Brief Report

An Agentic AI Architecture for General Practitioners in Primary Care

Valentina Carbonari , Pierangelo Veltri , Pietro Hiram Guzzi *

Posted Date: 3 February 2026

doi: 10.20944/preprints202602.0164.v1

Keywords: LLM; Agentic AI; clinical support systems

 doi:10.20944/preprints202602.0164.v1

Brief Report

An Agentic AI Architecture for General Practitioners in Primary Care

Valentina Carbonari [1], Pierangelo Velti [2] and Pietro Hiram Guzzi [1,*]

[1] Department of Surgical and Medical Sciences, University "Magna Græcia" of Catanzaro, Catanzaro, Italy
[2] DIMES, University of Calabria, Rende, Italy
* Correspondence: hguzzi@unicz.it

Abstract

Primary care physicians face increasing challenges in managing multimorbidity, continuous home-monitoring data, and fragmented access to specialty input. We propose an agentic artificial intelligence (AI) architecture for Medici di Medicina Generale (MMG) that combines a central planning large language model (LLM) with a bank of small, domain-specialized language models (SLMs) and deterministic tools. The system is designed for patient–home integration, safety triage, and transparent interoperability with regional infrastructures such as FSE 2.0 and ePrescription. Here, we describe the functional schema and workflow of this architecture.

Keywords: LLM; Agentic AI; clinical support systems

1. Introduction

Primary care delivery is progressively shifting from clinic-based encounters toward distributed, home-centered care, particularly for older, multi-morbid patients residing in rural or otherwise underserved contexts.[1–3] Within this setting, general practitioners (Medici di Medicina Generale; MMG) retain primary responsibility for longitudinal care coordination but operate under severe constraints in terms of time, data integration, and consistent access to specialist expertise. In parallel, consumer-grade sensors and home medical devices generate continuous, high-dimensional streams of physiological and behavioral data that are only sporadically integrated into clinical decision-making and rarely transformed into operational recommendations.[4] Existing clinical decision support systems remain predominantly rule-based, exhibit poor robustness in the presence of multimorbidity, and are weakly coupled to the real-world generalist workflow. Consequently, there is a critical requirement for decision support that is simultaneously **broad**—able to span the heterogeneous case-mix of primary care—and **trustworthy**—transparent, auditable, and aligned with local care pathways (PDTA) and national guideline ecosystems.[5–8]

To address this gap, we introduce an **agentic AI architecture** for MMG-led care that integrates a **central planning LLM** with a **bank of small, discipline-specific language models (SLMs)** and a suite of deterministic clinical tools. End users—patients at home and MMG/community teams—interact with the system via lightweight interfaces that acquire symptoms, vital signs, medication adherence information, and consent metadata. All incoming signals are normalized to FHIR resources and processed by a **safety triage mini-LLM** combined with a deterministic rules engine for red-flag detection (e.g., chest pain, suspected diabetic ketoacidosis, acute focal neurological deficit), with immediate escalation when predefined safety thresholds are met.[9] For non-urgent cases, a **Planner/Router LLM** decomposes the clinical query into sub-tasks, instantiates an explicit call-graph, and dispatches each sub-task to the appropriate SLM expert (e.g., Diabetology, Cardiology, Respiratory, Mental Health), with each SLM adapted via lightweight parameter-efficient tuning and grounded through a Retrieval-Augmented Generation (RAG) layer over PDTA documents and guideline corpora. Deterministic calculators (eGFR, ASCVD risk, CHA_2DS_2-VASc, QTc, drug–drug interaction checking,

 doi:10.20944/preprints202602.0164.v1

renal/hepatic dose adjustment) are exposed as tools within this framework, providing numerically reliable and unit-consistent computations.

