Demo: Orchestrating Large Language Model Agents and Resources for Medical Deep Research

Yuan Li*, **Matthew Pan***, **Claire Liu***, **Hengjin Zhu***, **Xijing Wang***, **Yingtao Luo**
yingtaol@andrew.cmu.edu, yuanli4@andrew.cmu.edu
Carnegie Mellon University
*Equal contribution.

Abstract

Deep Research represents the convergence of large language models (LLMs), advanced reasoning, and information retrieval for expert-level inquiry. Existing Deep Research systems are constrained by reliability issues, limited integration with specialized resources, and inflexible output formats. In this paper, we introduce **Medical Deep Research**, an open, agentic system designed for in-depth medical and clinical investigations. Our framework features a multi-agent research module and a resource module that integrates curated medical tools and can dynamically discover Model Context Protocols (MCPs). Through fine-grained design for planning, tool orchestration, query processing, report formatting, and MCP integration, the system supports comprehensive medical information retrieval and customizable output generation. We evaluate this system across four key aspects: resource coverage, tractability, correctness, and helpfulness. Our evaluation results demonstrate the potential of Medical Deep Research to serve as a reliable and powerful platform for medical research. Code available at https://github.com/Clinical-Copilot/Medical_Deep_Research

1 Introduction

Recent advances in large language models (LLMs) have endowed them with impressive reasoning abilities, as demonstrated by OpenAI o-series [1] and DeepSeek R1 [2]. The enhanced reasoning ability unlocks the potential of LLM agents—systems that not only generate text but also tackle complex, real-world tasks. A downstream application of these agents is **Deep Research** [3]. Leveraging powerful reasoning abilities, Deep Research devises in-depth research plans, searches, retrieves, and filters vast online information, then synthesizes the findings into a comprehensive report.

However, existing Deep Research systems face the following challenges that limit their practical use in clinical and medical applications: (1) *Reliability in high-stakes domains*: They rely mainly on general search engines, retrieving only surface-level information (e.g., abstracts and metadata of papers), whereas critical medical insights often reside in full-text articles and specialized repositories; (2) *Limited integration of specialized resources*: Although current Deep Research systems can invoke external tools and Model Context Protocols (MCPs), their connections remain largely hard-coded. As medical and AI research grow increasingly intertwined, we need a dynamic, systematic framework for incorporating specialized resources—one that adapts to task—specific demands and supports complex workflows; (3) *Inflexible output formats*: Outputs of existing Deep Research systems are typically long reports with a uniform tone and structure, which lack the flexibility to present findings in versatile ways based on user requirements.

To address these challenges, we introduce **Medical Deep Research**, an agentic system that performs comprehensive medical investigations by executing searches, leveraging specialized medical tools and MCPs, and accessing biomedical databases. It delivers verifiable information that supports evidence-

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: The Second Workshop on GenAI for Health: Potential, Trust, and Policy Compliance.

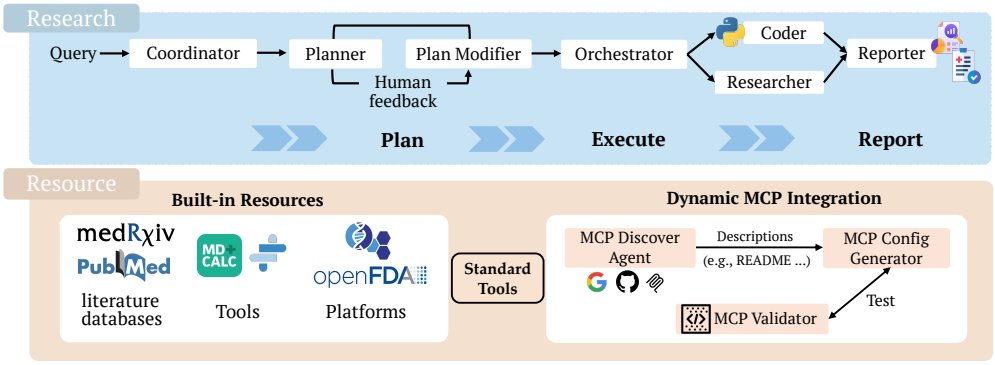

Figure 1: Overview of Medical Deep Research system. The research module is a multi-agent framework that plans, conducts research, and generates reports. The resource module includes a static built-in resource pool and a dynamic MCP-integration pipeline.

