iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80986

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Multi-Model AI Agent for Health Diagnostics: A LangGraph-Orchestrated Pipeline for Medical Report Analysis with Retrieval-Augmented Generation

Jamparangi Likith Sagar[1], Carlos Antonio Cherindza[2], Tathapudi Vamsi[3], Dr. Bharati Bidikar[4]
[1, 2, 3]*Department of Information Technology and Computer Application Engineering*
[4]*Department of Computer Science and System Engineering*
Andhra University College of Engineering (AUCE), Visakhapatnam – 530003, India

Abstract: ***Medical laboratory reports carry dense clinical data that both physicians and patients struggle to interpret efficiently. Clinicians need clear pattern recognition across test values to inform diagnosis and care, while patients need accessible explanations in daily used language. Existing tools either fail when report formats change or rely on a single AI model with no mechanism to validate medical accuracy. This paper presents a multi-model AI pipeline that ingests lab reports as scanned images or PDFs, extracts clinical values through OCR and AI-based parsing, and benchmarks them against standard reference ranges. Each extracted value passes through three sequential AI models to cross-check interpretations and reduce single-model error. The system produces a daily used language health summary with context-specific observations, and exposes a retrieval-augmented generation (RAG) chat layer that allows users to query their own report in natural language. The pipeline runs on FastAPI, LangGraph, Groq LPU, and FAISS, completing end-to-end processing of a standard Complete Blood Count (CBC) report in 15 to 30 seconds.***

Keywords: ***Medical Report Analysis, Large Language Models, LangGraph, Retrieval-Augmented Generation, FAISS, Multi-Model AI, Health Informatics, FastAPI, Groq LPU.***

I. INTRODUCTION

Patients receiving a Complete Blood Count (CBC) or metabolic panel report are typically presented with numerical values and directional indicators without contextual interpretation. Clinicians also require an efficient means of reviewing every parameter, identifying patterns, and determining what the results imply for a given patient. Consultation time constraints often preclude a detailed walkthrough of each number. Patients frequently resort to online searches, which yield either alarmist or overly simplistic explanations that do not support informed decision-making.

Recent advances in large language models (LLMs) have made it feasible to automatically explain medical text in plain language [1]. State-of-the-art models such as GPT-4 now approach the performance of medical professionals on standard clinical examination questions [3]. However, raw model capability alone is insufficient for clinical utility. A robust medical report interpretation tool must additionally handle diverse report formats, extract values from unstructured text, compare those values to clinical reference ranges, identify patterns across multiple parameters, adjust for patient age and sex, and answer questions grounded in the specific report rather than general knowledge.

This paper describes a system that meets these requirements through a structured, staged AI pipeline built with LangGraph [7]. The objective is direct: accept a raw lab report and return a clear, personalised health summary with actionable guidance, without requiring domain expertise from the end user.

II. RELATED WORK

Early medical report tools relied on rule-based parsers and named entity recognition, which performed acceptably on known layouts but degraded sharply on new templates or non-standard documents. Domain-specific pretrained models such as BioBERT [2] improved accuracy on clinical text but were designed primarily for classification and information-extraction tasks rather than patient-facing explanation. General-purpose large language models demonstrate broad medical knowledge yet can produce confidently incorrect statements, a failure mode with particular consequences in clinical settings [3].

Retrieval-augmented generation (RAG) was introduced to ground generated output in a specific document corpus rather than relying solely on parametric memory [4], reducing hallucinations and improving traceability. Workflow-orchestration frameworks have since emerged for composing LLM calls into deterministic, inspectable pipelines: LangGraph [7] extends LangChain with stateful, graph-structured workflows in which each node passes a shared state object to the next, a design well suited to staged report analysis, while FAISS [5] provides high-throughput dense-vector similarity search and serves as the retrieval substrate for user queries in the proposed system.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

The proposed system is realised as a LangGraph [7] directed graph comprising eight nodes, grouped into four functional stages: Ingestion, Analysis, Output Generation, and the Chat Interface. A shared state object is passed between nodes, allowing each stage to be tested or replaced independently without affecting the rest of the pipeline. Fig. 1 depicts the complete pipeline.

