

MediCare-Agent: A Multi-Agent AI System for Post-Discharge Nephrology Patient Care Using LangGraph, RAG, and Large Language Models

Prince¹, Ritesh Kumar²

¹Scholar of B.Tech 4th Year (AIDS), Department of Artificial Intelligence & Data Science, Dr. Akhilesh Das Gupta Institute of Professional Studies, Delhi, India.

²Ritesh Kumar, Department of Artificial Intelligence & Data Science, Dr. Akhilesh Das Gupta Institute of Professional Studies, Delhi, India.

¹princeg.pg39@gmail.com ²riteshchandel@gmail.com

ABSTRACT

Post-discharge care represents a critical gap in the healthcare journey of patients with chronic kidney conditions. Patients discharged from nephrology departments often struggle with unanswered clinical queries, complex medication regimens, and uncertainty about symptom severity — challenges that contribute significantly to preventable hospital readmissions. The proposed MediCare-Agent is an AI-driven, multi-agent system designed to provide 24/7 intelligent post-discharge support to nephrology patients. Built using a stateful LangGraph 0.2.28 workflow, the system routes patient queries through a four-node agent pipeline comprising a Router Node, Patient Identification Node, Medical Consultation Node, and General Response Node. Medical queries are answered using Retrieval-Augmented Generation (RAG) over a 1,547-page Comprehensive Clinical Nephrology textbook indexed via FAISS-cpu, with query results personalized using patient-specific context retrieved from a SQLite database. A locally hosted HuggingFace all-MiniLM-L6-v2 embedding model generates 384-dimensional dense vectors for semantic retrieval, while Meta's Llama 3.3 70B — served through Groq's free inference tier — powers all natural language generation and intent classification. A seven-entry nephrology drug reference database and DuckDuckGo real-time web search provide supplementary knowledge without any API key or subscription cost. Developed using FastAPI on the backend and Streamlit on the frontend, the system delivers structured, triage-formatted medical responses with end-to-end latency of approximately 3–4 seconds at zero monthly operational cost. This paper demonstrates how open-weight LLMs, domain-specific retrieval, and multi-agent orchestration can be combined into a deployable, patient-facing post-discharge care system.

Abbreviations

AI: Artificial Intelligence

API: Application Programming Interface

CKD: Chronic Kidney Disease

eGFR: Estimated Glomerular Filtration Rate

FAISS: Facebook AI Similarity Search

JWT: JSON Web Token

LLM: Large Language Model

NLP: Natural Language Processing

PDF: Portable Document Format

RAG: Retrieval-Augmented Generation

REST: Representational State Transfer

SQLite: Self-Contained Lightweight SQL Database Engine

UUID: Universally Unique Identifier

1. INTRODUCTION

Chronic kidney disease (CKD) affects an estimated 850 million people worldwide and is among the most prevalent chronic conditions driving preventable mortality and hospital readmissions globally [1]. When nephrology patients are discharged from hospital care, they are handed extensive post-discharge instructions covering dietary restrictions, fluid intake limits, multi-drug regimens with dose adjustments tied to estimated glomerular filtration rate (eGFR), and symptom monitoring guidelines. Despite this guidance, access to their treating nephrologist between appointments — often scheduled weeks apart — is severely limited, leaving patients vulnerable to medication errors, non-adherence, and avoidable clinical deterioration.

Traditional approaches to bridging this gap — such as nurse-led follow-up calls or written patient education materials — are resource-intensive and non-scalable. Meanwhile, general-purpose AI chatbots lack the domain-specific grounding needed to provide reliable nephrology guidance, and most commercial document-querying tools offer no mechanism for patient personalization or clinical triage classification.

The MediCare-Agent system addresses these limitations by introducing a centralized, intelligent post-discharge support platform built entirely on freely available, open-source infrastructure. By combining LangGraph-orchestrated multi-agent routing, FAISS-indexed Retrieval-Augmented Generation over a comprehensive nephrology textbook, Groq-hosted Llama 3.3 70B language generation, and SQLite-backed patient identification, the system delivers accurate, personalized, and clinically structured responses to patient queries — at zero recurring cost. Unlike standard search tools or general chatbots, MediCare-Agent applies a structured triage classification to every medical response, distinguishing queries requiring IMMEDIATE clinical attention from those suitable for ROUTINE follow-up.

The system further mitigates the limitations of isolated document-based analysis by incorporating DuckDuckGo real-time web search for queries outside the scope of the nephrology textbook, and a local nephrology drug reference database for rapid medication guidance without external API calls. Developed with a decoupled architecture using FastAPI on the backend and Streamlit on the frontend, the application provides a scalable, session-aware, and patient-friendly solution aimed at improving post-discharge outcomes for nephrology patients.

