

Article

Leveraging MCP and Corrective RAG for Scalable and Interoperable Multi-Agent Healthcare Systems

Dimitrios Kalathas ^{1,*} , Andreas Menychtas ² , Panayiotis Tsanakas ¹  and Ilias Maglogiannis ² ¹ School of Electrical and Computer Engineering, National and Technical University of Athens, Iroon Polytechniou 9, 15772 Zografou, Greece; panag@cs.ntua.gr² Department of Digital Systems, University of Piraeus, Karaoli Ke Dimitriou 80, 18534 Piraeus, Greece; amenychtas@unipi.gr (A.M.); imaglo@unipi.gr (I.M.)

* Correspondence: dkalathas@mail.ntua.gr

Abstract

The rapid evolution of Generative AI (GenAI) has created the conditions for developing innovative solutions that disrupt all fields of human-related activities. Within the healthcare sector, numerous AI-driven applications have emerged, offering comprehensive health-related insights and addressing user questions in real time. Nevertheless, most of them use general-purpose Large Language Models (LLMs); consequently, the responses may not be as accurate as required in clinical settings. Therefore, the research community is adopting efficient architectures, such as Multi-Agent Systems (MAS) to optimize task allocation, reasoning processes, and system scalability. Most recently, the Model Context Protocol (MCP) has been introduced; however, very few applications apply this protocol within a healthcare MAS. Furthermore, Retrieval-Augmented Generation (RAG) has proven essential for grounding AI responses in verified clinical literature. This paper proposes a novel architecture that integrates these technologies to create an advanced Agentic Corrective RAG (CRAG) system. Unlike standard approaches, this method incorporates an active evaluation layer that autonomously detects retrieval failures and triggers corrective fallback mechanisms to ensure safety and accuracy. A comparative analysis was conducted for this architecture against Typical RAG and Cache-Augmented Generation (CAG), demonstrating that the proposed solution improves workflow efficiency and enables more accurate, context-aware interventions in healthcare.

Keywords: multi-agent; MCP; eHealth; corrective RAG; AI assistant

Academic Editors: Jinwen Liang and Jixin Zhang

Received: 14 December 2025

Revised: 16 February 2026

Accepted: 19 February 2026

Published: 21 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The integration of Artificial Intelligence (AI) into healthcare is hindered by two critical bottlenecks: the lack of interoperability between data silos and the tendency of models to hallucinate in high-stakes scenarios [1]. Currently, at the forefront of AI technology is GenAI, which can be integrated into many daily human needs. Specifically, a very important domain is the field of healthcare and wellness. Many applications currently leverage LLMs to develop ChatBots and AI Assistants that enrich conversations [2] and offer the user a more complete experience integrated into eHealth platforms [3]. Simultaneously, a new terminology has emerged: AI Agents [4]. Agents represent a technique defining how the model will act and what action it must take before providing the final answer. They refer to autonomous software systems designed to perform specialized tasks and achieve predefined goals using AI techniques, while requiring minimal human oversight [5].

However, solutions that integrate these technologies in the healthcare domain still present limitations, which become critical when the primary goal is to monitor, supervise, and enhance individuals' health. Single-agent systems are often unable to address multi-tasking requirements because they rely on a main agent that focuses on a single objective and is enhanced only with tools to help manage it [6]. Moreover, systems that follow a single-agent architecture struggle with the "Context Window" bottleneck [7]. This means that the system often loses track of long-term patient history during the conversation. In the healthcare field, decisions made today depend on data from months or years ago, and single-agent implementations struggle to maintain and process this context across different domains. Another limitation of the single-agent approach is that attempting to fulfill multiple roles simultaneously—acting as a nutritionist, a pharmacist, and a scheduler—is more likely to lead to errors or hallucinations in the answers. This happens because it is difficult to define more than one role with high accuracy using only a single system prompt [8]. Nevertheless, digital systems in the healthcare market aim to cover as many scenarios as possible with high accuracy in their results while being as scalable as possible. Thus, despite their potential, current LLM-based systems face critical challenges that limit their scalability and effectiveness. This is precisely where Multi-Agent (MA) architectures and the Model Context Protocol (MCP) come into play [9]. A Multi-Agent System (MAS) is a complex system composed of multiple interacting intelligent agents, capable of simulating social interactions and teamwork in the real world, thereby enhancing overall adaptability and efficiency through decentralized decision-making processes and information sharing [10]. Furthermore, the MCP is an open standard developed by Anthropic that connects AI models with external tools, data sources, and APIs [11]. Specifically, the proposed system utilizes a central 'Planner Agent' designed to decompose complex clinical queries into manageable sub-tasks. Rather than relying on a single model to process all information, this Planner divides the initial query to identify and assign specific reasoning tasks to specialized agents. Crucially, MCP serves as the technical backbone for this collaboration. MCP allows these distinct agents to connect with external medical tools and databases via a standardized interface. This eliminates the need for custom integration code for each data source and ensures that new medical modules can be attached and supported for decision-making seamlessly, facilitating long-term scalability and better performance.

In addition, in the field of healthcare, accuracy and clarity of health-related information are vital. Despite the high-level performance and capabilities of LLM-based applications, they struggle with insufficient factual accuracy and generic responses [12]. To address this limitation, researchers present methods such as fine-tuning [13], which retrain LLMs with a dataset containing details about the subject matter provided. However, this approach requires a significant amount of resources, time, and data to retrain LLMs; furthermore, since developers do not have full control over the answers, the result may not align with users' cognitive processes as desired. One promising solution is to enhance LLMs with external knowledge that can be inspected, interpreted, and kept up-to-date. One of the most recent and effective methods to achieve this functionality is called Retrieval-Augmented Generation (RAG) [14]. RAG models, introduced by Lewis and Perez, represent a technique for enhancing the accuracy and reliability of generative AI models with information from specific and relevant data sources [15]. This is often supported by a Vector database, which is a database filled with vector embeddings that convert textual data into dense, continuous representations to enhance both retrieval and generation processes [16,17].

Two critical subdomains of healthcare are nutrition and medicine. Many digital platforms offer LLM-based applications that provide details related to these fields, but it is common that the accuracy of their outputs is often limited. This is largely because such subdomains are typically not considered critical enough to justify substantial investment in

further model optimization. The RAG method could mitigate this situation and provide innovative solutions for more accurate and targeted results. In the proposed architecture, the combination of RAG and MAS is critical for resolving conflicts between these sensitive subdomains. By assigning dedicated agents to specific knowledge bases, the system can retrieve data to ensure patient safety and correctness in the outcomes provided. For instance, consider the interaction between a Nutrition RAG Agent and a Medicine RAG Agent. A Typical RAG system might simply retrieve a healthy diet plan; however, the Medicine RAG Agent, simultaneously accessing pharmacological databases via RAG, can identify if the patient is allowed to follow the diet based on their personal drug treatment. Through MA communication, the system detects this specific ‘nutrient-drug interaction’ and proactively filters the suggestion. This layered verification ensures that advice is not only factually correct according to medical literature but also safe within the specific context of the patient’s pharmacological profile. To the best of our knowledge, there are currently limited implementations that effectively integrate MCP within a MAS specifically focused on personalized healthcare monitoring.

This paper bridges that gap by introducing a novel prototype architecture that orchestrates specialized agents, enhanced by Agentic RAG [18] combined with Corrective RAG (CRAG) [19] into one approach named as Agentic Corrective RAG (AC-RAG), a previously unintroduced architectural synthesis, to manage medical domains autonomously. LangChain is a well established and widely adopted framework designed for developing applications powered by LLMs, and the proposed prototype utilizes this framework to streamline the RAG implementation tools [20]. In contrast to generic single-agent ChatBots that struggle with conflicting constraints, our approach creates an active, context-aware system capable of reasoning across healthcare domains, such as nutritional advice and pharmacological instructions. Unlike Typical RAG implementations that blindly trust retrieved data, our approach incorporates an active evaluation layer. This layer autonomously detects retrieval failures and triggers corrective fallback mechanisms via MCP tools to ensure safety. Thus, the main innovation of AC-RAG lies in its ability to transform passive retrieval into an active, self-correcting process, ensuring that clinical advice is dynamically validated rather than merely retrieved. Furthermore, we validate this architecture through a comparative analysis against Typical RAG and Cache-Augmented Generation (CAG) [21]. By leveraging MCP as the interoperability layer, the proposed solution not only ensures high accuracy and safety through cross-agent verification but also achieves scalability that allows for the integration of future medical tools without architectural restructuring.