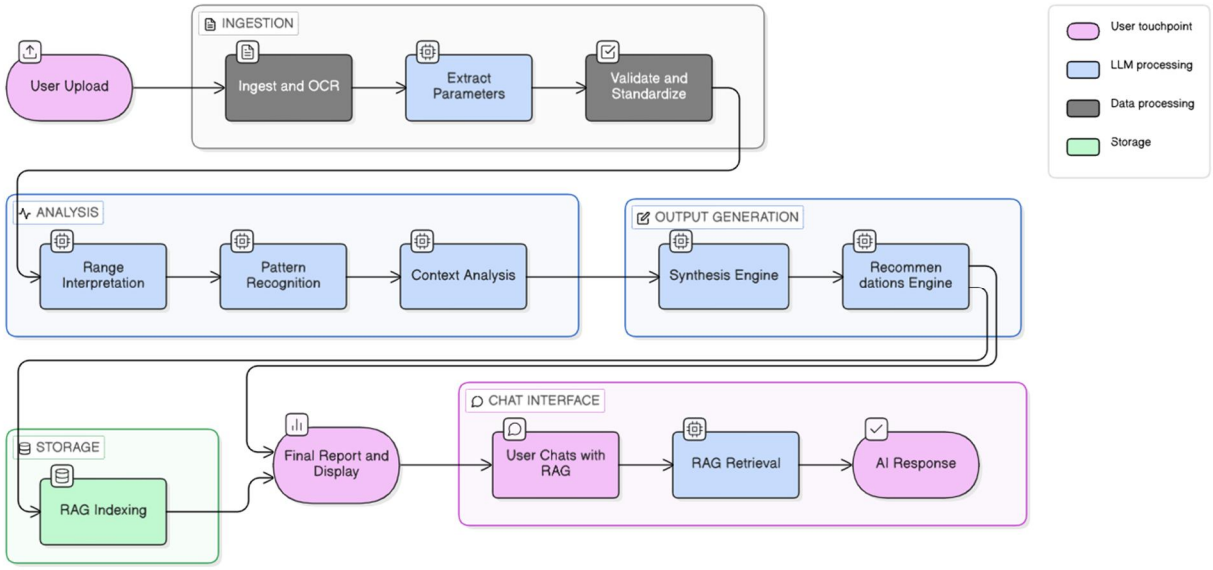


Fig. 1. Full LangGraph pipeline: Ingestion, Analysis, Output Generation, and Chat Interface.

A. Document Ingestion and OCR (Node 1)

The system takes PDF files or uploaded images in JPG and JPEG format. PDFs are read directly using PyMuPDF, which gets the text without scanning the page as an image. For image files, the system straightens the image, fixes the contrast, and removes noise before running Tesseract OCR [6] to read the text. Both paths then go through a cleaning step that fixes OCR mistakes, corrects encoding issues, and puts a space between the numbers and units.

B. Parameter Extraction and Validation (Nodes 2 and 3)

An AI model reads the cleaned text in overlapping chunks and returns a list of blood test values, each with a name, number, unit, and normal range. Using an AI model here instead of fixed pattern rules means the system works on many different lab report formats without needing custom rules for each one. A retry loop catches most formatting errors, usually in one or two tries. The validation step then compares each value to a medical reference database, picks the right range for the patient's age and sex, and marks each value as Low, Normal, or High.

C. Three-Model Analysis (Nodes 4, 5, and 6)

Three AI models run one after another on the checked values. The first model looks at each abnormal value on its own and explains it in plain language with a confidence level. The second model looks at all values together and finds patterns that point to a health condition, because a group of values often tells more than any one value alone. The third model re-reads those findings with the patient's age and sex in mind, adjusts anything where normal ranges differ by age or sex, and gives a risk score from 0 to 10 with a written explanation.