Each SLM produces a structured output comprising therapeutic and monitoring recommendations, explanatory rationale, literature citations with page-span handles, confidence estimates, and standardized clinical codes (e.g., SNOMED, ATC). A downstream **Aggregator/Critic LLM** reconciles the set of specialty outputs, resolves inconsistencies, calibrates confidence, and emits two distinct artifacts: a concise, actionable summary for MMG (including taskable alerts and titration instructions) and a lay-oriented explanation tailored to the patient. Final artifacts are exported to regional infrastructures (FSE 2.0/EHR, DEM/ePrescription, laboratory and imaging systems) via FHIR-based interfaces, while an immutable audit log records prompts, model and tool versions, execution traces, and key decisions to support transparency, post-hoc evaluation, and regulatory oversight.[10]

The architecture is explicitly optimized for safety, governance, and evaluability. Model cards, adapters, prompt templates, and guardrails are managed within a versioned registry, and run-time telemetry continuously tracks calibration metrics (e.g., expected calibration error, Brier score), citation coverage, and refusal/hallucination rates. The clear separation of concerns—planning (LLM), domain reasoning (SLMs), retrieval (RAG), and arithmetic/tooling (deterministic functions)—enables modular verification with specialty clinicians and supports controlled, stepwise deployment in operational primary care environments.[6,11]

2. Related Work

2.1. Decision Support in Primary Care and Multimorbidity

Early generations of clinical decision support systems (CDSS) for primary care were predominantly rule-based and embedded within electronic health record (EHR) platforms, focusing on alerts for drug–drug interactions, preventive care reminders, and guideline-based recommendations for single diseases.[5,6] While these systems achieved measurable improvements in adherence to individual guidelines, their performance degraded in the presence of multimorbidity, polypharmacy, and heterogeneous data sources typical of general practice. The underlying knowledge bases were often static, manually curated, and poorly aligned with evolving PDTA and national guidelines, leading to alert fatigue, limited clinician trust, and low sustained adoption in routine primary care workflows.[6]

A parallel line of work has investigated risk prediction and monitoring models based on structured EHR data, registries, and claims, yielding tools for cardiovascular risk estimation, hospitalization prediction, and chronic disease control metrics.[5] These models, frequently built using logistic regression or gradient boosting, are optimized for specific endpoints and populations, and typically assume relatively clean, curated input features. As such, they provide limited support for the unstructured, multi-source, and streaming data now emerging from home monitoring devices and patient-generated health data. Moreover, they do not directly address the orchestration of complex care plans across multiple conditions or the need for transparent, patient-facing explanations in general practice.

In the context of multimorbidity, recent work has emphasized the need for systems that can reconcile multiple disease-specific guidelines and reason over competing risks, treatment interactions, and patient preferences.[6] Proposed approaches include rule-based conflict resolution, ontological models of co-occurring conditions, and probabilistic frameworks for treatment trade-offs. However, these systems remain heavily expert-engineered, are difficult to maintain at scale, and rarely support continuous integration of new evidence or localized PDTA variants. None provide the kind of agentic, multi-step reasoning and tool orchestration that our architecture targets for MMG-led care in home-centered primary care settings.

2.2. Large Language Models in Clinical Decision Support

The advent of large language models (LLMs) has triggered a wave of work on generative decision support, clinical summarization, and patient communication.[1,2,4] Foundational models have demonstrated strong performance on medical question answering, guideline summarization, and

 doi:10.20944/preprints202602.0164.v1

synthetic vignette generation, including in primary care contexts. At the same time, they exhibit well-documented failure modes, such as hallucinations, overconfidence, and sensitivity to prompt phrasing, which are particularly problematic in safety-critical domains like clinical care.[5] As a result, current recommendations emphasize constrained use cases (e.g., draft generation, coding support) and human-in-the-loop oversight rather than autonomous decision-making.

Several studies have investigated retrieval-augmented generation (RAG) as a mechanism to ground LLM outputs in external knowledge sources such as PubMed, clinical guidelines, and institutional policies.[5,6] RAG-based systems have been shown to improve factuality and citation quality, and to facilitate rapid adaptation to new evidence without full model retraining. Nonetheless, most existing implementations treat RAG as a monolithic black box, without explicit modeling of multi-step clinical workflows, modular specialty reasoning, or integration with deterministic tools. They also seldom address the full pipeline from patient-generated data ingestion to interoperable outputs compatible with FHIR-based infrastructures like FSE 2.0 and DEM/ePrescription.

Recent work has begun to explore structured prompting, schema-constrained generation, and calibration techniques to enhance the reliability of LLM outputs in healthcare.[6] Approaches include forcing models to emit JSON conforming to predefined schemas, using chain-of-thought or tool-augmented reasoning, and post-hoc calibration via temperature scaling or conformal prediction. These contributions inform several design choices in our architecture, notably the use of schema contracts for all intermediate and final outputs, explicit confidence estimates, and telemetry over calibration metrics such as expected calibration error (ECE) and Brier score.

