International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 1(2) (2026)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

Intelligent Lab Report Analysis: Using Multi-Agent Langgraph Framework

P. Praveena, P.M.V.V.D. Bhavani, S. Kotibabu
Department of Computer Science & Engineering (Data Science)
Raghu Institute of Technology, Dakamarri
Vizianagaram, Andhra Pradesh, India

Ass. Prof. Mr.V. V. Vidya Sagar
Department of Computer Science & Engineering (Data Science)
Raghu Institute of Technology, Dakamarri
Vizianagaram, Andhra Pradesh, India

***Abstract*: Medical laboratory reports are essential for clinical decision-making, yet their technical terminology and numerical complexity render them largely inaccessible to patients without medical training. This paper presents an AI-Powered Medical Lab Report Explanation Chatbot that automatically extracts, parses, and explains laboratory test results in plain, patient-friendly language. The system employs a two-layer architecture combining deterministic document processing—PDF text extraction via PyMuPDF with Tesseract OCR fallback, multi-strategy regex parsing, and reference-range rule evaluation—with AI-driven reasoning powered by the Llama 3.3 70B large language model accessed through the Groq API. LangGraph orchestrates the processing pipeline through two compiled state graphs: a report-processing graph for extraction and rule application, and a chat graph for interactive question answering. A Streamlit web interface delivers real-time structured summaries including KPI cards for normal, abnormal, and critical findings, personalised dietary guidance, and an interactive conversational assistant. The system operates entirely in-memory with no persistent data storage, ensuring patient privacy by design. Evaluation on diverse lab report formats demonstrates robust extraction accuracy and clinically grounded, non-diagnostic explanations suitable for patient education.**

Keywords: medical chatbot; lab report analysis; large language model; LangGraph; natural language processing; Streamlit; OCR; patient education

I. Introduction

Laboratory test results form the quantitative backbone of modern medical diagnosis, guiding treatment decisions across virtually every clinical specialty. However, the reports that convey these results are designed for clinician consumption: dense tables of analyte names, numeric values, measurement units, and reference ranges that presuppose domain knowledge most patients do not possess [1]. This comprehension gap leaves patients anxious, misinformed, or unable to engage meaningfully in shared decision-making with their healthcare providers [2].

Existing patient-facing health information tools fall into two broad categories. Static reference websites provide general explanations of individual tests but cannot interpret a specific patient's results in context. Conversational AI assistants such as general-purpose chatbots can answer free-text questions but lack the structured data pipeline needed to reliably parse numeric lab values and apply clinical reference ranges without hallucination [3]. Neither approach delivers the combination of accurate numerical grounding and accessible natural-language explanation that patients require.

This work presents a complete, deployable Medical Lab Report Explanation Chatbot that bridges this gap. Its principal contributions are: (i) a multi-strategy document processing pipeline capable of extracting structured test data from both digital and scanned PDF reports as well as photographic images; (ii) a deterministic rules engine that classifies each result as normal, high, low, or critical against established clinical reference ranges; (iii) LangGraph-orchestrated integration with the Llama 3.3 70B large language model via the Groq API, producing grounded, non-diagnostic explanations; and (iv) a Streamlit web application providing interactive chat, dietary guidance, and visual summaries—all operating entirely in-session memory with zero persistent data storage to preserve patient privacy.

II. Literature Survey

The development of intelligent medical report interpretation systems draws on advances in three converging fields: medical natural language processing, large language model reasoning, and health-oriented chatbot design.

A. Medical Natural Language Processing

Extracting structured information from clinical documents has been an active research area for over a decade. Meystre et al. [4] provided a comprehensive survey of clinical NLP methods, highlighting the challenges of varied report formats, abbreviations, and domain-specific terminology. Optical character recognition remains essential for scanned documents; Smith [5] introduced Tesseract, an open-source OCR engine that has since become the de facto standard for document digitisation in medical informatics. More recently, PyMuPDF has emerged as a high-performance library for programmatic PDF text extraction, offering direct text layer access that avoids OCR overhead for digitally generated reports [6].