2. LITERATURE REVIEW

The evolution of AI-assisted healthcare communication has progressed through several distinct phases. Early clinical decision support systems (CDSS) were rule-based, relying on hand-crafted if-then logic trees for symptom triage. While deterministic, these systems failed to handle the variability of natural patient language and required extensive expert curation to maintain [2].

The introduction of transformer-based large language models fundamentally changed this landscape. BioBERT [3], pre-trained on PubMed abstracts and PubMed Central full-text articles, demonstrated that domain-adapted pre-training significantly outperforms general BERT on biomedical named entity recognition and question-answering benchmarks. Subsequently, Med-PaLM 2 [4] achieved expert-level performance on the United States Medical Licensing Examination (USMLE), establishing that sufficiently scaled LLMs can encode substantial clinical knowledge without explicit fine-tuning on clinical QA pairs. More recently, Meta's Llama family of open-weight models [5] has brought this generative capability to zero-cost deployments, removing the proprietary API dependency that previously restricted clinical AI to well-funded institutions.

Retrieval-Augmented Generation (RAG), introduced by Lewis et al. [6], addresses the hallucination problem inherent in purely generative LLMs by combining a retrieval index with a generative decoder. In medical contexts, RAG has been applied to drug-drug interaction querying, clinical note summarization, and patient FAQ systems [7]. FAISS

(Facebook AI Similarity Search) [8] provides the efficient approximate nearest-neighbour retrieval underlying most production RAG deployments. Despite these advances, existing medical chatbot systems largely remain either fully rule-based (lacking generative flexibility) or purely generative (lacking grounded, verifiable responses). Few systems integrate personalized patient context, domain-specific retrieval, and clinical triage into a unified multi-agent architecture — the precise gap MediCare-Agent fills.

In the domain of multi-agent AI systems, LangGraph [9] has emerged as the standard framework for building stateful agent workflows over language models, providing conditional routing, typed state management, and persistent memory — capabilities essential for multi-turn patient interactions requiring identification and contextual memory across conversation sessions.

2.1 Objectives

The primary objective of MediCare-Agent is to develop an AI-driven, multi-agent post-discharge support system specifically tailored to nephrology patients, achieving zero monthly operational cost without sacrificing response quality. The system aims to provide personalized, citation-grounded medical responses by combining a patient-context-aware FAISS RAG pipeline with structured clinical triage classification. A key objective is to ensure high reliability through a three-step hierarchical routing strategy that minimizes unnecessary LLM API calls. The system is further designed to provide seamless external knowledge via DuckDuckGo real-time web search and rapid medication guidance through a local nephrology drug database. All of this is achieved within a secure, session-persistent architecture using FastAPI backend session management, LangGraph MemorySaver state persistence, and UUID-keyed Streamlit frontend sessions.

3. METHODOLOGY

The architecture of MediCare-Agent follows a modern, decoupled three-tier model to ensure scalability, clinical accuracy, and session-aware personalization. The system is divided into three primary layers: the presentation layer, the application layer, and the AI and data layer.

The presentation layer is developed using Streamlit 1.41.1, chosen for its rapid development capabilities and built-in session state management. This layer provides a web-based chat interface where patients enter their name for identification and submit medical or general queries. A UUID session identifier, generated via Python's `uuid.uuid4()` on first load and persisted in Streamlit's `session_state`, uniquely identifies each patient session. Custom CSS (dark theme: background #0F172A, accent border #00FFB2) provides a visually distinct medical interface. All communication with the backend is handled via HTTP POST requests to the FastAPI `/chat` endpoint, carrying the user message and session UUID as JSON payload.

The application layer is implemented using FastAPI 0.115.0, served by Uvicorn 0.30.6. The backend exposes four REST endpoints: POST `/chat` (primary interaction), GET `/health` (liveness probe), GET `/patients` (patient enumeration for testing), and DELETE `/session/{session_id}` (session reset). An in-memory Python dictionary maps each UUID session identifier to the currently identified patient record, ensuring patient context persists across conversation turns. Request and response schemas are validated by Pydantic 2.9.2. The backend instantiates the LangGraph workflow at startup and invokes it per request, passing the user message and current session patient context.

The AI and data layer integrates the LangGraph 0.2.28 multi-agent workflow, a FAISS-cpu 1.7.4 vector store over the nephrology corpus, a SQLite patient database, a local drug reference database, and DuckDuckGo real-time search — all operating at zero cost. Patient records and chat interaction data are stored in SQLite. The LangGraph MemorySaver

checkpoint, keyed by session UUID as `thread_id`, provides automatic multi-turn state persistence for the identified patient record without repeated database queries.

3.1 Data Collection and Preprocessing

The primary medical knowledge source is the Comprehensive Clinical Nephrology textbook (1,547 pages), covering CKD pathophysiology, diabetic nephropathy, hypertensive nephropathy, renal replacement therapy, drug dosing in renal impairment, and post-discharge care protocols. The PDF is ingested using PyPDF 4.3.1 via LangChain's PyPDFLoader. Documents are segmented using RecursiveCharacterTextSplitter with `chunk_size=1,000` characters, `chunk_overlap=200`, and a hierarchical separator list that preserves paragraph and sentence boundaries, yielding approximately 12,000 semantic chunks.

Each chunk is encoded into a 384-dimensional normalized dense vector using the HuggingFace all-MiniLM-L6-v2 sentence embedding model running locally on CPU (`encode_kwargs: normalize_embeddings=True`). The FAISS index is built once via `setup_rag.py`, saved to disk, and loaded in subsequent runs via `FAISS.load_local()` — reducing cold-start time from 5–10 minutes to under one second. Patient records are stored in a SQLite schema with fields: `id`, `name`, `diagnosis`, `medications`, `discharge_date`, `doctor`, and `notes`. Name lookup uses a `LOWER()`-normalized SQL LIKE query for case-insensitive partial-match search. The local drug database covers seven nephrology medications: lisinopril, furosemide, metformin, amlodipine, telmisartan, atenolol, and enalapril — each with drug class, indication, eGFR considerations, and interaction data.

3.2 Evaluation and Testing

System evaluation focused on routing accuracy, retrieval relevance, response clinical structure, and end-to-end latency. Patient identification was tested with name variations — partial names, case variations, and common misspellings — to validate the `LOWER()` LIKE SQL approach. RAG retrieval was evaluated by posing specific nephrology questions and verifying that retrieved chunks contained clinically relevant page-sourced content. The three-step router was tested across all routing categories: short-message patient name entries (Step 1 heuristic), temporal and historical web-search queries (Step 2 keyword detection), and mixed medical and general intent queries (Step 3 LLM classification).

The triage template was validated by posing urgent symptom queries (e.g., severe chest pain, sudden vision loss) and confirming IMMEDIATE classification with emergency guidance, alongside routine queries (e.g., mild fatigue, dietary questions) yielding ROUTINE classification. Web search fallback was evaluated by posing queries about external information not present in the nephrology textbook (e.g., macroeconomic health policy, recent research trials) and confirming DuckDuckGo integration returned properly sourced responses. End-to-end latency was measured across 50 test interactions per response category on a standard development workstation (Python 3.11, 16 GB RAM, CPU inference).

3.3 Implementation Workflow

The operational workflow of MediCare-Agent proceeds as follows:

- Session Initialization:** The Streamlit frontend generates a UUID session identifier and presents the chat interface. The patient is prompted to enter their name or ask a general medical question.
- Patient Identification:** The user's first short message is routed by the Router Node (Step 1 heuristic: token count ≤ 3) to the Patient Identification Node, which performs a `LOWER()` LIKE SQLite query. On match, the patient record is stored in LangGraph MemorySaver under the session UUID and a structured welcome message is returned.

1. **Query Routing:** Each subsequent message passes through the three-step Router: (i) short-message heuristic for re-identification; (ii) `needs_web_search()` keyword detection for web routing; (iii) Groq Llama 3.3 70B classification call (`temperature=0`, `max_tokens=10`) for MEDICAL vs. GENERAL routing.
2. **Medical Consultation (RAG Pipeline):** For MEDICAL queries, the Medical Node constructs a patient-context-enhanced query string (patient name + diagnosis + medications + raw query), performs FAISS similarity search ($k=4$) over the nephrology corpus, checks the local drug database, assembles a structured clinical prompt, and invokes Llama 3.3 70B to generate a five-section triage response.
3. **General Response or Web Search:** For GENERAL queries, the General Node either executes DuckDuckGo search (`max_results=3`) with formatted source citations, or invokes Llama 3.3 70B directly with an empathetic conversational prompt for non-informational inputs.
4. **Response Delivery and State Persistence:** The FastAPI backend returns the generated response and `patient_found` flag to the Streamlit frontend. LangGraph MemorySaver automatically persists the full session state, ensuring the identified patient record is available across all subsequent turns.

4. RESULTS AND DISCUSSION

The MediCare-Agent system was successfully implemented as a fully functional, zero-cost post-discharge nephrology care prototype. Testing demonstrated significant improvement in information accessibility for simulated patient interaction scenarios across all four registered patient profiles — Rahul Sharma (Diabetic Nephropathy), Priya Singh (CKD Stage 3), John Clark (Diabetic Nephropathy), and Amit Patel (Hypertensive Nephropathy).

Patient identification was highly reliable across all tested name variations, with SQLite's `LOWER()` `LIKE` partial-match lookup correctly identifying patients from first-name-only entries, mixed-case inputs, and partial surname matches at approximately 50 ms latency. The three-step routing strategy demonstrated high routing accuracy: the short-message heuristic correctly intercepted all patient name entries without requiring an LLM call, and the keyword-based web detection reliably identified temporal and historical queries before reaching the classification step — minimizing Groq API usage and reducing average routing latency to well under one second for heuristic-resolvable queries.

The FAISS RAG pipeline delivered clinically relevant responses consistently. Patient-context query enhancement — injecting diagnosis and medication data into the retrieval query — produced perceptibly more relevant chunk retrieval compared to raw query-based retrieval, particularly for medication-specific questions where the patient's current drug list provided crucial context for FAISS similarity scoring. End-to-end medical response latency averaged 3–4 seconds, dominated by the Groq LLM generation step (1–2 seconds) and FAISS retrieval (100 ms). The local drug database intercepted all seven covered medication queries sub-100 ms, confirming the value of the local-first drug lookup strategy.

The mandatory five-section triage template proved highly effective in structuring outputs. IMMEDIATE triage triggers (severe chest pain, sudden vision loss, extreme breathlessness) were correctly classified with explicit emergency service guidance. ROUTINE triage was correctly applied to dietary inquiries, mild symptom queries, and medication schedule questions. The DuckDuckGo web search fallback functioned correctly for all tested web-keyword queries, returning properly formatted responses with numbered sources and hyperlinked references. The Streamlit frontend maintained a smooth chat experience, with the patient banner updating correctly on identification and the "New Patient" session reset functioning cleanly across all tested scenarios.

5. CONCLUSION AND FUTURE WORK

The MediCare-Agent system demonstrates the transformative potential of multi-agent AI in the domain of post-discharge nephrology patient care. By leveraging LangGraph, FAISS-indexed RAG, Groq-hosted Llama 3.3 70B, SQLite patient management, and DuckDuckGo real-time search, the project achieved its goal of building a clinically

structured, personalized, and fully zero-cost post-discharge support system. Features such as patient-context query enhancement, structured triage classification, mandatory medical disclaimers, and a three-step hierarchical routing strategy effectively address the limitations of general-purpose chatbots in clinical patient-facing applications. The system delivers GPT-4-class response quality at 3–4 seconds end-to-end latency with no recurring infrastructure cost — a property critical for adoption in resource-constrained healthcare settings.

Future enhancements will focus on integration with HL7 FHIR-compliant Electronic Health Record (EHR) systems for real-time patient record access, replacing the current static SQLite database. A formal randomized controlled clinical trial with real nephrology patients will be conducted to obtain quantitative accuracy, patient satisfaction, and health outcome metrics. Multi-source RAG augmentation incorporating PubMed abstracts, KDIGO clinical guidelines, and FDA drug label data will broaden knowledge coverage. Streaming token responses will reduce perceived latency. Voice interface support will improve accessibility for elderly patients with limited digital literacy. A longitudinal analytics dashboard tracking patient symptom trends and medication adherence will provide clinicians with population-level post-discharge insights.

6. REFERENCES

- [1] GBD Chronic Kidney Disease Collaboration, "Global, regional, and national burden of chronic kidney disease, 1990–2017: A systematic analysis for the Global Burden of Disease Study 2017," *The Lancet*, vol. 395, no. 10225, pp. 709–733, Feb. 2020.
- [2] Shortliffe, E. H. and Cimino, J. J. (2014). *Biomedical Informatics: Computer Applications in Health Care and Biomedicine*, 4th Edition, Springer.
- [3] Lee, J. et al. (2020). BioBERT: A Pre-trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240.
- [4] Singhal, K. et al. (2023). Large Language Models Encode Clinical Knowledge. *Nature*, vol. 620, pp. 172–180.
- [5] Touvron, H. et al. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.
- [6] Lewis, P. et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474.
- [7] Zakka, C. et al. (2024). Almanac — Retrieval-Augmented Language Models for Clinical Medicine. *NEJM AI*, vol. 1, no. 2.
- [8] Johnson, J., Douze, M., and Jégou, H. (2021). Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547.
- [9] LangChain Inc. (2024). LangGraph: Building Stateful, Multi-Actor Applications with LLMs. Documentation v0.2.28. Available: <https://langchain-ai.github.io/langgraph>
- [10] Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. *Proceedings of EMNLP*, pp. 3982–3992.
- [11] Groq Inc. (2024). Groq LPU Inference Engine — Technical Overview. Available: <https://groq.com>
- [12] Coleman, E. A. et al. (2006). The Care Transitions Intervention. *Archives of Internal Medicine*, vol. 166, no. 17, pp. 1822–1828.