllamaModel "mistral:7b"

Stop all services by port:
- demo_stop.ps1

Quick health check:
- demo_verify.ps1

Scoring and Aggregation Logic

- Scores are parsed from the SCORES section even if headers are missing.
- Aggregate Scores: average accuracy, insight, and total for each response.
- Aggregate Rankings: average rank position across reviewers.
- Tie handling: if average rank ties, display uses aggregate scores as tie breaker.
- Recommendation: use 3+ council workers for meaningful rankings with exclude-self.

Output Language and Consistency

- Workers enforce English-only output via a high priority system rule.
- If a model returns non-English text, the backend performs a best-effort rewrite.

UI and Observability

- Stage 1 compare and diff view with horizontal scroll.
- Stage 2 merged Scores + Ranking table plus full raw output for transparency.
- Worker status panel with reachability, Ollama health, and busy state.
- Token counts display real Ollama token numbers when available; fallback to estimates.

Persistence

- Conversations are stored as JSON files in data/conversations.
- Assistant messages store stage results plus metadata (label mapping, aggregates).
- Older conversations are upgraded on load for backward compatibility.

Compliance Summary

- No cloud APIs or external inference services.
- All inference runs locally via Ollama.
- Distributed REST architecture with configurable worker topology.
- Chairman runs as a separate service and exposes only /api/synthesize.
- Full Stage 1-2-3 workflow implemented end to end.

Testing and Verification

- Orchestrator health: GET http://localhost:8001/
- Worker health: GET http://localhost:8001/api/workers/health
- Stage 2 robustness: scripts/smoke_test_stage2.py

Known Limitations

- Requires Ollama to be running on each worker host.
- With only 2 council workers, exclude-self causes ranking ties.
- Token counts can differ across models if measured by different tokenizers.

Generative AI Usage Statement

- Tools: Cursor (GPT-5.2 family)
- Usage: code review, debugging, UI updates, and documentation edits