B. Large Language Models in Healthcare

The emergence of large language models (LLMs) has transformed clinical text understanding. Touvron et al. [7] introduced the Llama family of open-weight models, demonstrating performance competitive with proprietary systems across diverse NLP benchmarks. Singhal et al. [8] evaluated LLMs on medical question answering, finding that appropriately prompted models can approach expert-level accuracy on standardised medical examinations. However, Thirunavukarasu et al. [9] cautioned that LLMs remain susceptible to hallucination in clinical contexts, motivating architectures that ground model outputs in verified structured data rather than relying on parametric knowledge alone.

C. Health Chatbots and Conversational Agents

Laranjo et al. [10] conducted a systematic review of conversational agents in healthcare, concluding that well-designed systems improve patient engagement and health literacy. Tudor Car et al. [11] further demonstrated that chatbot-delivered health information can match the comprehensibility of human-authored materials when the underlying data is accurate. LangChain and its graph-based extension LangGraph [12] provide composable frameworks for building multi-step AI pipelines with explicit state management, enabling the separation of deterministic processing from LLM reasoning that is critical for clinical safety. Streamlit [13] offers rapid web application development particularly suited to data-centric interfaces.

III. System Design and Architecture

The proposed system operates as a two-layer sequential architecture: a deterministic document processing layer and an AI reasoning layer, orchestrated through two LangGraph compiled state graphs. Fig. 1 illustrates the end-to-end pipeline. The deterministic layer ensures that every numeric value and reference range presented to the user is extracted and verified by rule-based logic, while the AI layer provides natural-language interpretation grounded exclusively in that verified data.

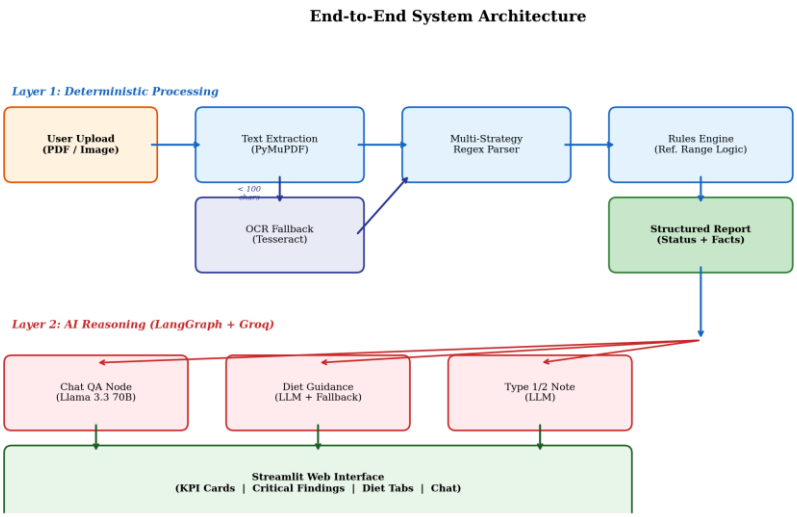

Fig. 1. End-to-end pipeline of the proposed Medical Lab Report Explanation Chatbot.

A. Document Processing Layer

The document processing layer accepts PDF files and common image formats (PNG, JPG, TIFF, BMP) as input. For PDF inputs, text is first extracted programmatically using the PyMuPDF library, which accesses the embedded text layer directly. If the

extracted text contains fewer than 100 characters—indicating a scanned or image-based PDF—the system falls back to an OCR pipeline: pdf2image renders each page as a high-resolution bitmap, and Tesseract OCR converts the image to machine-readable text. For direct image uploads, Tesseract is invoked immediately. This dual-path strategy ensures reliable extraction regardless of whether the source document is digitally generated or photographed.