D. Synthesis, Recommendations, and Safety (Nodes 7 and 8)

A combining step merges the outputs of all three models into one clear health story, fixes any conflicts between them, and puts findings in order of importance. A suggestions step then turns those findings into specific, everyday lifestyle and diet tips. A safety check goes over the final output and swaps out any drug names, dose instructions, or anything that sounds like a medical diagnosis with a note to talk to a doctor, keeping the tool within safe limits.

E. Retrieval-Augmented Chat Interface

After the pipeline finishes, the report text and analysis outputs are segmented into overlapping chunks, converted into dense vector embeddings using sentence-transformers, and saved in a FAISS index for that session. When a user asks a question, it gets turned into a vector and matched against that index. The matching pieces and chat history are put together into a prompt that tells the AI to answer only from the given context. If there is not enough information to answer, the model returns a fallback response indicating insufficient context rather than generating an unsupported answer [4].

IV. RESULTS AND DISCUSSION

A standard CBC report takes about 15 to 30 seconds from upload to final output. Most of that time is spent waiting for the AI models to run. Groq LPU was chosen because it is faster than GPU-based services. The RAG index takes about 3 to 5 extra seconds to build, and chat answers come back in under 2 seconds.

OCR works well on clear scanned images. Blurry or tilted scans cause more errors, but the retry loop in the reading step catches most of them. Using three models gives better results than using just one, especially when values are on the edge but the overall picture points to a health issue.

Fig. 2 shows the AI chat being used. A user with Anemia, Thrombocytopenia, and Leukopenia asked what Anemia means. The system explained it in simple words and pointed to the patient's hemoglobin value of 11.2 g/dL, which is below the normal range of 13.0–17.5 g/dL. The report received a Risk Score of 7/10 because all three conditions showed up together.