based medical practice. Designed for maximum flexibility, our system offers many configurations, including the choice of LLMs, research depth, human feedback loop, and output formats. Overall, Medical Deep Research comprises two core components — a research module and a resource module:

- **Research module.** Built on a multi-agent framework, this module first decomposes each query and drafts a plan that can be refined with human feedback. The agents then execute that plan by orchestrating tasks and invoking medical resources, including tools, MCPs, databases, and platforms. Finally, an specialized agent synthesizes the results and report the findings in the requested format.
- **Resource module.** Medical Deep Research delivers high-quality answers through two complementary resource pillars: (1) *Built-in resources*: A curated catalog of 10+ pre-integrated tools and MCPs–including literature databases, clinical-trial registries, pharmacovigilance repositories, and genomics portals–are embedded in the system for immediate use. (2) *Dynamic MCP integration pipeline*: It automatically discovers new MCP servers from search engines and GitHub repositories. The pipeline then handles authentication and validates discovered MCP servers through a code sandbox before adding it to the resource catalog. This dynamic pipeline ensures that the system remains synchronized with the latest specialized tools and information.

In addition to two core modules, Medical Deep Research offers the following advantages:

- *Comprehensiveness:* Drawing on an extensive pool of medical resources–including literature databases, clinical-trial registries, and other specialized tools, the system could address medical and broader healthcare questions.
- *Usability.* The system is user-friendly. Users can select the LLM backend, set the desired research depth, provide feedback on research plans, and tailor outputs to designated formats. Each configuration is accompanied by clear instructions, making the platform easy for medical practitioners to use.
- *Flexibility.* The system features an agentic, plug-and-play architecture that is easy to extend. In the research module, developers can add agents to modify workflows and assemble custom research pipelines, while in the resource module they can integrate additional tools and MCPs or register new MCP servers. This system provides great flexibility for adaptation and extension.

Impact. Medical Deep Research is an open-source, medically oriented Deep Research system. It demonstrates its potential to serve as a transparent co-pilot for clinicians, researchers, and healthcare stakeholders. It could also serve as a data engine for curating high-quality training corpora with verifiable medical evidence [4]. Furthermore, by incorporating source credibility index such as impact factor, Medical Deep Research enhances the reliability of its evidence base. This integration allows users to prioritize findings from credible papers, improving trust in automated research outcomes. In future iterations, these metrics could also guide model fine-tuning by weighting higher-impact sources more heavily in corpus curation.

2 Medical Deep Research

This section provides a detailed introduction to Medical Deep Research, focusing on the research and resource modules. The overall framework is illustrated in Figure 1.

2.1 Research Module

The research module employs a multi-agent architecture to plan and execute high-quality, in-depth medical research. It can access a wide range of medical tools and MCPs, synthesize findings, and generate outputs in designated formats.

2.1.1 Agentic Architecture

We implement the agentic architecture using the `LangGraph` framework, where specialized agents—Coordinator, Planner, Plan Modifier, Orchestrator, Researcher, Coder, and Reporter—operate as nodes in a directed state graph. The system uses structured message-passing via `Command` objects to manage state transitions and inter-node communication. **Coordinator.** As the system's entry point, the Coordinator initiates the workflow upon receiving user input. It either delegates tasks to the Planner or responds directly to straightforward queries. **Planner.** The Planner agent decomposes tasks and devises execution strategies. Using the argument `--mode`, which accepts `low`, `medium`, or `high`, it controls the granularity of subplans and investigation depth, and determines whether further research is needed. **Plan Modifier.** After the initial plan is generated, the Plan Modifier incorporates user feedback to finalize the research plan. **Orchestrator.** The Orchestrator coordinates the execution phase by routing tasks between the Researcher and Coder agents. It also aggregates information from both and delivers it to the Reporter. **Researcher.** This agent utilizes tools and MCPs to retrieve and analyze medical information. Beyond general web search, it accesses specialized MCP servers and domain-specific databases. Further details on available resources are provided in Sections 2.1.2 and 2.2. **Coder.** Handling code-based tasks such as data analysis and execution, the Coder operates using the `PythonREPL` tool inherited from `LangChain`. **Reporter.** As the final-stage processor, the Reporter consolidates all findings and generates research reports. Implementation specifics on output enrichment are discussed in Section 2.1.3.

