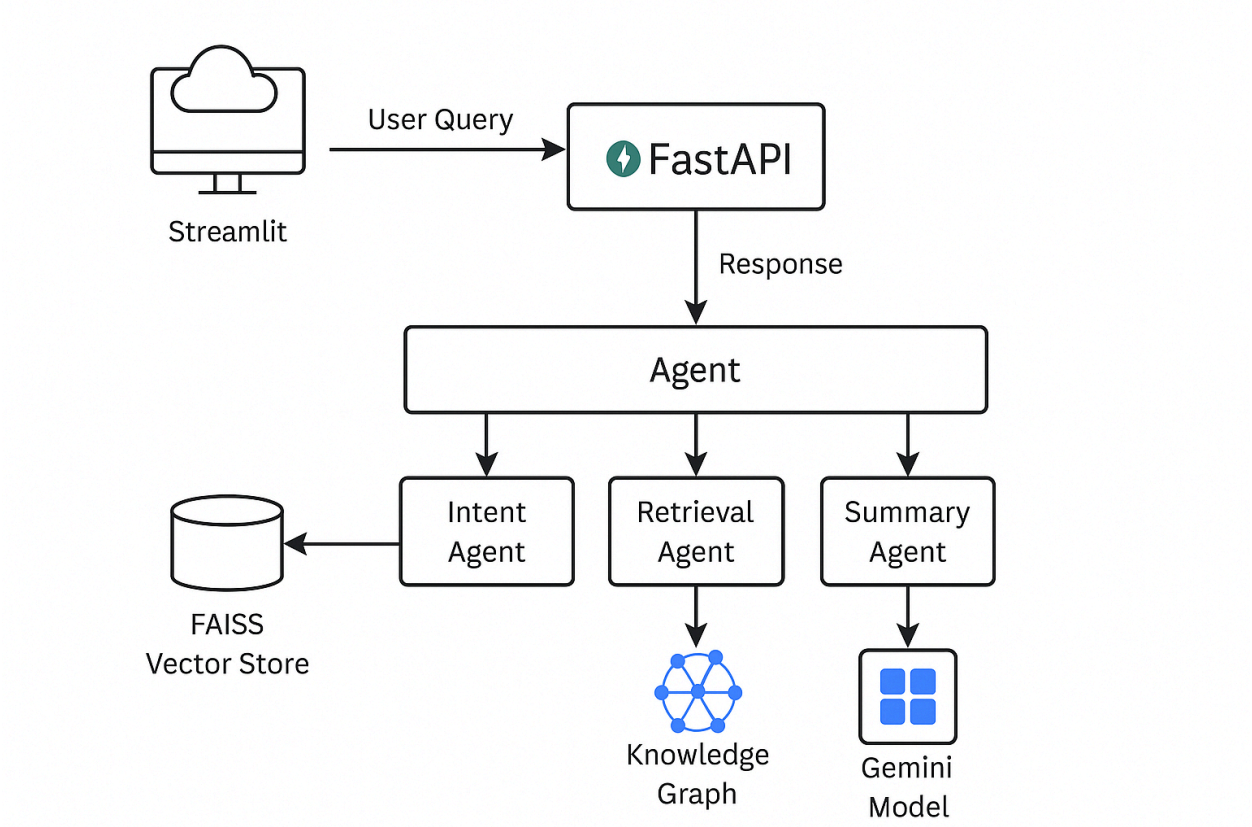
1. System architecture overview



2. Domain chosen + dataset source

Domain:

