PHARMA SAGE - Project Documentation

# Overview

PHARMA SAGE is an intelligent question-answering system designed for pharmaceutical PDF documents using a Graph + RAG (Retrieval-Augmented Generation) architecture. It integrates Milvus (vector DB), Neo4j (graph DB), and Google's Gemini (LLM & embeddings) to provide intelligent insights and responses.

# Tools & Technologies Used

- Google Gemini (Generative AI)  
- LangChain (for LLM abstraction)  
- Milvus (Zilliz Cloud) - Vector Database  
- Neo4j - Graph Database  
- Streamlit - Frontend Chat UI  
- FastAPI - Backend REST API  
- PyMuPDF / PyPDF2 - PDF Parsing  
- Uvicorn - ASGI Server  
- Python Virtual Environment (venv)

# What Problem Does It Solve?

In the pharmaceutical domain, researchers, clinicians, and decision-makers deal with an overwhelming amount of complex data buried in long research papers, clinical trial documents, drug information sheets, and regulatory filings. Traditional keyword-based search is inefficient in understanding **context**, **relationships**, or **deep scientific semantics**.

**PHARMA SAGE** solves this by introducing an **AI-powered Knowledge Graph Assistant** that enables:

* 📚 **Intelligent document understanding** using LLM-powered semantic embeddings
* 🔗 **Relationship discovery** through a Neo4j knowledge graph built from document chunks
* 🤖 **Natural language question answering** powered by Google Gemini (Gemini Pro)
* 🔍 **Hybrid search** combining semantic relevance + graph traversal
* 📈 **Graph analytics** to explore key entities, connections, and trends

It allows users to upload pharmaceutical PDFs, generate embeddings, construct a graph, and interact with the enriched data using plain English — all via a modern web UI.

**How It Works – High-Level Workflow**

 PDF **Upload & Pre-processing**

* Users upload multiple PDF documents via a Streamlit frontend.
* Each PDF is chunked into manageable segments using a text splitter.
* Duplicates are skipped using content hashing.

 Vector **Embedding Generation**

* Each chunk is embedded using **Google’s Gemini embedding model (embedding-001)**.
* Embeddings (dim=768) are stored in **Milvus vector DB** with metadata.

 Knowledge **Graph Creation**

* Entities and document structure are extracted.
* A **Neo4j graph** is constructed with nodes for documents, chunks, and relationships (e.g., CONTAINS, NEXT).
* Entity linking and graph traversal are possible.

