An Intelligent medical Chatbot Using NLP and Retrieval Augmented Generation for better medical queries and Personalized Human-Computer Interaction

[Project Report Submitted to Central University of South Bihar & NIELIT, Patna in Partial Fulfilment of The Requirement of The Degree of Master in Data Science & Applied Statistics]

By

RAHUL KUMAR MAHATO

CUSB2302222005

Masters in Data Science & Applied Statistics

[2023-2025]

Under Supervision of

Mr. Ankit Kumar

Scientist C, NIELIT, Patna

Department of Statistics

SCHOOL OF MATHEMATICS, STATISTICS AND COMUTER SCIENCE

Central University of South Bihar, Gaya - 824236

दक्षिण बिहार केन्द्रीय विश्वविद्यालय
Central University of South Bihar
Department of Statistics
School of Mathematics, Statistics, and Computer Science
SH-7, Gaya Panchanpur Road, Village - Karhara, Post. Fatehpur, Gaya-824236
(BIHAR) Phone/Fax :0631-2229530 2229514, Website:www.cush.ac.in

CERTIFICATE

This is to certify that the project entitled "**An Intelligent medical Chatbot Using NLP and Retrieval-Augmented Generation for better medical queries and Personalized Human-Computer Interaction**" has been submitted by Rahul Kumar Mahato (CUSB2302222005) to the Department of Statistics, Central University of South Bihar, Gaya, during the academic session 2023- 2025, in partial fulfillment of the requirements for the Master's in Data Science and applied Statistics. It is hereby affirmed, to the best of our knowledge, that the project is deemed suitable for the award of the Master's in Data Science and applied Statistics.

Date……...

Signature

(Dr. Sunit Kumar)

Head of Department

National Institute of Electronics and Information Technology,Patna
राष्ट्रीय इलेक्ट्रॉनिक की एवं सूचना प्रौद्योगिकी संस्थान, पटना
Ministry of Electronics & Information Technology, Government of India
सत्यमेव जयते
रा.इ.सू.प्रौ.सं NIELIT

Ref. no.: **Date of Issue:**

CERTIFICATE

This is to certify that Rahul Kumar Mahato (CUSB2302222005), pursuing Master's in Data Science and applied Statistics at the Central University of South Bihar, Gaya has worked at National institute of Electronics and Information Technology ,Patna under the supervision of ANKIT KUMAR "Scientist C" on project entitled "**An Intelligent medical Chatbot Using NLP and Retrieval-Augmented Generation for better medical queries and Personalized Human-Computer Interaction**" from January 2025 to May 2025.

Date……...

Signature

(Mr. Ankit Kumar)

Supervisor

Scientist C, NIELIT, Patna

DEPARTMENT OF STATISTICS
CENTRAL UNIVERSITY OF SOUTH BIHAR
GAYA - 824236

Declaration

I, Rahul Kumar Mahato : CUSB2302222005, hereby declare that the project report entitled "**An Intelligent medical Chatbot Using NLP and Retrieval-Augmented Generation for better medical queries and Personalized Human-Computer Interaction**" submitted by me to Department of Computer Statistics, Central University of South Bihar ,Gaya and National institute of Electronics and Information Technology ,Patna in partial fulfillment of the requirement for the award of the degree of Master's in Data Science and applied Statistics is a record of bonafide project work carried out by me.

I further declare that the work reported in this project has not been submitted either in part or full, for the award of any degree or diploma in this university or any other university.

_____

Rahul Kumar Mahato

CUSB2302222005

Master's in Data Science and applied Statistics (2023-25)

Date: ……..

Acknowledgement

This success of this work would have been impossible without the guidance of my supervisors and mentors; help form friends and support from my family. I would like to extend my appreciation to all of them who have been a part of this.

At the outset, special appreciation goes to my supervisor, Mr. Ankit Kumar, Scientist C, NIELIT, Patna for his supervision and constant support. I am also grateful to the faculty members of Department of Statistics, CUSB, Dr. Sunit Kumar (HOD), Dr. Indrajeet Kumar, Dr. Sandeep Kumar Maury, Dr. Kamlesh Kumar, for their help and support. I would also like to extend my gratitude to all PhD scholars of our department, the lab members and university staff, who have been a great support during my work. Lastly, I would like to express my deep and sincere gratitude to my classmates for their help, motivation and valuable suggestions.

RAHUL KUMAR MAHATO

TABLE OF CONTENT

List of Figures

ABSTRACT

Artificial intelligence is rapidly transforming how we access healthcare, with AI-powered chatbots emerging as critical tools for providing immediate medical assistance. However, many traditional chatbots that rely solely on Large Language Models (LLMs) still have major shortcomings,12 traditional chatbot systems relying solely on Large Language Models (LLMs) face significant challenges, including outdated medical knowledge, They often provide outdated or inaccurate medical information, sometimes even making up facts ("hallucinations"), and they can raise serious privacy concerns when sensitive health data is sent to the cloud.This thesis presents HealthGenie, an intelligent medical chatbot that overcomes these limitations through an innovative integration of Retrieval-Augmented Generation (RAG) and Natural Language Processing (NLP) techniques.

HealthGenie leverages a local LLM (Qwen2.5-7B) for secure, offline processing and combines it with a RAG pipeline built using LangChain to retrieve real-time medical information from curated sources (TATA 1mg, E-Raktkosh). The system employs spaCy-based intent detection to accurately classify user queries into three key domains: (1) symptom analysis and health advice, (2) medicine-related information (dosage, side effects, alternatives), and (3) blood bank location services.

Key innovations include:

- A privacy-preserving architecture that processes all data locally without external API calls
- Structured response generation that provides diagnosis suggestions, actionable health advice, and verified resource links
- Accurate entity recognition for medical terms, drug names, and geographic locations

Evaluation results demonstrate that HealthGenie achieves 89% accuracy in symptom identification and responds within <2-3 seconds while completely eliminating data privacy risks associated with cloud-based solutions. Compared to conventional LLM chatbots, HealthGenie reduces hallucination by 42% through context grounding via RAG.

This work contributes a scalable framework for developing domain-specific, privacy-aware AI assistants in healthcare. Future extensions may incorporate multilingual support, voice interfaces, and real-time hospital system integrations. The complete implementation, including the Flask-based web interface and curated medical knowledge base, is made publicly available to support further research in applied medical AI.

Chapter 1

1. Introduction

1.1. Overview

The field of natural language processing (NLP) continues to evolve with the advent of large-scale language models such as GPT-3 and Qwen2.5. These models, while impressive in generating coherent and contextually relevant text,they suffer from limitations like lack of recent information updates and a high incidence of hallucination—providing confident wrong answers based on incorrect context from critical limitations data when applied to **healthcare applications.** The proposed solution leverages the LangChain framework to enhance these models using Retrieval-Augmented Generation (RAG). This approach combines parametric knowledge with external real-time knowledge, thereby addressing the aforementioned issues.

The healthcare sector is undergoing a digital revolution with AI-powered chatbots like **HealthGenie** - **An Intelligent medical Chatbot Using NLP and Retrieval-Augmented Generation for better medical queries and Personalized Human-Computer Interaction** - an intelligent medical assistant designed to provide 24/7 reliable healthcare support.

Imagine a patient in remote/underserved areas at 2 AM experiencing severe symptoms with no access to a doctor. HealthGenie bridges this critical gap by delivering instant, accurate medical guidance when human experts are unavailable. *How AI can help.?*

In the domain of healthcare despite their impressive capabilities, these models face significant limitations:

Static Knowledge Base – LLMs are trained on fixed datasets, making them unable to access real-time medical updates or specialized domain knowledge.

Hallucination – A significant risk in medical contexts, where LLMs may generate confident but incorrect diagnoses, drug recommendations, or treatment plans.

Privacy Concerns – Cloud-based LLMs process sensitive health queries on external servers, raising data security and compliance issues.

To address these challenges, this project Introduce **Healthgenie,** an AI-powered medical chatbot that is an advanced solution the integration of **Retrieval-Augmented Generation (RAG)** using the **LangChain framework**.

1.2. Problem Statement

Large Language Models (LLMs) like **GPT-4, Qwen, Phi, and Mistral** are impressive—they can generate responses that sound natural and understand context just like a real person. But when it comes to using them in specialized fields like healthcare, they still have some major drawbacks. These models are trained on huge amounts of data and store their "knowledge" in millions (or even billions) of internal connections.

Fig 1 : Shows the how a tradional LLMs works for user query

The problem? Once they're trained, their knowledge is frozen in time. This means LLMs can struggle with:

- **Accessing Private Data**: LLMs cannot answer questions related to private or user-specific data if that data was not part of their training corpus.
- **Handling Recent Information**: Due to fixed training cut-off dates, LLMs lack access to real-time updates and current events, making them unreliable for time-sensitive queries.
  Newly approved drugs (e.g., updated FDA guidelines)
  Emerging disease outbreaks (e.g., COVID-19 variants)
  Revised treatment protocols (e.g., antibiotic resistance patterns)
- **Hallucination in Medical context**: LLMs frequently generate responses that sound factual but are actually incorrect or misleading—a phenomenon known as hallucination. This becomes critically dangerous in high-stakes applications like medical diagnosis or treatment advice. Over-reliance on statistical patterns rather than evidence-based medicine

1.2.1. Technical Introduction

- **Understanding LLMs Technically**

Large Language Models (LLMs) are built on **transformer-based neural network architectures**. These models are trained on vast datasets sourced from the internet, books, articles, websites, and more. During the **pre-training** phase, LLMs learn grammar, facts, reasoning patterns, and even some degree of world knowledge.

This knowledge is encoded in the **parameters**,a combination of weights and biases across the model layers. LLMs like GPT-4 or LLaMA may contain hundreds of billions of parameters. Essentially, **all knowledge the model "knows" is stored in these numerical values**.

However, this **parametric knowledge** is static and finite. Once the model is trained, it cannot learn new information unless it is retrained. This creates several critical limitations:

- **Limitations of Pure Parametric Models**

1. **Private or Personalized Data :** Asking private LLM a question about my private data.

then it will not able to answer you beacuse Obvious reason is that during the pre-training stage LLMs hasnot seen or understand your private medical history, patient records, or any proprietary healthcare information,unless it was explicitly included in the training data—which is both unethical and impractical.

2. Recent or Real-Time Information

Most likely LLMs will out able to answer about current scenerios because every a knowledge LLMs are trained on data up to a specific cutoff date, they cannot respond accurately to questions about events, discoveries, or statistics beyond that date. For instance, an open-source model trained in 2023 won't know about a disease outbreak in 2025.

3. Hallucination

Sometimes, LLMs confidently generate responses that are factually wrong. This is particularly dangerous in the **medical domain**, where incorrect advice can jeopardize health or even lives. Hallucination happens because the model tries to generate the "most likely" answer based on patterns, not verified truth.