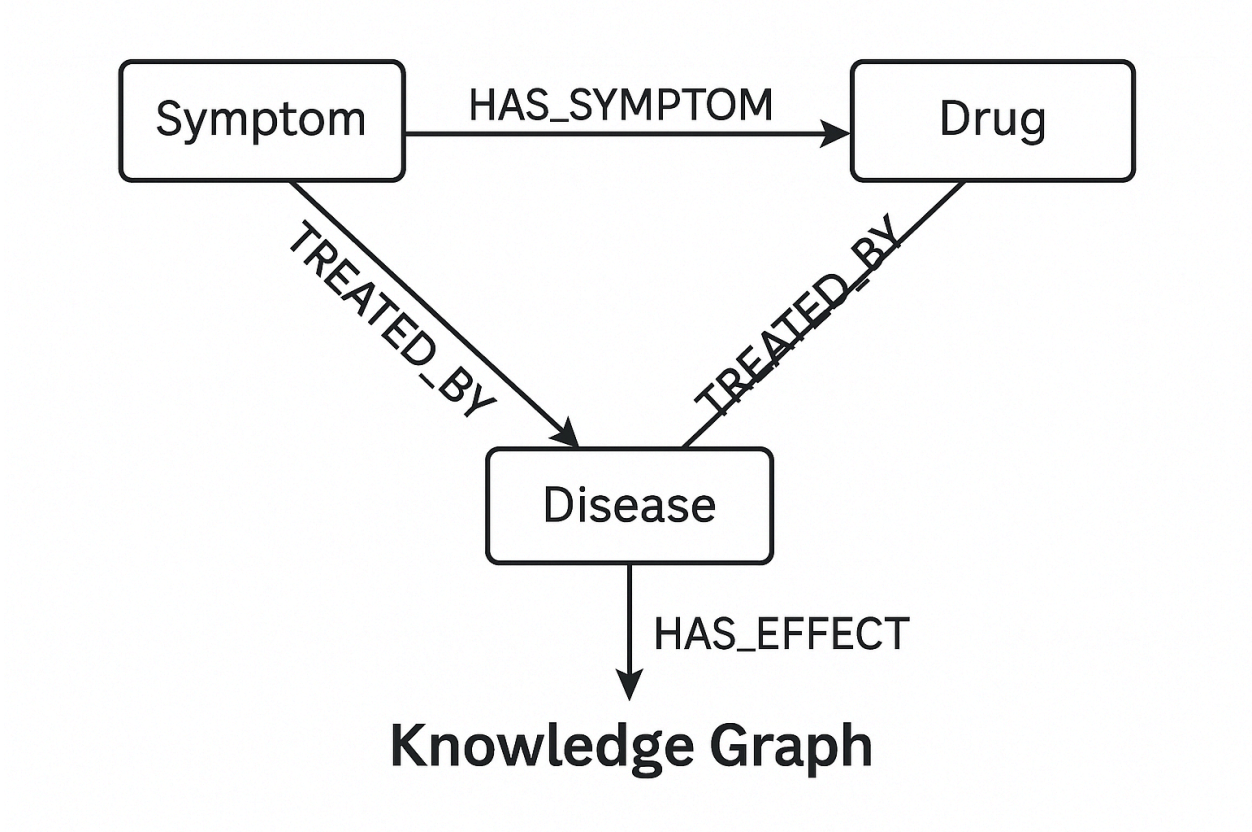
Healthcare – Diabetes Awareness, Treatment, and Management

Dataset Sources Used:

- World Health Organization (WHO) — Diabetes Fact Sheets
- National Institutes of Health (NIH) — Diabetes treatment guidelines
- American Diabetes Association (ADA) — Care Standards
- MedlinePlus — Diabetes patient education
- Additional public health articles, medical descriptions, and clinical notes

All documents were converted into text and placed in `data/raw/`.

