MULTI-AGENT SYSTEMS, LLMS, RESEARCH AUTOMATION

Multi-Agentic R&D Research Tool: An AI-Driven Multi-Agent System for Automating the Research and Development Lifecycle

M A. Pria Nandhini, S. Atchaya, G. Dharshan, T S. Yugaram

Department of Artificial Intelligence and Data Science, Rathinam Technical Campus, Coimbatore, India

Published October 2026

Abstract

Research and development (R&D) work—surveying literature, identifying gaps, forming hypotheses, running experiments, analysing results, and writing up findings—remains largely manual, repetitive, and time-consuming. This paper presents the design of a Multi-Agentic R&D Research Tool, a system of specialised, cooperating large-language-model (LLM) agents, each responsible for one stage of the research pipeline and coordinated by a central Orchestrator Agent. Given a research topic, the system retrieves and summarises relevant literature through retrieval-augmented generation (RAG) over arXiv and Semantic Scholar, identifies open problems, proposes candidate hypotheses, scaffolds and executes baseline experiments in a sandbox, analyses results, and assembles a structured, cited draft report. A Critique Agent screens intermediate outputs, and four human-in-the-loop checkpoints keep the researcher in control of key decisions. We describe the architecture, agent roles, control flow, and technology stack, and outline an evaluation protocol that compares time-to-first-draft and literature coverage against a manual baseline. This paper reports the system design and evaluation plan; quantitative results will be added after the prototype evaluation is completed.

Keywords: multi-agent systems; large language models; retrieval-augmented generation; research automation; human-in-the-loop; literature review; agent orchestration

1 Introduction

Large language models (LLMs) can summarise papers, write code, and reason over data when used individually. A single "do-everything" agent, however, tends to lose focus and accuracy over long, multi-step research workflows. Multi-agent architectures address this by splitting the workflow into narrow roles—each agent specialised for one job—that communicate through an orchestrator and shared memory [1, 6]. This work applies that pattern to the R&D lifecycle used in academia and industry laboratories: literature review, hypothesis formation, experimentation, analysis, and reporting.

The proposed tool targets students, research scholars, and small R&D teams who need a structured assistant that accelerates the early, labour-intensive stages of a project without replacing the researcher's judgement. The main contributions of this paper are:

- a hub-and-spoke multi-agent architecture in which each agent owns one stage of the R&D pipeline;
- an Orchestrator Agent that plans the workflow, passes context between agents, and enforces human-in-the-loop checkpoints;
- RAG-based grounding over open academic sources to reduce hallucinated citations [4]; and

- an evaluation protocol comparing time-to-first-draft and literature coverage against a manual baseline.

2 Problem statement

Manually surveying literature, tracking related work, designing experiments, and drafting reports consumes a disproportionate share of a researcher's time relative to the novel contribution. Existing tools—reference managers, single-purpose summarisers, and chatbots—each address one narrow piece of the problem and do not exchange information, leaving the researcher to stitch literature findings, data, and analysis into a coherent narrative. There is no lightweight, extensible system that coordinates multiple specialised agents across the full research pipeline while keeping the researcher in control at each checkpoint.

3 Related work

Literature search and summarisation tools built on services such as Semantic Scholar [5] automate retrieval and summarisation but stop there; their findings are not carried forward into hypothesis generation or experimentation. Single-agent LLM chatbots can answer topical questions but lose coherence over long, multi-stage tasks and cannot execute or track experiments. Reasoning-and-acting prompting [7] improves tool use within one agent, yet does not by itself provide role specialisation.

General-purpose multi-agent frameworks—AutoGen [1], LangGraph [2], and CrewAI [3]—supply orchestration primitives such as agent roles, message passing, and tool calling, but are domain-agnostic. Role-based software-engineering agent teams such as MetaGPT [6] and end-to-end research-automation efforts such as The AI Scientist [8] demonstrate the promise of this direction. The present work differs in emphasis: it is a lightweight, extensible pipeline configured specifically for the R&D workflow, with explicit human checkpoints after each major stage and a dedicated Critique Agent.

