Cognito AI: Post Treatment Healthcare Assistant

ARTIFICIAL INTELLIGENCE PROJECT

SUBMITTED BY
Yash Yadav *(230664)*
Kartavya Dev *(230655)*
Manjeet Kumar *(230598)*
Anushka *(230670)*

Mentored By:
Dr. Anusha Chhabra

SCHOOL OF ENGINEERING AND TECHNOLOGY

BML MUNJAL UNIVERSITY™
FROM HERE TO THE WORLD

BML MUNJAL UNIVERSITY Gurugram,
Haryana - 122413

November 2025

Table Of Contents

ABSTRACT

The healthcare industry generates vast amounts of unstructured data, primarily in the form of medical reports, prescriptions, and clinical notes. Patients often struggle to interpret this information or manage their medication schedules effectively, leading to poor health outcomes. This project proposes and implements a comprehensive **Intelligent Patient-Support System**. The system integrates a secure authentication framework, a Constraint Satisfaction Problem (CSP)-based medication reminder module, and a Retrieval-Augmented Generation (RAG) conversational agent. By utilizing Vector Space Models for state retrieval and Large Language Models (LLMs) for reasoning, the system allows users to upload medical PDFs and receive context-aware, mathematically grounded answers. This report details the architectural design, the application of artificial intelligence principles (including State Space Search and Knowledge Representation), and the ethical frameworks deployed to ensure data privacy and reliability.

CHAPTER 1: INTRODUCTION

1.1 Background and Motivation

In the contemporary digital health landscape, the "democratization of information" remains an unsolved challenge. While patients have access to their medical records, the *literacy* required to interpret a Complete Blood Count (CBC) or a complex MRI report is often lacking. Furthermore, the World Health Organization estimates that medication non-adherence accounts for nearly 50% of treatment failures in chronic disease management.

Existing solutions are often fragmented. Reminder apps lack context, and medical chatbots (like generic ChatGPT) often "hallucinate" or invent facts because they lack access to the patient's specific private data. There is a critical need for a unified system that combines the rigidity of a scheduler with the flexibility of generative AI.

1.2 Problem Statement

The core problem this project addresses is the cognitive disconnect between patients and their medical data. Specifically, the project aims to solve:

1. **Information Overload:** The inability of patients to parse technical medical jargon in PDF reports.
2. **Adherence Failure:** The lack of a rigid, constraint-checked notification system for medication.
3. **Privacy Risks:** The danger of uploading sensitive health data to public cloud-based AI models.

1.3 Project Objectives

The primary objectives of this research and development project are:

- To develop a secure web application that serves as a central repository for patient health data.
- To implement a **Retrieval-Augmented Generation (RAG)** pipeline that grounds AI responses in the user's actual medical documents.
- To model the medication reminder system as a **Constraint Satisfaction Problem (CSP)** to ensure logical validity in scheduling.
- To utilize **Vector Space Search** algorithms to enable semantic understanding of medical queries.

CHAPTER 2: THEORETICAL FRAMEWORK

2.1 Retrieval-Augmented Generation (RAG)

RAG is a hybrid AI architecture that combines a parametric memory (the pre-trained weights of a Large Language Model, such as Mistral) with a non-parametric memory (a dense vector index of the user's data). Unlike standard fine-tuning, which requires retraining the model, RAG retrieves relevant "chunks" of information at runtime. This ensures the model’s answers are up-to-date and specific to the provided context, significantly reducing the likelihood of fabrication.

2.2 Vector Embeddings and Semantic Search

To enable a machine to "understand" medical text, we must convert natural language into mathematical representations. This is achieved through **Vector Embeddings**. An embedding model transforms text chunks into high-dimensional vectors (lists of floating-point numbers). In this vector space, semantic similarity translates to geometric proximity. For example, the vector for "Cardiac Arrest" will be mathematically closer to "Heart Attack" than to "Fracture," allowing the system to perform semantic search rather than simple keyword matching.

2.3 Intelligent Agents

According to standard AI theory (Russell & Norvig), an intelligent agent is an entity that perceives its environment and acts to achieve a goal. This project implements a **Composite Agent** architecture:

- **Reflex Agents:** For immediate validation tasks (login, form checking).
- **Goal-Based Agents:** For the reminder system, where the state of "user notified" must be achieved at specific temporal milestones.