3. Knowledge graph structure



4. RAG/GraphRAG design

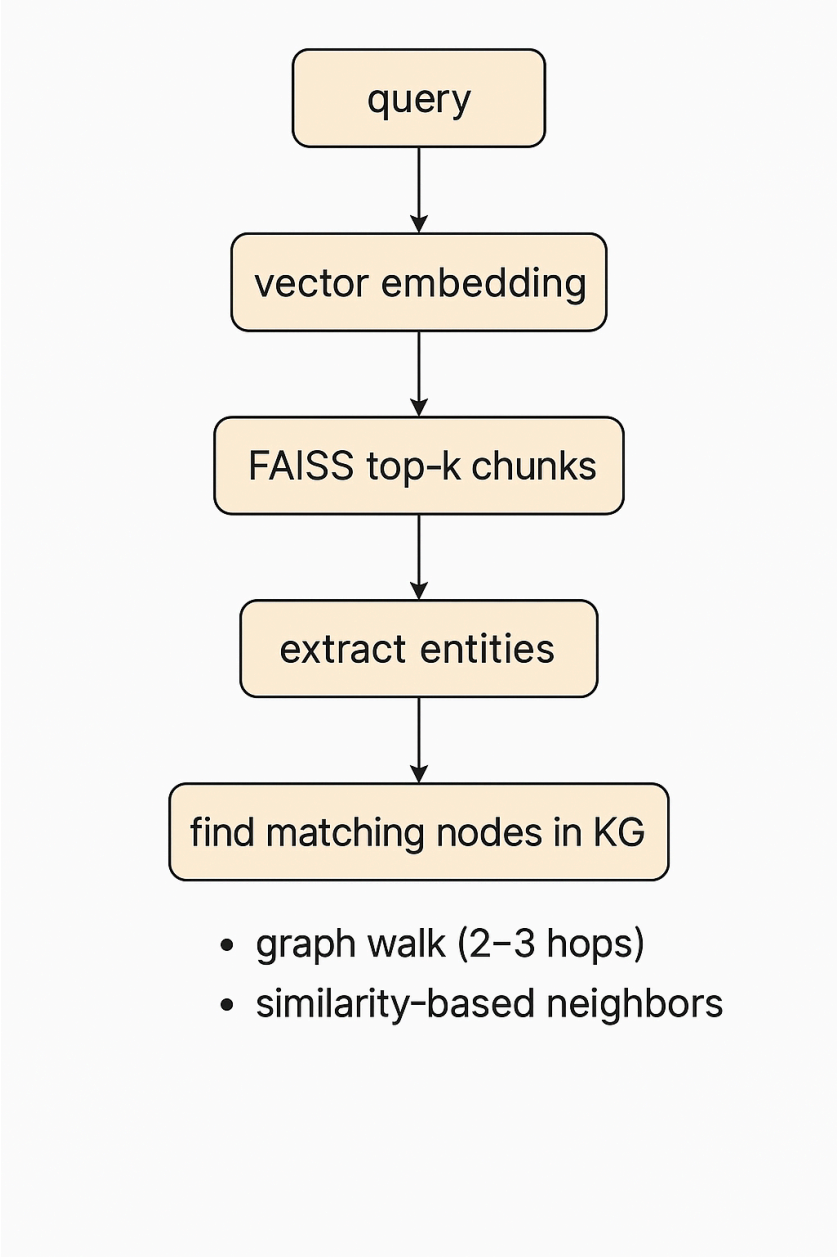
Standard RAG (Vector-based)

- Uses **FAISS** to retrieve **top-k** semantically similar chunks.
- Good for general Q&A but limited for reasoning over relationships.

GraphRAG (Enhanced Retrieval)

Adds graph reasoning:

- Identify entities related to the query
- Expand through graph neighbors using multi-hop search
- Retrieve semantic + structural context
- Combine both to improve accuracy and explainability



5. Agent workflow explanation

The agent uses a **multi-stage pipeline**:

1. Intent Classification

Detects the type of query:

- Symptoms
- Diagnosis
- Medication
- Lifestyle
- Complications
- General medical questions

2. GraphRAG Retrieval

Gets:

- Vector-based similar chunks
- Graph neighbors
- Relevant entities
- Multi-hop relationships

3. Summarizer (LLM)

Condenses retrieved chunks into a clear, accurate summary.

4. Graph Reasoning Module

Analyzes entity relationships and context from KG.

5. Final Answer Generator

Combines:

- Query
- Summaries
- Graph context
- Medical reasoning

Outputs a structured and clinically safe answer.

6. API endpoints

POST /query

Retrieve GraphRAG output + final answer.