The core innovation of this study lies in the synergistic combination of an Agentic MAS architecture with a CRAG mechanism. The system leverages the Agentic approach to decompose complex user queries, autonomously determining which specialized agents must cooperate to address specific medical needs. A critical requirement is the synthesis of the user’s personal health records—specifically nutrition plans and drug treatments—with broader medical knowledge. To ensure the highest level of accuracy, we introduce a ‘Judge’ validation layer. If this evaluator determines that the retrieved internal data is insufficient or outdated, it triggers a corrective loop via the MCP. This mechanism activates web-based tools to perform on-the-fly retrieval of up-to-date clinical data, ensuring that any associations between the user’s nutrition and treatment plan are grounded in the most current and verified information available.

The remainder of the paper is organized as follows: In Section 2, we analyze the Scientific Background of the proposed system. Section 3 focuses on the Design and Implementation of the system. Section 4 presents the System in Practice. Section 5 illustrates the Results of this work, highlighting the research findings and the evaluation tests conducted, while Section 6 demonstrates the Discussion, followed by the Conclusion.

2. Scientific Background

The rapid evolution of GenAI has led to significant solutions in various sectors, particularly in healthcare, using LLM-based applications to monitor, support, and provide assistance in people's daily routines. ChatGPT, operating on the GPT-3.5 architecture, has been evaluated on its capacity to process complex medical information using questions from the United States Medical Licensing Examination (USMLE) [22]. Notably, it achieved a performance level approaching the passing threshold even in the absence of specialized prompting [23]. Another approach in medical informatics uses LLMs for report compilation [24]. Some studies have also explored the utility of other foundational models fine-tuned on medical data. Med-PaLM [25] and Med-PaLM2 [26] are two fine-tuned versions of the Flan-PaLM model by Google. Other examples of specialized LLMs include initiatives such as ChatDoctor [27], MedAlpaca [28], and BioGPT [29]. These approaches are tailored to specific healthcare applications, enhancing the capabilities of AI in areas like medical diagnostics, personalized treatment, and clinical decision-making. LLMs have also attracted substantial research attention for their use as the core processing engine of AI agents. Numerous frameworks have been introduced in the literature that help create agents capable of taking advantage of the impressive reasoning capability of LLMs to carry out complex tasks autonomously. AutoGen and LangChain are some of the most prominent frameworks that integrate various libraries for interaction with external tools [30]. For example, openCHA is an open-source LLM-powered framework designed to empower conversational agents to generate personalized responses for users' healthcare queries, such as assistance and diagnosis [31]. ChatDev demonstrates how LLMs can be used to instantiate multiple collaborative agents that emulate the structure and workflow of a software company, completing an end-to-end development cycle to produce functional software [32]. Other agent-based systems similarly exploit the language understanding and reasoning capabilities of LLMs to interact with and manipulate their operating environment, including graphical user interfaces [33,34]. Moreover, there is an interesting approach that uses a single agent combined with RAG technology to assist with the user's drug treatment [35]. A foundational two-agent system was proposed by Alghamdi and Mostafa, consisting of one agent dedicated to generating medical guidance and a second agent responsible for validating the trustworthiness of the generated responses [36]. Other systems, like RxLens, provide a Multi-Agent LLM-powered solution for scanning and ordering prescriptions for pharmacies [37]. Recent optimization techniques have largely remained fragmented. For instance, CAG approaches focus exclusively on preloading contexts to bypass retrieval entirely [38], while CRAG is typically implemented as a standalone mechanism to simply filter irrelevant documents within linear pipelines [39]. Although these methods and autonomous agents exist individually in the state of the art, the literature lacks a cohesive strategy that unifies these advanced retrieval and correction mechanisms within a scalable system. These advancements pave the way for a more intelligent, data-driven, and responsive healthcare ecosystem. This work focuses on an innovative architecture that integrates MCP into a MAS, providing scalability for high-demand healthcare workflows and leveraging the CRAG methodology to increase the accuracy of outcomes based on specific user scenarios.

3. Design and Implementation

The system is divided into four entities: the User Interface (UI), the Platform, the MAS, and the MCP. This research examines the combination of MCP and RAG models into one concept, which leads to a MAS that could be easily integrated into eHealth platforms as an assistant for personal medical treatment and nutrition. In order for the application to be

considered functional, all four entities must be properly combined and interact with each other, taking into account the scalability of the system, as illustrated in Figure 1.

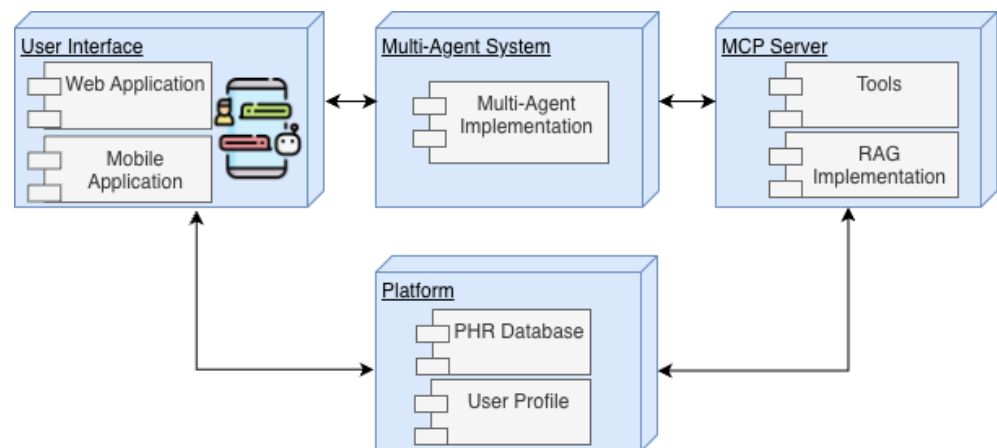


Figure 1. The high-level system architecture illustrating the integration of the four entities (UI, Platform, MAS, and MCP) to support the eHealth assistant.

3.1. User Interface

Web and mobile applications constitute the primary interaction channels for patients, healthcare professionals, and the general public, offering intuitive environments for monitoring health indicators and daily routines. Within this interface, users can manage their medication treatments, record their nutrition programs, and monitor their health progress. Our prototype is an external implementation that allows users to interact directly with the MAS through a minimal ChatBot user interface. This system handles questions about drug treatments or nutrition and generates results with more detailed and accurate information.

3.2. Platform

The Platform is a cloud-based component that hosts services crucial to our framework. The PHR Database stores the personal health records for a user, such as daily measurements from wearable devices (blood pressure, steps, stress, heart rate, etc.), drug treatments (medicines, daily dose, schedule, etc.), and nutrition data. Crucially, this component ensures full compliance with the General Data Protection Regulation (GDPR), implementing strict security protocols to safeguard the privacy and integrity of all sensitive health data. Furthermore, users can create a personal account to personalize the Telehealth process with a healthcare professional. Moreover, there is a notification service that informs patients about the progress of their health and keeps the supervising healthcare professional updated, if one is assigned.

3.3. Multi-Agent System

The MAS is developed through AutoGen, a framework that provides predefined classes and numerous connectors for integration with various external components. Furthermore, it offers the opportunity to develop custom classes that incorporate more flexibility into the system. To process multilingual datasets, the system utilizes LLMs from the Ollama API, particularly the “gpt-oss:120b” model [40]. Notably, we adopt a self-hosted approach within a fully local infrastructure, effectively excluding the use of external cloud-based LLM services from the scope of this paper. This model is also used in every agent because it possesses Agentic features and supports native capabilities for function calling, Python 3.12 tool calls, and structured outputs. The architecture that the prototype adopts consists of six agents, as illustrated in Figure 2.

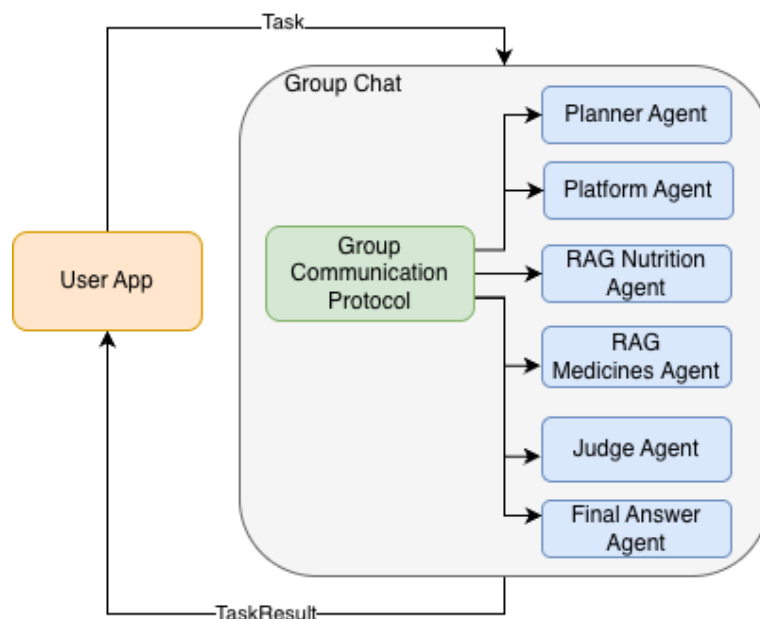


Figure 2. The MAS—Multi Agent System architecture, illustrating the six autonomous agents developed using the AutoGen framework and powered by the gpt-oss:120b model.