2.3. Agentic and Tool-Augmented Architectures

Beyond single-call LLM usage, a growing body of work has proposed *agentic* architectures in which LLMs act as planners that decompose complex tasks into sub-tasks, call external tools, and iteratively refine outputs.[10] In the clinical domain, this includes systems that orchestrate medical calculators, drug interaction checkers, and guideline retrieval engines under the control of a central LLM that plans and routes across tools. These approaches improve modularity and extensibility, allowing verified deterministic components to handle numerically sensitive or high-liability computations, while the LLM focuses on coordination, explanation, and integration of heterogeneous evidence.

Frameworks such as LangChain and related orchestration toolkits provide abstractions for building these multi-component pipelines, introducing standardized interfaces for models, retrievers, memory modules, and tools, as well as tracing and evaluation utilities.[10] The LangChain Expression Language (LCEL) further enables declarative specification of workflows as compositions of runnables, with built-in support for batching, streaming, and branching. Prior work has applied these frameworks to generic AI assistants, question-answering over proprietary corpora, and code-generation workflows, but only limited examples exist in regulated healthcare settings, and even fewer in the context of primary care and national health infrastructures.

Our work builds on this agentic paradigm but tailors it specifically to MMG-led primary care. In contrast to generic agent frameworks, the proposed architecture enforces a clear separation of concerns between planning (central LLM), domain reasoning (specialty SLMs), retrieval (RAG over PDTA and guidelines), and deterministic arithmetic tools. It also couples agentic workflows with explicit governance mechanisms, including versioned registries for model cards and prompts, runtime telemetry on safety and calibration metrics, and an immutable audit log for all model and tool interactions.[6,10] To our knowledge, this combination of agentic orchestration, specialty-specific language models, and governance-aware telemetry has not been systematically explored in primary care decision support.

2.4. Patient-Generated Data and Home Monitoring

A substantial literature addresses the integration of patient-generated health data (PGHD) and home monitoring streams into clinical workflows.[4] Studies in diabetes, heart failure, chronic obstructive pulmonary disease, and hypertension have demonstrated that remote monitoring can reduce

 doi:10.20944/preprints202602.0164.v1

hospitalizations and improve disease control when coupled with structured care pathways and responsive clinical teams. However, most operational systems rely on threshold-based alerts, simple dashboards, and manual triage, limiting scalability as device penetration and data volume increase.[5] These systems also frequently operate outside the core EHR or FHIR ecosystem, leading to fragmentation and duplicative documentation.

When PGHD is used for predictive modeling, it is often aggregated into summary features (e.g., daily means, variability indices) and fed into conventional machine learning models for risk prediction or intervention targeting.[4] Such models can be effective for specific endpoints but do not directly address the need for multi-step reasoning over complex, temporally rich data streams or for generating personalized, multi-condition care plans. Importantly, they typically do not expose intermediate reasoning steps, citations, or structured justifications, limiting their utility as transparent decision-support tools for MMG.

Our architecture differs from prior PGHD systems in two key respects. First, it treats home monitoring data as a first-class input to an agentic workflow that includes safety triage, specialty routing, and retrieval-grounded reasoning, rather than as a separate alert stream. Second, it mandates FHIR-based normalization and interoperability with regional infrastructures (FSE 2.0, DEM/ePrescription, labs/imaging), allowing recommendations derived from PGHD to be integrated seamlessly into existing clinical records and care pathways.[6] This design aims to reduce the integration burden on MMG while providing a principled substrate for evaluation, governance, and incremental deployment.

2.5. Positioning of This Work

In summary, existing CDSS for primary care primarily offer rule-based, single-disease support with limited robustness to multimorbidity and streaming home data, while recent LLM-based systems demonstrate impressive generative capabilities but lack structured integration with tools, governance, and national infrastructures.[1,4,5] Agentic AI frameworks and orchestration toolkits have introduced powerful abstractions for multi-step, tool-augmented workflows, yet their application to safety-critical, regulated primary care settings remains embryonic.[6,10]

The proposed agentic AI architecture for MMG-led care sits at the intersection of these strands. It extends traditional CDSS by supporting multimorbidity, integrating PGHD, and providing structured, bidirectional interfaces with regional infrastructures. It advances LLM-based decision support by combining a central planning LLM with a bank of specialty SLMs, RAG over PDTA and guidelines, and deterministic clinical tools under explicit schema and safety contracts. Finally, it operationalizes agentic principles within a governance-aware framework that emphasizes calibration, auditability, and controlled rollout in real-world primary care environments.