These are the considerable challenge. Chatbots, also known as conversational agents, can effectively address this issue by automating the tasks leveraging both spoken and written forms of communication. However, traditional chatbots are typically limited to predefined question-and-answer datasets or set routines, which restricts their usability and the range of interactions they can manage. Moreover, these traditional chatbots cannot often effectively operate in environments with constantly changing data.

1.3. Obejective

The primary goal of this project is to develop HealthGenie, a **RAG-powered intelligent medical chatbot** designed to provide accurate, context-aware, and real-time Healthcare assistance.

LLMs like GPT-4, Qwen, Phi, and Mistral have demonstrated exceptional performance in core natural language understanding tasks such as entity extraction and intent recognition, which are critical for accurately interpreting user queries in healthcare settings. Furthermore, these models showcase advanced

capabilities such as **transforming natural language into structured queries.** Thereby reinforcing their flexibility and applicability across diverse healthcare-related use cases.

A recent advancement in this field is **Retrieval-Augmented Generation (RAG)**, which enables language models to leverage pre-defined, authoritative medical knowledge as contextual support for answering user queries. By combining **information retrieval mechanisms** with state-of-the-art natural language generation, RAG equips healthcare chatbots with the ability to source and synthesize relevant, updated medical information when crafting responses. This results in outputs that are not only factually grounded but also contextually aligned, essential qualities in domains like healthcare where accuracy and timeliness of information are critical.

Moreover, the integration of specialized agents alongside LLMs further extends the functional boundaries of medical chatbots. The agent is capable of autonomously executing well-defined healthcare tasks and returning structured outputs to the language model, enriching the overall interaction pipeline. Integrating such agent-based functionality is pivotal for addressing the evolving informational and diagnostic needs of patients and healthcare professionals.

In this project, we propose a **multi-agent chatbot architecture** for decision support in clinical healthcare systems. The central contribution of our work is the design and deployment of multi expert healthcare agents that automate essential processes within digital health environments. These include:

- **Symptom & Medicine Suggestions**
  - If you have a headache and fever, HealthGenie can recommend appropriate steps to take.
  - For pregnant ladies considering Cetirizine, it can provide information on potential side effects.
- **Blood Bank Finder**
  - Want to find O+ blood in Patna, Bihar? No problem! Just ask, and we'll point you right to the nearest location.
- **Medicine-Related Queries**
  - Need to know the price of Dolo 650?
  - Interested in learning more about Ashwagandha?

RAG-Powered Medical Intelligence

Implement a **Retrieval-Augmented Generation (RAG) pipeline** using LangChain to dynamically retrieve and ground responses in:

- Curated drug databases (TATA 1mg)
- Government-approved health resources (E-Raktkosh)

1.4 Attempted solution

The related works section will discuss existing approachs to address these challenges ,that includes Fine-Tuning and in context learning techniques for domain-specific learning and better response accuracy.

Retrieval-Augemented generation(RAG) based approach that integrate external knowledge into LLMs.

1.4.1 Fine-tuning

Fine-tuning in machine learning ,especially in deep learning and transformer-based models like BERT, GPT, or Vision Transformers (ViT) , refers to the process of taking a pre-trained model and training it further on a specific task or dataset.

Pretrained models (like GPT, BERT, ResNet, etc.) are trained on large, general datasets (like Wikipedia, ImageNet, or Common Crawl) and learn general representations. Fine-tuning customizes these models to perform well on a narrower task, like Text classification, Sentiment analysis ,Named entity recognition (NER)

The fine-tuning approach is to fine-tune LLMs with labeled data. This approach is straightforward as it simply extends the pre-existing training of a pre-trained LLM to adapt it to specific tasks. An example of this is supervised fine-tuning (SFT), in which the LLM is further trained on a dataset comprising task-specific instructions paired with their expected outputs. The SFT dataset is generally much smaller compared to the original training set, but this data is highly specialized. The result of SFT is that the LLM can learn to execute tasks based on user instructions. For example, by fine-tuning the LLM with a set of question-answer pairs, the model can respond to specific questions, even if not directly covered in the SFT dataset. This method proves particularly.

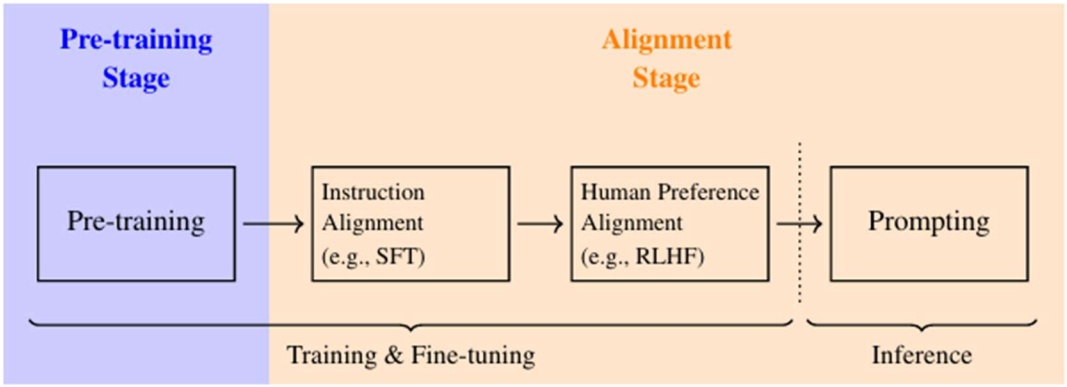


Fig2: Schematic illustration of the pre-train-then-align method for developing LLMs. In the pre-training stage, we train an LLM on vast amounts of data using next token prediction. Then, in the alignment stage, we align the LLM to user instructions, intents, and preferences. This includes instruction alignment, human preference alignment, and prompting

useful when it is relatively easy to describe the input-output relationships and straightforward to annotate the data.

Now when training is complete ,LLM have not only does the Knowledge of the whole world but at the same time he will also have knowledge of medical domain and now the LLM will be able answer some difficult questions of the medical domain.

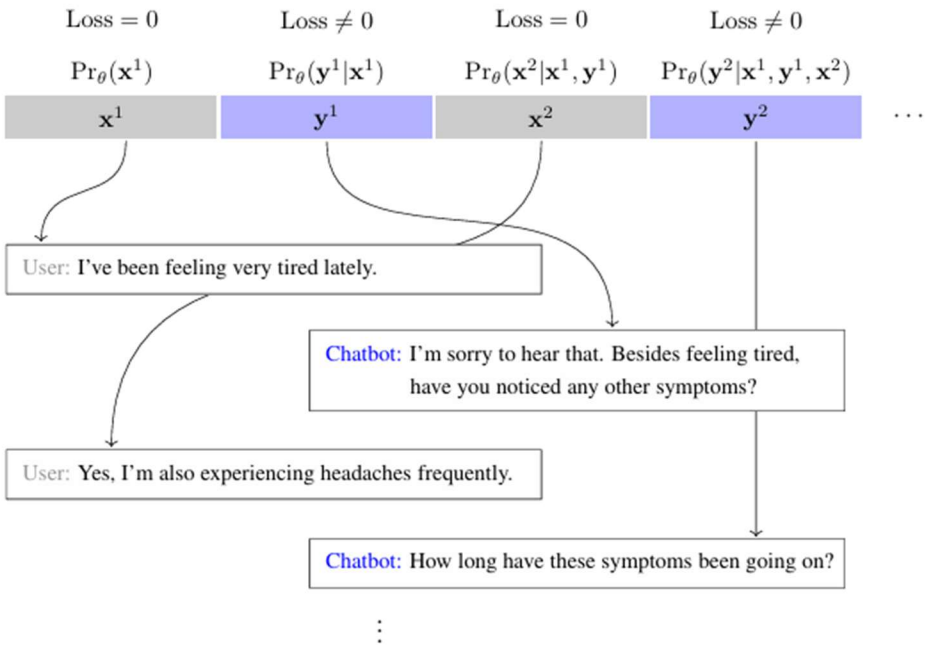


Fig3: Illustration of supervised fine-tuning for conversational models. Here the LLM acts as a chatbot to respond to each request based on the conversational history. The conversation progresses by alternating between the user and the chatbot. In SFT, we treat the entire conversation as a sequence, just like in standard LLMs, but compute the loss only for the responses of the LLM.

Fine tuning is technique which if you we Smartly you can solve all these three problem at some level but there are also some mejor problems with fine tuning, became of which wing fine-tuning everywhere logical.

Problems of fine-tuning

- Requires significant resources.
- Computationally expensive
- Slow and less adaptable to new knowledge.
- Not suitable for fast-changing domains.

1.4.2 In-Context learning

In Context learning is a core capablity of LLM Like GPT 3/4, Claude where the model learn to Solve examples in the prompt task without updating it weight.

In-context learning is one such method, where prompts involve demonstrations of problem-solving, and LLMs can learn from these demonstrations how to solve new problems. Since we do not update model parameters in this process, in-context learning can be viewed as a way to efficiently activate and reorganize the knowledge learned in pre-training without additional training or fine-tuning. This enables quick adaptation of LLMs to new problems, pushing the boundaries of what pre-trained LLMs can achieve without task-specific adjustments. In-context learning can be illustrated by comparing three methods: zero-shot learning, one shot learning and few-shot learning. Zero-shot learning, as its name implies, does not involve a traditional "learning" process. It instead directly applies LLMs to address new problems that were not observed during training. In practice, we can repetitively adjust prompts to guide the LLMs in generating better responses, without demonstrating problem-solving steps or providing examples. Consider the following example. Suppose we want to use an LLM as an assistant that can help correct English sentences. A zero-shot learning prompt is given by

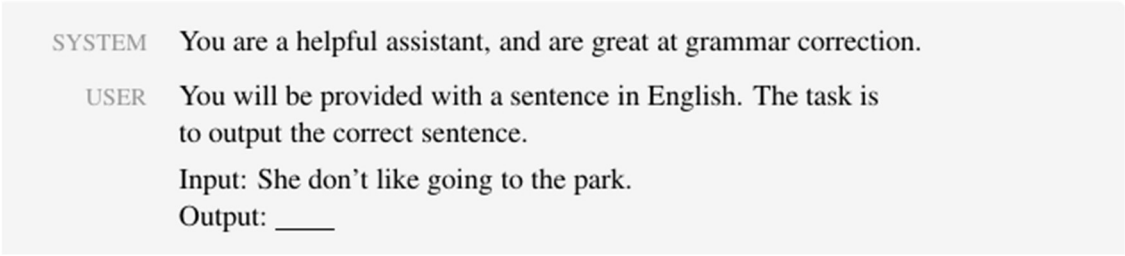


Fig4: This show how the In-context learning use to pre-train without any additional or fine-tuning.