3.4. Model Context Protocol

The MCP server is designed to expose the necessary tools required by each agent to perform its designated tasks. To elaborate further, for the Platform Agent, two specialized tools are provided, both functioning as connectors to self-hosted User Database, which serves as the backend infrastructure. The first tool enables access to the PHR collection, allowing the agent to retrieve nutrition and medication data whenever such information is needed by the MAS. The second tool provides access to the User Profile collection, ensuring that the system can obtain up-to-date user information to support personalized interactions [41].

Moreover, there are RAG tools that support the two RAG Domain Agents. RAG models are developed with the LangChain framework, which serves as the orchestrator, simplifying the development of applications driven by LLMs with tools for model selection, monitoring, and retrieval-based enhancement. These agents can use the retriever tools, which have the functionality to retrieve documents from the vector database based on the initial query. Specifically, the MCP allows the retrieval of extra knowledge if the MAS judges it necessary; to achieve this, it uses semantic search on the initial user's query within the vector database that has been created for this purpose. We utilize the Facebook AI Similarity Search (FAISS) library to index and retrieve vector embeddings [42]. To ensure the reliability of our similarity search results, the embeddings indexed in FAISS are generated strictly from verified, open-access medical and nutritional datasets. Specifically, we aggregated global dietary standards by collecting public guidelines from the World Health Organization (WHO) [43], focusing on the “Healthy Diet” section of their official repository. To complement this with precise nutritional quantification, we utilized the U.S. Department of Agriculture (USDA) database [44] to source comprehensive food composition data, including caloric and vitamin content across diverse food types. Furthermore, we obtained pharmacological data from the publicly accessible databases of the National Library of Medicine (NLM), utilizing the DailyMed SPL Resources [45] for drug labels and MedlinePlus XML Files [46] for health topics. Crucially, this data aggregation is not static; we employed automated web scraping techniques to construct the dataset dynamically. Most of the vector embeddings are generated on-the-fly based on the specific context of the user's initial query, allowing the system to scrape, process, and properly update the index

in real time with targeted information from these repositories. These sources are strictly open-access and do not require special permissions for research use. Furthermore, for the personal health scenarios, we utilized synthetic patient profiles containing randomized but clinically realistic data points (e.g., medication schedules, nutrition logs). This approach allowed us to simulate high-risk scenarios without utilizing real patient data or compromising user privacy. Moreover, the model for the embedding is “text-embedding-3-large” from OpenAI, which is trained on multilingual data; this is important because it can adjust to the language of the user. Last but not least, for semantic search, we use cosine similarity, which measures the angle between vectors to find the semantic similarity between texts, determining how closely the meaning of a query matches that of a document. To further enhance the system’s reliability, we introduce a validation layer managed by a Judge Agent. This agent is equipped with a specialized MCP ‘Web Search Tool’ configured for the NLM database, focused on a specific drug at the time and designed to handle cases where the internal vector data do not meet the requirements. When triggered, this tool performs an on-the-fly RAG search to fetch real-time details—such as recent contraindications or new side effects—based specifically on the drug name identified in the user’s query. This capability ensures that the system does not rely solely on the static vector database but can actively verify and update its knowledge from online sources before delivering a response. Through this process, the corresponding agent can introduce up-to-date information, accurate data, and contextually relevant evidence into the group, thereby enhancing the overall quality and reliability of the system’s response. The MCP protocol uses JSON-RPC 2.0 messages to enable structured communication between MCP servers and agents, which consume these resources to perform tasks [47].

3.5. Implementation of MAS

This section details the practical implementation of the MAS, focusing on the specific roles, responsibilities, and interaction patterns of the agents that cooperate and coexist. It outlines the internal communication protocols that enable autonomous collaboration and describes the workflow from the initial user query decomposition by the Planner to the final synthesis of the response. **Group Chat—Internal Communication Protocol** establishes the foundation where all agents coexist in groups, providing the advantage of sharing a common thread of messages. As previously mentioned, each participant agent is specialized for a particular task, such as the planner, nutrition specialist, or final editor, in a collaborative writing task. We implement the Group Chat into the system, which features a model-based next-speaker selection mechanism. It supports the team by analyzing the current conversation context, including the conversation history and participants’ names and description attributes, to determine the next speaker using a model. This interaction is further strengthened through the internal communication protocol, which relies on the shared conversation history, the group chat orchestration, and AutoGen’s internal message-passing system between agents; all of these collectively ensure coherent and well-coordinated multi-agent collaboration. The **Planner Agent** is the initial agent that interacts with the user query. Through prompt engineering and the use of an LLM with reasoning capabilities like ‘gpt-oss’, we enable the system to understand the main requirements of the user and to define and split the tasks among the sub-agents. It is critical for the Planner to be fully informed about the agents that coexist in the system and their capabilities. Thus, the system prompt describes exactly the specializations of each agent, their names, and some well-structured examples of the tasks they can solve. Following this, the **Platform Agent** is specialized for the PHR of each user. More specifically, if the Planner understands that the initial query concerns personal data details, it transfers the information to this agent. By invoking targeted MCP tools, the agent can securely access nutrition and treatment-

related data necessary for the reasoning process. Once the relevant information has been collected, the Platform Agent communicates these details to the rest of the agent group, ensuring that all downstream components operate with accurate and up-to-date patient context. Simultaneously, **RAG Domain Agents** are specialized for specific domains and designed to interface with domain-relevant functions and RAG tools. The prototype uses two RAG Domain Agents, one for medicines and one for nutrition. For instance, if a user submits a query regarding a specific medication, the Planner Agent assigns the task to the RAG Medicine Agent. However, unlike standard implementations, this agent focuses on entity extraction and initial retrieval; it parses the query to identify the specific drug name and invokes the retriever to fetch preliminary documents from the vector database. Subsequently, the workflow introduces a critical validation step via the **Judge Agent**. This agent acts as an active evaluator between the retrieval and generation phases. It analyzes the details provided by the RAG Domain Agent to determine their relevance and completeness. If the Judge identifies gaps, ambiguities, or outdated information, it autonomously invokes a specialized MCP 'Web Search Tool'. Uniquely, the Judge Agent performs an on-the-fly search on verified medical websites to fetch the latest details and adds this validated information back into the system's shared context via the MCP. This corrective mechanism allows the system to self-repair knowledge gaps in real time before an answer is formulated. Finally, the **Final Answer Agent** is the last agent in the group, and its role is to sum up all the details and inputs from the other agents to generate the final outcome that will be presented to the user. Moreover, the Final Answer Agent is also responsible for validating that the final response produced by the system aligns with the user's initial query. This ensures that the generated output is accurate, relevant, and within the scope of the request. Once this verification step is complete, the Final Answer Agent notifies the agent group that the workflow for the current query can be terminated. An end-to-end extended architecture is illustrated in Figure 3. Moreover, to formally describe the orchestration logic and the interaction between the agents and the MCP, the **End-to-End Process Representation** is provided in Algorithm 1. This algorithm illustrates the complete workflow, detailing how a user query is decomposed, processed, and validated. Specifically, the process commences with the Initialization Phase, where the system registers the Planner, Platform, RAG, Judge, and Final Answer agents and establishes secure connections to external MCP tools, such as the PHR Database and Vector Database. Upon receiving the user input (U), the system enters the Main Orchestration Loop. Here, a model-based selection mechanism dynamically determines which agent should act next based on the evolving conversation history (H). If the Planner is selected, it analyzes the user's intent to broadcast specific task assignments. Subsequently, depending on the request, the Platform Agent or RAG Agents are triggered to fetch personalized health records or domain-specific medical knowledge. If necessary, the **Judge Agent** intervenes to correct or expand this knowledge via web retrieval. Finally, the workflow converges at the Final Answer Agent, which synthesizes the accumulated information (R_{draft}) and performs a critical alignment check against the original query. The process terminates only when the response is validated as accurate and complete, ensuring a reliable output.