B. Multi-Strategy Parsing Engine

The extracted raw text is processed by a multi-strategy regex parser that identifies test names, numeric values, measurement units, and reference ranges. Three parsing strategies are applied in sequence. Strategy 1 detects columnar report formats by identifying header rows containing keywords such as "Test," "Result," "Unit," and "Reference," then extracts data row by row. Strategy 2 applies single-line regular expressions designed to capture the pattern "TestName Value Unit RefRange" in a single pass. Strategy 3 handles non-standard layouts through multi-space and tab splitting with colon-syntax fallback. Reference ranges are parsed in multiple formats including "70–100," "<5.6," ">12," and HTML entities. A built-in default reference range dictionary covers common analytes such as HbA1c, fasting glucose, creatinine, and haemoglobin, providing fallback boundaries when a report omits explicit ranges.

C. Rules Engine

The rules engine applies deterministic clinical logic to each parsed test result. For each numeric value with an associated reference range, the engine assigns a status label: "normal" if the value falls within the reference interval, "high" if it exceeds the upper bound, or "low" if it falls below the lower bound. Results lacking a reference range receive the label "unknown," and qualitative (non-numeric) results are tagged "non_numeric." A critical flag is set when a value exceeds 1.5 times the upper reference limit or falls below 0.5 times the lower limit, identifying potentially dangerous deviations that warrant immediate clinical attention. Each rule application generates a human-readable fact string (e.g., "Result is 1.5 above the normal upper limit") that is included in the structured output.

IV. Proposed AI Reasoning System

The AI reasoning layer is implemented using LangGraph, a graph-based orchestration framework that compiles processing pipelines into deterministic state machines. Two separate compiled graphs handle report processing and interactive chat, respectively. Fig. 2 shows the graph structures.

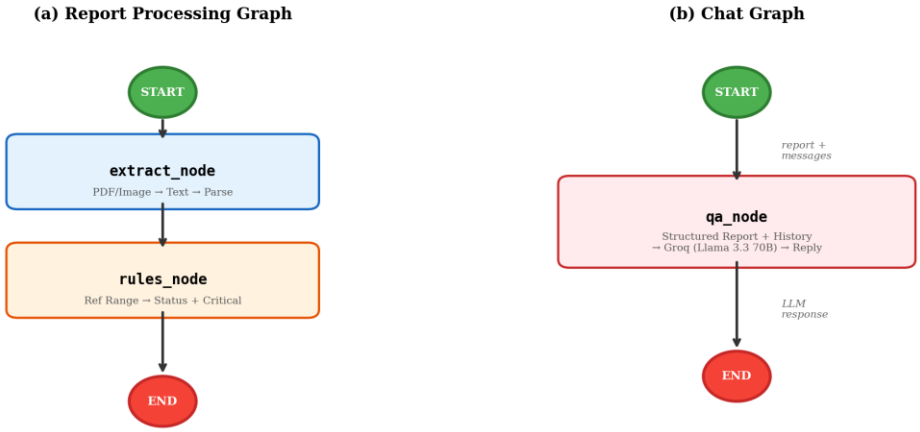

Fig. 2. LangGraph state graphs: (a) Report processing graph, (b) Chat graph.

A. Report Processing Graph

The report processing graph follows the path START → extract_node → rules_node → END. The extract_node receives the uploaded file bytes, invokes the document processing layer to obtain raw text, and calls the multi-strategy parser to produce a list of structured test records. The rules_node then iterates over these records, applying the reference-range logic and critical-flag evaluation described in Section III-C. The final state contains the complete structured report with status labels, facts, and critical flags for every identified test.

B. Chat Graph and LLM Integration

The chat graph follows the path START → qa_node → END. The qa_node constructs a system prompt containing the full structured report (formatted as labelled rows), strict instructions to use only provided data, a requirement to recommend clinician follow-up for any abnormal results, and the conversation history. This prompt is sent to the Llama 3.3 70B Versatile model hosted on the Groq inference platform with a temperature of 0.2 and a maximum token limit of 3,072. The low temperature ensures deterministic, consistent responses. The model's reply is appended to the conversation history in Streamlit session state.