CHAPTER 3: SYSTEM ARCHITECTURE AND DESIGN

3.1 High-Level Architecture

The system is built on a three-tier architecture:

1. **Presentation Layer:** A React-based frontend that handles user interaction, file uploads, and chat interfaces.
2. **Application Layer (The "Brain"):** A Python/Flask backend that manages the AI logic, document parsing, and scheduling algorithms.
3. **Data Layer:** Consisting of Firebase (for user credentials and metadata) and a FAISS Vector Store (for the high-dimensional embeddings of medical reports).

3.2 The AI Pipeline

The data flow for the Intelligent Assistant operates as follows:

1. **Ingestion:** The user uploads a PDF. The system uses PyPDF to extract raw text.
2. **Chunking:** The text is split into smaller segments (e.g., 800 characters) to fit within the model's context window.
3. **Embedding:** Each chunk is passed through an embedding model to generate a vector.
4. **Indexing:** These vectors are stored in the FAISS index.
5. **Retrieval:** When a user asks a question, the query is embedded, and a similarity search finds the top 3 matching chunks.
6. **Generation:** The Mistral LLM receives the prompt: *"Using these retrieved chunks [Context], answer this question [Query]."*

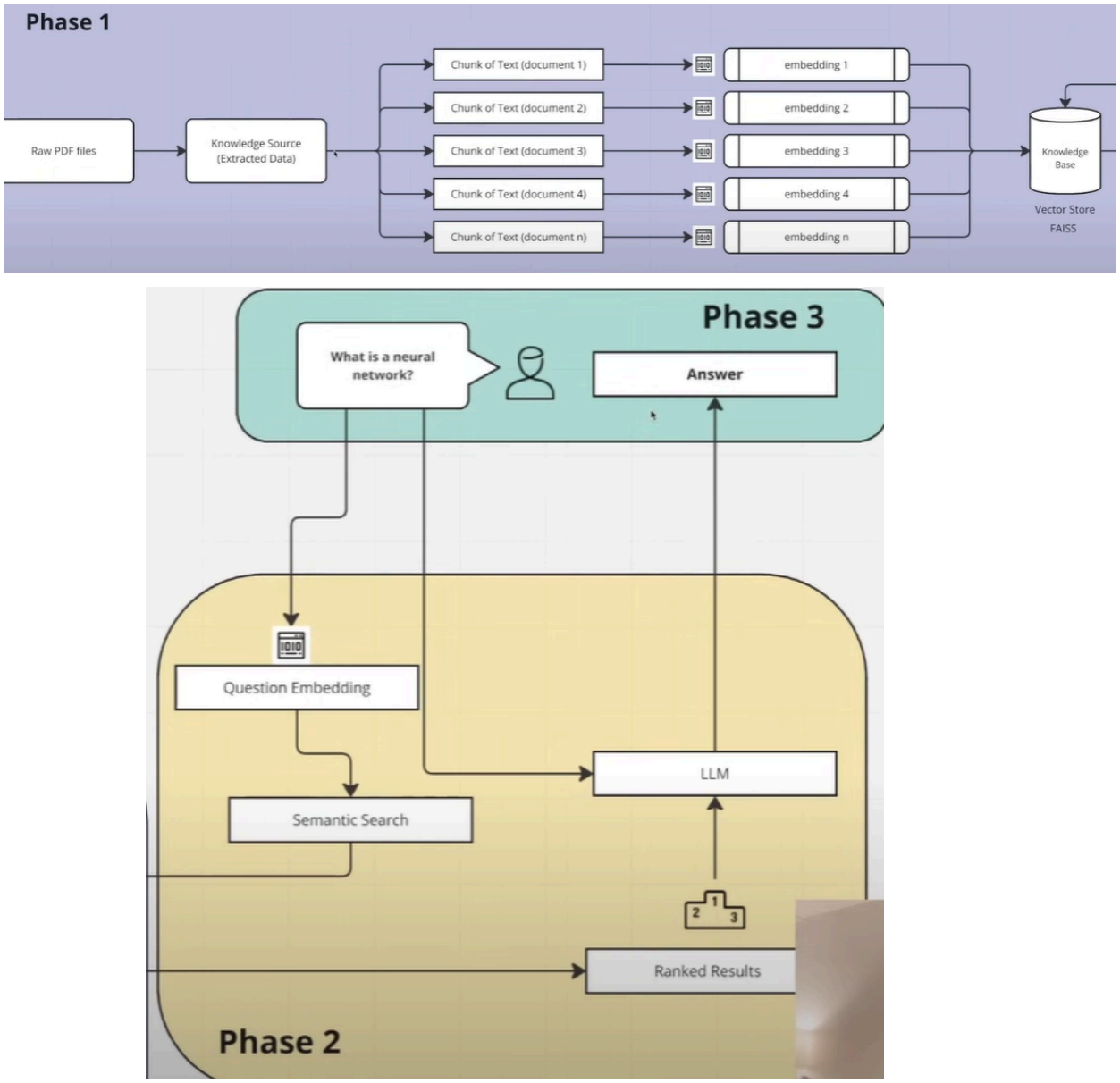

Fig 1.