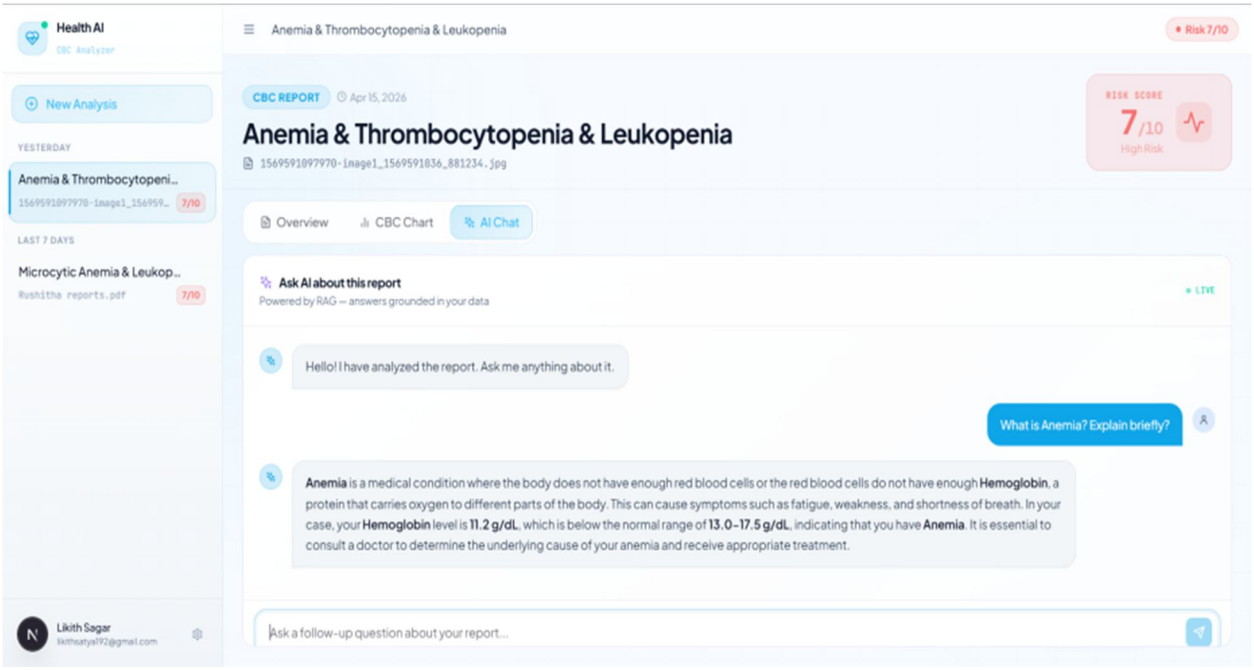


Fig. 2. AI Chat interface explaining Anemia from the patient's own CBC values (Risk Score: 7/10 – High Risk).

Fig. 3 shows the Overview tab for a complex report with Microcytic Anemia, Leukopenia, Neutropenia, Lymphopenia, and Thrombocytopenia. Each blood value shows up as a card with a colored dot: green for normal, yellow for low, red for high. Hemoglobin was 6.8 g/dL, MCV was 56, and Total WBC was 7 all well below normal. The system correctly grouped these together as Microcytic Anemia with several other problems present at the same time.

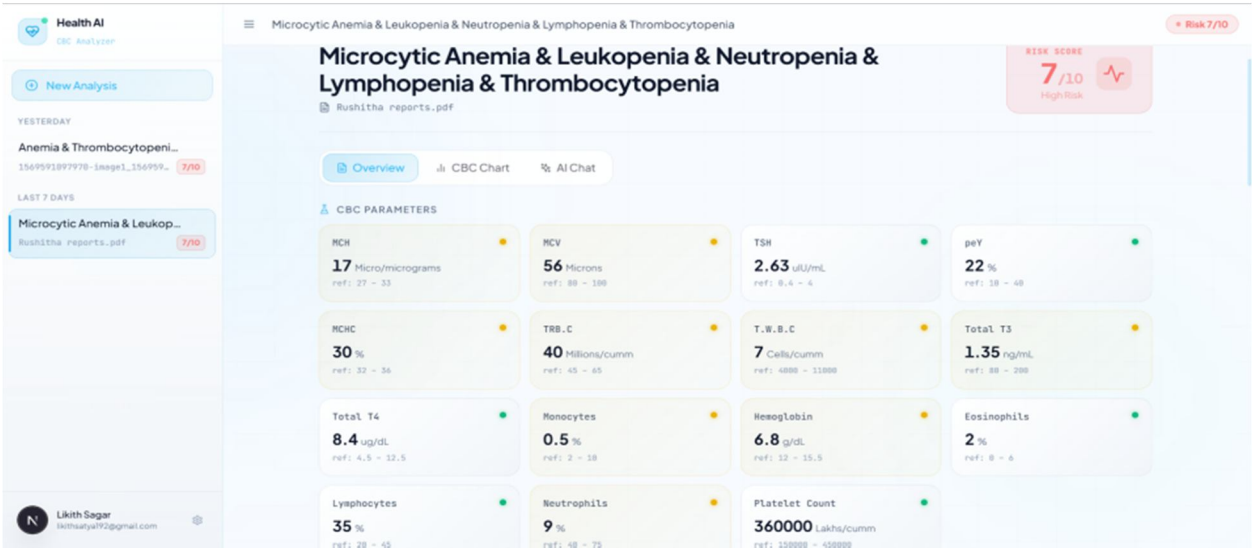


Fig. 3. CBC Overview tab with color-coded parameter cards showing values and reference ranges for a Microcytic Anemia case

Fig. 4 shows the CBC Chart for the same report. Each value is shown as a bar, where 100% means the value sits at the middle of the normal range. Yellow bars are low, green are normal, and red are high. This report had 9 low values and 0 high values. The chart helps patients quickly see which values are off without having to read every number.