4 Proposed system architecture

The system follows a hub-and-spoke design (Figure 1). A central Orchestrator Agent receives the research topic and constraints, builds a task plan, and delegates sub-tasks to specialised agents. Agents communicate through a shared context and memory store (a vector database), so later agents can use the findings of earlier ones. A human checkpoint follows each major stage.

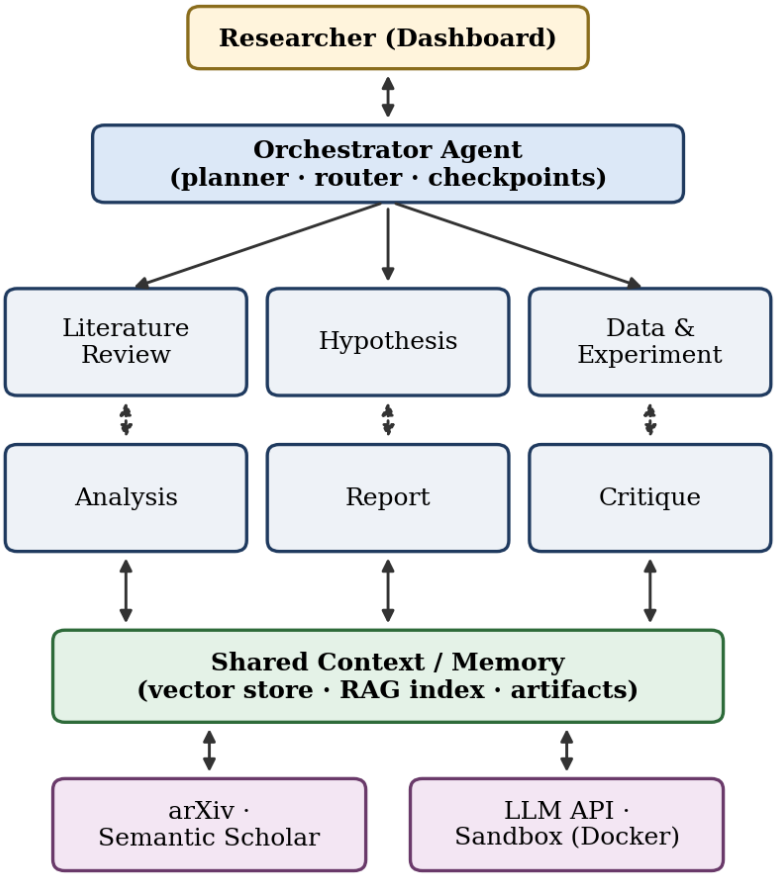


Figure 1. Hub-and-spoke architecture of the Multi-Agentic R&D Research Tool.

4.1 Agent roles

Table 1 summarises the seven agents and their responsibilities.

Table 1. Agent roles and responsibilities.

Agent	Responsibility
Orchestrator	Interprets topic; builds task plan; delegates; merges outputs; enforces checkpoints.
Literature Review	Queries arXiv / Semantic Scholar; retrieves papers; produces structured, cited state-of-the-art summary.
Hypothesis	Analyses literature gaps; proposes 2–3 candidate research questions / hypotheses.
Data & Experiment	Locates / prepares datasets; scaffolds baseline code; executes it in a sandbox.
Analysis	Processes results; computes statistics; generates charts and tables.
Report	Assembles review, hypothesis, method and results into a cited draft report.
Critique	Reviews outputs for coherence, unsupported claims and missing citations.

4.2 Data and control flow

A research session proceeds through the following steps:

- *Input.* The user submits a topic and constraints (domain, time available, preferred sources) through the dashboard.

- *Literature review.* The Orchestrator invokes the Literature Review Agent, which retrieves papers, stores embeddings in the vector store, and returns a cited summary. **Checkpoint 1:** user reviews the summary.
- *Hypothesis.* The Hypothesis Agent proposes 2–3 candidate hypotheses grounded in the identified gaps. **Checkpoint 2:** user selects or edits one.
- *Experiment.* The Data & Experiment Agent prepares data and a baseline experiment for the chosen hypothesis. **Checkpoint 3:** user approves before execution.
- *Analysis and report.* The Analysis Agent summarises results, the Critique Agent checks consistency, and the Report Agent compiles the draft. **Checkpoint 4:** final review.