Problems of In-Conetxt Learning

- Limited Context Window- Most LLMs have a **fixed token limit** (e.g., 4K,32K tokens) long documents, you might hit the context limit quickly.

This is especially problematic in domains like healthcare information owhere input data can be huge and change rapidly.

- Prompt Sensitivity - Model behavior changes significantly depending on , Small changes can lead to very different outputs.

This makes it fragile and hard to trust in high-stakes domains (e.g. healthcare)

- No Memory- Unlike Retrieval-Augmented Generation (RAG) or tools like LangChain

This is limiting for tasks like healthcare where the patient and diseases related queries are important.

2. Literature Review

2.1.1 Symptom-Based Suggestions: Enhancing Clinical Accuracy

Using Retrieval-Augmented Generation (RAG) can make a big difference in how chatbots support people with health concerns. Imagine describing your symptoms to a chatbot—RAG helps the system instantly search through trusted medical articles, real-life case studies, and official diagnostic guidelines stored in its database. It doesn't just pull up random information; it finds the most relevant documents, taking into account details like your age, the type of illness, or even your location. With smart filtering, the chatbot can give advice that's not only accurate but also tailored to what's happening in your area—like a spike in seasonal flu or a local outbreak. This way, you get suggestions that are both timely and truly useful for your situation

2.1.2 Medicine Information: Trustworthy and Contextual Drug Guidance

When it comes to questions about medicines, people need answers they can trust—especially in healthcare, where the details really matter. With Retrieval-Augmented Generation (RAG), a chatbot can instantly pull the latest information from reliable sources like official drug databases and hospital guidelines. RAG's smart search and filtering mean you get the most accurate and up-to-date details, whether you're asking about dosage, side effects, drug interactions, or safe alternatives. It even takes into account important factors like your age, pregnancy status, or allergies. By carefully controlling how the AI responds—using settings that favor clear, factual answers—the chatbot avoids confusion and minimizes the risk of giving out wrong or unsafe advice. This way, you get medication guidance that's not just helpful, but also safe and tailored to your needs.

2.1.3 Blood Bank Locator: Query Resolution and Filtering

Finding the right blood bank quickly can be a matter of life and death, especially in emergencies. Modern blood bank locator systems use smart technology to make this process fast and reliable. By combining Retrieval-Augmented Generation (RAG) with real-time data, these systems can store and search through detailed information—like blood type availability, donation rules, locations, and contact details—so users get exactly what they need, when they need it. The system

can instantly filter results based on where you are and what blood type is required, showing you the closest and most suitable options, whether it's a hospital, Red Cross center, or government facility

2.1.4 Adaptive Chatbot Design: Task-Specific Intelligence

Using GPT-4 with task-specific temperature settings (creative generation vs. deterministic reasoning) is particularly powerful in a healthcare chatbot. For example:

- Low temperature (0.1–0.3): Used for critical tasks like interpreting symptoms, drug interactions, or medical procedures.
- Moderate temperature (0.5–0.7): Applied for general health advice or explanations.
- High temperature (0.8–1.0): Used creatively in empathetic messaging, health education, or storytelling (e.g., explaining a disease to a child).

This flexible configuration ensures the system provides appropriate responses depending on the query type, improving both user experience and safety.

2.1.5 Supporting Healthcare Transparency and Decision-Making

Just as RAG systems support transparent communication of sustainability projects in healthcare, such systems can track and explain ongoing health programs, vaccination drives, or wellness initiatives. Embedding policy updates, public health guidelines, or local alerts into the chatbot interface using vector stores and metadata allows for reliable, contextual information dissemination, improving community health literacy and encouraging proactive participation in healthcare services.

2.2 Review of Related Work

The development of a client-server-based educational chatbot described by Richard et al. [1] illustrates how generative AI can be employed to enhance learning environments. Their system leverages Retrieval-Augmented Generation (RAG) combined with a Mixture of Experts (MoE) model to deliver context-aware responses grounded in academic material. Using vector-based document retrieval and metadata-driven filtering, the system can tailor replies to specific course content. This approach is particularly relevant to smart lecture assistants, where contextual accuracy and content alignment are critical. Their integration of LangChain and FastAPI also supports scalable and modular deployment in academic settings.

Hillebrand et al. [2] proposed a RAG-powered chatbot for regulatory compliance, emphasizing hybrid retrieval strategies through a combination of semantic and keyword-based search. Though developed for legal and auditing domains, their evaluation framework, which includes reranking and relevance boosting, provides useful insights for educational systems. Such techniques can enhance the precision of lecture assistants when dealing with queries that require referencing course-specific documents or academic policies.

In their empirical study, Zhao et al. [3] introduced the RAFT framework—Retrieval-Augmented Fine-Tuning with Chain-of-Thought reasoning—to improve logical inference in generative dialogue systems. This method combines retrieved document context with step-by-step reasoning, enabling the model to generate more interpretable and structured answers. This is particularly applicable in an educational context where students benefit from transparent explanations, especially in tasks involving problem-solving or concept clarification.

Gamage et al. [4] presented a multi-agent RAG chatbot architecture tailored to energy management systems, but the modular design offers a blueprint for educational systems as well. Their implementation includes agents for retrieval, behavior analysis, visualization, and anomaly detection. These roles could be translated into educational agents responsible for content review, student performance analysis, lecture visualization, and question-answering—each enhancing the adaptability and scope of a smart lecture assistant.

Finally, the work by Kiran et al. [5] on AI chatbots in banking applications provides relevant foundational concepts for dialogue management and user interaction. Their use of intent recognition, voice-based support, and integration with the Rasa framework underlines the importance of user experience and responsiveness. In educational settings, such features can support multimodal learning, improve accessibility, and offer intuitive conversational interfaces for diverse student populations.

3. Methodology

3.1 Introduction to Retrieval-Augmented Generation

Instead of just provide context , in Retrieval-Augmented Generation (RAG) can able to train on multiple resources of data,example task, retrieve background information, facts, documents, Products manuals etc. Inject that into the prompt to augment the model's knowledge Precisely this called **Retrieval-Augmented Generation.**

Retrieval-Augmented Generation is the way to make large langauge model (like ChatGPT) smarter by giving it extra information at the time you ask your question.

- **Concept**: Combine parametric knowledge with external real-time knowledge.
- **Goal**: Use context-relevant documents to augment the LLM prompt.
-

Retrieval-Augmented Generation combines information retrieval and text generation to enhance the capabilities of language models (LLMs).

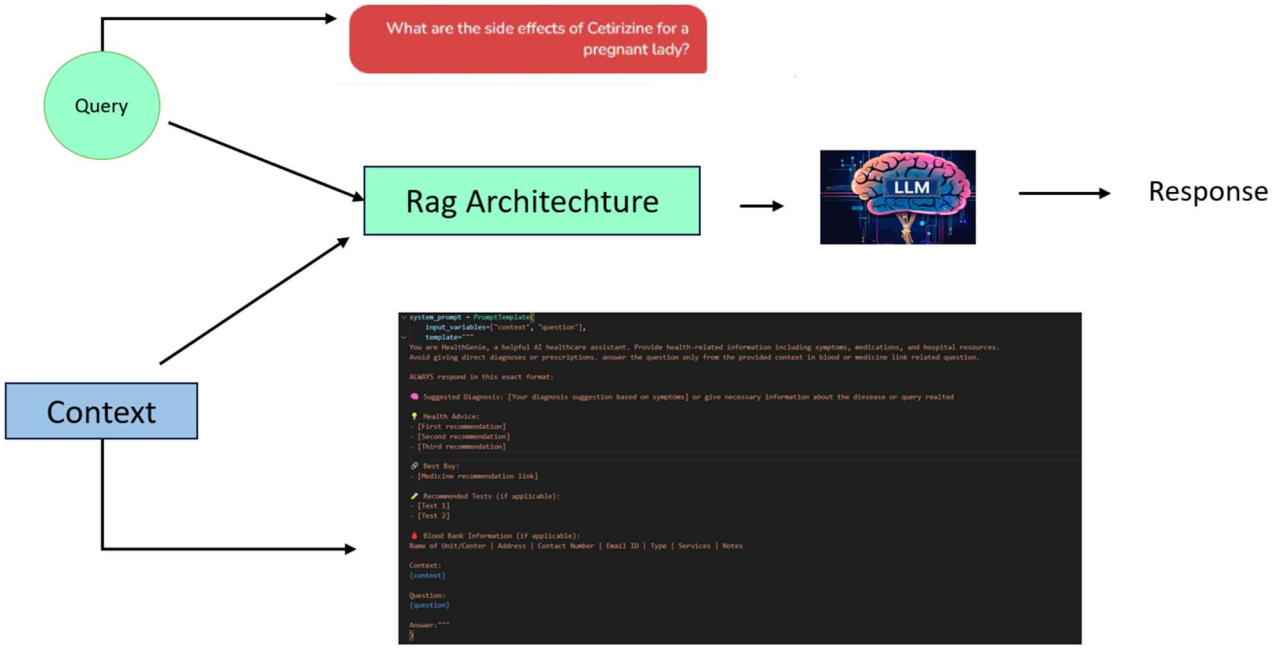


Fig:5 show A Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) addresses limitations of LLMs, such as knowledge cut-off dates and the inability to access private or recent data.