3. System Overview

We designed an agentic AI system for primary care decision support with the goal of assisting general practitioners (MMG) in monitoring patients at home. The architecture separates functions into ten modules (F1–F10; Figure 1), each corresponding to a distinct responsibility within the clinical workflow. This modular decomposition enables controlled evaluation, versioning, and integration into existing health infrastructures.

 doi:10.20944/preprints202602.0164.v1

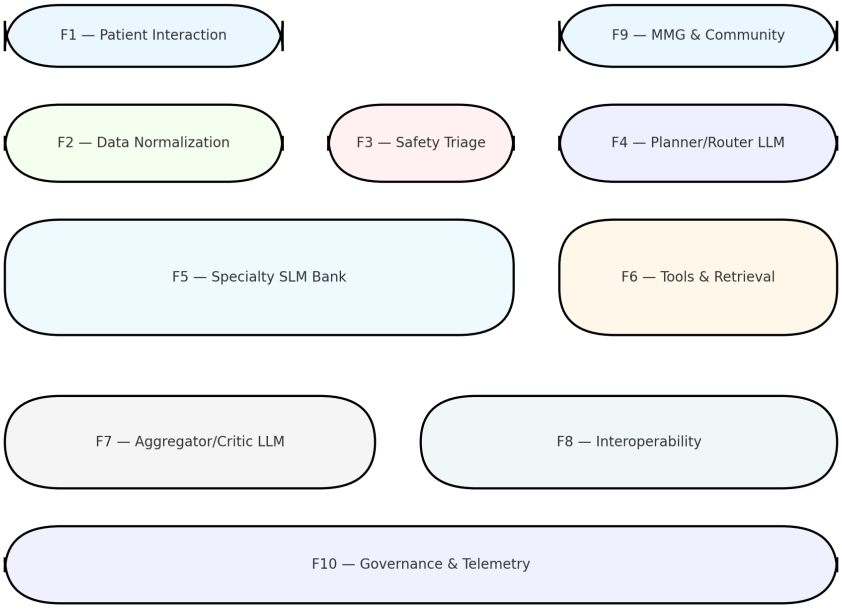

Figure 1. Functional schema of the agentic AI system for primary care. (a) The functional schema decomposes the architecture into ten modules (F1–F10). Patient and MMG/community interfaces (F1, F9) capture symptoms, vitals, and medication adherence, and deliver actionable alerts and care tasks. Data are normalized into FHIR resources with standardized terminologies (F2), then screened by a safety triage layer (F3) combining deterministic thresholds with a mini-LLM for red-flag detection and escalation. If no urgent event is detected, a planner/Router LLM (F4) constructs an explicit call-graph, routing subtasks to domain-specific SLMs (F5) or deterministic tools and retrieval services (F6). Outputs are merged and calibrated by an aggregator/critic LLM (F7), producing structured clinician-facing recommendations and plain-language patient summaries. These artifacts are exported through interoperability services (F8) to regional infrastructures such as the FSE 2.0 and ePrescription platforms. Governance functions (F10) provide registries, telemetry, and audit trails to support transparency, model versioning, and regulatory compliance. This modular decomposition allows each component to be validated independently, ensures that clinical recommendations remain traceable to guidelines and calculators, and facilitates safe integration into heterogeneous national health systems.

Data ingestion and normalization

Patients interact with the system via a mobile application (F1), capturing self-reported symptoms, medication adherence, and data from sensors (blood glucose, blood pressure, SpO_2, ECG). These inputs are normalized (F2) into FHIR R4 resources with terminology binding to SNOMED CT, LOINC, and ATC/RxNorm. Automated unit conversion and plausibility checks ensure consistency and safety.

Safety triage

Inputs are screened by a safety triage layer (F3), consisting of deterministic rules and a mini-LLM trained to classify free-text reports into predefined escalation categories. Acute risks (e.g., suspected DKA, chest pain) trigger immediate escalation.