4.3 Grounding and quality control

To limit hallucinated references, the Literature Review Agent retrieves candidate papers from arXiv and Semantic Scholar and answers from the retrieved passages using retrieval-augmented generation [4]; every citation in the draft report is required to map to a retrieved record. The Critique Agent reviews each agent's output for coherence, unsupported claims, and missing citations before it is shown to the researcher.

5 Methodology and implementation plan

The prototype will be developed incrementally: (i) requirement analysis and finalisation of agent roles and workflow; (ii) comparison of LangGraph, CrewAI, and AutoGen and selection of the base framework; (iii) design of orchestration logic, the shared-memory schema, and the checkpoint interface; (iv) implementation in the order Literature Review, Hypothesis, Data & Experiment, Analysis, Report, and Critique agents; (v) end-to-end integration testing on sample topics; and (vi) evaluation against a manual baseline. Table 2 lists the planned technology stack.

Table 2. Planned technology stack.

Layer	Technology
Orchestration	LangGraph / CrewAI / AutoGen (selected in design phase)
Language model	Claude / GPT API, or open-source model for low-cost runs
Retrieval / memory	FAISS or Chroma; arXiv and Semantic Scholar APIs
Backend	Python, FastAPI
Frontend	Streamlit or React dashboard
Execution	Sandboxed Python in Docker
Versioning	Git / GitHub

6 Evaluation plan

The system will be evaluated on two to three sample research topics in computer science and engineering, where structured digital sources are freely queryable. For each topic, a researcher will produce a first draft manually and another using the tool. Planned measures are: (a) time-to-first-draft; (b) literature coverage, measured as the overlap between papers retrieved by the system and a reference set compiled by domain experts; (c) citation validity, the fraction of cited references that

resolve to real, relevant papers; and (d) expert-rated quality of the generated hypotheses and report. Results of this evaluation will be reported in the final version of this work.

7 Expected outcomes, advantages and limitations

The expected outcome is a working prototype that takes a topic and produces a literature summary, a selected hypothesis, a baseline experiment, and a structured draft report, with a measured reduction in time-to-first-draft. Its main advantages are modularity (each agent can be improved or replaced independently), human control through checkpoints, grounded outputs through RAG, and reusability for other domains.

Limitations are as follows. Literature quality depends on the coverage of the APIs used, and paywalled sources are out of scope. Generated hypotheses and reports require expert review; the tool assists research rather than replacing the researcher. Experiment execution is limited to lightweight, sandboxable code and cannot control physical laboratory equipment.

8 Conclusion and future work

This paper presented the design of a Multi-Agentic R&D Research Tool that coordinates specialised LLM agents across the research pipeline, from literature review to first-draft reporting, with explicit human checkpoints and retrieval-grounded outputs. Future work includes completing the prototype, executing the evaluation protocol described above, and extending the framework to domains beyond computer science and engineering.

Acknowledgments

The authors thank Dr. Dhurgadavi, Professor, Department of Artificial Intelligence and Data Science, Rathinam Technical Campus, for her guidance and support.

References

[1] Wu Q, et al. AutoGen: enabling next-gen LLM applications via multi-agent conversation [preprint]. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2308.08155

[2] LangChain. *LangGraph documentation* [Internet]. https://python.langchain.com

[3] CrewAI. *CrewAI documentation* [Internet]. https://docs.crewai.com

[4] Lewis P, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: *Advances in Neural Information Processing Systems (NeurIPS)*; 2020.

[5] Allen Institute for AI. *Semantic Scholar API documentation* [Internet]. https://api.semanticscholar.org

[6] Hong S, et al. MetaGPT: meta programming for a multi-agent collaborative framework [preprint]. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2308.00352

[7] Yao S, et al. ReAct: synergizing reasoning and acting in language models. In: *International Conference on Learning Representations (ICLR)*; 2023.

[8] Lu C, et al. The AI Scientist: towards fully automated open-ended scientific discovery [preprint]. *arXiv*. 2024. https://doi.org/10.48550/arXiv.2408.06292