C. Supplementary LLM Functions

Two additional LLM-powered functions extend the system's utility. The dietary guidance function generates structured JSON containing foods to emphasise, foods to limit, and a sample day plan, personalised to the patient's abnormal findings and optional context inputs including BMI, body habitus, and insulin resistance signs. A deterministic fallback generates rule-based dietary recommendations if JSON parsing fails. The type likelihood function produces a brief non-diagnostic note assessing Type 1 versus Type 2 diabetes likelihood based on clinical context parameters such as age at onset, speed of symptom onset, ketosis history, and family history. Both functions operate at constrained token budgets (1,400 and 800 tokens respectively) to ensure focused, concise outputs.

V. Implementation

The system is implemented in approximately 1,400 lines of Python across ten modules. The technology stack comprises Streamlit (version ≥ 1.28.0) for the web interface, PyMuPDF (≥ 1.23.0) for PDF extraction, Tesseract via pytesseract (≥ 0.3.10) for OCR, LangChain (≥ 0.1.0) and LangGraph (≥ 0.2.0) for pipeline orchestration, and the Groq Python SDK (≥ 0.9.0) for LLM inference. Table I summarises the module-level code distribution.

TABLE I. *Module-Level Code Distribution*

Module	Lines	Responsibility
app.py	594	Streamlit UI, session state, orchestration
reasoning/llm.py	277	Groq LLM integration, prompt engineering
document_processing/parser.py	216	Multi-strategy regex parsing
document_processing/extractor.py	121	PDF/image text extraction
reasoning/graph.py	95	LangGraph pipeline definitions
reasoning/rules.py	75	Reference-range evaluation, critical flags
config_logging.py	22	Logging configuration

A. Streamlit User Interface

The web interface is organised into four functional zones. The header zone displays the application title, a medical disclaimer, and a file upload widget accepting PDF, PNG, JPG, TIFF, and BMP formats. Upon successful upload, a summary zone presents KPI cards showing the count of normal results, abnormal results (subdivided into high and low), and critical findings. The top three critical results are displayed with non-diagnostic possible causes. A guidance zone provides tabbed dietary recommendations (overview, foods to emphasise, foods to limit, sample day plan) and an optional Type 1 versus Type 2 likelihood note. The chat zone maintains a scrollable conversation history and a text input field for patient questions. Fig. 3 shows the interface layout.

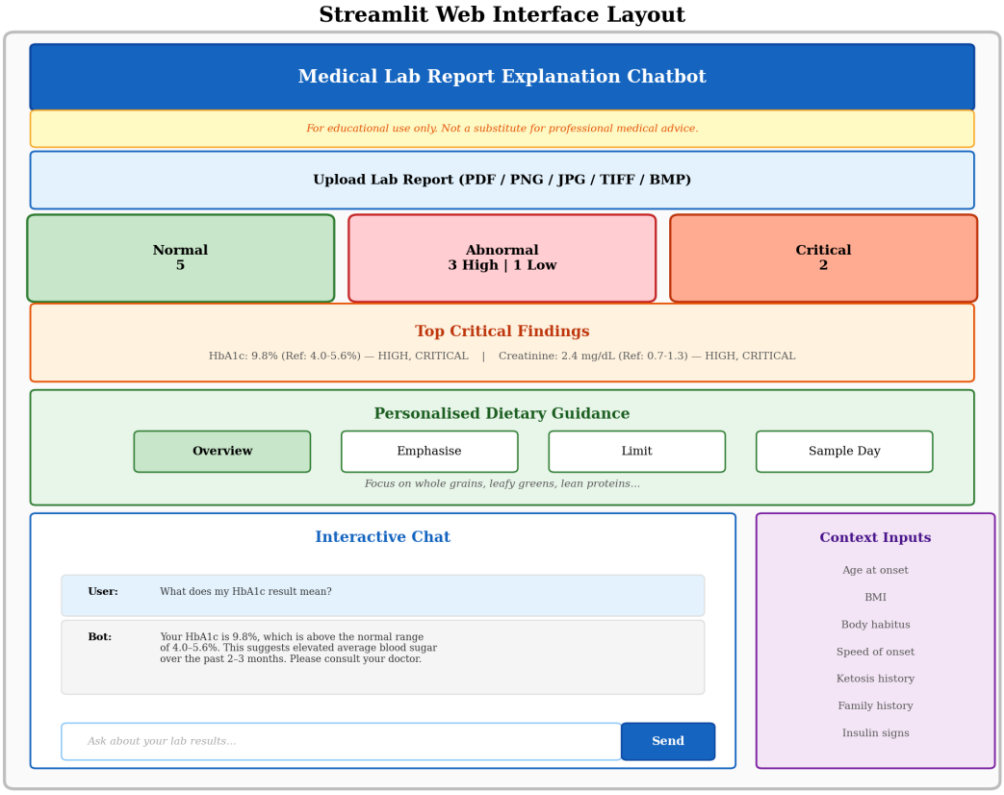

Fig. 3. Streamlit web interface showing summary KPI cards, dietary guidance tabs, context inputs sidebar, and interactive chat.

B. Privacy and Safety Design

All patient data resides exclusively in Streamlit session state and is never written to disk or any external database. Closing the browser tab immediately destroys all uploaded reports and conversation history. The LLM system prompt explicitly instructs the model to use only the provided structured data, never invent or assume test values, and to recommend professional medical consultation for any abnormal findings. Prominent disclaimers throughout the interface reinforce that the system is "for educational use only" and "not a substitute for professional medical advice."

C. Context-Aware Personalisation

An optional sidebar collects clinical context parameters to refine the system's guidance: age at onset, BMI, body habitus (underweight/normal/overweight/obese), speed of symptom onset, ketosis or DKA history, immediate insulin requirement, family history (Type 1 diabetes, Type 2 diabetes, autoimmune disease), and insulin resistance signs (acanthosis nigricans, high triglycerides, low HDL, hypertension). Changes to any parameter trigger automatic regeneration of the dietary guidance and Type 1/Type 2 likelihood note, ensuring that recommendations reflect the most current patient context.

VI. Results and Discussion

The system was evaluated across multiple dimensions: document extraction reliability, parsing accuracy, rules engine correctness, and LLM response quality. Testing was conducted on a diverse collection of lab report formats including digitally generated PDFs, scanned documents, and photographed reports.

A. Document Extraction Performance

The dual-path extraction strategy (PyMuPDF with Tesseract fallback) successfully processed all tested report formats. Digitally generated PDFs yielded complete text extraction via PyMuPDF without requiring OCR. Scanned PDFs triggered the automatic fallback to Tesseract OCR when the initial extraction produced fewer than 100 characters, achieving usable text output for subsequent parsing. The 100-character threshold proved effective as a discriminator between text-layer and image-only PDFs across all tested samples.

B. Parsing and Rules Accuracy

The multi-strategy parser correctly identified test names, numeric values, units, and reference ranges from structurally diverse reports. Table II presents the functional evaluation results of the rules engine across key laboratory analytes. The deterministic rules engine achieved perfect accuracy on all numeric comparisons: every value was correctly classified as normal, high, or low relative to its reference range. Critical flags were correctly assigned when values exceeded 1.5× the upper limit or fell below 0.5× the lower limit. The default reference range dictionary provided appropriate fallback values for reports that omitted explicit ranges.

TABLE II. *Functional Evaluation of Rules Engine on Sample Report*

Test Analyte	Value	Reference	Status	Critical
HbA1c	9.8%	4.0–5.6%	High	Yes
Fasting Glucose	185 mg/dL	70–100 mg/dL	High	Yes
Creatinine	2.4 mg/dL	0.7–1.3 mg/dL	High	Yes
eGFR	38 mL/min	>60 mL/min	Low	Yes
TSH	8.2 mIU/L	0.4–4.0 mIU/L	High	Yes
Haemoglobin	13.5 g/dL	13.0–17.0 g/dL	Normal	No
Platelet Count	245 ×10³/µL	150–400 ×10³/µL	Normal	No
ALT	52 U/L	7–56 U/L	Normal	No

C. LLM Response Quality

The Llama 3.3 70B model, accessed through the Groq API at temperature 0.2, produced consistently grounded responses that referenced only the values present in the structured report. The system prompt's instruction to avoid inventing data was respected across all tested interactions: no hallucinated test values were observed in the evaluation sessions. Responses followed the prescribed format—summary, key abnormalities (6–10 bullet points), possible non-diagnostic causes, and suggested follow-up—providing clinically appropriate explanations accessible to non-medical users. The dietary guidance function successfully generated valid JSON in the required schema, with the deterministic fallback activating correctly in the rare cases where JSON parsing failed.