Algorithm 1 Process_User_Query(*User_Input*)**Input:** User Query (*U*)**Output:** Final Response (*R*)**1. Initialization Phase**

```

1:  $H \leftarrow []$  ▷ Initialize empty conversation history
2:  $A \leftarrow \{\text{Planner, Platform, RAG}_{\text{Med}}, \text{RAG}_{\text{Nutri}}, \text{Judge, Final}\}$  ▷ Register Agent Ensemble
3:  $S_{\text{MCP}} \leftarrow \text{Connect}(\text{Tools: } [\text{PHR\_Database, RAG\_Search, RAG\_Web\_Search}])$  ▷ Establish MCP Tool Connections

```

2. Context Ingestion

```

4:  $H.\text{append}(U)$  ▷ Inject user query into context

```

3. Main Orchestration Loop

```

5: while Process_Is_Active do
6:   # A. Next Speaker Selection
7:    $A_{\text{current}} \leftarrow \text{Select\_Next\_Speaker}(H, A)$  ▷ Model-based dynamic routing
8:   # B. Agent Execution Logic
9:   if  $A_{\text{current}} = \text{Planner\_Agent}$  then
10:     $P \leftarrow \text{Analyze\_Intent}(H)$  ▷ Decompose query into sub-tasks
11:     $\text{msg} \leftarrow \text{"Task Assigned: " + } P$ 
12:     $H.\text{broadcast}(\text{msg})$ 
13:   else if  $A_{\text{current}} = \text{Platform\_Agent}$  then
14:     if query_requires_personal_data then
15:       # Secure retrieval of Patient Health Records (PHR)
16:        $D_{\text{raw}} \leftarrow S_{\text{MCP}}.\text{invoke}(\text{"get\_user\_phr\_database"})$ 
17:        $\text{Analysis} \leftarrow \text{Analyze\_Data}(D_{\text{raw}})$ 
18:        $H.\text{broadcast}(\text{Analysis})$ 
19:     end if
20:   else if  $A_{\text{current}} \in \{\text{RAG}_{\text{Med}}, \text{RAG}_{\text{Nutri}}\}$  then
21:     if query_requires_domain_knowledge then
22:       # 1. Standard Retrieval via MCP
23:        $D_{\text{ctx}} \leftarrow S_{\text{MCP}}.\text{invoke}(\text{"rag\_vector\_retriever"})$  ▷ Retrieve top-k documents
24:       # 2. Judge Validation (CRAG Logic)
25:        $\text{Score, Gaps} \leftarrow \text{Judge\_Agent.Evaluate}(D_{\text{ctx}}, U)$ 
26:       if  $\text{Score} < \text{Threshold}$  then ▷ If context is insufficient
27:         # 3. Corrective Web Search
28:          $D_{\text{web}} \leftarrow S_{\text{MCP}}.\text{invoke}(\text{"web\_search\_tool"}, \text{Gaps})$ 
29:          $D_{\text{final}} \leftarrow \text{Merge}(D_{\text{ctx}}, D_{\text{web}})$  ▷ Enrich context with web data
30:       else
31:          $D_{\text{final}} \leftarrow D_{\text{ctx}}$ 
32:       end if
33:        $R_{\text{draft}} \leftarrow \text{Synthesize\_With\_LLM}(D_{\text{final}})$ 
34:        $H.\text{broadcast}(R_{\text{draft}})$ 
35:     end if
36:   else if  $A_{\text{current}} = \text{Final\_Answer\_Agent}$  then
37:      $R_{\text{draft}} \leftarrow \text{Summarize}(H)$  ▷ Synthesize all agent outputs
38:     if  $R_{\text{draft}}$  aligns with  $U$  then ▷ Validate relevance and safety
39:        $R \leftarrow R_{\text{draft}}$ 
40:        $\text{Process\_Is\_Active} \leftarrow \text{false}$  ▷ Terminate loop
41:     else
42:        $H.\text{broadcast}(\text{"Refinement Needed"})$  ▷ Trigger re-evaluation
43:     end if
44:   end if
45: end while

```

4. Output Generation

```

46: return  $R$ 

```


Medicine Agent connects to an MCP Server to query the Medicines VectorDB, effectively retrieving static, verified medical knowledge regarding the drug. Simultaneously, on the nutrition path, the **Platform Agent** connects to a separate MCP Server to fetch live patient health records from the Database. This personal context is then passed to the **RAG Nutrition Agent**, which queries the Nutrition VectorDB for relevant dietary guidelines.

3.6.3. Phase 3: Active Validation (The Judge-Loop)

The retrieved contexts from both agents are forwarded to the central **Judge Agent**. This component represents the system's "Judge Layer." The Judge evaluates the combined information for completeness and safety. Crucially, if the Judge detects a "knowledge gap"—for instance, if the internal databases lack specific data on the interaction between Ibuprofen and a high-protein diet—it triggers the **Corrective Loop**. As shown in the diagram, the Judge activates the "Search on the fly Tool" via an MCP Server. This enables the system to fetch real-time, up-to-date clinical details from external web sources to fill the identified gap.

3.6.4. Phase 4: Final Synthesis

Once the data is validated and any missing information is retrieved, the **Final Answer Agent** synthesizes the pharmacological facts, personal health data, and real-time findings. It constructs a clinically relevant answer, ensuring that the final response delivered to the user is both accurate and personalized.

4. System in Practice

4.1. User Requirements

This prototype is designed to aid users in searching for details regarding two core scenarios. The first concerns their pharmaceutical treatments, potential interactions between medicines in their daily routine, and general information about drugs. The second relates to their personal nutrition, how to improve it, and general information about a healthy diet. Primary users of this tool include medical researchers, healthcare professionals, nutritionists, patients under observation, and the general population wishing to monitor their drug treatment, nutrition program, and daily health progress.

User requirements, pivotal in designing medical prototypes, dictate both the user interface design and model selection criteria, ensuring alignment with healthcare professionals' and patients' needs. In our case, the tool is characterized by the flexibility of integration into an existing platform that is already established and specializes in remote patient monitoring. Users create a personal account and add information about their drug treatment and nutrition program. The platform offers privacy, authentication, and a user-friendly environment where users can ask for details from their AI Assistant.

For research purposes, the user interface developed for integrating the AI Assistant has two functionalities. The primary one is a ChatBot capable of answering questions about the aforementioned scenarios; the second involves monitoring the workflow of the Agents coordinating to produce the final answer, their interaction with the MCP server, and the total processing time. This implementation was available only to the main developers of the prototype to assist in the evaluation methods.

4.2. Prototype

All previously discussed technologies, ideas, and architectures have been implemented in a cutting-edge application that serves as both an end-user product and a proof-of-concept prototype. The "Personal Medical Assistant" (PMA) client app is available and integrated within a digital health platform across iOS, Android, and Web applications. This is an extension of the existing version capable of integrating with eHealth platforms [35], but this

approach offers enhanced capabilities and innovative architectures with high scalability. This subsection presents the user interface for the two aforementioned scenarios. Figure 5 illustrates the main PMA interface.

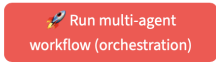
Multi-Agent Assistant using MCP

UI to observe the AutoGen multi-agent coordination in real time:

- SelectorGroupChat with planner, RAG, and final-answer agents
- Step-by-step log of **who speaks when**
- Optional internal events (tool calls / thoughts)
- Timeline + JSON export per run

And a chat-style interface to talk to the multi-agent assistant.

Orchestration View (full coordination log)



This will show the full internal coordination:

- All agent messages
- Optional internal events
- Timeline + per-step response time

Chat with the Multi-Agent Assistant

Ask something about your treatment, nutrition, or health tips... 

Figure 5. The main user interface of the PMA client application.

4.2.1. User Scenario

To begin with, the user, through the digital health app and after authentication, can open the PMA and ask something about their treatment, nutrition, or health tips. Moreover, it is important that the user has an updated PHR, so the MCP server will be capable of accessing these details. Figure 6 presents the UI of the user with an initial query. After the user inputs the question, the system seems to think about the answer, which means that the agents are cooperating and using the MCP server to create the final answer, as illustrated in Figure 7. The final outcome is in Figure 8, and it is important to mention that the PMA proposes to the user to always take the advice of their personal doctor to support their treatment.

Chat with the Personal Medical Assistant

Is Advil in my treatment plan? Give me the side effects. 

Figure 6. The PMA interface displaying the user's initial health query.