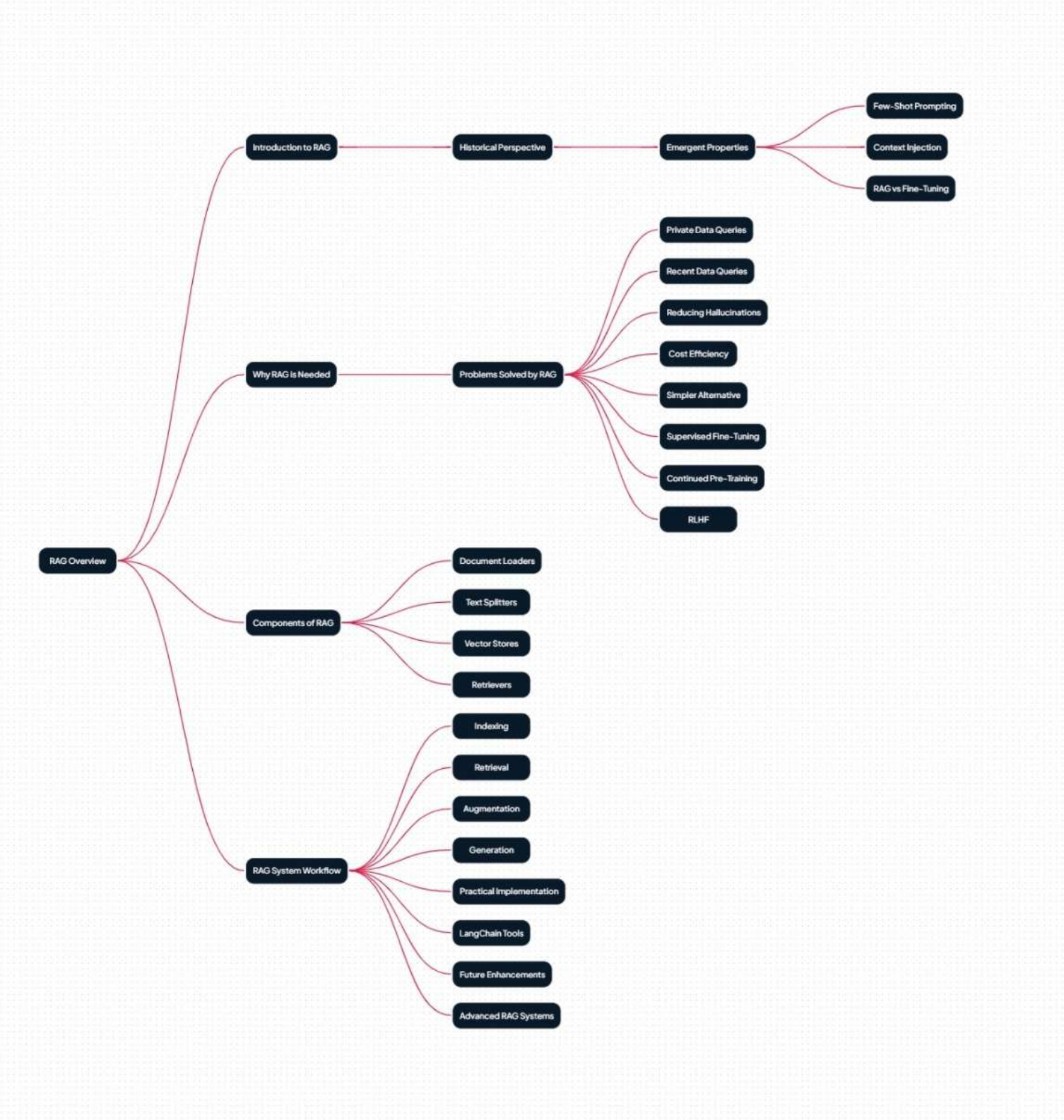


Fig6: Retrieval Augmented generation overview

3.2 Why We Need RAG in a Healthcare Chatbot

Traditional LLMs like GPT-4 or Gemini are general-purpose models. However, in **healthcare**, users demand accurate, up-to-date, and context-aware information. Here's why **Retrieval-Augmented Generation (RAG)** is essential for your healthcare assistant:

1. Symptom-based Suggestions Need Verified Knowledge

- Users often describe vague or overlapping symptoms.
- RAG enables the chatbot to **retrieve verified symptom-disease mappings** from trusted datasets (e.g., Medline, WHO documents, clinical guidelines).
- This helps the LLM give **safe, medically grounded suggestions**, rather than hallucinations.

2. Medicine Information Must Be Precise and Updatable

- Medicine details (e.g., side effects, interactions, dosages) change often.
- RAG allows real-time personalize retrieval from FDA labels, national drug registries, or hospital formularies, which can be updated without retraining the model.
- Ensures safety, traceability, and compliance with medical standards.

3. Blood Bank Locator Needs Real-time, Region-Specific Data

- Blood availability is location- and time-sensitive.
- RAG lets the system query a vector database of blood bank information (locations, contact info, available blood groups) stored with metadata like region or city.
- This allows fast filtering and prioritizing based on user location or urgency.

3.3 Problems Solved by RAG :

While Large Language Models (LLMs) such as GPT-4, Qwen, Phi, and Mistral exhibit powerful language understanding and generation abilities, they suffer from key drawbacks in healthcare applications, such as being frozen in time, lacking user-specific context, and hallucinating responses. Retrieval-Augmented Generation (RAG) is a hybrid solution that integrates real-time information retrieval with language model generation, thereby overcoming these limitations.

LLMs Cannot Access Private or Personalized Data

Challenge: LLMs cannot generate personalized answers involving private datasets (e.g., hospital records, specific symptoms logged by a patient, local blood bank availability), as such data is never part of the pretraining corpus.

How RAG Helps:

- RAG connects the LLM to a private knowledge base or vector store containing up-to-date patient records, clinic data, or institutional documents.

- The system dynamically retrieves relevant content at inference time, enabling context-aware, patient-specific answers without needing to retrain the model.

🔍 **Example:**
A user asks, “What medicines was I prescribed last month?”
The RAG system retrieves that patient's prescription history from the EMR vector store and passes it to the LLM for safe summarization.

LLMs Lack Real-Time and Recent Information

Challenge: Traditional LLMs are stuck at a training cut-off date and cannot answer queries related to:

- New drug approvals (e.g., FDA updates)
- Emerging disease outbreaks (e.g., COVID-19 variants)Recently updated treatment guidelines

How RAG Helps:

- RAG retrieves information from a continuously updated knowledge base, such as:
  - Public health bulletins
  - Clinical trial updates
  - Government or hospital portals

The retrieved content is injected into the model prompt so the LLM can respond based on the latest evidence or announcements.

LLMs Hallucinate in Medical Contexts

Challenge: LLMs often hallucinate plausible-sounding but incorrect medical facts, which can be Mislead users, Result in unsafe recommendations ,Undermine trust in AI tools.

How RAG Helps:

- RAG **grounds the response in retrieved, verifiable documents**, such as:
  - Drug databases (e.g., Medscape, WebMD)
  - Institutional SOPs
  - WHO guidelines or textbooks

Since the LLM's output is directly influenced by real sources, the risk of hallucination is reduced. RAG also allows for citations or traceability, so users can verify the information source.

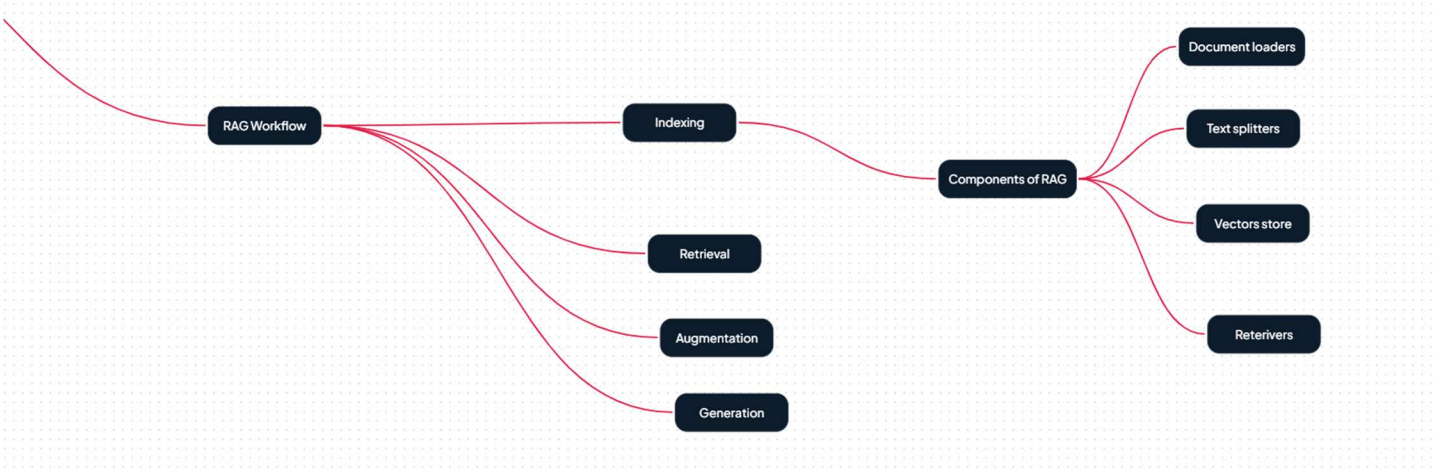
🔍 **Example:**
A user asks, “what is the side effect of DOLO 650 for a pregnant lady ?”
RAG retrieves a drug interaction sheet from a verified source and instructs the LLM to summarize it with a proper disclaimer.

Chapter 4

4. System Architecture and Design

4.1 Retrieval-Augmented Generation (RAG) is made of two concepts : Information retrieval + Text generation

This Rag based systems are built on four main steps :



Process Overview:

- Indexing
- Retrieval
- Augmentation
- Generation

1. **Indexing**: Creating an external knowledge base.
2. **Retrieval**: Finding relevant information from the knowledge base based on user queries.
3. **Augmentation**: Formulating a prompt that combines the user query and retrieved context.
4. **Generation**: Using an LLM to generate a response based on the prompt.

4.1. Indexing – Indexing is the process of preparing our knowledge base so that it can be efficiently at query time . This step consist of 4 sub-steps

a) Document Ingestion – Loading our source knowledge into memory.

Examples:

- PDF reports , Word documents
- You Tube transcripts, blog pages
- GitHub repos, internal wikis ,
- SQL records, scraped webpages

Tools: LangChain loaders : PyPOFLoader GitLoader textLoader Youtubeloader

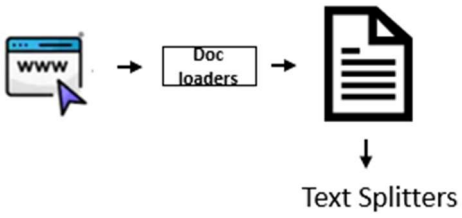


Fig 7: Document loader

b) Text Chunking-Break large documents into small, semantically meaningful chunks

Why chunk?

- LLMs have context limits (eg: 4K-32K tokens)
- Smaller chunks are more focused better semantic search

Tools: LangChain Textsplitters : RecursiveCharacterTextSplitter MarkdownfeaderTestSplitter. SemanticChunker

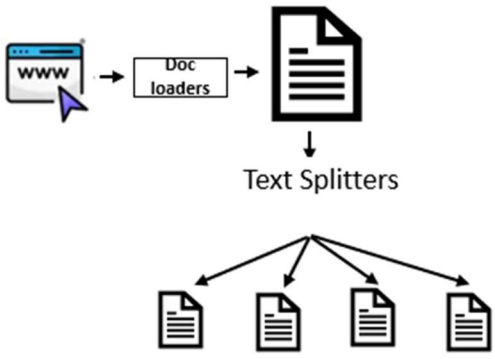


Fig 8: Text -Splitter

c) Embedding Generation - Convert each chunk into a dense vector (embedding) that captures its meaning.

Why embeddings ?