Planning and routing

If safe, the Planner/Router LLM (F4) decomposes the clinical query into sub-tasks and constructs a call-graph specifying which specialty components are required. This LLM is constrained to output structured JSON conforming to a schema and cannot generate clinical recommendations directly.

Specialty SLMs and tools

The Specialty SLM Bank (F5) consists of domain-focused models (3–7B parameters) with LoRA adapters. Each SLM is aligned with PDTA and guideline sources. Deterministic tools (F6) handle numeric calculations (eGFR, ASCVD, CHA_2DS_2-VASc, QTc) and a retrieval layer provides validated citations.

 doi:10.20944/preprints202602.0164.v1

Aggregation and summarization

Outputs are merged by the Aggregator/Critic LLM (F7), which resolves conflicts, calibrates confidence, and generates a clinician-facing report and a patient-facing explanation.

Interoperability and governance

Final artifacts are exported (F8) as FHIR resources for bidirectional integration with the FSE 2.0 and ePrescription systems. Governance functions (F10) maintain a model registry, track evaluation metrics (ECE, Brier, citation coverage, refusal rates), and archive audit logs.

Evaluation

We validated the system in shadow-mode using de-identified vignettes of diabetes–hypertension comorbidity. Functional correctness was evaluated at each stage. Calibration metrics were computed, and clinicians rated recommendation quality and alignment to PDTA.

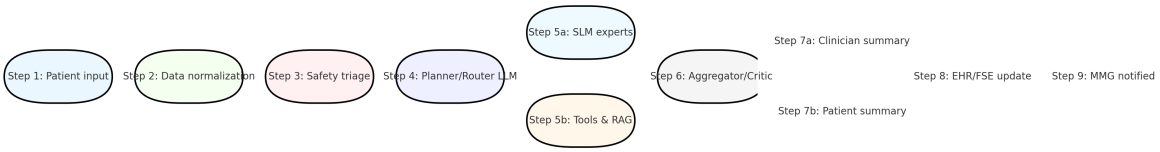

Figure 2. The workflow illustrates the patient journey from input and normalization, through safety triage, orchestration, specialty reasoning, and aggregation, to the generation of tailored outputs. The system enforces layered safety gates, explicit schema contracts, and mandatory citation checks, ensuring that clinician and patient summaries are both actionable and auditable.

4. Case Study: A First Implementaiton in a LangChain-Based Workflow

The proposed agentic architecture for primary care decision support has been instantiated in a first working prototype built on *LangChain*. This case study illustrates how LangChain and the LangChain Expression Language (LCEL) operationalize the conceptual modules of the system into an executable workflow, providing a concrete path from abstract design to reproducible code.[10] The implementation focuses on a reduced but representative scenario in which clinical vignettes are routed to cardiology or nephrology experts, enriched with domain knowledge through retrieval-augmented generation, and transformed into structured, auditable recommendations.

The proposed system is implemented using LangChain [12], an open-source framework for developing applications around large language models (LLMs) through the composition of modular building blocks, including prompts, models, retrievers, memory modules, and external tools [12]. By providing a unified abstraction over heterogeneous model providers (e.g., OpenAI, Anthropic, or locally hosted models) and data backends (e.g., vector databases or relational stores), LangChain enables a single high-level workflow specification to be instantiated across different concrete implementations without altering the overall control logic [12]. In this study, LangChain functions as an orchestration layer that integrates domain-specific data sources and custom tooling with general-purpose LLMs within a cohesive and reproducible pipeline.

LangChain Workflow Structure

A LangChain workflow is formalized as a directed sequence of components—referred to as a *chain*—that transforms an initial input (such as a user query or structured request) into a final output through a series of intermediate processing steps. Each component exposes a standardized execution interface (e.g., `invoke`, `batch`, or `stream`) and exchanges structured data objects, typically represented as key–value mappings, with subsequent components. This design facilitates modular testing, logging, and reuse of individual stages within the pipeline. In the present application, the workflow is structured into four main stages: (i) input preprocessing, (ii) optional retrieval and tool

 doi:10.20944/preprints202602.0164.v1

invocation, (iii) LLM-driven reasoning, and (iv) structured output parsing. Each stage is implemented as a LangChain component and composed using the LangChain Expression Language (LCEL).