2.1.2 Enhancing Researcher Agent's Ability to Use Resources

The Researcher agent is a core component of the system, responsible for retrieving information from a broad range of resources, including various tools and MCPs. Given the scale of available resources, the agent must have a clear understanding of its objectives and make informed selections accordingly. To support this, we adopt the ReAct [5] framework to enhance its reasoning capabilities. Additionally, since different resource calls require specific input formats, a query processor is implemented to ensure input alignment. **ReAct framework.** Researcher agent takes in the research plan and resource descriptions in the prompt, and is responsible for intelligently selecting which resources to use to accomplish the research goals. ReAct [5], a well-established framework, integrates reasoning and action in LLMs. It promotes a "think-then-act" paradigm, where the agent first reasons about the optimal action before executing it. With explicit reasoning traces, this framework enhances agent's ability to make decisions about which resources to utilize. Our implementation is adapted from `LangChain.create_react_agent` function, with the corresponding prompt shown in Appendix B.

Query Processor. Given the diversity of resources, ranging from search engines and medical databases to specialized tool, it is crucial to align tool invocation queries with each tool's expected input format. To facilitate this, we implement a `@query_processor` decorator that wraps tools/MCPs, allowing LLMs to generate dynamic queries based on tool-specific requirements. Our system supports multiple query processing options defined in the `QueryStrategy` enumeration, including `DIRECT`, `PARAPHRASE`, and `EXPAND` option. For example, we use `PARAPHRASE` for general search engines to rephrase a given keyword into semantically equivalent alternatives, since it will improve retrieval of more relevant and comprehensive information [6]. For tools like LiteSense 2.0–which require sentence- or paragraph-level biomedical queries–we apply `EXPAND` to enrich prompts with related biomedical facts, hypotheses, or contextual observations. These query processes ensure that each query is precisely tailored to its target resource, thereby enhancing both retrieval accuracy and coverage.

2.1.3 Enriching Output format

We equip the system with the ability to generate versatile reports tailored to diverse user needs. Specifically, the Reporter agent employs a dynamic prompt generation mechanism, controlled via the `output` argument, to adapt outputs to user-defined formats. It supports three options: `long report`, `short summary`, and a `custom` option that accepts user-specified format descriptions. `long report` and `short summary` use predefined prompt templates. When the `custom`

option is selected, a two-stage process is triggered. First, a prompt engineering LLM interprets the user's intent as a meta-prompt and constructs a tailored prompt. This prompt then guides the Reporter agent in generating the final report with the desired structure, style, and content. This design enables the generation of varied output formats and tones.

2.2 Resource Module

The depth of research depends on the abundance of resources available in the resource module. We first enhance the system by curating a collection of built-in resources, including widely used medical tools and platforms. Also, recognizing that hard-coded resources may not cover the full spectrum of medical topics–and that the number of medical MCP servers is rapidly increasing–we introduce a *dynamic MCP integration pipeline*. This pipeline discovers relevant medical MCP servers and automatically integrates them into the system, enabling continuous expansion and up-to-date access to biomedical resources.

2.2.1 Built-In Resources

We hand-code a curated set of essential resources so the system can reliably access core medical and healthcare information. These resources include literature databases such as `LitSense 2.0` [7], which connects to PubMed and PMC; drug-discovery tools like `ToolUniverse` [8]; clinical-trial repositories such as ClinicalTrials.gov; and regulatory knowledge bases such as DrugBank [9] and OpenFDA [10]. The system could also access authoritative practice guidelines and clinical decision-support tools, grounding its recommendations in the latest clinical research and standards. Together, this curated resource pool provides broad coverage across key medical domains.

The system also incorporates a journal-credibility pipeline (`src/tools/journal_scraper.py`) that scrapes SCImago Journal Rank (SJR) data to build a structured database of over 10,000 journals with ISSN, quartile, and impact-factor metadata. The resulting `biomedical_journals.json` file supports an ISSN-matching module (`src/utils/issn_matcher.py`), verifying whether cited studies originate from peer-reviewed biomedical journals. This integration standardizes journal metadata and enables credibility scoring, distinguishing verified literature from unverified or preprint sources.