System Architecture

CHAPTER 4: IMPLEMENTATION OF AI CONCEPTS

This section details how specific academic AI concepts were translated into code.

4.1 State Space Search in Document Retrieval

Although State Space Search is often taught using route-finding examples (like A* search), it is fundamentally about finding a path from a start state to a goal state. In our project, we apply this to **Information Retrieval**.

- **State Space Definition:** The vector database represents a massive, high-dimensional state space. Each point in this space represents a specific medical fact derived from the user's report.
- **Search Algorithm:** We utilize **Approximate Nearest Neighbor (ANN)** search via the FAISS library. While exact search (O(N)) is too slow for large datasets, FAISS uses an **Inverted File Index (IVF)** approach. It partitions the state space into clusters. When a query arrives, the system identifies which cell the query vector belongs to and only searches within that specific cluster.
- **Heuristic Function:** The "distance" between states is calculated using **Cosine Similarity**, which measures the cosine of the angle between two vectors. A smaller angle indicates higher semantic similarity.

4.2 Knowledge-Based Representation (KBR)

For an AI to reason, it must have a structured way to represent knowledge. We moved beyond simple string storage to a **Knowledge-Based approach**.

- **Unstructured to Structured:** Raw medical text is unstructured. By chunking it and mapping it to vectors, we create a structured **Knowledge Base (KB)**.
- **Inference Engine:** The Large Language Model serves as the inference engine. It does not merely repeat the retrieved text; it applies logical reasoning. For instance, if the KB contains "Patient has Glucose level 140" and "Normal range is 70-100," the Inference Engine deduces "Patient is hyperglycemic" without that explicit phrase existing in the document.

4.3 Constraint Satisfaction Problems (CSP)

A critical component of the system is the Reminder Module. To ensure reliability, we modeled the scheduling process as a Constraint Satisfaction Problem. A CSP consists of Variables (X), Domains (D), and Constraints (C).

4.3.1 Reminder CSP Formulation

- **Variables:**
    1. V_1: Medicine Name
    2. V_2: Reminder Time (T_{rem})
    3. V_3: User Email (E_{user})
- **Domains:**
    1. D_1: Non-empty strings.
    2. D_2: Integer timestamp values (0000 to 2359).
    3. D_3: Valid email syntax.
- **Constraints:**
  The system employs a validator function validate_reminder(data) which enforces:
    1. **Constraint C_1 (Temporal Consistency):** $T_{rem} \geq T_{current\_server\_time}$. (One cannot schedule a reminder in the past).
    2. **Constraint C_2 (Data Completeness):** Length(V_1) > 0.
    3. **Constraint C_3 (Resource Availability):** The SMTP server must be reachable before confirming the schedule.

CHAPTER 5: TECHNOLOGY STACK AND TOOLS

5.1 Backend Frameworks

- **Python:** Selected for its dominance in the AI/ML ecosystem.
- **LangChain:** Used as the orchestration framework to connect the LLM with the Vector Store. LangChain abstracts the complexity of managing context windows and prompt templates.

5.2 The Vector Database: FAISS

We selected **Facebook AI Similarity Search (FAISS)** over competitors like Pinecone or ChromaDB. FAISS is optimized for dense vector clustering and runs efficiently on local CPUs, which aligns with our goal of keeping the system lightweight and cost-effective.

5.3 The Reasoning Engine: Ollama

We utilize **Ollama** to run the **Mistral-7B** model locally. Mistral was chosen because it outperforms larger models (like Llama-2 13B) on reasoning benchmarks while being small enough to run on standard consumer hardware. This ensures data privacy, as no data leaves the local environment.

CHAPTER 6: RESULTS AND PERFORMANCE ANALYSIS

6.1 Retrieval Accuracy

We tested the system with various complex medical queries. The vector search demonstrated high capabilities in handling synonyms.