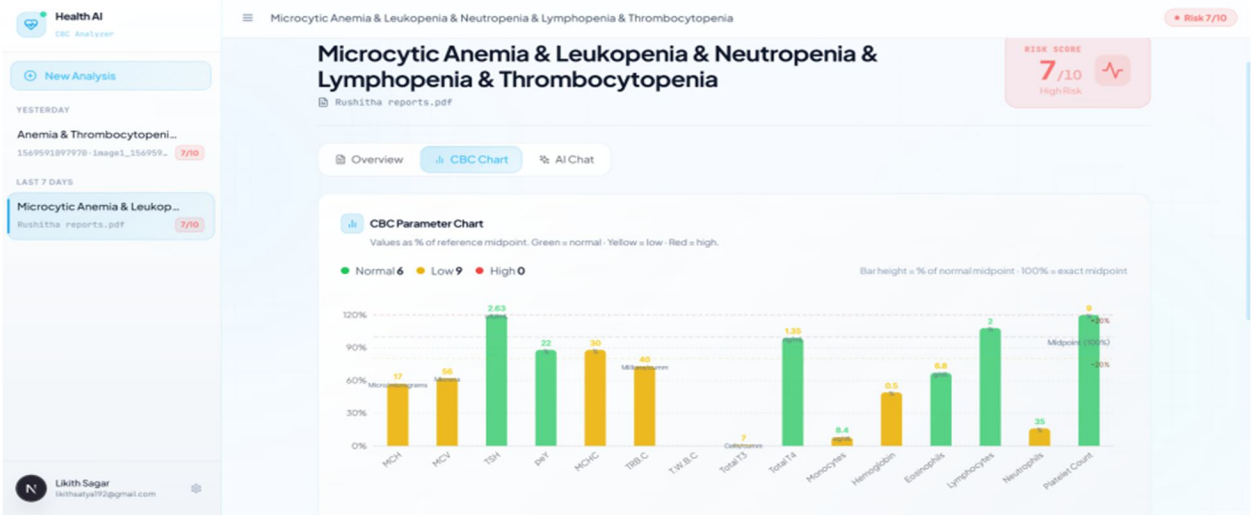


Fig. 4. CBC Parameter Chart showing each value as a percentage of the reference midpoint (Yellow = Low, Green = Normal, Red = High)