2.2.2 Dynamic MCP Integration Pipeline

Beyond the built-in resources, we develop a dynamic MCP integration pipeline that provides an end-to-end solution for seeking and integrating medical MCP servers into our research platform. At its core, this pipeline leverages a multi-agent architecture to search and analyze MCP server documentation and automatically generate configuration files. The pipeline consists of three key components: an MCP Discovery Agent for identifying candidate URLs, an MCP Config Generator for producing standardized configuration files, and an MCP Validator for verifying the usability of the integrated MCP servers.

MCP Discovery Agent. The seeking of MCP server is initiated by the `discover_mcp_servers()` function. It leverages search engines such as Google Search and curated MCP hubs[1] to identify potential medical MCP servers along with their corresponding documentation.

MCP Config Generator. Upon discovering URLs by MCP Discovery Agent, `generate_mcp_config_from_markdown()` function parses the retrieved documentation, which usually includes installation instructions and usage examples. This function employs LLMs to analyze and extract relevant configuration details such as `server_name`, `args`, `command`, and `enabled_tools`.

MCP Validator. The configurations generated by MCP Config Generator must be validated before integration. MCP Validator provides a Docker-based sandbox environment to operate such validation. The validator creates isolated containers for each MCP server candidate, attempts installation using various package managers (`pip`, `npm`, `uvx`), tests multiple command execution patterns, and validates tool availability through the MCP servers. This containerized approach ensures security isolation while preventing local environment pollution during testing. The system implements an iteration strategy with a maximum of 5 attempts per server candidate. If validation fails, detailed error feedback is passed back to the MCP Config Generator, enabling iterative refinement of configuration parameters. The system fails fast for unreachable servers while persisting through recoverable configuration errors. Upon successful validation, the configuration is integrated into the resource pool and becomes immediately deployable for medical research workflows.

[1] `https://github.com/modelcontextprotocol/servers` and `https://github.com/wong2/awesome-mcp-servers`

3 Evaluation

Settings. We evaluate Medical Deep Research through a combination of automatic and expert human assessment, focusing on four key aspects: *completeness*, *tractability*, *correctness*, and *helpfulness*. We invite three board-certified medical specialists, an orthopedic surgeon, an ophthalmologist, and a cardiologist, to evaluate the generated reports for all metrics.

- *Completeness.* For each generated report, we log all references and compute the proportion of unique sources to gain a qualitative understanding of reference distribution.
- *Tractability.* While full LLM explainability [11] is hard to measure, we verify whether each cited reference in both intermediary findings and final reports is accessible and correctly linked (e.g., to a valid paper or URL). Any inaccessible or mismatched reference is marked as a hallucination.
- *Correctness.* We also evaluate the clinical accuracy of the report content. Three human experts are asked to rate reports on 5-point Likert scales using the rubric in Appendix C. As an auxiliary evaluation, we adopt an LLM-as-a-judge approach with the same rubric using Gemini 2.5 Pro [12].
- *Helpfulness.* To assess practical utility for clinical decision-making, the same group of human experts evaluates each report based on the helpfulness score elaborated in Appendix C. Similar LLM-as-a-judge approach is applied to evaluate *helpfulness*, following the same procedure as in the evaluation of *correctness*.

We test five challenging medical queries–three queries supplied by the specialists, and two from HEALTHBENCH [13]. The queries cover treatment planning, diagnostic reasoning, risk screening, and health management. We use OpenAI o1 model as the backend, generating reports in `long report` format with `medium` mode to balance depth and latency.

Results. The evaluation of the system, from the aspects of source coverage, tractability, correctness, and helpfulness, demonstrates that Medical Deep Research effectively leverages a wide range of information sources and produces reliable, verifiable medical reports. We present the findings below:

Medical Deep Research utilizes comprehensive access to medical information sources. Due to its extensive built-in resources and MCP integration pipeline that dynamically adds MCPs, we find that the references of our system are diverse. As shown in Figure 2, PMC and PubMed–major medical literature databases–account for the largest proportions of referenced sources, at 25.7% and 11.5%, respectively. In addition to frequently accessing literature databases, the system occasionally visits clinical guideline platforms such as `Medscape` and specialized sites like `Radiopaedia`. This demonstrates the system's ability to orchestrate and integrate information from diverse sources during the research process.