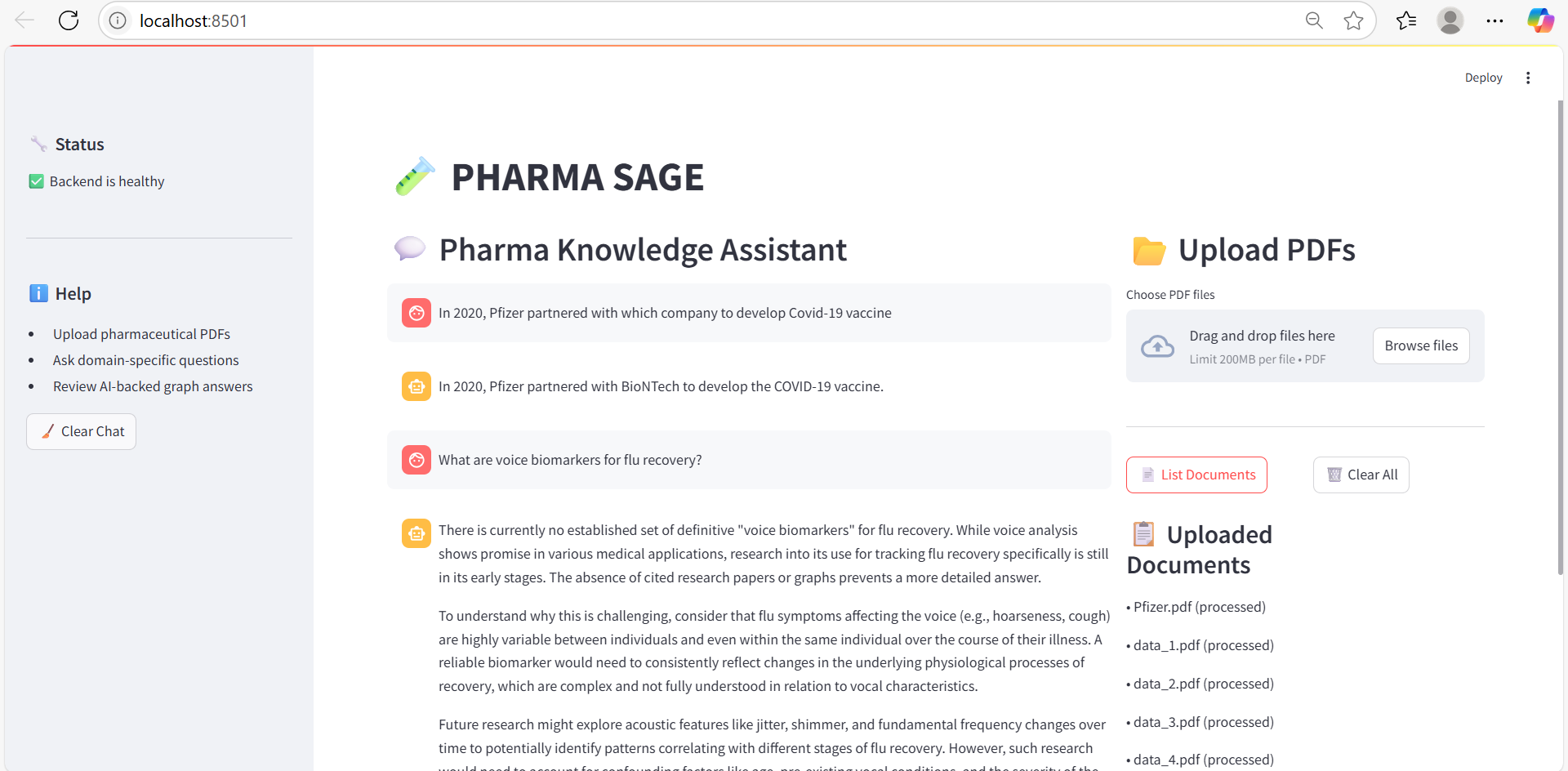
 Natural **Language Question Answering**

* User queries are handled using three modes:
  + **Semantic search** using similarity from Milvus
  + **Graph querying** using Cypher-based traversal from Neo4j
  + **Hybrid querying** combining both with a contextual Gemini LLM response