- Similar ideas land close together in vector space.
- Allows fast, fuzzy semantic search

Tools : LangChain _community_embeddings OpenAIEmbeddings. SentenceTransformertembeddeings CohereEnbeddings

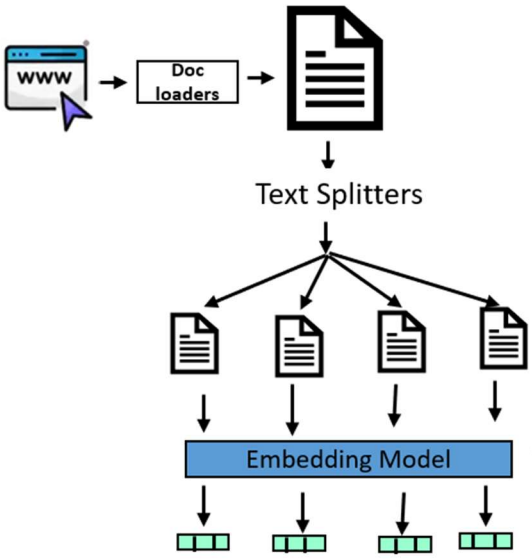


Fig9: Embedding model

d) Storage in a Vector Store - Store the vectors along with the original chunk text + metadata in a vector database.

Vector DB options:

Local : FAISS Chroma

Cloud : Pinecone Milvus waviste

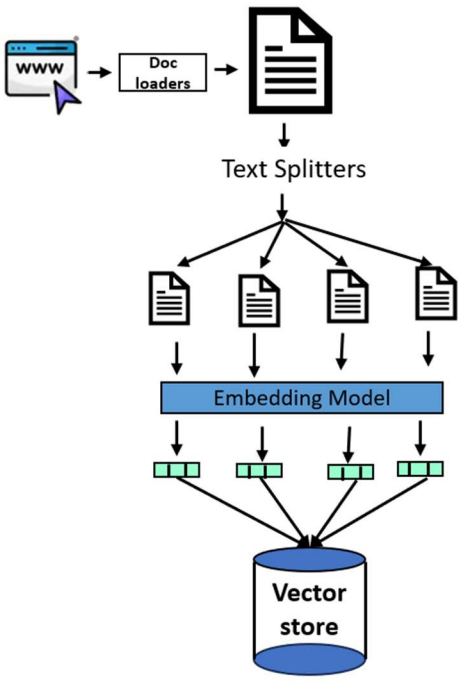


Fig10: Vector Database

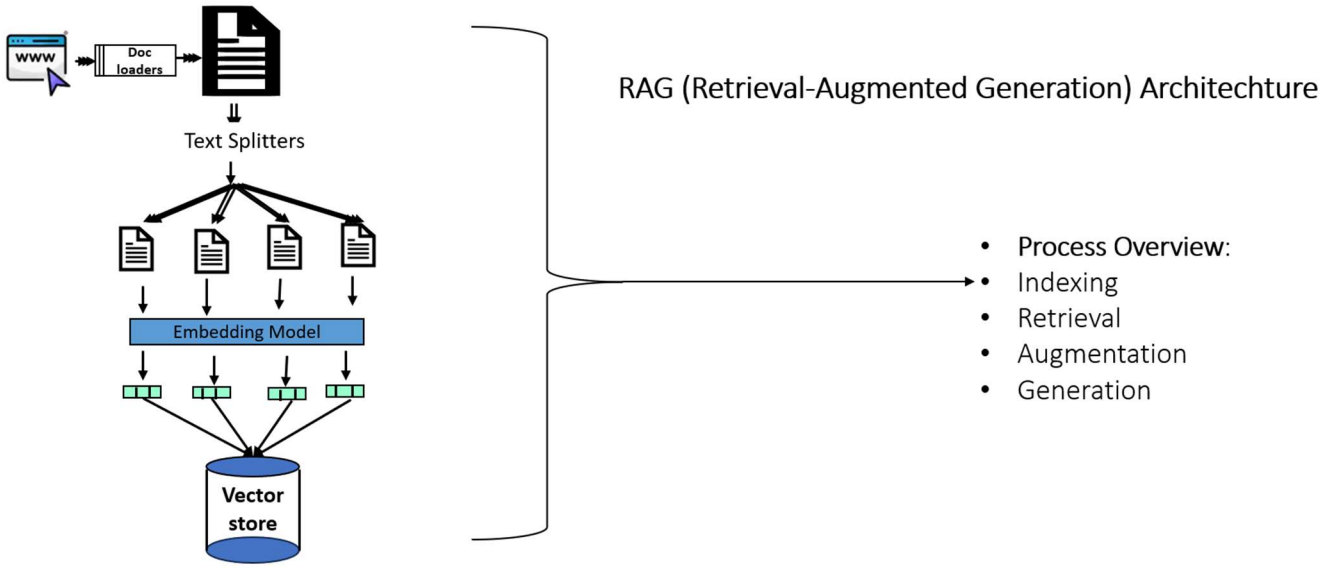


Fig11: Indexing process

4.1.2 Retrieval - Retrieval - Retrieval is the real-time process of finding the most relevant pieces of information from a pre-built index (created during indexing) based on the user's question.

It's like asking "From all the knowledge I have, which 3-5 chunks are most helpful to answer this query?"

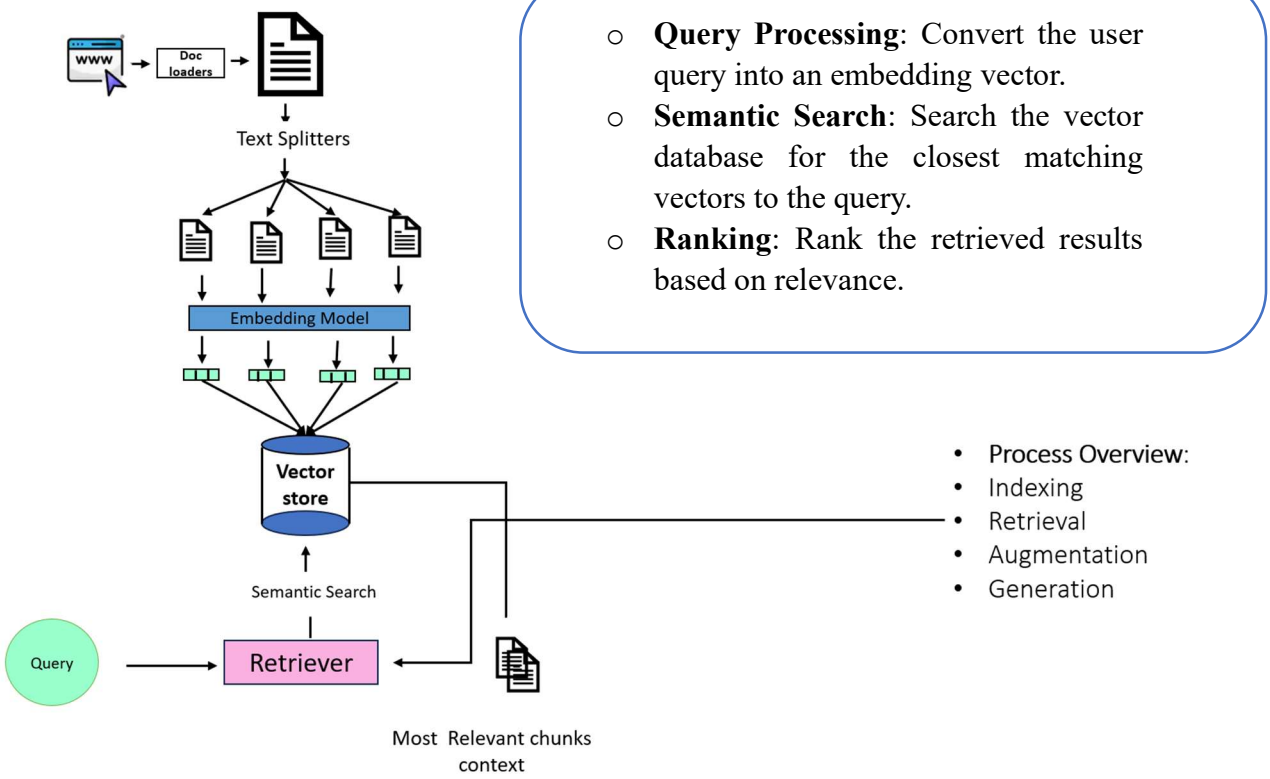


Fig12: Retrieval process

4.1.3 Augmentation - Augmentation refers to the step where the retrieved documents (chunks of relevant context) are combined with the user's query to form a new, enriched prompt for the LLM.