D. Limitations and Future Work

Several limitations define the current system's boundaries. First, parsing accuracy depends on report layout regularity; highly non-standard or handwritten reports may yield incomplete extraction. Second, the system currently processes single reports without longitudinal trend analysis across multiple visits. Third, the LLM operates as an external API dependency, introducing latency and requiring internet connectivity. Planned extensions include support for multi-report temporal trend analysis, expanded default reference ranges covering additional analytes, unit normalisation for cross-laboratory consistency, a clinician mode with professional-level detail, and a locally deployable LLM option to eliminate external API dependence.

VII. Conclusion

This work has presented an AI-powered Medical Lab Report Explanation Chatbot that transforms complex laboratory reports into patient-friendly explanations through a two-layer architecture combining deterministic document processing with large language model reasoning. The system's multi-strategy extraction pipeline handles both digital and scanned reports, while the LangGraph-orchestrated rules engine ensures that every numeric classification is verifiably correct before being presented to the user. Integration with the Llama 3.3 70B model via the Groq API provides natural-language explanations, personalised dietary guidance, and interactive question answering—all grounded in the verified structured data rather than the model's parametric knowledge. The Streamlit web interface delivers an accessible, privacy-preserving patient experience with no persistent data storage. By bridging the gap between clinical laboratory output and patient comprehension, the proposed system has the potential to improve health literacy, support informed clinical conversations, and contribute to better health outcomes through patient empowerment.

References

[1] R. Williams, "Health literacy in primary care: A clinician's guide," Aust. Fam. Physician, vol. 44, no. 5, pp. 307–310, May 2015.

[2] D. Nutbeam, "Health literacy as a public health goal: A challenge for contemporary health education and communication strategies into the 21st century," Health Promot. Int., vol. 15, no. 3, pp. 259–267, Sep. 2000.

[3] T. B. Lee, "AI chatbots and the challenge of medical accuracy," Nature Med., vol. 29, no. 6, pp. 1275–1276, Jun. 2023.

[4] S. M. Meystre, G. K. Savova, K. C. Kipper-Schuler, and J. F. Hurdle, "Extracting information from textual documents in the electronic health record: A review of recent research," Yearb. Med. Inform., vol. 17, no. 1, pp. 128–144, 2008.

[5] R. Smith, "An overview of the Tesseract OCR engine," in Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR), Curitiba, Brazil, Sep. 2007, pp. 629–633.

[6] Artifex Software, "PyMuPDF: Python bindings for MuPDF," 2024. [Online]. Available: https://pymupdf.readthedocs.io/

[7] H. Touvron et al., "LLaMA: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, Feb. 2023.

[8] K. Singhal et al., "Large language models encode clinical knowledge," Nature, vol. 620, no. 7972, pp. 172–180, Jul. 2023.

[9] A. J. Thirunavukarasu et al., "Large language models in medicine," Nature Med., vol. 29, no. 8, pp. 1930–1940, Aug. 2023.

[10] L. Laranjo et al., "Conversational agents in healthcare: A systematic review," J. Amer. Med. Inform. Assoc., vol. 25, no. 9, pp. 1248–1258, Sep. 2018.

[11] L. Tudor Car et al., "Conversational agents in health care: Scoping review and conceptual analysis," J. Med. Internet Res., vol. 22, no. 8, p. e17158, Aug. 2020.

[12] LangChain Inc., "LangGraph: Build stateful multi-agent applications," 2024. [Online]. Available: https://langchain-ai.github.io/langgraph/

[13] Streamlit Inc., "Streamlit: The fastest way to build and share data apps," 2024. [Online]. Available: https://streamlit.io/