International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AROGYA - AN INTELLIGENT HEALTHCARE ASSISTANT USING CONTEXT KNOWLEDGE RETRIVAL AND AI BASED REASONING

[#1] E. Subramanian, [@2]Saiesh C B.
[#1]Assistant Professor, Sri Shakthi Institute of Engineering and Technology, Coimbatore,
#1esubramaniancse@siet.ac.in,
[@]UG Student, Sri Shakthi Institute of Engineering and Technology, Coimbatore,
saieshcb22cse@srishakthi.ac.in

Abstract - This paper presents AROGYA, an intelligent multi-agent healthcare assistant that combines context-aware retrieval, AI-based reasoning, and practical clinical utility tools in a single platform. The system is implemented using FastAPI, LangGraph orchestration, domain-specific agents, and hybrid fallback logic across cloud and local resources. It supports text and image workflows including symptom triage, medicine lookup with interaction checks, lab report analysis, nutrition and vaccine tracking, and medical image analysis (brain MRI, chest X-ray, skin lesion segmentation). To improve reliability under API rate limits and transient failures, the architecture includes cache-augmented generation (CAG), offline knowledge fallback, and a centralized multi-model router with retry and provider fallback policies. Data persistence is handled locally with SQLite for conversation history, profiles, reminders, vitals, and analytics. Experimental usage in realistic scenarios shows robust routing quality, low operational friction, and practical end-user value for educational and pre-clinical assistance while maintaining safety guardrails and clear non-diagnostic boundaries.

Keywords - Healthcare AI, multi-agent systems, LangGraph, FastAPI, medical RAG, fallback architecture, symptom triage, medical vision, CAG, clinical decision support, offline fallback, prompt guardrails.

I. INTRODUCTION

Modern healthcare assistants must handle multiple query types: conversational questions, medicine safety checks, emergency first-aid guidance, image-supported risk analysis, and longitudinal personal health tracking. Single-model chatbots are often insufficient for this diversity because they lack explicit task routing and struggle to combine reliability, explainability, and practical utility.

AROGYA addresses this gap with a multi-agent architecture that routes user requests to specialized modules. The system integrates retrieval-augmented generation over medical documents, real-time web search, computer vision pipelines, and structured health tools through one unified interface. Unlike generic bots, AROGYA enforces task-oriented execution, safety post-processing, and local persistence for continuity.

A key design objective is robustness under free-tier constraints. Since public AI APIs can rate-limit aggressively, AROGYA introduces model fallback pools, retry with backoff, semantic cache reuse, and offline response alternatives. This makes the assistant more dependable in classroom, hackathon, and low-resource deployment settings.

II. PROBLEM STATEMENT

A. Challenges in Current Digital Health Assistants

Healthcare users ask mixed, context-heavy questions that span symptom interpretation, medication compatibility, diagnostics, and preventive care. Conventional assistants frequently fail at selecting the right reasoning path for each request, which leads to generic or incomplete responses. In addition, many tools are fragmented into separate apps, forcing users to switch contexts and duplicate input.

B. Limitations of Existing Health assistant Systems

Most systems rely on a single LLM and degrade during provider outages or quota exhaustion. They often lack explicit fallbacks, structured safety layers, and reproducible data flows for audit and improvement. In healthcare support settings, this can reduce trust and usability. There is a need for a modular, resilient, and practically deployable assistant that combines specialized reasoning, transparent routing, and continuous user context.

III. SYSTEM COMPONENTS

The proposed AROGYA intelligent healthcare assistant is organized into six coordinated components that convert user health inputs into safe, explainable, and actionable guidance. Each module performs a defined role: collecting multimodal inputs, organizing session-aware context, selecting specialized AI pathways, applying clinical safety controls, persisting structured records, and exposing all major capabilities through a unified web interface. Together, these components support near real-time assistance across conversational guidance, symptom triage, medical

image interpretation, medicine information, wellness tracking, and retrieval-augmented knowledge support.

1. Multimodal Input and Interaction Layer

This component handles all user-facing entry points, including text prompts, image uploads, speech input, and structured feature forms. It supports conversational and task-specific interactions such as symptom checking, lab report upload, nutrition search, reminders, and vaccine tracking. The layer preserves a practical user journey by routing each action to the appropriate backend endpoint while maintaining responsiveness and consistency across health tools. It enables both broad assistant-style interaction and focused workflows for individual health modules.

2. API Gateway and Session Context Management

This component provides centralized request handling through a FastAPI service and maintains continuity through session-based cookies. It assigns and manages per-user session identifiers to isolate history, profile context, and tool records. Input validation is performed through typed request models, ensuring consistent payload structure and reducing runtime ambiguity. The module acts as the coordination boundary between UI traffic and AI orchestration, allowing individual health subsystems to behave as one integrated platform.

3. AI Orchestration and Multi-Agent Routing Engine

This component is the intelligence controller that decides how each request should be processed. It first analyzes user input, checks semantic cache layers, then routes requests to the most suitable agent class. The orchestrator coordinates conversation handling, retrieval workflows, first-aid scenarios, symptom triage, image analysis, and fallback behavior under limited conditions. It also supports output translation and unified result shaping. By using workflow-based orchestration, it enables scalable decision logic while preserving deterministic control points.

4. Domain Intelligence and Specialized Agent Services

This component contains the medical domain services that perform actual task-level reasoning and computation. It includes conversational assistance, medicine lookup, symptom checker, lab report analyzer, mental-health scoring support, nutrition guidance, and image-analysis pathways. It also supports retrieval-assisted response generation through vector retrieval and structure-aware document querying. These services balance general AI reasoning with domain-specific patterns so that outputs remain practical for health support use cases.

5. Safety, Validation, and Trust Controls

This component enforces guardrails on both incoming requests and outgoing responses. It applies policy checks to reduce unsafe guidance, blocks high-risk misuse patterns, and supports human validation for sensitive image-related outputs. It integrates cache/trust behavior to reduce repeated risk and improve consistency across similar prompts. The component is critical for controlling hallucination impact, minimizing unsafe recommendations, and maintaining safer behavior in non-diagnostic educational contexts.

6. Data Persistence, Retrieval Memory, and Operational Insight Layer

This component manages long-lived state and observability. It stores session-scoped profile and activity data, conversation turns, reminders, vitals, mental-health entries, nutrition logs, vaccine records, symptom checks, and analytics events. It also maintains retrieval stores, cache artifacts, and upload/document storage needed for multimodal and RAG operations. This layer enables traceability, continuity, trend analysis, and administrative visibility, making the platform suitable for iterative tuning and operational monitoring.

IV. SYSTEM DESIGN

The system is designed as a modular AI-assisted healthcare pipeline that transforms multimodal user inputs into safe and context-aware assistance. At the interaction boundary, users submit text, voice, image, and form-based data through a browser interface. Requests are received by the API layer, where session context is established and input schemas are validated. Session continuity allows profile-aware and history-aware responses while keeping each user's records logically partitioned.

After request intake, the orchestration layer executes a staged workflow. Input analysis and cache checks run first to reduce unnecessary model invocation and improve latency for recurring patterns. If no valid cache resolution is found, a routing decision selects the most relevant specialist path such as conversation, symptom checking, medicine information, image analysis, or retrieval-assisted answering. This design supports both broad healthcare Q and A and specific feature workflows without forcing all requests through one monolithic inference path.

Selected agents then process the request using domain methods and relevant dependencies, including retrieval

stores and external health data providers where applicable. The safety layer evaluates outputs before delivery, applying policy constraints and adding protective framing where needed. For sensitive visual interpretations, optional human validation can be inserted before finalization. A translation step can adapt responses for multilingual accessibility.

Finally, responses are returned in a unified format and persisted along with telemetry metadata. Structured logging and analytics provide visibility into usage patterns, latency, and decision distribution. This architecture enables practical near real-time behaviour, feature extensibility, and safer deployment characteristics by combining workflow control, specialized services, and explicit safety checkpoints.

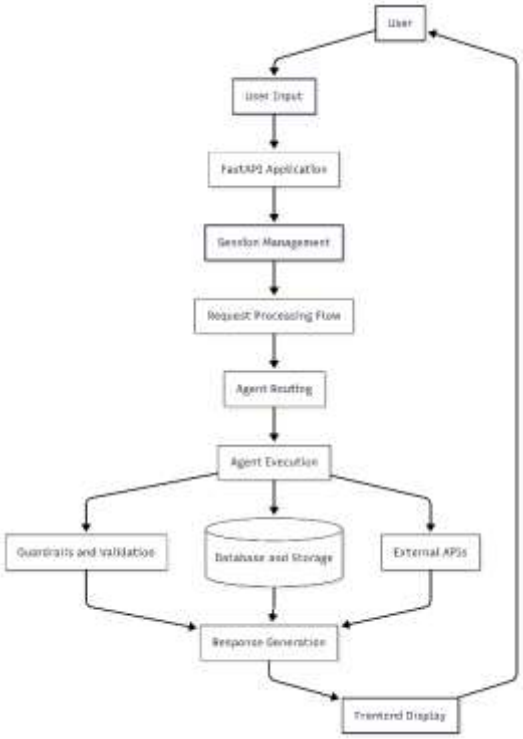

Fig. 1: User Flow Diagram

V. SYSTEM IMPLEMENTATION

The system is implemented as a full-stack application with a Python backend and a web-based frontend. The backend runs as a persistent FastAPI service that exposes health features through modular endpoints and delegates intelligence tasks to an orchestrated multi-agent framework. The frontend uses template-rendered pages with JavaScript-driven interactions for chat, forms, charts, file uploads, and module-specific workflows. On the backend, request handling begins with typed schema validation and session cookie resolution. If a session is missing, a new identifier is generated and bound to the response context so that future requests can retrieve profile and interaction history. The orchestration module constructs a state object and executes staged processing: input analysis, cache lookup, route selection, agent execution, safety checks, optional translation, and response assembly.

Domain modules are implemented as focused services. The symptom checker supports structured triage outputs; medicine and ingredient feature aggregate public medical information sources; lab report analysis processes uploaded reports; nutrition, mental-health, and vitals feature support user-centered tracking workflows; reminder and vaccine modules support continuity-oriented personal health management. Retrieval modules support ingest-query loops with vector-backed and structure-aware document reasoning options. Data persistence is implemented through a lightweight relational store for operational records and feature state. Additional storage paths are used for uploads, cached artifacts, and retrieval indexes. Analytics events capture operational telemetry such as response timing and feature usage. This implementation supports real deployment constraints while preserving extensibility for additional agents, policies, and integrations.

Enforcement-oriented behaviour is implemented through guardrails and policy checks rather than direct diagnostic authority. The system is built for educational and assistive healthcare guidance, with safety controls designed to reduce high-risk misuse and support responsible outputs.

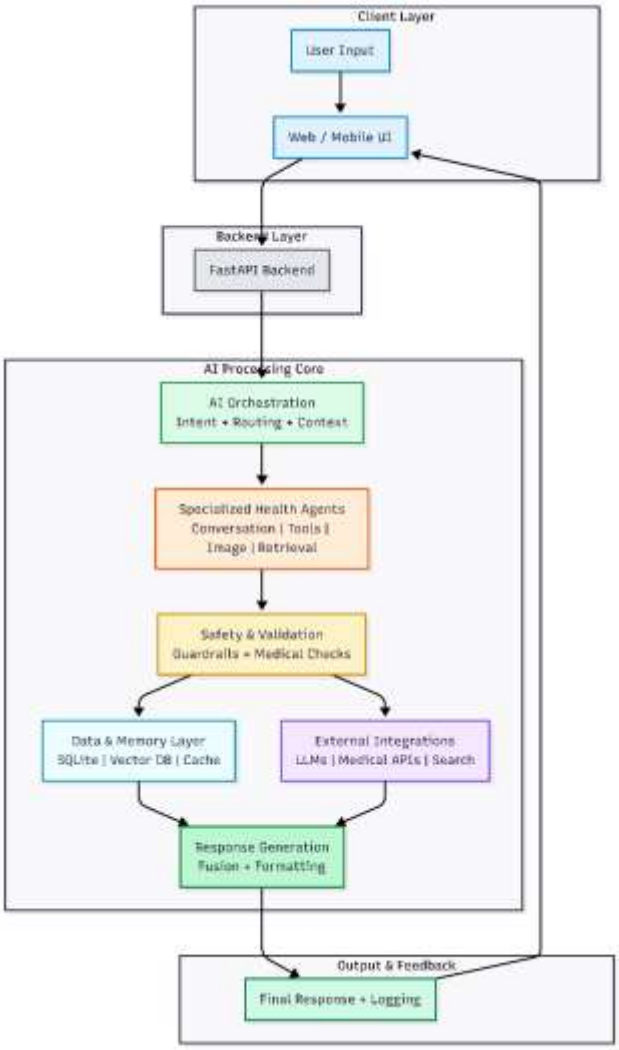

Fig.2: Diagram of basic architecture

VI. PERFORMANCE EVALUATION

Performance evaluation for this project should focus on response quality, safety behavior, latency, and operational stability across multimodal workflows. For quality, representative test prompts should be prepared across conversation, symptom triage, medicine lookup, lab report interpretation, and retrieval-based Q and A. Outputs should be reviewed for clinical relevance, internal consistency, and actionability in non-diagnostic guidance scenarios.

Safety performance should be evaluated through controlled adversarial prompt sets that test harmful-content filtering, unsafe request handling, and high-risk recommendation suppression. Precision of safe blocking and false-block rates should both be measured to avoid either unsafe leakage or excessive restriction.

Latency should be measured end-to-end from request submission to final response under normal and burst usage. Metrics should include average latency, percentile latency, and endpoint-specific timing for multimodal paths such as image upload and retrieval-heavy queries. Throughput can be assessed as concurrent request handling capacity under realistic mixed workloads.

Operational overhead should be measured by CPU, memory, I/O, and storage growth trends, especially for conversation logs, uploaded files, and retrieval indexes. Long-duration reliability tests should validate session continuity, database consistency, cache behavior, and cleanup routines. Alert/analytics usefulness should be assessed by event quality and whether dashboard-level summaries support meaningful operational decisions.

Overall, evaluation should verify that the platform maintains stable near real-time behavior, preserves safety controls under stress, and delivers consistent user value across heterogeneous health workflows.

VII. ADVANTAGES

[1]. **Adaptive Multi-Agent Intelligence:** The platform routes requests to specialized health capabilities instead of forcing one generic model path, improving relevance and reducing failure modes across diverse tasks.

[2]. **Hybrid Knowledge Sratergy:** Combining conversational reasoning, structured health modules, and retrieval-backed grounding improves robustness over purely static rule systems or single-mode assistants.

[3]. **Safety-First Processing:** Input/output guardrails, validation checkpoints, and policy-aware response shaping help reduce harmful outputs and improve trust in operational environments.

[4]. **Longitudinal User Context:** Session-aware storage for profile, reminders, vitals, nutrition, and mental-health entries supports continuity and more personalized assistance over time.

[5]. **Operational Visibility and Extensibility:** Structured analytics and modular service boundaries enable easier monitoring, tuning, and incremental feature expansion without redesigning the full stack.

VIII. CONCLUSION AND FUTURE WORK

This project demonstrates a practical AI-assisted healthcare architecture that integrates workflow orchestration, specialized domain services, retrieval mechanisms, and safety controls into a unified user-facing platform. By combining multimodal input handling with session-aware context and modular health capabilities, the system delivers broader utility than a single-purpose chatbot. Its design supports

scalable evolution, operational traceability, and safer response handling for educational and assistive healthcare contexts.

Future work can improve the system in several directions. First, broaden benchmark-driven validation with standardized test suites for safety, factual grounding, and multilingual response quality. Second, strengthen explainability by exposing concise evidence traces and confidence cues for key decisions. Third, extend adaptive learning pipelines with robust drift monitoring and secure model update strategies. Fourth, improve privacy and governance through stronger authentication, policy segmentation, and data lifecycle controls. Fifth, integrate enterprise observability and incident workflows for production-grade reliability and compliance readiness.

REFERENCES

[1]. *Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.*

[2]. *Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schutze. Ret-llm: Towards ¨ a general read-write memory for large language models. arXiv preprint arXiv:2305.14322, 2023.*

[3]. *Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485– 5551, 2020.*

[4]. *Daniel E. Rose; Danny Levinson; "Understanding User Goals in Web Search", WWW, 2004.Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.*

[5]. *Eugene Agichtein; Eric Brill; Susan Dumais; "Improving Web Search Ranking by Incorporating User Behavior Information", SIGIR, 2006.*

[6]. *Fabio Petroni Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? arXiv preprint arXiv:1909.01066, 2019.*

[7]. *Gargari, O. K.; Menon, A.; "Enhancing Medical AI With Retrieval-Augmented Generation: A Narrative Review," npj Digital Medicine, 2025.*

[8]. *Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Wang, H.; "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv preprint arXiv:2312.10997, 2023*

[9]. *He, Xia; Peng, Yunchao; "Fine-grained image classification via combining vision and language", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017*

[10]. *Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.*

[11]. *Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. Ares: An automated evaluation framework for retrieval-augmented generation systems. arXiv preprint arXiv:2311.09476, 2023.*

[12]. *Krishnendu Rarhi; Abhisek Bhattacharya; Abhishek Mishra; Krishnasis Mandal; "Automated Medical Chatbot", 2017.*

[13]. *Lekha Athota; Vinod Kumar Shukla; Nitin Pandey; Ajay Rana; "Chatbot for Healthcare System Using Artificial Intelligence", 2020 8TH INTERNATIONAL CONFERENCE ON RELIABILITY, INFOCOM, 2020.*

[14]. *Noor Nashid, Mifta Sintaha, and Ali Mesbah. Retrieval-based prompt selection for code-related few-shot learning. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pages 2450– 2462, 2023.*

[15]. *Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. arXiv preprint arXiv:2112.09332, 2021.*