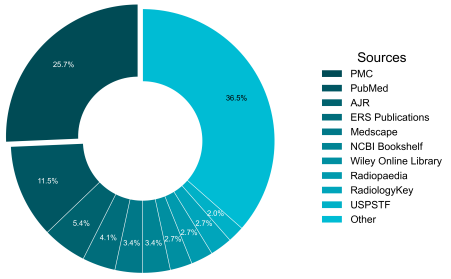

Figure 2: Distribution of referenced sources.

The system exhibits high reliability by providing tractable sources of information. We employ an automated script to examine all references appearing in intermediary findings and final reports, including website links and paper titles. On average, each query is supported by 29.6 references, all of which are verifiable. In other words, none of the listed references are hallucinated. This result is critical for fostering trust among medical practitioners in the reliability of our system.

Human experts and LLM raters confirm the system's high correctness and helpfulness. As shown in Table 3, the Gemini 2.5 Pro model and three board-certified specialists rated the reports near the maximum score of 5 for both correctness and helpfulness. Human experts gave average scores of 4.8 in both aspects, indicating strong clinical accuracy and practical utility. While the system occasionally makes logical errors, the orthopedic surgeon noted that Medical Deep Research shows promising potential to support real-world clinical decision-making.

4 Conclusion

We introduce Medical Deep Research, an agentic system designed to extend Deep Research capabilities into medical and clinical domains. By orchestrating multi-agent workflows for the research module and constructing an extensive and dynamic pool of medical resources, Medical Deep Research enables thorough, evidence-based investigations and delivers customized outputs for diverse clinical and scientific needs. We hope this work lays the groundwork for autonomous research and human-AI collaboration in the medical domain.

References

[1] OpenAI. OpenAI o1 System Card, December 2024.

[2] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[3] OpenAI. Deep Research System Card. Technical report, OpenAI, February 2025.

[4] Rui Yang, Yilin Ning, Emilia Keppo, Mingxuan Liu, Chuan Hong, Danielle S Bitterman, Jasmine Chiat Ling Ong, Daniel Shu Wei Ting, and Nan Liu. Retrieval-augmented generation for generative artificial intelligence in medicine. *arXiv preprint arXiv:2406.12449*, 2024.

[5] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[6] Google Gemini. Gemini Fullstack LangGraph Quickstart. `https://github.com/google-gemini/gemini-fullstack-langgraph-quickstart`, 2025.

[7] Lana Yeganova, Won Kim, Shubo Tian, Donald C Comeau, W John Wilbur, and Zhiyong Lu. Litsense 2.0: Ai-powered biomedical information retrieval with sentence and passage level knowledge discovery. *Nucleic Acids Research*, page gkaf417, 2025.

[8] Shanghua Gao, Richard Zhu, Zhenglun Kong, Ayush Noori, Xiaorui Su, Curtis Ginder, Theodoros Tsiligkaridis, and Marinka Zitnik. Txagent: An ai agent for therapeutic reasoning across a universe of tools. *arXiv preprint arXiv:2503.10970*, 2025.

[9] Craig Knox, Mike Wilson, Christen M Klinger, Mark Franklin, Eponine Oler, Alex Wilson, Allison Pon, Jordan Cox, Na Eun Chin, Seth A Strawbridge, et al. Drugbank 6.0: the drugbank knowledgebase for 2024. *Nucleic acids research*, 52(D1):D1265–D1275, 2024.

[10] Taha A Kass-Hout, Zhiheng Xu, Matthew Mohebbi, Hans Nelsen, Adam Baker, Jonathan Levine, Elaine Johanson, and Roselie A Bright. Openfda: an innovative platform providing access to a wealth of fda's publicly available data. *Journal of the American Medical Informatics Association*, 23(3):596–600, 2016.

[11] Jiaxing Zhang, Jiayi Liu, Dongsheng Luo, Jennifer Neville, and Hua Wei. Llmexplainer: Large language model based bayesian inference for graph explanation generation. *arXiv preprint arXiv:2407.15351*, 2024.

[12] Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Technical report, Google DeepMind, June 2025.

[13] Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

[14] Google. Gemini deep research — your personal research assistant, 2024.

[15] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[16] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[17] Manus AI. Manus — a general ai agent, 2025.