Chat with the Personal Medical Assistant

 Is Advil in my treatment plan? Give me the side effects.

 Thinking with the agent team...

Figure 7. The interface during the processing phase ("Thinking"), indicating active cooperation between the agents and the MCP server.

Chat with the Personal Medical Assistant



Is Advil in my treatment plan? Give me the side effects.



- **Advil (ibuprofen)** is not part of your current treatment plan based on the provided information.
- **Side effects of Advil:**
 - Common side effects:
 - Gastrointestinal discomfort (e.g., stomach pain, heartburn, nausea).
 - Dizziness.
 - Headache.
 - Severe side effects (less common):
 - Gastrointestinal bleeding.
 - Kidney problems.
 - Severe allergic reactions (e.g., rash, swelling, difficulty breathing).
 - **Caution:** People with a history of gastrointestinal issues or hypersensitivity to ibuprofen should use Advil carefully and consult a healthcare provider.

This information is for general guidance only and does not replace medical advice. Always consult your doctor or pharmacist for decisions about your treatment.

Figure 8. The final response generated by the MAS, which includes a recommendation to consult a personal doctor.

4.2.2. Monitoring Scenario

The MAS is an autonomous implementation in which agents cooperate together to solve the user’s requirements. To evaluate the communication between agents and how they interact with the MCP server, an extended feature was created within the main app, visible only to the researchers of this prototype. As presented in Figure 9, the evaluator can open the sidebar, add a query, and run the orchestrator workflow.

 **Settings (Orchestration View)**

Task to send to the team

Tips about improve heath diet.



☒ Show internal AgentEvents (thoughts, tool calls, etc.)

 Reset all orchestration runs

Multi-Agent Assistant using MCP

UI to observe the AutoGen multi-agent coordination in real time:

- SelectorGroupChat with planner, RAG, and final-answer agents
- Step-by-step log of **who speaks when**
- Optional internal events (tool calls / thoughts)

And a **chat-style interface** to talk to the multi-agent assistant.

Orchestration View (full coordination log)

 Run multi-agent workflow (orchestration)

This will show the **full internal coordination**:

- All agent messages
- Optional internal events
- Timeline + per-step response time

Figure 9. The evaluation interface showing the sidebar used to enter queries and trigger the orchestrator workflow.

Figure 10 illustrates the table generated by the system to monitor the operational workflow the MAS undertakes to address user queries. Furthermore, this view enables the observation of specific MCP tools invoked by each agent, as well as the execution time required for each process.

<https://doi.org/10.3390/electronics15040888>

	Step #	Agent	Preview	Response time (s)
	0	1 user	Tips about improve heath diet.	1.065
	1	2 PlanningAgent	1. RAGNutritionAgent : Retrieve general nutrition information and tips for improving a healthy diet. 2	1.087
	2	3 RAGNutritionAgent	[FunctionCall(id="call_KggIMIE0EPke8rWuef1caoEH", arguments={}, name="buildNutritionRAGIndex	0.991
	3	4 RAGNutritionAgent	[FunctionExecutionResult(content="[{"type": "text", "text": "\n\n \\\"status\\\": \\\"loaded\\\""}], "ann	0.033
	4	5 RAGNutritionAgent	[FunctionCall(id="call_geehngDZYIOHRIWdrQ6jcPO", arguments={"query": "tips for improving a healt	0.685
	5	6 RAGNutritionAgent	[FunctionExecutionResult(content="[{"type": "text", "text": "\n\n \\\"matches\\\": \\\"\\n\n\n \\\"rank\\	0.404
	6	7 RAGNutritionAgent	``json { "route": "RAGNutritionAgent", "items_considered": ["healthy diet", "nutrition tips"], "re	10.751
	7	8 FinalAnswerAgent	Here are some practical tips to improve your diet and overall health: - Start Small : Begin with sin	11.813

Figure 10. The system-generated interface for monitoring the MAS operational workflow. This table details the sequential order of agent execution, the specific data and functions retrieved via MCP tools (e.g., building and querying the RAG index), and the precise response duration for each message. *Note:* The truncated text in the ‘Preview’ column is a standard UI artifact of logging long JSON structures and does not affect the scientific understanding of the workflow. Additionally, the double asterisks (**) in Step 8 represent standard Markdown formatting for bold text generated by the language model.

5. Evaluation and Results

5.1. Evaluation Methodology

Through this iterative process, we identified **seven** key use cases that the prototype is designed to address.

1. **Presentation of Personal Data:** The MAS retrieves the user’s PHR and profile information and presents only the essential attributes, without exposing additional sensitive details such as nutrition logs, medication history, or physiological measurements.
2. **Personal Nutrition Information with Additional Knowledge:** The MAS accesses the user’s nutrition records from the PHR and enriches them using the RAG module to generate validated and accurate outcomes.
3. **Personal Drug Treatment Information with Additional Knowledge:** The MAS fetches medication-related information from the PHR and applies RAG-based augmentation to provide validated, evidence-informed insights and up-to-date information.
4. **General Nutrition Information with Additional Knowledge:** For general, non-personal nutrition queries, the MAS uses the RAG module for nutrition-based knowledge without referencing the user’s personal PHR data.
5. **General Drug Information with Additional Knowledge:** The MAS responds to general medication-related questions by leveraging RAG to generate validated knowledge, again without adding any personal health information from the user’s PHR.
6. **Combined Nutrition and Pharmacological Analysis:** In this advanced scenario, the MAS integrates data from both the Nutrition RAG Agent and the Medicine RAG Agent simultaneously. This allows the system to identify potential drug-nutrient interactions (e.g., verifying if a prescribed diet is safe given the user’s current medication) and provide comprehensive safety warnings that neither agent could generate in isolation.
7. **Real-Time Corrective Retrieval (AC-RAG):** This scenario addresses cases where the internal vector database contains insufficient, ambiguous, or outdated information. The **Judge Agent** actively evaluates the initial retrieval quality; if the relevance score falls below a predefined threshold of 0.7 [48], it autonomously triggers a **Web Drug Search** (e.g., NLM databases) via the MCP server. This corrective mechanism fetches the latest clinical data on-the-fly, which is then synthesized with the internal context to ensure the final response is factually current and accurate. Crucially, the system implements a dynamic update loop: validated external findings are automatically indexed into the vector database, allowing the assistant to incrementally expand its reusable knowledge base and preventing redundant external searches for future queries.

Table 1 illustrates the number of queries executed in each scenario and identifies which agent was responsible for resolving each task. Note that the **Judge Agent** is active in all RAG-related scenarios to validate content, but plays a critical active role in Scenario 7 by triggering external search tools.

Table 1. Matrix of agent assignments per scenario. The **Judge** column indicates the active participation of the validation layer.

Scenario	Queries	Planner	Platform	RAG Nutri	RAG Med	Judge	Final
1	10	X	X				X
2	15	X	X	X		X	X
3	15	X	X		X	X	X
4	20	X		X		X	X
5	20	X			X	X	X
6	15	X		X	X	X	X
7	15	X			X	X	X

Note: ‘X’ denotes that the corresponding agent is assigned and active during that specific scenario.

5.2. MCP Evaluation

A key component of the prototype is the MCP server and its integration within the MAS. Many applications hesitate to implement MCP due to its novelty and the lack of established evaluation protocols. One promising approach is the use of the DeepEval framework, which introduced the “MCP Use” metric. This metric evaluates how effectively an MCP-based LLM agent utilizes the available MCP servers, providing insight into whether the agent is leveraging external capabilities optimally during its reasoning and decision-making processes. Specifically, the metric assigns discrete scores ranging from 0 to 1, mapped to qualitative performance levels: 0 (Very Low), 0.25 (Low), 0.5 (Moderate), 0.75 (High), and 1 (Very High). It employs an LLM-as-a-judge approach to evaluate the invoked MCP primitives and the arguments generated by the application [49].

However, this method has the limitation that it typically evaluates one agent at a time and cannot calculate aggregate scores for Multi-Agent approaches. For the scope of this research, this method is extended with extra functionalities and adjusted to observe the entire process of the MAS and its interaction with the MCP server. Thus, it calculates MCP Use metrics per scenario, accounting for interactions where multiple agents utilize MCP tools. It also calculates an average score that corresponds to the general performance of the MCP usage.