LangChain Expression Language (LCEL)

LCEL is a declarative expression language introduced by LangChain to construct workflows from composable *runnables* using a pipe operator, thereby defining an explicit left-to-right dataflow graph. In LCEL, every building block—including prompt templates, model wrappers, routers, and output parsers—implements a common `Runnable` interface. This abstraction automatically provides support for synchronous and asynchronous execution (`invoke`, `ainvoke`), batching (`batch`, `abatch`), and token-level streaming (`stream`, `astream`). A representative LCEL chain employed in this work can be succinctly expressed as

$$\text{chain} = \text{prompt} \mid \text{llm} \mid \text{output\_parser},$$

where the `prompt` runnable maps structured inputs to a formatted textual prompt, the `llm` runnable performs generative inference, and the `output\_parser` runnable converts the raw model response into a task-specific structured representation.

Workflow Design with LCEL

The end-to-end system is implemented as a single LCEL expression that integrates preprocessing, retrieval, reasoning, and post-processing through sequential composition and, where required, branching runnables. Initially, an input preprocessing runnable validates and normalizes incoming requests, optionally enriching them with metadata (e.g., task identifiers or domain cues) used for downstream routing decisions. The resulting structured object is then passed to a prompt-construction runnable, which instantiates a parameterized template encoding system-level instructions and domain-specific context.

For retrieval-augmented generation, the workflow leverages LCEL constructs such as `RunnableBranch` or parallel subchains to query external knowledge sources (e.g., vector-store retrievers), integrate retrieved documents into the prompt, and forward the context-enhanced input to a chat model runnable. The generated response is subsequently processed by an output parser that enforces a predefined schema (e.g., JSON or domain-specific fields) and incorporates error handling or fallback strategies in cases of parsing failure.

Execution and Reproducibility

Because all components adhere to the `Runnable` abstraction, the same LCEL-defined workflow naturally supports single-instance inference, batched execution, and streaming outputs by simply selecting the appropriate execution method (e.g., `invoke`, `batch`, or `stream`). This flexibility is advantageous for both interactive applications and large-scale offline processing. Importantly, the declarative LCEL specification is independent of specific model providers, retrievers, or hyperparameter choices; consequently, modifications to the underlying configuration do not affect the logical structure of the workflow. This separation facilitates controlled experimentation, ablation studies, and deployment in production environments where configuration is externalized. Furthermore, the workflow can be instrumented using LangChain-associated tooling (e.g., LangSmith) for tracing, evaluation, and error analysis, thereby enhancing transparency and reproducibility in LLM-centric systems.

4.1. LangChain as Orchestration Layer

LangChain is used as the orchestration layer that glues together large language models, retrieval components, and deterministic tools into a single, traceable pipeline.[10] In this prototype, LangChain abstractions (prompts, models, retrievers, and tools) mirror the logical components of the architecture: the safety gate, the planner/router, the bank of specialty models, and the aggregation and formatting stages. By leveraging LangChain's provider-agnostic interfaces, the same pipeline can run on different

 doi:10.20944/preprints202602.0164.v1

underlying LLMs (e.g., general-purpose chat models or smaller domain-specific models) and different vector stores without changing the control logic.

A LangChain workflow is formalized as a directed sequence of components, or "chains", that transform an initial input (here, a structured clinical case) into a final, schema-conforming output. Each component implements a standardized execution interface (such as `invoke`, `batch`, or `stream`) and exchanges key–value data objects with downstream components, which naturally supports modular testing, logging, and substitution. In the present application, the high-level workflow is divided into four stages: input preprocessing, optional retrieval and tool invocation, LLM-driven reasoning, and structured output parsing.

4.2. LCEL: Declarative Workflow Construction

The LangChain Expression Language (LCEL) provides a declarative way to compose these building blocks into an explicit dataflow graph.[10] Every building block—prompt templates, model wrappers, routers, retrievers, and output parsers—implements a common `Runnable` interface, which automatically enables synchronous and asynchronous execution (`invoke/ainvoke`), batching (`batch/abatch`), and token-level streaming (`stream/astream`). Conceptually, a simple LCEL chain can be written as

```
chain = prompt | llm | output_parser
```

where the prompt runnable maps structured inputs to a formatted prompt, the LLM runnable performs generative inference, and the output parser runnable enforces a downstream schema.

