Assignment 1 - LLMs in Multi-Agent Environment

Group Code: LLM_Agents_Tom_Igor_Roie

Submitters:

Tom Ron, ID 301020723

Igor Nazarenko, ID 322029158

Roie Gilad, ID 312169543

GitHub Repository:

https://github.com/tomron87/LLM_Agent_Orchestration_HW1

> ## Self-Assigned Grade: 92/100

Self-Evaluation Justification Form

Students:

Tom Ron, ID 301020723
Igor Nazarenko, ID 322029158
Roie Gilad, ID 312169543

Project Title: Assignment 1 - LLMs in Multi-Agent Environment
Submission Date: November 2025
Self-Assigned Grade: 92/100

Justification for Our Self-Evaluation

We believe our project merits a 92/100 grade based on comprehensive evaluation across all required criteria. Our primary strengths include production-grade three-layer architecture with 89% test coverage across 35 tests, comprehensive documentation suite of 10 files including detailed PRD with KPIs, C4 architecture diagrams at 4 levels, and 7 industry-standard ADRs. We demonstrated academic rigor through systematic parameter sensitivity analysis with Jupyter notebook visualizations. Our security implementation is exemplary with zero hardcoded secrets, comprehensive .gitignore (359 lines), and entropy-enforced preflight validation.

However, we acknowledge honest weaknesses preventing a perfect score. Our prompt engineering documentation in `Prompting_and_Developing.md` lacks depth and detailed examples—we inadequately captured actual prompts and iterative refinement processes. Initially, our weak prompt engineering skills caused substantial time loss as we struggled with Claude and ChatGPT, requiring frustrating iterations before achieving results. Additionally, our research documentation lacks academic citations (3-5 needed), and the Jupyter notebook's LaTeX formulas need expansion. These represent genuine improvement areas identified through self-reflection.

We invested approximately 70 hours on this project, distributed across architecture design, implementation, testing, and documentation. The time investment reflects not just coding hours but a steep learning curve—particularly in mastering AI-assisted development workflows and understanding production-grade patterns. Significant time was spent on iterative refinement: experimenting with three-layer architecture decisions, achieving comprehensive test coverage with proper unit/integration separation, and creating documentation that genuinely serves future developers rather than checking boxes. While the hour count may seem modest, the density of learning and quality-over-speed approach resulted in production-grade deliverables.

Our genuine innovations go beyond the basic Ollama integration requirement. The `Helper Module Pattern` (`ui/components.py`) solves Streamlit's testability problem by separating visual rendering from business logic. Our documentation-as-code approach with `CLAUDE.md` creates a living development guide enabling effective AI-agent collaboration. The sophisticated test architecture with `DummyResp` fixtures enables true unit testing without mocking libraries—a pattern we developed through experimentation. The notice field pattern returns helpful Hebrew guidance within successful responses rather than throwing errors, prioritizing user experience. Finally, our fail-fast configuration validation and Makefile-driven workflow transform a student project into a system you could hand to a junior developer without explanation. These patterns represent architectural thinking, not just feature completion.

The most valuable outcome was our transformation in AI-assisted development. We progressed from vague requests to precise, mission-oriented prompts with clear success criteria. We mastered breaking complex tasks into structured missions, planning architecture through AI conversations, and validating outputs systematically. Designing three-layer architecture through AI-guided iterations taught us both software engineering principles and leveraging AI as a force multiplier, making this a transformative learning experience in next-generation development practices.

Assignment 1: LLM Agent Orchestration

Grade Report

Student ID:	38961
Team:	LLM_Agents_Tom_Igor_Roie
Repository:	https://github.com/tomron87/LLM_Agent_Orchestration_HW1
Assessment Date:	December 01, 2025

FINAL GRADE ■	86%
STATUS	VERY GOOD

■ Your Strong Performance

Great job! You showed strong technical capabilities in this assignment. Your work demonstrates strong understanding of software engineering principles.

■ Key Strengths

■ Perfect research analysis (Jupyter notebook + parameter sensitivity analysis + visualizations)

■ Outstanding documentation (54KB README, full C4 architecture, 36 screenshots)

■ Perfect security (no hardcoded secrets)

■ Excellent testing (48 tests with coverage config)

■ Perfect UI/UX documentation (36 screenshots)

■ Strong version control (12 meaningful commits + prompt docs)

■ Extensive cost analysis (277 mentions across docs)

■ Dedicated extensibility guide

■ Areas for Improvement

• Add quality tools (linting, CI/CD, pre-commit hooks)

• Consider adding plugin architecture for future extensibility

Keep up the good work! With attention to the improvement areas noted above, you can reach the highest tier of performance.

Assignment 1: LLM Agent Orchestration

Grade Report

Student ID:	38961
Team:	LLM_Agents_Tom_Igor_Roie
Repository:	https://github.com/tomron87/LLM_Agent_Orchestration_HW1
Assessment Date:	December 01, 2025

FINAL GRADE ■	86%
STATUS	VERY GOOD

■ Your Strong Performance

Great job! You showed strong technical capabilities in this assignment. Your work demonstrates strong understanding of software engineering principles.

■ Key Strengths

■ Perfect research analysis (Jupyter notebook + parameter sensitivity analysis + visualizations)

■ Outstanding documentation (54KB README, full C4 architecture, 36 screenshots)

■ Perfect security (no hardcoded secrets)

■ Excellent testing (48 tests with coverage config)

■ Perfect UI/UX documentation (36 screenshots)

■ Strong version control (12 meaningful commits + prompt docs)

■ Extensive cost analysis (277 mentions across docs)

■ Dedicated extensibility guide

■ Areas for Improvement

• Add quality tools (linting, CI/CD, pre-commit hooks)

• Consider adding plugin architecture for future extensibility

Keep up the good work! With attention to the improvement areas noted above, you can reach the highest tier of performance.