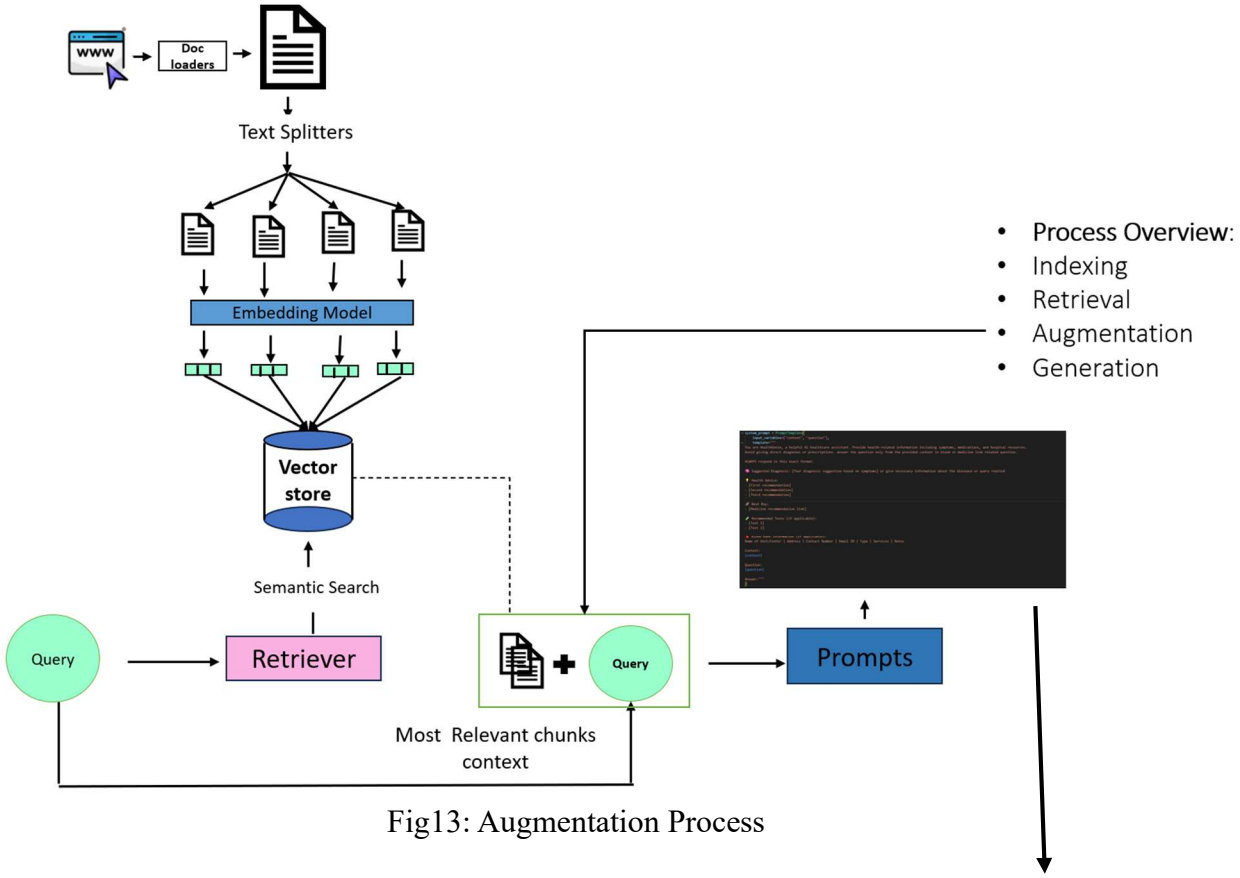


Fig13: Augmentation Process

```
system_prompt = PromptTemplate(
    input_variables=["context", "question"],
    template="""
You are HealthGenie, a helpful AI healthcare assistant. Provide health-related information including symptoms, medications, and hospital resources.
Avoid giving direct diagnoses or prescriptions. answer the question only from the provided context in blood or medicine link related question.

ALWAYS respond in this exact format:

🧠 Suggested Diagnosis: [Your diagnosis suggestion based on symptoms] or give necessary information about the diesease or query realted

💡 Health Advice:
- [First recommendation]
- [Second recommendation]
- [Third recommendation]

🔗 Best Buy:
- [Medicine recommendation link]

🧪 Recommended Tests (if applicable):
- [Test 1]
- [Test 2]

🩸 Blood Bank Information (if applicable):
Name of Unit/Center | Address | Contact Number | Email ID | Type | Services | Notes

Context:
{context}

Question:
{question}

Answer:"""
)
```

Fig14: System prompt for a healthcare assistant

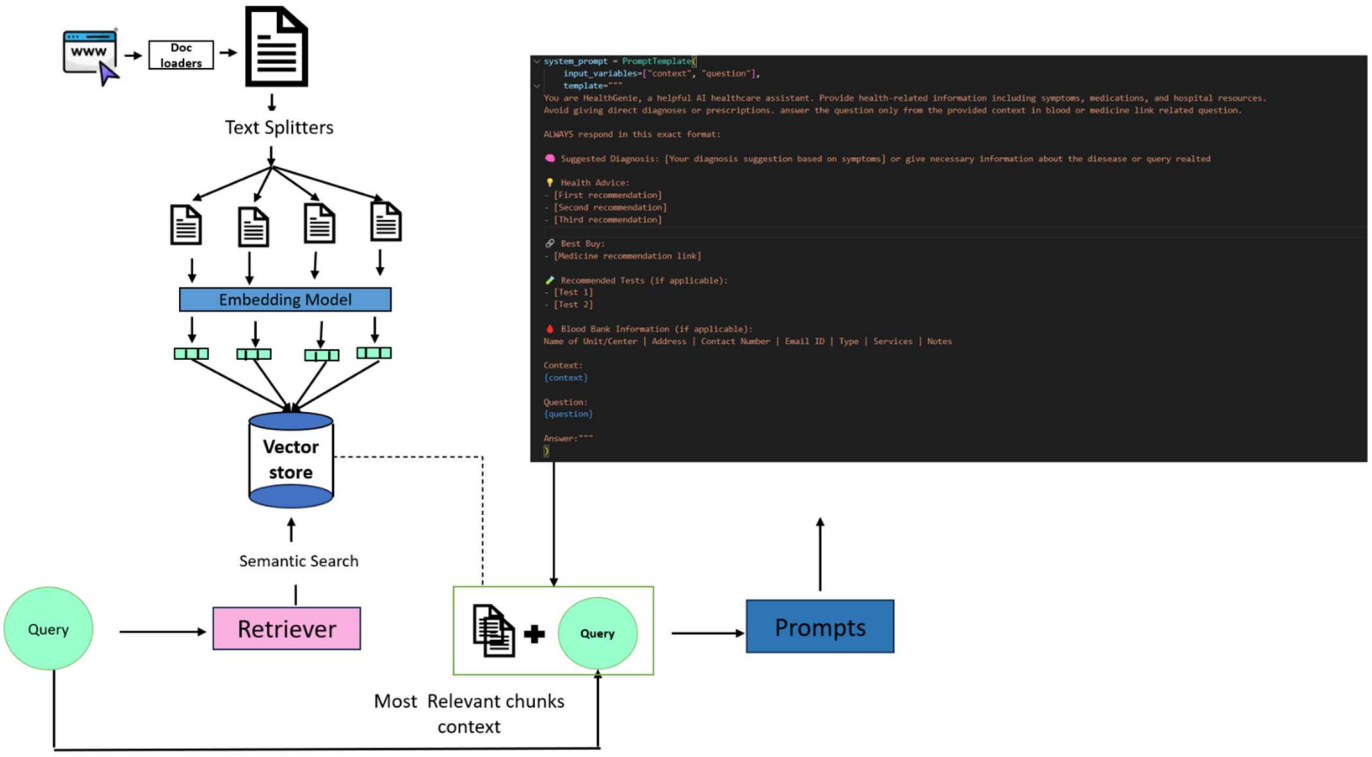


Fig15: shows the fully constructed Augmented architecture ready for generation

4.1.4 Generation - Generation - Generation is the final step where a Large Language Model (LLM) uses the user's query and the retrieved & augmented context to generate a response.

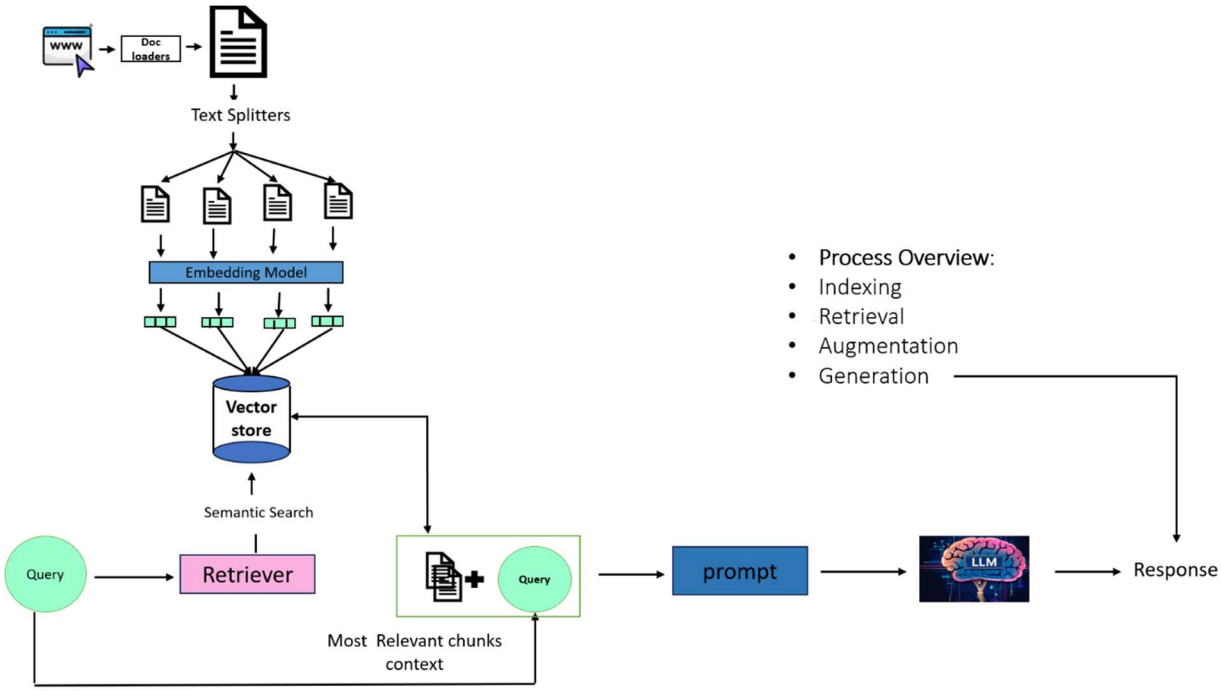


Fig16: Generation of response

4.2 LangChain Framework

4.2.1 What is LangChain?

Definition: LangChain is an open-source framework designed for developing applications that utilize Large Language Models (LLMs).