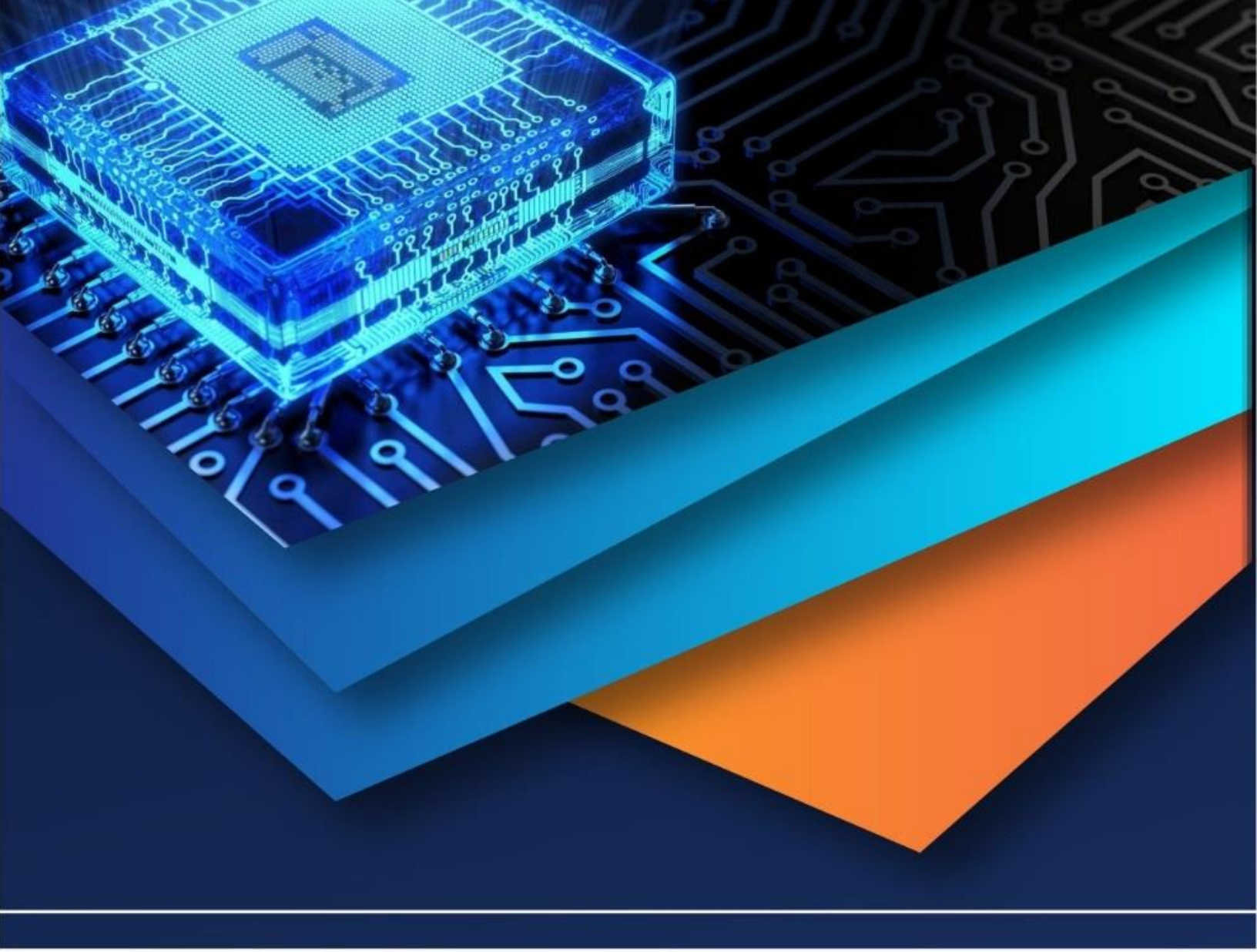
The three-model pipeline works best on cases like Microcytic Anemia, where a correct reading needs low MCV, low MCH, and low hemoglobin to be seen together. One model on its own might flag each value separately but miss what they mean as a group. The second model is built to catch these kinds of patterns. The chat layer lets users ask questions about their own report and get answers based on their actual data.

V. CONCLUSION

This work shows that breaking an AI pipeline into stages can read a lab report, check each value, and explain the results to both doctors and patients in plain words. Older rule-based tools and single model calls usually miss this kind of task. Splitting the work makes each part easy to test and fix on its own, and the FAISS setup keeps each person's data apart so reports stay separate. The same approach fits other fields too. Any document that needs to be read, checked, and explained to a non-expert suits this pipeline. Next steps: add Redis so sessions survive a server restart, a task queue so big reports do not block the site, and a routine that clears stored FAISS index files once a user removes their report.

REFERENCES

[1] P. Rajpurkar, E. Chen, O. Lehman, and A. Ng, "AI applications in healthcare and medicine," *Nature Medicine*, vol. 28, pp. 31–38, 2022.
[2] J. Lee et al., "BioBERT: a pretrained biomedical language model for text mining in biomedicine," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.
[3] K. Singhal et al., "Clinical knowledge encoded in large language models," *Nature*, vol. 620, pp. 172–180, 2023.
[4] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive natural language tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
[5] J. Johnson, M. Douze, and H. Jégou, "GPU-accelerated similarity search at billion scale," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.
[6] R. Smith, "Tesseract OCR engine: an overview," in *Proc. 9th Int. Conf. on Document Analysis and Recognition (ICDAR)*, 2007.
[7] H. Chase et al., "LangGraph: framework for stateful, multi-actor LLM applications," LangChain Documentation, 2024. [Online]. Available: https://langchain-ai.github.io/langgraph/
[8] N. Reimers and I. Gurevych, "Sentence-BERT: sentence embeddings using Siamese BERT networks," in Proc. EMNLP-IJCNLP 2019, pp. 3982–3992.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY