HemaChat: Lessons Learned from Developing a Multi-Agent Healthcare Chatbot for Sickle Cell Disease Care

David Eduardo Pereira
Systems and Computing, Federal University of Campina Grande
Campina Grande, Brazil
david.pereira@ccc.ufcg.edu.br

Herman Martins Gomes
Systems and Computing, Federal University of Campina Grande
Campina Grande, Brazil

Claudio E. C. Campelo
Systems and Computing, Federal University of Campina Grande
Campina Grande, Brazil

Eliane Cristina Araújo
Systems and Computing, Federal University of Campina Grande
Campina Grande, Brazil

Abstract

Software development and applications have changed significantly with the widespread adoption of Artificial Intelligence (AI), particularly Large Language Models (LLMs). Conversational chatbots now enable intuitive interaction through natural language. This industrial experience report presents the development of a healthcare platform centered on an AI-powered chatbot, named Hemachat, built on an LLM-based multi-agent system. This solution aims to support both healthcare professionals and patients with Sickle Cell Disease (SCD) by assisting with medical follow-up, tracking laboratory examinations and consultation schedules, answering general questions about the disease, and supporting patient triage. Furthermore, it generates treatment plans that can assist physicians in the clinical decision-making process. This report focuses on the development steps, the software architecture, and the major software engineering decisions that shaped the system. It further discusses key challenges, including privacy and security, testing AI-powered conversational systems, and ensuring reliable and clinically safe behaviour in a safety-critical healthcare environment. While reporting the development experience and identifying open challenges, we aim to provide practical insights to support developers of multi-agent LLM-based healthcare applications.

Keywords

Large Language Model, Multi-Agent System, Sickle Cell Disease

1 Introduction

Software development in the AI era has changed over the past few years, particularly following the release of ChatGPT [7]. Developing software with the assistance of Large Language Models (LLMs) has become a reality. Research indicates a growing proportion of developers using AI-assisted tools, and software companies encourage AI-related skills as part of a developer's professional profile. Furthermore, important software engineering challenges arise when applications rely on LLMs, models which inherently exhibit stochastic behavior. "How can developers effectively test and trust software that exhibits non-deterministic outputs?" This has become one of the central questions of modern software engineering.

Chatbots are among the software applications that have gained popularity in recent years, as they facilitate human-machine communication through natural language [8]. Moreover, they play an important role in democratizing access to advanced solutions since they require little technical knowledge from users. Some chatbot systems even incorporate multi-modal interactions, further improving accessibility and usability by allowing users to input different media types, such as audio and images.

Following this trend, this paper presents an experience report on the development of software composed of multiple components, including an AI-based chatbot, integration with messaging services, and a web dashboard. Additionally, since our Hemachat operates within the healthcare domain, the development process had to address additional challenges related to clinical security, testing, and data privacy.

Our platform was designed to support individuals affected by Sickle Cell Disease (SCD), a rare genetic disorder that, according to the World Health Organization, affected approximately 7.74 million people worldwide in 2021. Due to its genetic inheritance pattern, SCD disproportionately affects individuals of African descent. Moreover, studies show that SCD is often underrepresented in medical education and is insufficiently understood by healthcare professionals, resulting in inadequate management in clinical practice. This lack of familiarity can be further explained by broader cultural and historical factors, including social structural inequalities [5]. This issue also extends to technological development, since SCD is underrepresented in healthcare evaluation datasets [4]. Consequently, AI models may demonstrate limited performance when addressing these conditions.

In this paper, we describe the main software engineering challenges encountered throughout the development of our solution. By documenting these experiences, we aim to help future developers overcome similar challenges more effectively and contribute to the emerging body of knowledge on software engineering strategies for developing LLM-based Multi-Agent Systems (MAS).

2 Problem Context and Requirements

During a hackathon, a physician specialized in SCD highlighted the challenges of providing effective care for patients, emphasizing that although the disease is manageable, maintaining regular follow-up through laboratory examinations and medical appointments is essential for preventing serious complications. Moreover, physicians often manage large numbers of patients, making it difficult to monitor individual follow-up schedules. In addition, patients and

caregivers may underestimate the importance of regular examinations, which can compromise patients' health.

To address these challenges, we implemented Hemachat an AI-powered chatbot for patient communication. That was created to assist patients and caregivers in managing their medical follow-up by tracking, through natural language, laboratory examinations and consultation schedules. Users can monitor the status of appointments, identifying whether they are overdue or not. Additionally, the system automatically sends reminders via a messaging service (i.e., WhatsApp), facilitating engagement with patients and caregivers. Furthermore, we implemented a web-based dashboard that enables physicians to monitor the clinical follow-up status of their patients.

As mentioned earlier, many healthcare professionals lack sufficient knowledge to manage patients with SCD. To handle this challenge, our solution includes an AI-assisted triage process. Patients can describe their symptoms and the chatbot conducts a structured clinical interview to collect symptoms data. Based on the collected data, the system identifies the nearest emergency healthcare service and provides initial guidance. In addition, it generates a medical care report that contains a suggested differential diagnosis, laboratory examinations, and clinical management recommendations. This report is intended to support, rather than replace, physicians' clinical decision-making.

3 Architecture and Implementation Details

This section describes the software architecture, which consists of four main components: a chatbot, a backend, a dashboard (frontend), and a WhatsApp gateway.

Chatbot Multi-Agent System: The system is coordinated by a *Supervisor Agent*, which routes each user request to the most appropriate specialized agent. In addition to task routing, the Supervisor Agent acts as a guardrail by identifying and rejecting out-of-scope requests, helping prevent misuse of the system and ensuring that conversations remain within the chatbot's intended clinical domain.

Furthermore, the *General Questions Agent* answers common questions about SCD (e.g., "How is the disease diagnosed?"). The *Follow-up Agent* retrieves and presents information related to the patient's medical history, including previous or scheduled examinations or procedures (e.g., "When is my next Doppler test?"). The *Greeting Agent* is responsible for handling social interactions, such as responding to greetings and casual conversational exchanges.

Finally, the *Triage Agent* conducts the patient triage process. During this interaction, patients describe their symptoms while Hemachat performs an anamnesis by asking follow-up questions to collect relevant clinical information. Once the anamnesis is completed, the collected information is forwarded to two specialized agents that conclude the triage workflow. The *Location Agent* is responsible for identifying the nearest and most appropriate emergency healthcare service based on the patient's location. The *Medical Plan Generator Agent* generates a structured care plan that summarizes the reported symptoms, relevant clinical findings, and recommended next steps. This report is intended to support healthcare professionals by providing a concise summary of the patient's condition and serving as a clinical decision-support resource.

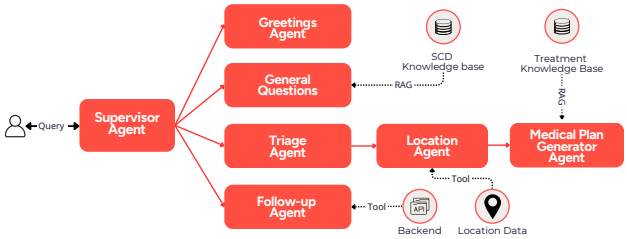

Figure 1: Chatbot MAS architecture.

The chatbot was built using LangGraph[1], a Python framework designed for developing multi-agent systems with explicit agent orchestration and workflow management. Furthermore, the architecture leverages Retrieval-Augmented Generation (RAG) to ground the agents in specialized clinical knowledge, reducing hallucinations and improving the safety of the generated responses [1].

In addition to RAG, the agents can invoke external tools to accomplish specific tasks. For example, the Follow-up Agent has access to an HTTP tool that communicates with the backend to retrieve patient's clinical history data. Figure 1 illustrates the proposed multi-agent architecture responsible for supporting Hemachat primary use cases, as well as their corresponding RAG knowledge sources and external tools.

Backend: The backend serves as the central data management and orchestration layer, responsible for managing patient information, healthcare professionals' data, medical exams, follow-up schedules, and conversation history. It provides RESTful API endpoints consumed by both the medical dashboard and chatbot, enabling personalized interactions through agent tool calls. Implemented in Python using the FastAPI[2] framework, the backend uses token-based authentication for secure access and MongoDB as the primary database due to its flexibility in handling heterogeneous clinical and conversational data.