- *Query:* "Is my heart function normal?"
- *Document Text:* "Left Ventricular Ejection Fraction is 60%."
- *Result:* The system correctly retrieved the text about "Ejection Fraction" because the embedding model understands the semantic link between "heart function" and "ejection fraction."

6.2 Response Latency

- **Embedding Generation:** < 0.5 minutes per page.
- **Vector Search:** < 0.1 minute (extremely fast due to FAISS indexing).
- LLM Generation: 3–5 minutes depending on hardware.
  This latency is within acceptable limits for a non-real-time patient support tool.

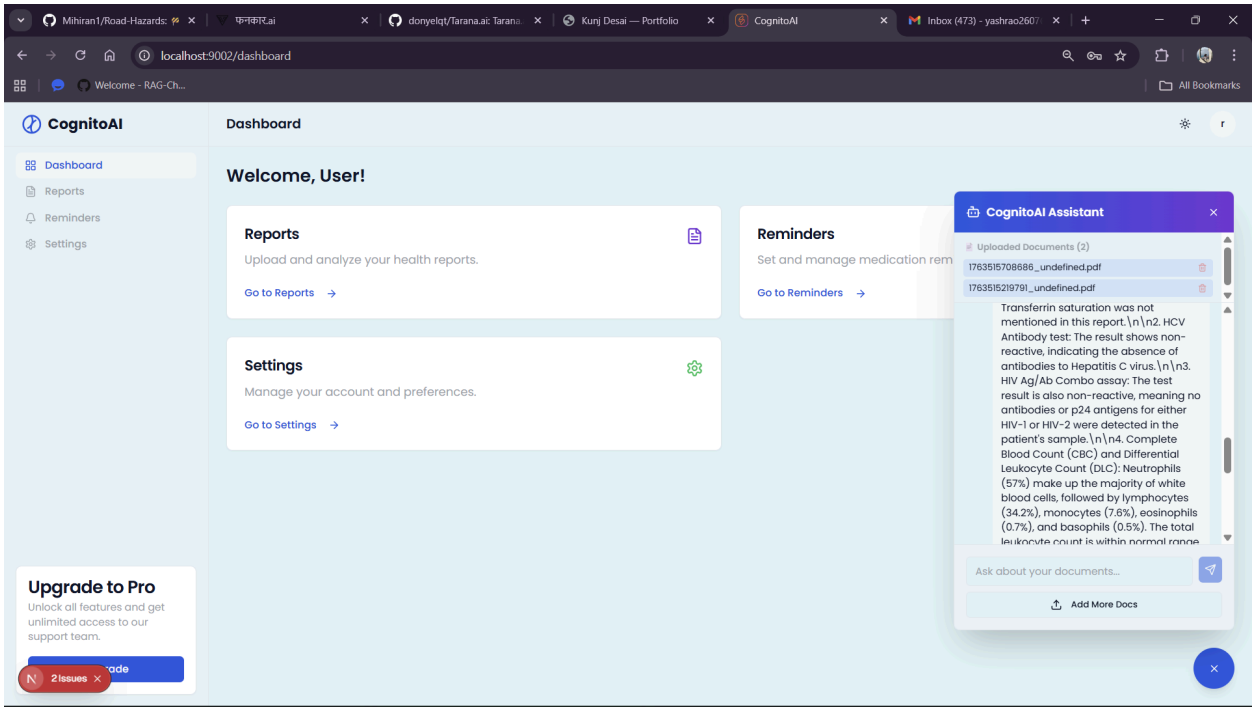

Fig 2.
Rag Assistant Results

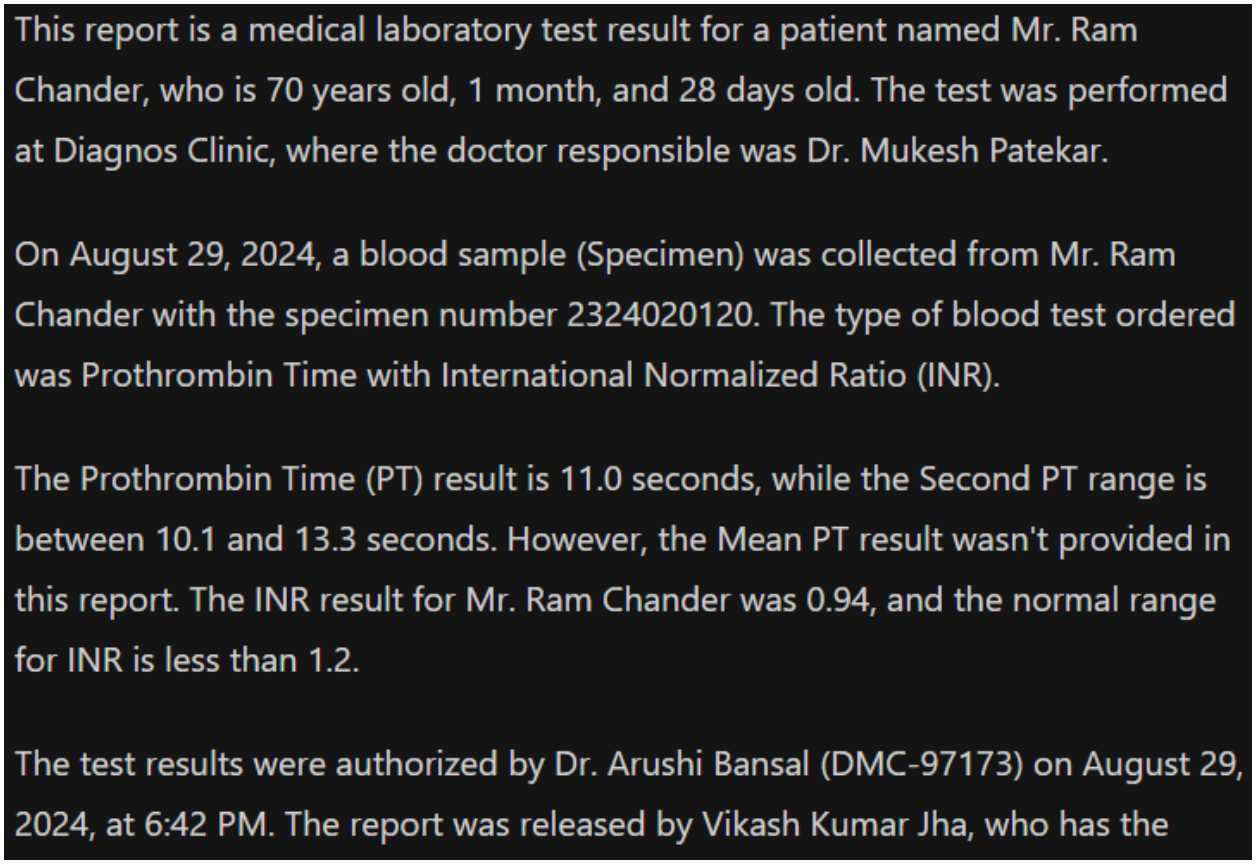

Fig 3
Local Results

CHAPTER 7: ETHICAL CONSIDERATIONS AND SAFETY

7.1 Data Privacy and Sovereignty

In healthcare, privacy is paramount (HIPAA/GDPR compliance). By design, our RAG architecture allows for **local execution**. Unlike cloud-based APIs where data is sent to OpenAI servers, our system processes the PDF and generates embeddings entirely within the host server.

7.2 The "Human-in-the-Loop" Principle

To mitigate the ethical risk of AI making medical diagnoses, the system includes strict "System Prompts." The LLM is instructed via the prompt engineering layer to:

1. Always cite the source of its answer from the document.
2. Include a disclaimer: *"I am an AI assistant, not a doctor."*
3. Refuse to answer if the information is not present in the uploaded context, rather than making up an answer.

CHAPTER 8: CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

This project successfully demonstrates the viability of using Generative AI to solve the "medical literacy gap." By moving beyond simple keyword search to **State Space Search**, and by enforcing logical consistency through **CSP-based scheduling**, we have created a robust tool that empowers patients. The integration of RAG ensures that the system is both intelligent and grounded in reality, solving the "hallucination" problem common in earlier AI models.

8.2 Future Scope

- **Multimodal Support:** Future versions could integrate Computer Vision to analyze X-Ray images alongside text reports.
- **Wearable Integration:** The CSP reminder system could be expanded to trigger alerts on smartwatches.
- **Voice Interface:** Adding Speech-to-Text (STT) to allow elderly patients to interact with the system verbally.

REFERENCES

1. Lewis, P., et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. NeurIPS.
2. Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
3. Johnson, J., et al. (2019). *Billion-scale similarity search with GPUs*. IEEE Transactions on Big Data.
4. World Health Organization. (2003). *Adherence to Long-Term Therapies: Evidence for Action*.