In the prototype, the full decision-support pipeline is expressed as a single LCEL graph that integrates preprocessing, routing, retrieval, specialty reasoning, and post-processing. The LCEL specification is independent of the concrete model providers, retriever configurations, or prompt variants, which are supplied as parameters or environment configuration. This separation supports ablation studies (e.g., swapping a generic model with a cardiology-finetuned model) and facilitates migration from experimental to production deployments without rewriting the core workflow.

4.3. From Architecture to LangChain Workflow

The implementation maps the conceptual modules F1–F7 of the architecture into a set of LCEL-defined chains. In the current case study, the focus is on a subset of tasks around specialty routing and retrieval-augmented reasoning rather than on full FHIR integration or safety triage. A simplified end-to-end flow can be summarized as follows:

- **Input preprocessing chain.** It receives a structured representation of the clinical vignette (e.g., demographics, comorbidities, current medications, and key symptoms), performs validation and normalization (such as ensuring that required fields are present and encoding disease labels or numeric values in a model-friendly format), and optionally enriches the input with metadata such as task identifiers and flags indicating known comorbidities.
- **Router chain.** It implements a small LLM-based classifier that maps the preprocessed case onto a target specialty, in this first prototype limited to cardiology or nephrology. The router uses a prompt template that exposes salient features of the case and asks the model to decide the most appropriate specialty under explicit constraints (e.g., "respond with one label only"), and is expressed as an LCEL runnable that takes the structured input and returns a structured routing decision for downstream branches.
- **Retrieval module.** It uses a vector-store retriever to obtain domain-specific context from curated corpora, including PubMed abstracts, clinical guidelines, and, when available, disease-specific multi-omics resources. The retriever is invoked conditionally depending on the selected specialty; for example, cardiology cases query a cardiology-specific index, and nephrology cases query a nephrology index, and the retrieved passages are integrated into the prompt supplied to the specialty language model.

 doi:10.20944/preprints202602.0164.v1

- **Specialty reasoning chains.** Two parallel LCEL sub-chains are instantiated: one for the Cardiology LM and one for the Nephrology LM. Each sub-chain takes as input the structured case plus retrieved context and applies a specialty-specific prompt template that encodes both general decision-support instructions and local care pathway conventions. The underlying models can be small domain-focused language models (SLMs) or suitably configured general-purpose LLMs; in either case, the chain is responsible for producing recommendations constrained to a predefined schema (e.g., diagnosis hypotheses, recommended tests, treatment suggestions, monitoring plan, and citations).
- **Output parsing and structuring.** The raw model response is processed by an LCEL output parser that enforces JSON or domain-specific field constraints, with embedded error handling logic (for example, re-prompting in case of schema violations or applying rule-based post-processing to standardize codes or ranges). The final output is a structured object suitable for downstream integration with the broader agentic architecture, including aggregation, logging, and interoperability modules, even if these are not fully implemented in this first prototype.

4.4. Evaluation

5. Conclusions

We have presented an agentic AI architecture for primary care that combines a central planning LLM, a bank of small specialty language models, and deterministic clinical tools to support Medici di Medicina Generale in managing complex patients at home.[1,4–6] By decomposing the workflow into modular functions—from data ingestion and FHIR-based normalization to safety triage, planning and routing, specialty reasoning, aggregation, and governance—the design aims to balance broad clinical coverage with transparency, auditability, and alignment to local care pathways and national infrastructures such as FSE 2.0 and ePrescription.[9,10] Shadow-mode evaluation on de-identified vignettes suggests that the components can be validated independently, with calibration metrics and clinician ratings indicating that recommendations can be made both actionable and traceable to underlying guidelines and calculators.

The first implementation in LangChain demonstrates that this architecture can be concretely instantiated using a provider-agnostic orchestration framework and a declarative expression language for workflow composition.[10] LCEL-based chains capture input preprocessing, router logic, retrieval-augmented specialty reasoning, and schema-constrained output parsing in a way that is amenable to systematic testing, tracing, and reconfiguration. This prototype, initially focused on cardiology and nephrology use cases, already exercises the core ideas of modular routing, retrieval grounding, and separation between planning, domain reasoning, and deterministic tools, laying the groundwork for expansion to additional specialties and integration with real-world data feeds.