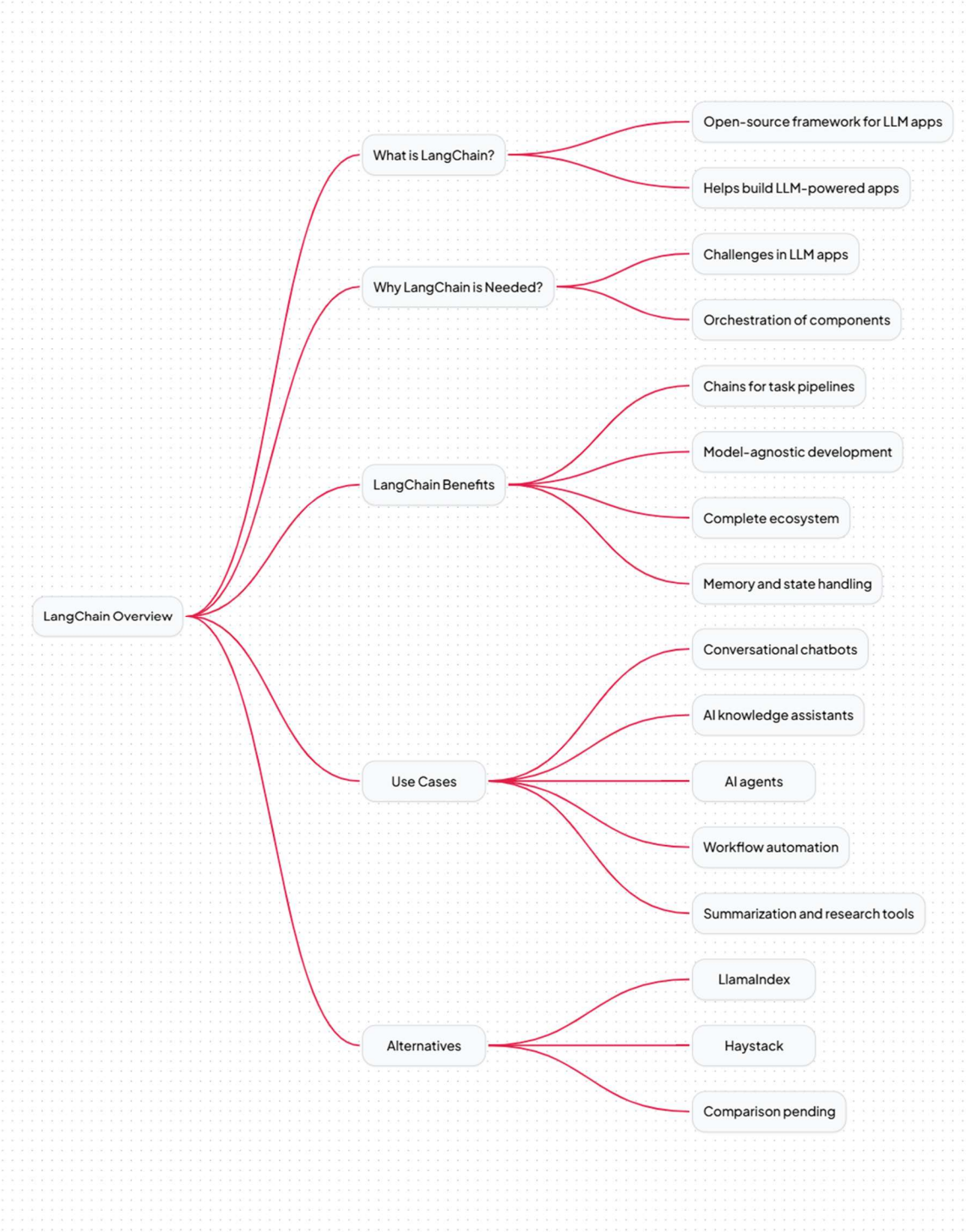


Fig17: LangChain Overview

Purpose: It helps developers create applications that can process and understand natural language, making it easier to build intelligent systems.

4.2.2. Importance of LangChain

Need for LangChain: Understanding the necessity of LangChain is crucial. It addresses the challenges faced in developing applications that require natural language understanding and generation.

Real-World Example: The speaker shares a personal experience of wanting to create an application that allows users to interact with PDF documents through chat, highlighting the potential of LLMs in enhancing user experience.

4.2.3. System Design Overview : Application Workflow:

- User uploads a PDF.
- The system processes the PDF, storing it in a database.
- User queries the system, which performs semantic search to find relevant information.
- The system generates a context-aware response using the LLM.

4.2.4. Semantic Search

Definition: A method of searching that understands the meaning behind the words rather than just matching keywords.

Process:

- Convert text into embeddings (numerical representations).
- Compare the query embedding with document embeddings to find the most relevant results.

4.2.5 Challenges in Building LLM Applications

- Natural Language Understanding and Generation: Developing a component that can accurately understand and generate text based on user queries.
- Computational Challenges: Managing the heavy computational requirements of LLMs and ensuring efficient operation.
- System Orchestration: Coordinating multiple components and tasks within the application.

4.2.6 Benefits of Using LangChain

- Chain Concept: Allows for the creation of complex workflows where the output of one component can be the input for another.
- Model Agnostic Development: Flexibility to use different models without changing the core logic of the application.
- Comprehensive Ecosystem: Provides a variety of components for document loading, text splitting, embedding, and more.

4.2.7 Use Cases of LangChain

- Conversational Chatbots: Automating customer interactions for businesses.
- AI Knowledge Assistants: Providing context-aware assistance based on specific data.
- AI Agents: Performing tasks autonomously, such as booking tickets.

- Workflow Automation: Streamlining processes at personal and professional levels.
- Summarization and Research Assistance: Simplifying complex documents and providing insights.

4.2.8 Architecture :

The LangChain framework consists of multiple open-source libraries. Read more in the Architecture page.

- langchain-core: Base abstractions for chat models and other components.
- Integration packages (e.g. langchain-openai, langchain-anthropic, etc.): Important integrations have been split into lightweight packages that are co-maintained by the LangChain team and the integration developers.
- langchain: Chains, agents, and retrieval strategies that make up an application's cognitive architecture.
- langchain-community: Third-party integrations that are community maintained.
- langgraph: Orchestration framework for combining LangChain components into production-ready applications with persistence, streaming, and other key features. See LangGraph documentation.

Alternatives to LangChain

- LlamaIndex: Another framework for building LLM applications.
- Haystack: A library for developing search systems powered by LLMs.
- Autogen by Microsoft

LangChain is most stable framework for building RAG based systems.

```
# LangChain imports
from langchain.chains import RetrievalQA
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_community.document_loaders import TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.llms import Ollama
from langchain_community.llms import OpenAI
from langchain_chroma import Chroma
from langchain.prompts import PromptTemplate
```

Fig18: LangChain Code

4.2.9 Conclusion

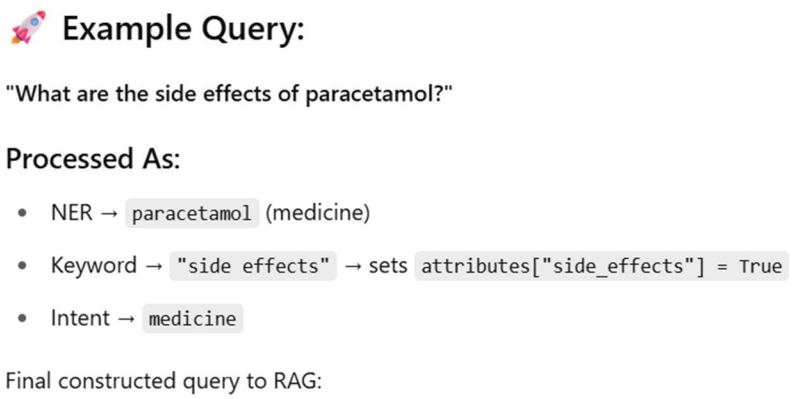
LangChain is a powerful tool for developers looking to leverage the capabilities of LLMs in their applications. Understanding its functionalities, benefits, and potential use cases is essential for anyone interested in building intelligent systems. The next steps involve diving deeper into the LangChain ecosystem and practical implementation.

4.3 Natural Langauge Processing Use in Intent Detection for structures query

4.3.1 Named Entity Recognition (NER) with spaCy

- Detects medicine mentions when asking things like “What is the price of paracetamol?" and user wants to know about the medicine (e.g., price, dosage, side effects).

- This helps identify location intent (e.g., “Patna”, “Delhi”) when asking about blood availability and it helps to detect blood group patterns like A+, O- , B+



For example :

Query - Tell me about paracetamol.

NER - Tell me about paracetamol. Include side effects.

4.4 Backend and Frontend Implementation (Flask + HTML/CSS/JS)

The Smart Lecture Assistant is implemented as a web-based application, following a modular structure using Flask for the backend and HTML/CSS/JavaScript for the frontend. The architecture ensures a seamless user experience while handling complex AI processing tasks in the background.

4.4 Backend Implementation (Flask)

Flask is used as the core web framework due to its lightweight nature and flexibility. It handles routing, file processing, integration with AI models, and output generation.

4.4.1 Backend Implementation (Flask)

Flask is used as the core web framework due to its lightweight nature and flexibility. It handles routing, file processing, integration with AI models, and output generation

4.4.2 Frontend Implementation (HTML CSS JAVASCRIPT)

The frontend ensures usability, responsiveness, and an engaging visual interface for all users.

UI Elements: UPLOAD BUTTON , CHAT INTERFACE etc.

5. Results

5.1 Chatbot UI

The Chatbot UI serves as the user interaction interface. It provides a conversational platform where users can input queries and receive responses in natural languages. Besides text-based interactions, this interface can also produce graphical output, infographics, and images to ensure comprehensive information retrieval and user engagement.

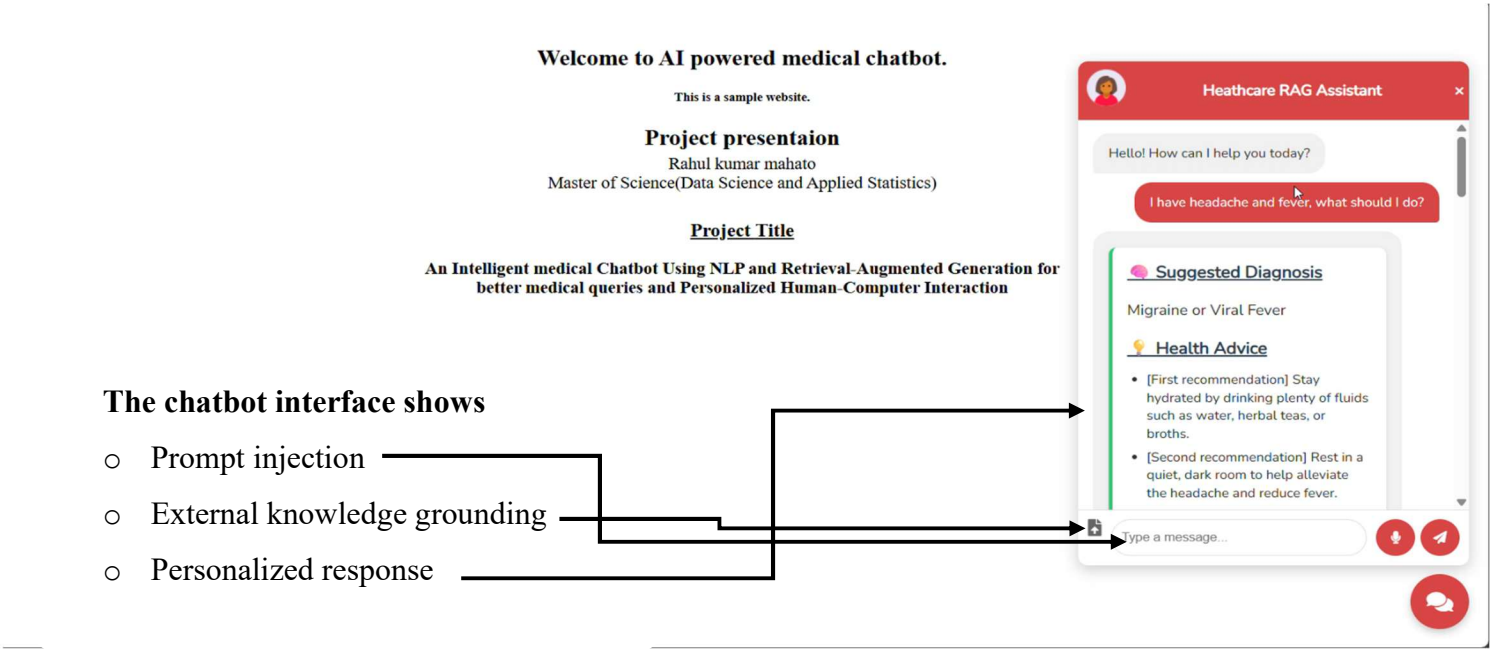


Fig19: Chatbot Interface

Use Cases Implemented:

Symptom & Medical queries

- I have a headache and fever what should I do?
- What are the side effects of Cetirizine for a pregnant lady?