Dashboard (Frontend): The dashboard is designed for healthcare professionals to manage patients and monitor clinical follow-up. It enables clinicians to register examinations and procedures, configure follow-up schedules, track overdue appointments, send notifications, and review patient-chatbot interactions. The interface consists of three main modules: a patient list with clinical information, detailed patient records including medical history and conversations, and system configuration for defining examination and procedure frequencies. The frontend was developed using React, Next.js, and Material UI.

WhatsApp Gateway: The WhatsApp Gateway is a Python FastAPI service that receives text or audio messages via webhooks, forwards them with user data to the chatbot service, and returns responses through the WhatsApp API. This service improves maintainability and supports future integration with other communication channels, such as email and Telegram.

4 Development Challenges and Lessons Learned

This section highlights the main technical and practical challenges encountered during the development of the solution.

[1] https://www.langchain.com/langgraph
[2] https://fastapi.tiangolo.com/

The Continuous Evolution of Multi-Agent Frameworks: One of the main challenges during development was selecting a suitable framework for implementing the MAS architecture. LangGraph and LangChain were adopted due to their support for multi-agent workflow orchestration and features essential for chatbot development, such as tool calling, human-in-the-loop interactions, memory management, and workflow control. However, the adoption of these emerging frameworks introduced maintenance challenges due to frequent updates and breaking changes. This experience reflects a common trade-off in adopting cutting-edge technologies: while they provide advanced capabilities, they may lack the stability and maturity of more established frameworks, particularly in rapidly evolving areas such as LLM-based multi-agent systems.

Privacy in Multi-Agent Systems: Because our solution operates in the healthcare domain, patient privacy has been a fundamental concern throughout the project. Ensuring privacy in LLM-based multi-agent systems is particularly challenging due to unrestricted natural language inputs, complex information flows between autonomous agents, and the use of cloud-based LLM services that may require sensitive data to leave local infrastructure.

To address these concerns, an anonymization pipeline was implemented that removes personally identifiable information (PII). Incoming messages are processed by a lightweight Named Entity Recognition (NER) model running locally on CPU, where detected entities are replaced with semantic placeholders (e.g., `<NAME>` and `<ADDRESS>`). This approach preserves the meaning of user requests while reducing exposure of sensitive information. The model[3] is developed based on Portuguese language resources and aligned with the Brazilian General Data Protection Law.

Despite these efforts, audio anonymization remains an open challenge. Speech must first be transcribed before anonymization, but high-quality Automatic Speech Recognition (ASR) models are often provided through cloud services, creating privacy concerns. Locally deployed ASR models provide stronger privacy but require computational resources. Therefore, healthcare AI systems must balance privacy, computational cost, and transcription quality, and this is still an important direction for future improvements.

Clinical Safety: Although LLMs demonstrate remarkable capabilities across a wide range of tasks, their stochastic nature makes them unsuitable for many safety-critical applications without additional safeguards. This is particularly true in the healthcare domain, where hallucinations—plausible but incorrect or fabricated information generated by an LLM—may directly affect patient safety. Among the approaches proposed in the literature, RAG has emerged as one of the most promising techniques for reducing hallucinations and improving the factual quality of LLM-generated responses.

While RAG improves response grounding, the development team decided not to rely on it as the sole mechanism for clinical safety. Instead, a second validation was made through a human evaluation. A total of 22 medical physicians evaluated the medical care plans generated by the proposed MAS. The study compared several retrieval strategies, including LLM-Guided Tree Retrieval, Metadata-Filtered Retrieval, Semantic Similarity Retrieval, and a baseline without RAG [3].

The evaluation showed no statistically significant differences in the overall quality of generated medical plans across the evaluated retrieval strategies, including the baseline. However, RAG-based approaches, particularly the **Tree-based strategy**, consistently produced safer recommendations according to human clinical safety metrics. Additionally, LLM-as-a-Judge evaluations did not always align with healthcare professionals' assessments, as LLM-based evaluators tended to overestimate the safety of generated care plans. These findings indicate that, although RAG is an effective strategy for reducing hallucinations and improving clinical safety, it should be considered part of a broader safety framework rather than a silver bullet: human expert evaluation remains essential.

Guardrails — Preventing Misuse Through a Hybrid Guardrail Architecture: Another important lesson concerned preventing chatbot misuse. Although HemaChat supports patients with SCD, it is publicly accessible to promote reliable disease information and must reject requests outside its scope. Initially, the Supervisor Agent assessed every request, but this required LLM processing and risked unnecessary computational costs from repeated irrelevant queries. We therefore introduced a two-stage guardrail: a lightweight local NLP model filters most out-of-scope inputs at no LLM cost, while ambiguous cases are forwarded to the Supervisor Agent for context-aware assessment, combining local-model efficiency with LLM reasoning.

5 Chatbot Testing Challenges

Testing conversational AI systems remains a software engineering challenge due to the unstructured nature of language and the stochastic behavior of LLMs. This section presents the lessons learned and remaining challenges.

Observability: Validating LLM-based conversational systems remains a challenging task, particularly when they are composed of multiple interacting agents. Unlike traditional software, where execution paths are generally deterministic and easier to inspect, MAS exhibit dynamic behaviors that emerge from agent coordination, prompt engineering, tool invocation, external knowledge retrieval, and the stochastic nature of LLMs. To support the development of Hemachat, we adopted Phoenix[4] as the primary observability platform. Phoenix provides detailed execution traces, including agent interactions, prompts, tool invocations, retrieved context, execution latency, and token consumption, allowing developers to inspect the complete execution flow, identify failure points, compare system versions, and perform cost-aware evaluations throughout the development life cycle.

Despite the benefits of observability platforms, effective validation still depends on representative testing datasets. Constructing such datasets is particularly difficult in healthcare, where developer-created test cases often fail to capture real patient behavior, while recruiting patients for evaluation introduces ethical, regulatory, and logistical constraints. Consequently, obtaining realistic patient interactions remains an open research challenge.

AI-Based Patient Simulation for Chatbot Testing: Hence, AI-based user simulation has recently emerged as a promising approach for testing conversational agents in a cost-effective, scalable,

[3]https://huggingface.co/urchade/gliner_multi_pii-v

[4]https://arize.com/phoenix/

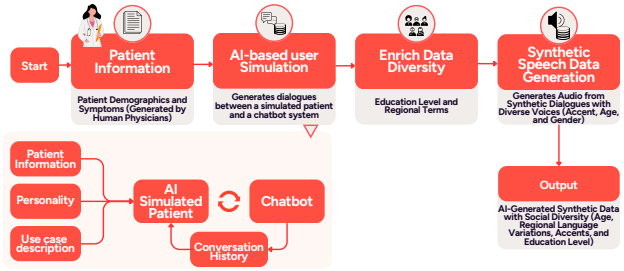

Figure 2: Pipeline for Synthetic Dialogue Generation for Chatbot Testing.

and safe manner. Although recent studies have explored this direction, the area is still in its early stages. Existing work has primarily focused on demonstrating the feasibility of AI-based simulation, while rigorous methodologies and evaluation remain limited.

Inspired by these studies, we have been investigating the use of AI-generated patient simulations to evaluate healthcare chatbots. In [6], the authors proposed generating conversations between AI agents acting as proxies for human users. Their work introduced multiple user personas, including cooperative, uncooperative, sarcastic, and hesitant users, demonstrating how personality diversity can expose different categories of chatbot failures. Another important contribution of this work was the definition of a taxonomy of chatbot response errors, providing a structured framework for analyzing chatbot behavior during testing.

Building upon these ideas, we are developing a framework for simulating diverse patient personas with different demographic characteristics, communication styles, and clinical backgrounds. The approach begins with patient profiles created by physicians specialized in SCD, describing relevant clinical conditions and demographic information. These **profiles**, together with the **chatbot use-case description** and the desired **persona**, are provided as input to an LLM through prompt engineering to generate realistic patient queries, which are automatically submitted to our solution. After each chatbot response, the conversation history is incorporated into the next prompt, enabling coherent multi-turn interactions while reducing repetitive or isolated queries and producing conversations that more closely resemble real patient interactions. The overall testing workflow is illustrated in Figure 2.