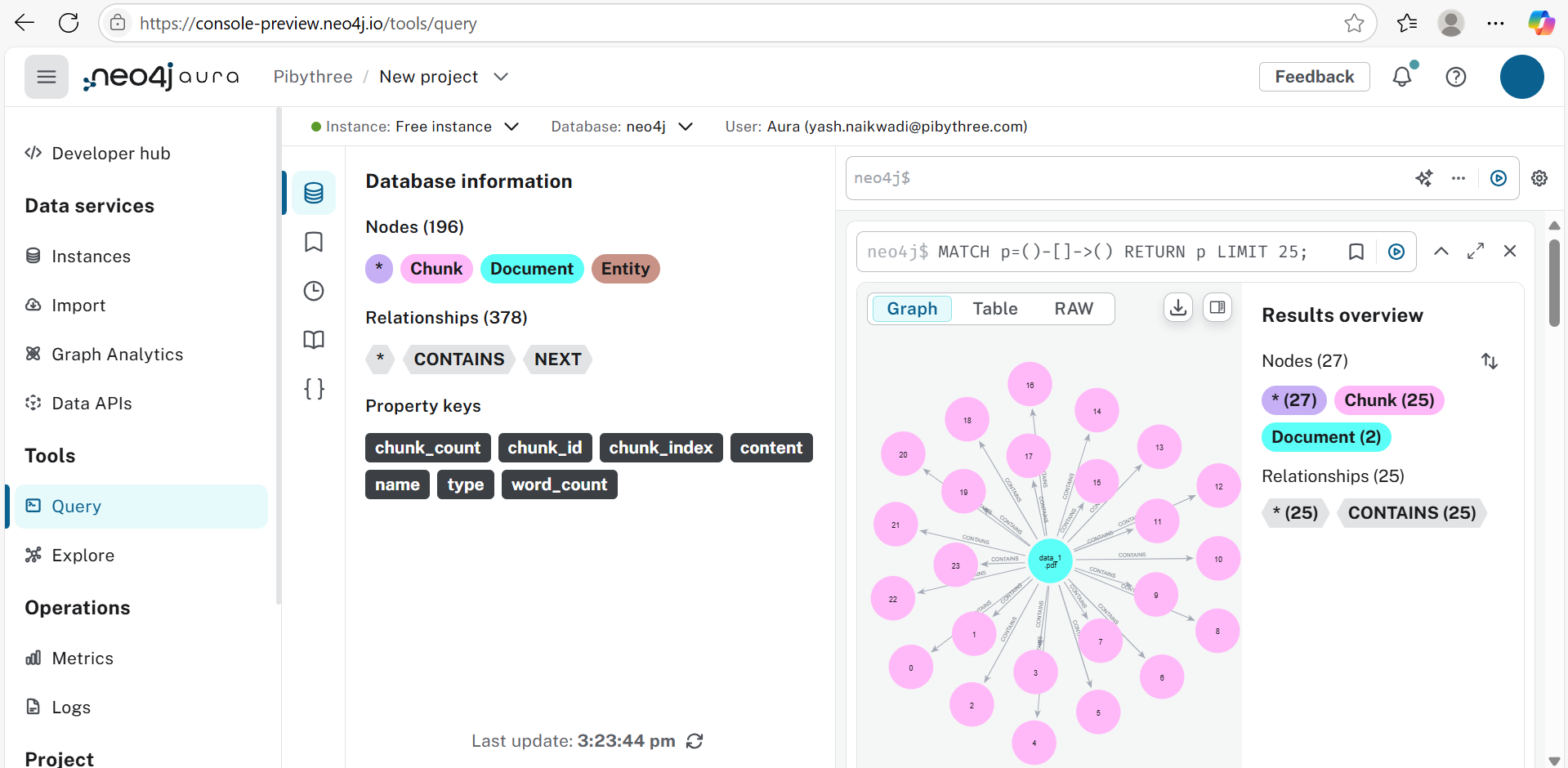
 Insights **& Analytics**

* The app also supports visualizing:
  + Entity connections
  + Relationship types
  + Document stats and metadata
  + AI-generated summaries and insights

# Chat UI response from PHARMA SAGE



# Knowledge Graph in neo4j



# Sample Questions to Ask to PHARMA SAGE

 📌*What are voice biomarkers for flu recovery?*

 📌*How did Pfizer collaborate on COVID-19 vaccines?*

 📌*Which acoustic features indicate nasal congestion?*

 📌*How does spectral entropy relate to decongestion?*

 📌*What is the duration of vaccine development in pharma?*

 📌*How are MFCCs used in respiratory illness analysis?*

 📌*What are key clinical trial parameters?*

# Core Python Modules

- `app.py`: Streamlit frontend UI with chat and file upload interface.  
- `main.py`: FastAPI application initializing all backend routes and services.  
- `milvus\_ops.py`: Handles connection and operations on Milvus vector database using Gemini embeddings.  
- `neo4j\_ops.py`: Graph logic for storing and querying document knowledge in Neo4j.  
- `upload.py`: Handles PDF ingestion, chunking, embedding, and graph creation pipeline.  
- `intelligent\_query.py`: Routes that implement semantic, graph-based, and hybrid query answering using Gemini LLM.

**Core Features of PHARMA SAGE**

| **Category** | **Feature Description** |
| --- | --- |
| Document Ingestion | Upload multiple PDFs at once |
| Deduplication | Skips reprocessing identical documents via hashing |
| Embeddings | Uses **Google Gemini Embedding API** (embedding-001) |
| Vector Store | Stores chunk embeddings in **Milvus (Zilliz Cloud)** |
| Knowledge Graph | Auto-constructs graphs in **Neo4j** from documents |
| Semantic Search | Retrieves relevant chunks based on vector similarity |
| Graph Search | Uses Cypher queries to retrieve related entities |
| Hybrid QA | Combines vector + graph search + LLM summarization |
| Chat UI | Interactive chat powered by **Gemini Pro** |
| Graph Analytics | Get entity stats, relationship types, and document summaries |
| Insight Discovery | Entity discovery, insight generation, and AI-guided reports |
| API Integration | REST APIs powered by **FastAPI backend** |
| Frontend | Modern chat and upload interface using **Streamlit** |
| Environment Config | .env and settings.yaml control secrets and runtime settings |
| Extensibility | Modular Python codebase (LLM models, DBs, prompts can be swapped) |

**Basic Architecture of PHARMA SAGE**