[18] n8n-io. n8n: Fair-code workflow automation. `https://github.com/n8n-io/n8n`, 2025.

[19] HuggingFace. Open deep research example — `smolagents`, 2025.

[20] ByteDance. Deerflow: A community-driven deep research framework, 2025.

[21] Zheng Dong. deep-research: An ai-powered research assistant, 2024.

[22] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

[23] Perplexity Team. Introducing perplexity deep research. `https://www.perplexity.ai/hub/blog/introducing-perplexity-deep-research`, February 2025.

[24] OpenEvidence. Openevidence, 2025.

A Related Work

Deep Research unifies LLMs, information retrieval systems, and reasoning to advance scholarly inquiry and real-world problem-solving. The effectiveness of such systems hinges on integrating reasoning capabilities with multi-agent architectures. `OpenAI/DeepResearch` [3] exemplifies this evolution, powered by the o-series reasoning model optimized for web browsing and data analysis. Similarly, `Gemini/DeepResearch` [14] leverages `Gemini 2.5 Pro` with a million-token context window to process large-scale information. These advancements build upon reasoning techniques such as Chain-of-Thought prompting [15] and Self-Consistency [16] to enhance information synthesis and decision-making. Other systems focus on automating tool use and orchestration. For example, `Manus` [17] demonstrates advanced API integration and tool selection mechanisms, while `n8n` [18] provides a flexible automation platform that integrates with external data and analytics services. In the open-source community, several notable Deep Research systems have emerged, including `smolagents/Open_Deep_Research` [19], `ByteDance/deer-flow` [20], and `dzhng/deep-research` [21]. Furthermore, as illustrated by Google Co-Scientist [22], AI holds significant potential for scientific research. The medical domain, in particular, appears both appealing and feasible for research automation. `Perplexity/DeepResearch` [23] claims enhanced citation mechanisms for medical statements, offering further evidence of the viability of Deep Research in the medical domain. `OpenEvidence` [24] supports evidence-based question answering by connecting LLMs to respected journal databases, but lacks agentic orchestration and dynamic tool integration, thereby offering a less flexible system. To the best of our knowledge, no open-source system offers comprehensive medical specializations that are reliable for clinical decision support. We compare our Medical Deep Research system with existing approaches in Table 1.

Table 1: Comparison between Medical Deep Research and existing systems. We focus on the following criteria: open-source, web search, MCP support, code execution, flexible output format, and dynamic MCP integration.

System	Domain	Open	Web Search	MCP Support	Code Exe.	Flexible Out.	Dynamic MCP
`OpenAI/DeepResearch`	General	×	✓	✓	✓	×	×
`Gemini/DeepResearch`	General	×	✓	×	✓	×	×
`Perplexity/DeepResearch`	General	×	✓	×	×	×	×
`Manus`	General	×	✓	✓	✓	✓	×
`DeerFlow`	General	✓	✓	✓	✓	✓	×
`mshumer/OpenDeepResearcher`	General	✓	×	×	×	×	×
`n8n`	General	✓	✓	✓	✓	✓	×
`dzhng/Deep-Research`	General	✓	✓	×	×	×	×
`OpenEvidence`	Medical	×	✓	×	×	×	×
Medical Deep Research (Ours)	Medical	✓	✓	✓	✓	✓	✓

B ReAct Prompt

```
You are ‘researcher‘ agent. You are dedicated to conducting thorough investigations
using tools and providing comprehensive solutions through systematic use of the
available tools, including both built-in tools and dynamically loaded tools.
# Research Strategy
1.  **Initial Analysis**:  - Forget your previous knowledge, and carefully read
the problem statement - Identify key aspects that need investigation and break
down complex queries into simpler, focused sub-queries - Consider different angles
or perspectives for comprehensive coverage - Assess all available tools, including
dynamically loaded ones, and think strategically about which combinations will provide
the most comprehensive coverage - Think carefully before deciding which tools to use
and in what order
2. **Tool Selection and Usage**: - Utilize relevant resources as much as possible:
Use tools relevant to the current step, and actively seek out multiple relevant tools
with the same queries since each tool might represent distinct sources of information
- Do not limit yourself to a single tool or source. Explore all available relevant
resources to ensure comprehensive coverage - Choose the most appropriate tool for
each subtask.  Prefer specialized tools over general-purpose ones when available -
For dependent operations, wait for first tool’s result before using in second tool.
For example, when you try to obtain the information online, you should obtain the
URL of the websites first before using crawler - Use parallel tool calls only for
```