Table 2 illustrates the results of the evaluation tests for the MCP, including the advanced CRAG scenario. The evaluation results demonstrate that the proposed system exhibits strong performance across all scenarios, even as the number of queries is **110**. The MCP Use metric shows a stable average value of **0.75 (High)**, indicating that the agents effectively leverage MCP servers to enrich their reasoning process. Notably, Scenario 7 achieves a “Very High” score due to the Judge Agent’s active invocation of both vector retrieval and web search tools; however, this comes with a slight trade-off in latency (11.50 s), which is expected given the real-time external data fetching. Despite this, the time required to complete the evaluations remains within a reasonable range, with an overall average of **7.06 s**, confirming that the system can process multiple queries efficiently without significant degradation in responsiveness. These outcomes suggest that the architecture is both reliable and scalable, achieving balanced performance in terms of MCP utilization and execution time, which is crucial for real-world deployment in healthcare environments. Finally, it is important to clarify that the reported duration corresponds strictly to the complete execution cycle of the MCP tools, encompassing processes such as external web

scraping or vector database retrieval, rather than the total end-to-end processing time of the system.

Table 2. MCP Usage per Scenario (Qualitative & Quantitative). Note that the reported duration reflects the full execution time of MCP tools, including web scraping and vector database retrieval tasks.

Scenario	Num of Queries	MCP Use Metric	Duration (s)
1	10	Very High	2.97
2	15	Moderate	7.44
3	15	High	8.20
4	20	High	5.24
5	20	High	5.10
6	15	High	9.25
7	15	Very High	11.50
Overall	110	High	7.06

Note: Bold text denotes overall aggregated values (totals and averages) across all scenarios.

5.3. Comparative RAG Evaluation: Standard vs. CAG vs. AC-RAG

GenAI is a technique at the cutting edge of technology, and every digital platform associated with healthcare attempts to integrate it through AI Assistants enhanced with LLMs. However, the critical challenge in medical informatics is not merely generating fluent text, but ensuring strict adherence to verified clinical protocols while maintaining responsiveness. To validate our proposed solution, we conducted a quantitative comparative evaluation against two prominent architectures: **Typical RAG** and **Cache-Augmented Generation (CAG)**. Crucially, this comparative study was executed entirely on a self-hosted local infrastructure, ensuring that all performance metrics reflect a privacy-centric, offline operational environment rather than cloud-based API.

To validate the proposed architecture, we conducted a comprehensive quantitative evaluation using the “LLM-as-a-judge” paradigm. It is important to clarify that this method replaces subjective human evaluation with a high-performance Large Language Model (specifically “gpt-oss:120b”) to grade the system’s outputs; consequently, no human participants were involved in the scoring process. The evaluation was performed on a dataset of 95 distinct queries, covering the RAG based use cases described previously in Table 1.

The primary metric, **Faithfulness (FF)**, employs the judge model to measure the quality of the generator by evaluating whether the actual output factually aligns with the contents of the retrieved context [49,50]. This serves as the most critical safety metric, measuring the system’s ability to strictly adhere to retrieved medical facts without hallucinating. To assess utility, we employed **Answer Relevancy (AR)**, which evaluates how relevant the actual output of the LLM application is compared to the provided input [49,50]. Furthermore, the quality of the retrieval pipeline was measured using **Context Relevance (CR)**. This metric evaluates the overall relevance of the information presented in the retrieval context for a given input, where higher scores indicate that contexts are effectively filtered and closely aligned with the user’s query [49,50]. Finally, to evaluate operational trade-offs, we analyzed **Latency and Token Usage**, measuring the time cost (seconds) and computational volume (tokens per request), including system prompts. It is pertinent to note that the observed higher latency is primarily attributed to the use of a self-hosted, local infrastructure for the 120 billion-parameter models, rather than cloud-based API services.

5.3.1. Results Analysis: The Accuracy-Efficiency Trade-off

Table 3 presents the averaged results across all test scenarios. The data reveals a critical distinction: while the baselines achieve technical “faithfulness” by adhering strictly to

provided text, they suffer from significant limitations in relevance and precision compared to the **AC-RAG**.

Table 3. Comparative Performance Statistics (Averaged).

Approach	Faithfulness	Answer Relevancy	Context Relevance	Avg Latency (s)	Tokens/Req
Typical RAG	0.85	0.64	0.55	30.30	37,057
CAG (Cache)	0.82	0.66	0.56	26.54	68,593
AC-RAG	0.88	0.86	0.78	33.40	52,500

Note: Bold text indicates the results of the proposed approach.

Regarding Typical RAG, although it achieves good **Faithfulness (0.85)**, this metric is misleading in isolation. The low **Context Relevance (0.55)** and **Answer Relevancy (0.64)** indicate a severe accuracy issue: the system retrieves and repeats large volumes of data (~37 k tokens) without effectively filtering for the user’s specific question. It is “faithful” to the retrieved noise rather than “accurate” to the user’s intent. Furthermore, this inefficient processing of irrelevant tokens results in the highest **Latency (30.30 s)**, rendering the approach both slow and imprecise.

Conversely, the Cache-Augmented (CAG) approach prioritizes speed (26.54 s) by pre-loading massive amounts of data (~68 k tokens). However, this “brute-force” strategy dilutes the signal with noise. Similar to Typical RAG, the **Answer Relevancy (0.66)** remains poor because the model struggles to locate the specific answer buried within the cached thousands of tokens. While faster than Typical RAG, it essentially sacrifices precision for speed.

In contrast, our proposed AC-RAG architecture addresses this accuracy gap by introducing the Judge Agent. By actively filtering context *before* generation, the system achieves significantly higher **Answer Relevancy (0.86)** and **Context Relevance (0.78)**. The **Faithfulness (0.88)** represents a more useful “real-world” accuracy—providing a specific, relevant answer rather than blindly regurgitating a document. The intermediate **Latency (~33.40 s)** reflects the necessary time taken to verify this relevance, ensuring that the final response is safe and clinically valid.

5.3.2. Detailed Breakdown of the Proposed Approach

To further analyze the capabilities of the AC-RAG, Table 4 breaks down performance by domain, including the complex “**Combined**” scenario.

In the specialized domains of Nutrition and Medicine, the system achieves high Answer Relevancy (0.98 for Nutrition), demonstrating the Planner’s ability to effectively route queries to the correct domain expert. Regarding Context Relevance (CR), the metric indicates that in the case of medicines, the retrieved contexts are highly relevant; however, for nutrition, the results are lower. This disparity occurs because nutrition queries often consist of general questions about healthy diets, whereas in medicine cases, users typically inquire about specific drugs. Consequently, when seeking additional information about a specific topic, it is easier for the retrieval system to identify and extract relevant knowledge.

Table 4. Proposed Approach: Performance by Scenario.

Scenario	Faithfulness	Answer Relevancy	Context Relevance	Latency (s)
Nutrition Agent	0.75	0.98	0.49	31.20
Medicine Agent	0.72	0.83	0.81	32.50
Combined (Nutri + Med)	0.82	0.91	0.85	33.50

In the **Combined Analysis** scenario, the system activates both the Nutrition and Medicine RAG agents simultaneously to check for drug-nutrient interactions. This results in a **Faithfulness score of 0.82** and **Context Relevance of 0.85**, suggesting that the system successfully merges disparate data sources into a coherent clinical recommendation. Notably, these metrics differ from those in Table 3 because this specific scenario does not always require the full activation of the Judge layer; consequently, the MAS sometimes bypasses the validation loop.

5.4. System Architecture Validation

For the purpose of this study, we evaluated feature-based and overall comparisons between existing architectures and the prototype presented in this paper. Key evaluation features were determined based on their direct relevance to healthcare-based AI systems, where reliability, accuracy, and operational safety are critical requirements. These features, such as memory, coordination mechanisms, scalability, and task allocation, were selected because they influence how effectively an architecture can support complex medical workflows. In healthcare environments, systems must manage evolving patient contexts, integrate medical data sources, operate autonomously under uncertainty, and maintain up-to-date knowledge. Therefore, the chosen features reflect the core functional and technical dimensions that impact performance, scalability, and usability in real-world eHealth scenarios.

Table 5 provides a comprehensive analysis, contrasting our proposed **AC-RAG + MCP** approach against both foundational architectures (Generic LLM, Single Agent, Standard MAS) and advanced retrieval techniques (Typical RAG, CAG). The comparison highlights distinct limitations in existing approaches. Specifically, **Generic LLMs** and **Single Agents** lack the distributed reasoning required for complex medical cases, while a **Standard MAS** without RAG often suffers from hallucination risks due to a lack of grounded knowledge. Regarding retrieval techniques, **Typical RAG** improves accuracy but remains “passive,” unable to detect when retrieved data is irrelevant. Similarly, **Cache-Augmented Generation (CAG)** offers speed but fails in dynamic healthcare settings where data, such as patient vitals or drug recalls, changes in real time.

In contrast, our proposed architecture overcomes these barriers by integrating **Agentic Orchestration** with **Corrective Active Retrieval**. The inclusion of a “Judge Agent” ensures that data is not only retrieved but dynamically validated and corrected via the web when necessary, making it the most suitable candidate for high-stakes healthcare applications.

Table 5. Comprehensive Comparative Architecture Analysis.

Feature	Generic LLM	Single Agent	Std. MAS	Std. RAG	CAG	Proposed (AC-RAG)
Memory	None	Tool-based	Local	Vector Only	Pre-loaded Context	Shared + Vector + Web
Context	Short	Extended	Distributed	Retrieval Dependent	Fixed/Cached	Active & Verified
Reasoning	Generic	Tool-aided	Fragmented	Fact-based	Context-based	Self-Correcting
Accuracy	Low	Moderate	Inconsistent	Variable	High (Static)	Very High
Scalability	Low	Moderate	Moderate	High	Limited	High (MCP)
Data Type	Static Weights	External Tools	Local Data	Vector Snapshot	Cached Context	Real-Time Hybrid
Correction	None	Manual	None	None	None	Auto (Judge)
Healthcare Fit	Low	Moderate	Moderate	Moderate	Low (Static)	Very High

Note: Bold text indicates the evaluated features (first column) and highlights the specific capabilities of the proposed AC-RAG architecture (final column).

6. Discussion

This work demonstrates the architecture and implementation effectiveness of a prototype approach for an extended version of a Personal Medical Assistant, which could easily integrate with existing platforms for Telehealth. Core aspects of the system include a MAS which integrates with an MCP server that focuses on medical information about drugs and nutrition. The primary feature of the approach is that the MAS uses the MCP to provide several tools to support the outcomes of the agents that take part in the MAS. The method of RAG is used to extend the knowledge base of the LLM to produce more accurate responses for personal treatment, drug details, and healthy nutrition. Moreover, the Agentic architecture enables autonomous communication and coordination among agents, allowing them to distribute tasks, exchange context, and collectively refine the final output with minimal human intervention. The adoption of the MCP architecture provides several key advantages for the proposed system. MCP enables standardized and secure communication between agents and external tools, ensuring consistent data exchange. Its modular design allows new capabilities to be added without altering the overall system structure, significantly improving extensibility and maintainability. Additionally, MCP isolates tool execution from the agent layer, enhancing reliability and reducing the risk of LLM-based errors from hallucinations. Crucially, the comparative evaluation highlights a distinct advantage of this Agentic Corrective RAG approach over traditional “brute-force” retrieval methods like Cache-Augmented Generation (CAG). While baselines such as CAG prioritize speed by flooding the context window with massive amounts of pre-loaded data, this often results in “context dilution,” where the specific answer is obscured by irrelevant noise. In contrast, our architecture introduces an active “Judge Agent” that filters and validates information before it reaches the generation phase. This ensures that the system does not merely retrieve *more* data, but *better* data, effectively prioritizing clinical relevance and safety over raw processing speed—a trade-off that is essential in high-stakes healthcare environments. Future research directions will focus on enhancing the adaptive capabilities of the “Judge Agent” through Reinforcement Learning from Human Feedback (RLHF). By incorporating iterative feedback from healthcare professionals into the validation loop, we aim to fine-tune the agent’s sensitivity thresholds for triggering external web searches. This would allow the system to dynamically learn from expert corrections, further optimizing the critical balance between autonomous decision-making and patient safety standards. The key contribution of this work is not the implementation of another Medical AI System, but the creation of an innovative approach that is more autonomous and scalable, but simultaneously accurate in the answers that it provides, using technologies that are at the state of the art. Hence, this proposed prototype highlights a novel approach to the literature gap of using commodity LLM platforms for enriching drug information for personal treatment and nutrition in the field of Telehealth.

7. Conclusions

This prototype integrates AI for a Personal Medical Assistant, which could be adopted easily by Telehealth and eHealth digital platforms. The approach incorporates a range of challenging scenarios which are crucial in patient monitoring and for human well-being. In this work, we identified a problem that existing solutions have, and with innovative methods, we present an approach that could handle it. Thus, the demonstrated innovative techniques have been integrated into a state-of-the-art prototype application in which a Multi-Agent System, enhanced by an MCP server, builds a stable and scalable AI Assistant that could easily coexist in digital platforms whose purpose is to support Telehealth. Most significantly, it can adjust the knowledge base that will be used to create a more accurate and valid answer. The RAG method is in the spotlight of the way that the prototype

builds the responses and presents them. Thus, it could handle with high accuracy two core scenarios. The first one is about drug treatment, and the second is about healthy diet and nutrition. Moreover, the innovative architecture that is adopted delivers a highly scalable system with reasoning capabilities and task allocation to achieve more efficient resource utilization. By introducing a self-correcting ‘Judge,’ we resolve the difficult balance in medical AI between avoiding false information and actually providing helpful answers. By empowering the system to autonomously verify its own retrieval quality and fetch real-time web data when internal vectors are insufficient, we bridge the gap between static knowledge bases and the dynamic nature of medical science. Furthermore, this modular architecture is not limited to healthcare; it can be readily adapted to other high-stakes domains that demand high accuracy and real-time verification. To sum up, this approach is an AI Assistant with extended capabilities, able to provide valid, effective, and accurate Q&A conversations based on medical treatment, drug details, and nutrition for patients, healthcare professionals, and the general population.

Author Contributions: D.K. contributed to the design process, implemented the algorithm, created the prototype application, performed the evaluation tests and wrote the initial draft of the manuscript. I.M., P.T. and A.M. provided the research direction, scientifically supervised the study, critically reviewed the script and shared observations and feedback. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code and data generation scripts used in this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kalathas, D.; Koulouris, D.; Menychtas, A.; Tsanakas, P.; Maglogiannis, I. Continuous machine learning for assisting AR indoor navigation. *SN Comput. Sci.* **2024**, *5*, 913.
2. Bulla, C.; Parushetti, C.; Teli, A.; Aski, S.; Koppad, S. A review of AI based medical assistant chatbot. *Res. Appl. Web Dev. Des.* **2020**, *3*, 1–14.
3. Pap, I.A.; Oniga, S. eHealth assistant AI chatbot using a large language model to provide personalized answers through secure decentralized communication. *Sensors* **2024**, *24*, 6140.
4. Ferrag, M.A.; Tihanyi, N.; Debbah, M. From LLM reasoning to autonomous AI agents: A comprehensive review. *arXiv* **2025**, arXiv:2504.19678.
5. Hadfield, G.K.; Koh, A. An economy of AI agents. *arXiv* **2025**, arXiv:2509.01063.
6. Masterman, T.; Besen, S.; Sawtell, M.; Chao, A. The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: A survey. *arXiv* **2024**, arXiv:2404.11584.
7. Tang, Q.; Xiang, H.; Yu, L.; Yu, B.; Lu, Y.; Han, X.; Sun, L.; Zhang, W.; Wang, P.; Liu, S.; et al. Beyond turn limits: Training deep search agents with dynamic context window. *arXiv* **2025**, arXiv:2510.08276.
8. Khatami, S.; Frantz, C. Prompt Engineering Guidance for Conceptual Agent-based Model Extraction using Large Language Models. *arXiv* **2024**, arXiv:2412.04056.
9. Li, X.; Wang, S.; Zeng, S.; Wu, Y.; Yang, Y. A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges. *Vicinagearth* **2024**, *1*, 9.
10. He, J.; Treude, C.; Lo, D. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. *ACM Trans. Softw. Eng. Methodol.* **2025**, *34*, 124.
11. Ehtesham, A.; Singh, A.; Gupta, G.K.; Kumar, S. A survey of agent interoperability protocols: Model context protocol (MCP), agent communication protocol (ACP), agent-to-agent protocol (A2A), and agent network protocol (ANP). *arXiv* **2025**, arXiv:2505.02279.