Several limitations remain. The current system has been evaluated only in shadow mode on curated vignettes, without prospective deployment in live clinical workflows or exposure to adversarial inputs and distribution shifts.[6] Safety triage and governance components will require further stress-testing, including monitoring of refusal behavior, hallucination rates, and citation coverage under realistic load and edge cases. Moreover, socio-technical factors such as MMG workload, patient digital literacy, liability, and regulatory requirements will shape acceptable deployment models and must be addressed through participatory design with clinicians, patients, and health authorities.

Future work will therefore focus on three directions. First, we plan to extend the specialty SLM bank and associated RAG corpora to cover a broader range of primary care conditions and multimorbidity patterns, with specialty societies involved in prompt and schema co-design. Second, we aim to integrate the system with operational FHIR pipelines and regional infrastructures, enabling closed-loop workflows where recommendations, orders, and monitoring plans are written back into FSE 2.0 and ePrescription in real time. Third, we envision a program of prospective, stepwise evaluation to quantify clinical impact, equity effects, and unintended consequences, and to iteratively refine the architecture in collaboration with stakeholders. If successful, this line of work could provide a reusable

 doi:10.20944/preprints202602.0164.v1

blueprint for safe, agentic AI systems that augment, rather than replace, general practitioners in the evolving landscape of home-centered primary care.

References

1. Goh, E.; Gallo, R.J.; Strong, E.; Weng, Y.; Kerman, H.; Freed, J.A.; Cool, J.A.; Kanjee, Z.; Lane, K.P.; Parsons, A.S.; et al. GPT-4 assistance for improvement of physician performance on patient care tasks: a randomized controlled trial. *Nature Medicine* **2025**, *31*, 1233–1238.
2. Thirunavukarasu, A.J.; Ting, D.S.J.; Elangovan, K.; Gutierrez, L.; Tan, T.F.; Ting, D.S.W. Large language models in medicine. *Nature medicine* **2023**, *29*, 1930–1940.
3. Carbonari, V.; Veltri, P.; Guzzi, P.H. Decoding Rarity: Large Language Models in the Diagnosis of Rare Diseases. *arXiv preprint arXiv:2505.17065* **2025**.
4. Liu, X.; Liu, H.; Yang, G.; Jiang, Z.; Cui, S.; Zhang, Z.; Wang, H.; Tao, L.; Sun, Y.; Song, Z.; et al. A generalist medical language model for disease diagnosis assistance. *Nature medicine* **2025**, *31*, 932–942.
5. 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. *Nature Medicine* **2025**, *31*, 943–950.
6. 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.
7. Lomoio, U.; Vizza, P.; Giancotti, R.; Petrolo, S.; Flesca, S.; Boccuto, F.; Guzzi, P.H.; Veltri, P.; Tradigo, G. A convolutional autoencoder framework for ECG signal analysis. *Heliyon* **2025**, *11*.
8. Lomoio, U.; Veltri, P.; Guzzi, P.H.; Liò, P. Design and use of a Denoising Convolutional Autoencoder for reconstructing electrocardiogram signals at super resolution. *Artificial Intelligence in Medicine* **2025**, *160*, 103058.
9. Williams, C.Y.; Zack, T.; Miao, B.Y.; Sushil, M.; Wang, M.; Kornblith, A.E.; Butte, A.J. Use of a large language model to assess clinical acuity of adults in the emergency department. *JAMA network open* **2024**, *7*, e248895–e248895.
10. Ronan, I.; Crowley, P.; Rombouts, E.; Cornally, N.; Saab, M.M.; Murphy, D.; Tabirca, S. A LangChain-based pipeline for one-shot synthetic text generation using generative pre-trained transformers in palliative care research. *Journal of Biomedical Informatics* **2025**, p. 104936.
11. Cannataro, M.; Guzzi, P.H.; Veltri, P. IMPRECO: Distributed prediction of protein complexes. *Future Generation Computer Systems* **2010**, *26*, 434–440.
12. Annam, S.; Saxena, M.; Kaushik, U.; Mittal, S. Langchain: Simplifying development with language models. *Textual Intelligence: Large Language Models and Their Real-World Applications* **2025**, pp. 287–304.