Although this research is still ongoing, initial experiments demonstrated the practical value of the proposed approach by revealing weaknesses in our chatbot that were not identified through conventional developer-driven testing. These observations suggest that AI-based patient simulation can uncover realistic interaction scenarios that are difficult to anticipate manually. However, an important open question remains regarding how to rigorously validate the effectiveness of simulated users, as limited evidence exists comparing AI-generated interactions with real patient conversations. Therefore, while AI-based simulation represents a promising and cost-effective alternative when access to patients is limited, it should currently be considered a complementary strategy rather than a replacement for evaluations with real users.

In conclusion, we believe that establishing standardized methodologies and evaluation protocols for AI-based patient simulation represents an important direction for future research in the testing of conversational agents, specially for health care scenarios.

Prompt Injection and Prompt Hacking: The robustness of our solution against prompt-based threats was evaluated through testing inspired by the *AI Agents Traps framework* [2], covering summary-based prompt injection, critic evasion, cross-session information leakage, biased phrasing, and jailbreaking. The results demonstrated robust behavior across most categories, with the exception of summary-based prompt injection, where malicious instructions were occasionally included in the conversation summary shared among agents. However, no evidence was found that these instructions affected the behavior or decisions of downstream agents, indicating that while the system remained robust, the summarization component could be further strengthened.

6 Conclusion

This report presented our experience building a MAS-based chatbot designed to support the clinical care of SCD patients. As an ongoing project, the proposed solution continues to evolve while addressing practical challenges associated with developing healthcare conversational systems, including the rapid evolution of multi-agent frameworks, privacy preservation (audio and text), clinical safety, and the validation of AI-based applications without direct access to representative patient populations. Our experience highlights several lessons that can benefit both researchers and practitioners when developing systems for critical environments in the current landscape of AI-based applications.

Artifact Availability

No artifacts are provided.

Acknowledgements

This research was supported by the Agents4Good Project, a collaboration between Kunumi[5] and Embrapii, through the Center for Electrical Engineering and Informatics (CEEI) at UFCG[6].

References

[1] Amugongo et al. 2025. Retrieval augmented generation for large language models in healthcare: A systematic review. *PLOS Digital Health* 4 (06 2025), 1–33.
[2] Franklin et al. 2026. AI Agent Traps. *SSRN Electronic Journal* (8 March 2026).
[3] Leite et al. 2026. LLM-Based Multi-Agent System with Retrieval-Augmented Generation for Medical Care Planning Generation in Sickle Cell Disease. In *Proceedings of the 17th International Conference on Computational Processing of Portuguese (PROPOR 2026) - Vol. 2.* Salvador, Brazil.
[4] Mutisya et al. 2025. Mind the Gap: Evaluating the Representativeness of Quantitative Medical Language Reasoning LLM Benchmarks for African Disease Burdens.
[5] Reich et al. 2022. An Integrative Review: The Evolution of Provider Knowledge, Attitudes, Perceptions and Perceived Barriers to Caring for Patients with Sickle Cell Disease 1970–Now. *Journal of Pediatric Hematology/Oncology Nursing* 40, 1 (July 2022), 43–64.
[6] Roman et al. 2025. ChatChecker: A Framework for Dialogue System Testing and Evaluation Through Non-cooperative User Simulation. arXiv:2507.16792 [cs.AI]
[7] Md. Asraful Haque. 2025. LLMs: A game-changer for software engineers? *BenchCouncil Transactions on Benchmarks, Standards and Evaluations* 5, 1 (2025), 100204.
[8] Li Liu and Vincent G. Duffy. 2023. Exploring the Future Development of Artificial Intelligence (AI) Applications in Chatbots: A Bibliometric Analysis. *International Journal of Social Robotics* 15, 5 (2023), 703–716.

[5] https://www.kunumi.com/br
[6] https://embrapii.org.br/unidades/software-e-automaco-ceei/