12. Angert, T.; Suzara, M.; Han, J.; Pondoc, C.; Subramonyam, H. Spellburst: A node-based interface for exploratory creative coding with natural language prompts. In Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, San Francisco, CA, USA, 29 October–1 November 2023; pp. 1–22.
13. Zhang, B.; Liu, Z.; Cherry, C.; Firat, O. When scaling meets LLM finetuning: The effect of data, model and finetuning method. *arXiv* **2024**, arXiv:2402.17193.
14. Sawarkar, K.; Mangal, A.; Solanki, S.R. Blended RAG: Improving RAG (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In Proceedings of the 2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR), San Jose, CA, USA, 7–9 August 2024; pp. 155–161.
15. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
16. Joshi, S. Introduction to Vector Databases for Generative AI: Applications, Performance, Future Projections, and Cost Considerations. *Int. Adv. Res. J. Sci. Eng. Technol.* **2025**, *12*, 79–93.
17. Yu, W.; Iter, D.; Wang, S.; Xu, Y.; Ju, M.; Sanyal, S.; Zhu, C.; Zeng, M.; Jiang, M. Generate rather than retrieve: Large language models are strong context generators. *arXiv* **2022**, arXiv:2209.10063.
18. Singh, A.; Ehtesham, A.; Kumar, S.; Khoei, T.T. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv* **2025**, arXiv:2501.09136.
19. Yan, S.-Q.; Gu, J.-C.; Zhu, Y.; Ling, Z.-H. Corrective retrieval augmented generation. *arXiv* **2024**, arXiv:2401.15884.
20. Joshi, S. Review of autonomous systems and collaborative AI agent frameworks. *Int. J. Sci. Res. Arch.* **2025**, *14*, 961–972.
21. Chan, B.J.; Chen, C.-T.; Cheng, J.-H.; Huang, H.-H. Don't do rag: When cache-augmented generation is all you need for knowledge tasks. In *Companion Proceedings of the ACM on Web Conference 2025*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 893–897.
22. Gilson, A.; Safranek, C.W.; Huang, T.; Socrates, V.; Chi, L.; Taylor, R.A.; Chartash, D. How does ChatGPT perform on the United States Medical Licensing Examination (USMLE)? The implications of large language models for medical education and knowledge assessment. *JMIR Med. Educ.* **2023**, *9*, e45312.
23. Kung, T.H.; Cheatham, M.; Medenilla, A.; Sillos, C.; De Leon, L.; Elepaño, C.; Madriaga, M.; Aggabao, R.; Diaz-Candido, G.; Maningo, J.; et al. Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models. *PLoS Digit. Health* **2023**, *2*, e0000198.
24. Koulouris, D.; Kalathas, D.; Menychtas, A.; Pasiadis, A.; Athanasiou, V.; Tsanakas, P.; Maglogiannis, I. A Web-Based Information System for Medical Forensics Incorporating Generative AI in Reports Compilation. In *Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations, Limassol, Cyprus, 26–29 June 2025*; Springer: Cham, Switzerland, 2026; pp. 342–352.
25. Singhal, K.; Azizi, S.; Tu, T.; Mahdavi, S.S.; Wei, J.; Chung, H.W.; Scales, N.; Tanwani, A.; Cole-Lewis, H.; Pfohl, S.; et al. Large language models encode clinical knowledge. *Nature* **2023**, *620*, 172–180.
26. Singhal, K.; Tu, T.; Gottweis, J.; Sayres, R.; Wulczyn, E.; Amin, M.; Hou, L.; Clark, K.; Pfohl, S.R.; Cole-Lewis, H.; et al. Toward expert-level medical question answering with large language models. *Nat. Med.* **2025**, *31*, 943–950.
27. Li, Y.; Li, Z.; Zhang, K.; Dan, R.; Jiang, S.; Zhang, Y. ChatDoctor: A medical chat model fine-tuned on a large language model Meta-AI (LLaMA) using medical domain knowledge. *Cureus* **2023**, *15*, e40895.
28. Han, T.; Adams, L.C.; Papaioannou, J.M.; Grundmann, P.; Oberhauser, T.; Löser, A.; Truhn, D.; Bressem, K.K. MedAlpaca—An open-source collection of medical conversational AI models and training data. *arXiv* **2023**, arXiv:2304.08247.
29. Luo, R.; Sun, L.; Xia, Y.; Qin, T.; Zhang, S.; Poon, H.; Liu, T.Y. BioGPT: Generative pre-trained transformer for biomedical text generation and mining. *Briefings Bioinform.* **2022**, *23*, bbac409.
30. Samdani, G.; Dixit, Y.; Viswanathan, G. Leveraging LangGraph and AutoGen for Agentic AI Frameworks. *World J. Adv. Eng. Technol. Sci.* **2023**, *8*, 402–411.
31. Abbasian, M.; Azimi, I.; Rahmani, A.M.; Jain, R. Conversational health agents: A personalized LLM-powered agent framework. *arXiv* **2023**, arXiv:2310.02374.
32. Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand, 2024; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 15174–15186.
33. Zhang, C.; Yang, Z.; Liu, J.; Li, Y.; Han, Y.; Chen, X.; Huang, Z.; Fu, B.; Yu, G. AppAgent: Multimodal agents as smartphone users. In Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 26 April–1 May 2025; pp. 1–20.
34. Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. AgentBench: Evaluating LLMs as agents. *arXiv* **2023**, arXiv:2308.03688.

35. Kalathas, D.; Menychtas, A.; Maglogiannis, I.; Tsanakas, P. Incorporating Medical Assistants in eHealth Environments Using an Agentic RAG Approach. In *Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations, Limassol, Cyprus, 26–29 June 2025*; Springer: Cham, Switzerland, 2026; pp. 154–167.
36. Alghamdi, H.M.; Mostafa, A. Towards reliable healthcare LLM agents: A case study for pilgrims during Hajj. *Information* **2024**, *15*, 371.
37. Jagatap, A.; Merugu, S.; Comar, P.M. RxLens: Multi-Agent LLM-powered Scan and Order for Pharmacy. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track), Online, 2025*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 822–832.
38. Agrawal, R.; Kumar, H. Enhancing Cache-Augmented Generation (CAG) with Adaptive Contextual Compression for Scalable Knowledge Integration. *arXiv* **2025**, arXiv:2505.08261.
39. Gangavarapu, R.; Srinivasan, A.R.A.; Moparthi, V. Evaluating Accuracy in Large Language Models: Benchmarking Corrective Rag vs. Naive Retrieval Augmented Generation Approach. In *Proceedings of the 2025 IEEE International Conference on AI and Data Analytics (ICAD), Medford, MA, USA, 24 June 2025*; pp. 1–7.
40. Marcondes, F.S.; Gala, A.; Magalhães, R.; Perez de Britto, F.; Durães, D.; Novais, P. Using Ollama. In *Natural Language Analytics with Generative Large-Language Models: A Practical Approach with Ollama and Open-Source LLMs*; Springer: Cham, Switzerland, 2025; pp. 23–35.
41. Şakar, T.; Emekci, H. Maximizing RAG efficiency: A comparative analysis of RAG methods. *Nat. Lang. Process.* **2025**, *31*, 1–25.
42. Douze, M.; Guzhva, A.; Deng, C.; Johnson, J.; Szilvasy, G.; Mazaré, P.E.; Lomeli, M.; Hosseini, L.; Jégou, H. The Faiss library. *IEEE Trans. Big Data* **2025**, 1–17. <https://doi.org/10.1109/TBDATA.2025.3618474>.
43. World Health Organization. Healthy Diet. Available online: <https://www.who.int/news-room/fact-sheets/detail/healthy-diet> (accessed on 10 December 2025).
44. U.S. Department of Agriculture, Agricultural Research Service. FoodData Central. Available online: <https://fdc.nal.usda.gov/> (accessed on 10 December 2025).
45. National Library of Medicine (US). DailyMed. Available online: <https://dailymed.nlm.nih.gov/dailymed/> (accessed on 10 December 2025).
46. National Library of Medicine (US). MedlinePlus XML Files. Available online: <https://medlineplus.gov/xml.html> (accessed on 10 December 2025).
47. Ray, P.P. A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions. *TechRxiv* **2025**. <https://doi.org/10.36227/techrxiv.174495492.22752319/v1>.
48. Han, S.; Junior, G.T.; Balough, T.; Zhou, W. Judge’s Verdict: A Comprehensive Analysis of LLM Judge Capability Through Human Agreement. *arXiv* **2025**, arXiv:2510.09738.
49. Alabdulwahab, A.; Japic, C.; Le, C.; Dubey, D.; Trivedi, D.; Hope, J.; Stone, P.; Srivastava, S.; Tashman, A.; Zhang, A. Comparative Study of Large Language Model Evaluation Frameworks with a Focus on NLP vs. LLM-As-A-Judge Metrics. In *Proceedings of the 2025 Systems and Information Engineering Design Symposium (SIEDS), Charlottesville, VA, USA, 2 May 2025*; pp. 410–415.
50. Es, S.; James, J.; Anke, L.E.; Schockaert, S. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, St. Julian’s, Malta, 2024*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 150–158.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.