completely independent operations - Validate all input values before making tool calls - Before each tool call, consider whether this tool will provide the most relevant and comprehensive information for the current research objective
3. **Information Gathering and Processing**: - Execute tools in the correct dependency order - Document reasoning for each tool call and store intermediate results properly - Actively seek out multiple sources and tools for each research aspect to ensure thorough coverage - Evaluate each tool result's relevance to the original query and filter out unrelated information - Combine and synthesize information from multiple sources to create a comprehensive understanding - Identify and address information gaps - Prioritize relevant and recent information - When task includes time range requirements: - Use appropriate time-based search parameters - Verify source publication dates - Ensure results meet time constraints
Available Tools
You have access to many tools, ranging from general web searching tools to domain-specific biomedical databases. For each query, you could use multiple tools to ensure you get multi-facet information.
Tool Usage Guidelines
- Read the tool documentation carefully before using it. Pay attention to required parameters and expected outputs - If a tool returns an error, analyze the error and adjust your approach accordingly - Often, the best results come from combining multiple tools. For example, use a Github search tool to search for trending repos, then use the crawl tool to get more details - If you need to use a tool's result in another tool, store the first result and use it directly
Output Format
- Provide a structured response in markdown format. - Include the following sections: - **Problem Statement**: Restate the problem and your analysis approach. - **Research Findings**: Organize by topic rather than by tool used. For each major finding: - Summarize the key information - Use inline citations with [tag] format immediately after each claim. Tag format: first author's surname (or first significant title word if no author) + last two digits of year, e.g. [smith24] - Add "-a", "-b"... if needed to keep tags unique - Reuse the same tag for repeat citations - Include relevant images if available - **Conclusion**: Provide a synthesized response based on the gathered information. - ** References**: List every unique tag in the order it first appears, one per line with a blank line between, formatted **[tag]** [Full Source Title](URL). Show URLs only here. - Use inline citations throughout the text and maintain a comprehensive References section. - **Reference Format Guidelines**: - When a URL is available: **[tag]** [Full Source Title](URL) - When only a paper title is available: **[tag]** [Full Paper Title] (Title only - no URL available) - When only search results are available: **[tag]** [Search Result Title] (Search result - no direct URL) - Always include the full title of the source, whether URL is available or not
Notes
- Always verify information relevance and credibility - Actively seek out and utilize all relevant resources available for comprehensive research coverage - Think carefully before selecting tools and ensure you're using the most appropriate combination of resources - If no URL is provided, focus on search results - Never perform mathematical calculations or file operations - Do not try to interact with pages - crawl tool is for content only - Use crawl_tool when URLs are provided in the search results, and you think it is helpful to gather more information - Always include source attribution with inline citations [tag] format - Clearly indicate which source provides each piece of information using the specified citation format - For time-constrained tasks, strictly adhere to specified time periods - Ensure you've explored multiple relevant sources and tools before concluding your research

C Rubrics for Evaluating Correctness and Helpfulness

Table 2: Rubric for evaluating correctness and helpfulness.

Score	Correctness	Helpfulness
1	Many significant factual errors or misinterpretations.	Unclear, disorganized; fails to address core questions.
2	Several factual errors or misinterpretations.	Incomplete or unclear answers; poor organization.
3	Mostly accurate with minor errors or omissions.	Addresses most points; some vague parts, acceptable organization.
4	Highly accurate with only a few minor issues.	Clear, comprehensive answers; well organized.
5	Impeccably accurate, demonstrates nuanced understanding.	Exceptional clarity, insight, and organization; fully addresses all aspects.

D Evaluation Results

Table 3: Evaluation scores (1–5).

Rater	*Completeness*	*Tractability*	*Correctness*	*Helpfulness*
Gemini 2.5 Pro	5.0	5.0	5.0	5.0
Orthopedic Surgeon	5.0	5.0	4.8	5.0
Ophthalmologist	5.0	5.0	5.0	4.6
Cardiologist	5.0	5.0	4.6	4.8
Human Expert Avg.	5.0	5.0	4.8	4.8