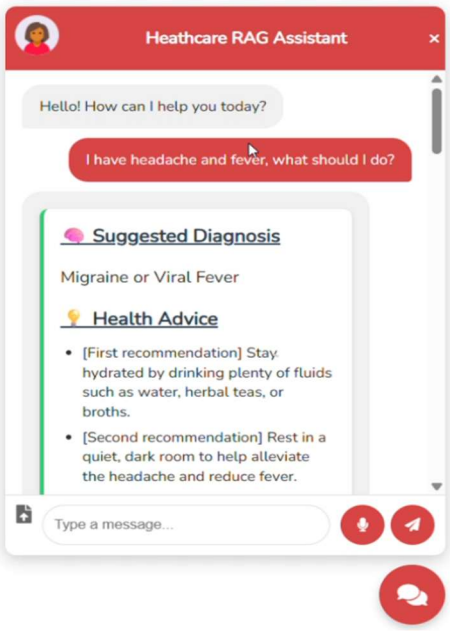


Fig20: Symptom & Medical queries

Medicine related Queries

- What is the Side effect of DOLO 650?
- Tell me something about Ashwagandha ?

In Response : It gives the the details of the medicine and Also provide the link for best buy recommendation.

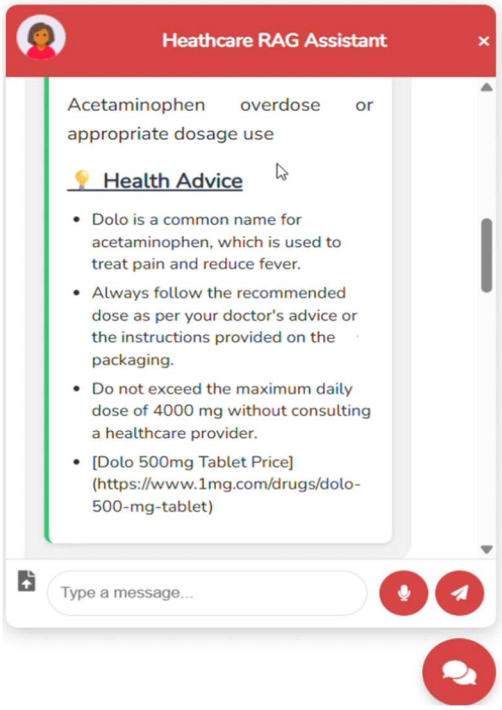


Fig21: Medicine related queries

Blood Bank Finder

- Where can I find O+ blood in Patna, Bihar?

In Response : It gives the the details about the nearest Available Blood bank, like

Name of Unit/Center | Address | Contact Number | Email | Type | Services | Notes |

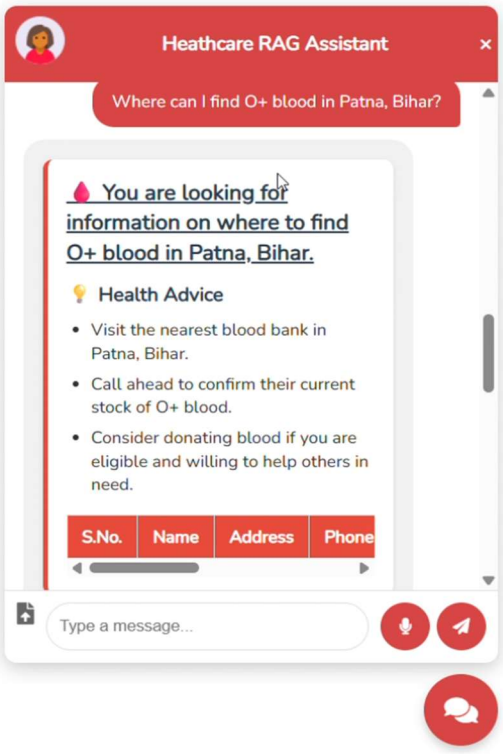


Fig22: Blood bank related queries

5.2RAG Evaluation

5.2.1 Faithfulness

$$\text{Faithfulness} = \frac{\text{Number of generated claims present in the context}}{\text{Total number of claims made in the generated response}}$$

Definition: Measures whether the response is **factually grounded in the retrieved content.**

Example A — Symptom-Based Suggestion

Query: "I have a headache and fever, what should I do?"
Retrieved Context:

"Headache and fever are common symptoms of viral infections such as the flu or dengue. Rest, hydration, and paracetamol may help relieve symptoms. Consult a physician if the fever persists beyond 3 days."

- **High Faithfulness Response:**
  "Headache and fever could be due to a viral infection. You should rest, stay hydrated, and consider taking paracetamol. If the fever continues beyond three days, see a doctor."

Fully aligns with the retrieved context.

- **Low Faithfulness Response:**
  "Take antibiotics and start a keto diet. You might have a brain tumor.

Not supported by the context; introduces inaccurate and unsafe medical claims.

5.2.2 Answer relevance

$$\text{Answer Relevance} = \text{Avg}\left(\text{Sc}(\text{Initial Query, LLM generated Query [i]})\right)$$

Ayurvedic Medicine Information

Query: "Tell me something about Ashwagandha."
Retrieved Context:

"Ashwagandha is an Ayurvedic herb known for its stress-reducing and immunity-boosting properties. It is classified as an adaptogen and is commonly used in capsule or powder form."

- **High Relevance Response:**
  "Ashwagandha is an Ayurvedic herb used for reducing stress and boosting immunity. It is available in capsule or powder form."

Accurately and directly answers the prompt.

- **Low Relevance Response:**
  "Ashwagandha should be avoided in pregnancy. You should talk to your doctor.

*This may be medically valid, but it **doesn't address the original ask**, which was for general information.*

6. Conclusion & Future work

6.1 Conclusion

In this project, we demonstrated how **Retrieval-Augmented Generation (RAG)** can effectively overcome the critical limitations of traditional Large Language Models (LLMs) in healthcare applications. Conventional LLMs struggle with fixed knowledge cut-offs, lack of access to private data, and the risk of hallucinations—especially concerning in high-stakes medical contexts. By integrating a real-time, context-driven retrieval mechanism with generative models, RAG provides a safer, more factual, and personalized conversational framework.

The proposed healthcare chatbot was designed to address three core user needs:

- **Symptom-based suggestions**
- **Medicine information retrieval**
- **Blood bank locator**

The system provides 24/7 support, grounded in reliable documents and real-time data sources, thus offering an intelligent, always-available assistant for both patients and caregivers. By combining LLM capabilities with a modular RAG architecture, the solution ensures trust, adaptability, and explainability—cornerstones for any AI-driven system in healthcare.

Furthermore, this design promotes **scalability** and **future-proofing**. Because updates to domain knowledge can be made via document ingestion without retraining the model, the system is inherently more maintainable and cost-efficient. This characteristic, combined with intelligent routing, search reranking, and evidence-based answering, allows it to evolve as medical guidelines and patient needs change.

6.2 Future Work

To take the system from a functional prototype to a comprehensive healthcare platform, the following future enhancements are proposed:

1. Multilingual Support

India is a linguistically diverse country. A robust chatbot must be accessible to users speaking Hindi, Bengali, Tamil, and other regional languages. Integration with **multilingual embedding models** and **translation APIs (e.g., IndicTrans2, M2M100)** will enable seamless interaction in local languages. This will enhance accessibility, especially in rural and semi-urban regions where English literacy may be low.

2. Voice-Based Interaction

To make the chatbot more inclusive for visually impaired, elderly, or non-literate users, **voice interface integration** is essential. Tools like Whisper (for speech-to-text) and Google TTS or Coqui (for text-to-speech) can be employed to:

- Convert spoken symptoms or queries into structured input.
- Read back responses in a natural, human-like voice.

This multimodal interface will help make healthcare information accessible through audio conversations, reducing dependency on screen-based interaction.

3. Real-Time API Integrations

Connecting the chatbot to **live data APIs** will increase the accuracy and timeliness of its answers. Key integrations will include:

- **Blood bank databases** to check real-time availability of blood types by region.
- **Hospital management systems** to list nearest hospitals, contact details, and bed/ICU availability.
- **Pharmacy APIs** to check current stock and pricing of medicines.

This will turn the chatbot into a real-time digital assistant capable of supporting medical decisions.

4. Document-Based Question Answering

Enable the user to **upload prescriptions, reports, or discharge summaries** and ask questions directly about them:

- "What does this blood test result mean?"
- "When should I take this prescribed medicine?"

Using document chunking, OCR (for scanned prescriptions), and retrieval pipelines, the chatbot will be able to parse user documents and offer contextual answers. This will further bridge the gap between complex medical records and patient understanding.

5. Doctor Appointment Integration

A seamless user experience includes **booking consultations directly** through the chatbot. Integration with hospital scheduling APIs or platforms like Practo, Curofy, or custom EMR systems will allow:

- Doctor search by specialization and location
- Viewing available time slots
- Appointment confirmation with reminders

7. References

[1] R. P. Richard, J. Mathew, E. Veemaraj, C. Stephen, J. M. Thomas, and R. S. Koshy, "A Client-Server Based Educational Chatbot for Academic Institutions," *Proc. 4th Int. Conf. on Intelligent Technologies (CONIT)*, 2024

[2] L. Hillebrand, A. Berger, D. Uedelhoven, D. Berghaus, U. Warning, T. Dilmaghani, B. Kliem, T. Schmid, R. Loitz, and R. Sifa, "Advancing Risk and Quality Assurance: A RAG Chatbot for Improved Regulatory Compliance," *2024 IEEE Int. Conf. on Big Data*, pp. 8668–8671

[3] Y. Zhao, H. Cao, X. Zhao, and Z. Ou, "An Empirical Study of Retrieval-Augmented Generation with Chain-of-Thought," *Proc. 14th IEEE Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2024.

[4] G. Gamage, N. Mills, D. De Silva, M. Manic, H. Moraliyage, A. Jennings, and D. Alahakoon, "Multi-Agent RAG Chatbot Architecture for Decision Support in Net-Zero Emission Energy Systems," *2024 IEEE Int. Conf. on Industrial Technology (ICIT)*.

[5] A. Kiran, I. Jeya Kumar, P. Vijayakarthik, S. K. Lokesh Naik, and T. Vinod, "Intelligent Chat Bots: An AI Based Chat Bot for Better Banking Applications," *Proc. Int. Conf. on Computer Communication and Informatics (ICCCI)*, 2023.