Request

```
{
  "query": "How does metformin work?",
  "k": 5,
  "hops": 2
}
```

Response

```
{
  "intent": "treatment",
  "retrieval": {
    "vector_hits": [...],
    "graph_nodes": {...}
  },
  "summary": "Metformin reduces hepatic glucose production...",
  "final": "Metformin works by..."
}
```

7. Setup & run instructions

Step 1

In **VS Code terminal (inside your project folder C:\Mindgraph)** run:

```
git init
git remote add origin
https://github.com/JOBIN02/GraphRAG-Healthcare.git
```

Step 2

Create a new file: `.gitignore` and paste:

```
# Python
__pycache__/
*.pyc

# Virtual env
.venv/

# Environment variables
.env

# Data files
data/raw/*
data/embeddings/*
data/graphs/*
data/processed/*

# Streamlit
.streamlit/

# VSCode
.vscode/
```

Save this before adding files to git.

Step 3 — Add all SAFE project files

Run this:

```
git add .
```

Git will automatically ignore everything listed in `.gitignore`.

Step 4 — Commit

```
git commit -m "Initial commit: GraphRAG Healthcare system"
```

Step 5 — Push to GitHub

GitHub gave you the command here — use:

```
git branch -M main
git push -u origin main
```

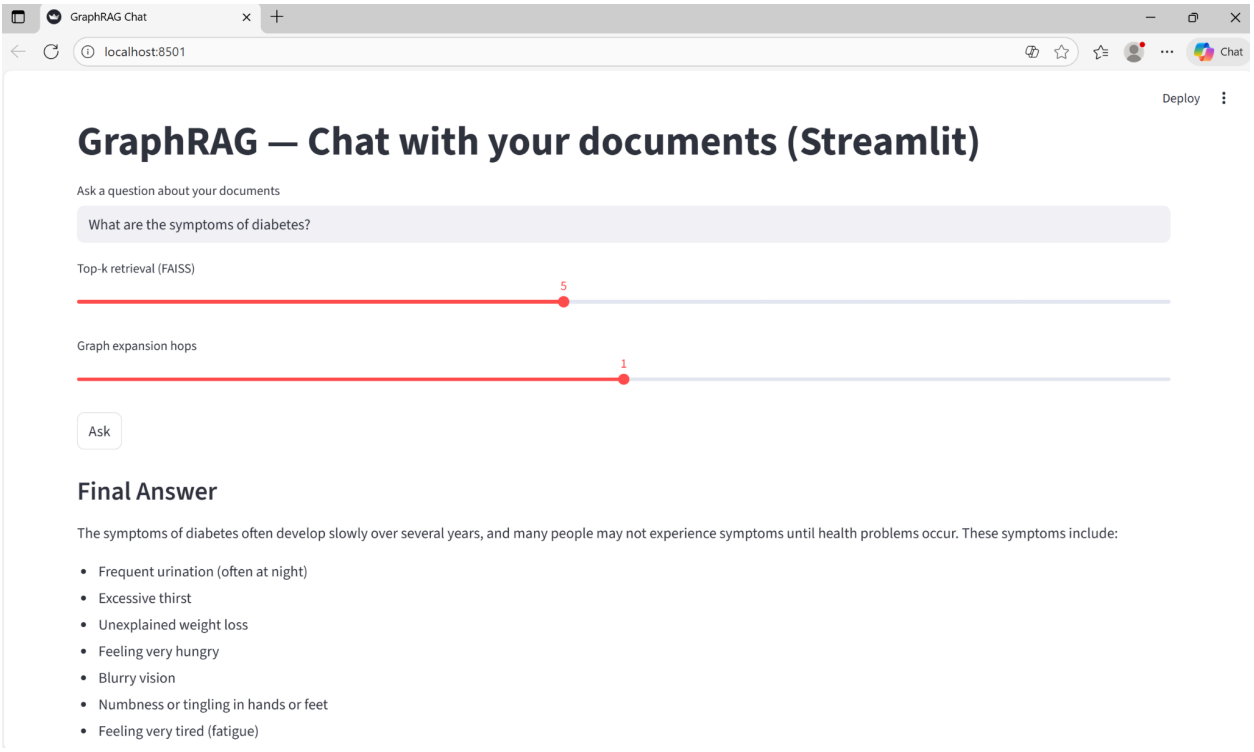
8. Sample queries + sample responses

Query 1:

“What are the symptoms of diabetes?”

Response:

- Excessive thirst
- Frequent urination
- Blurred vision
- Fatigue
  (Graph nodes: Symptoms cluster)



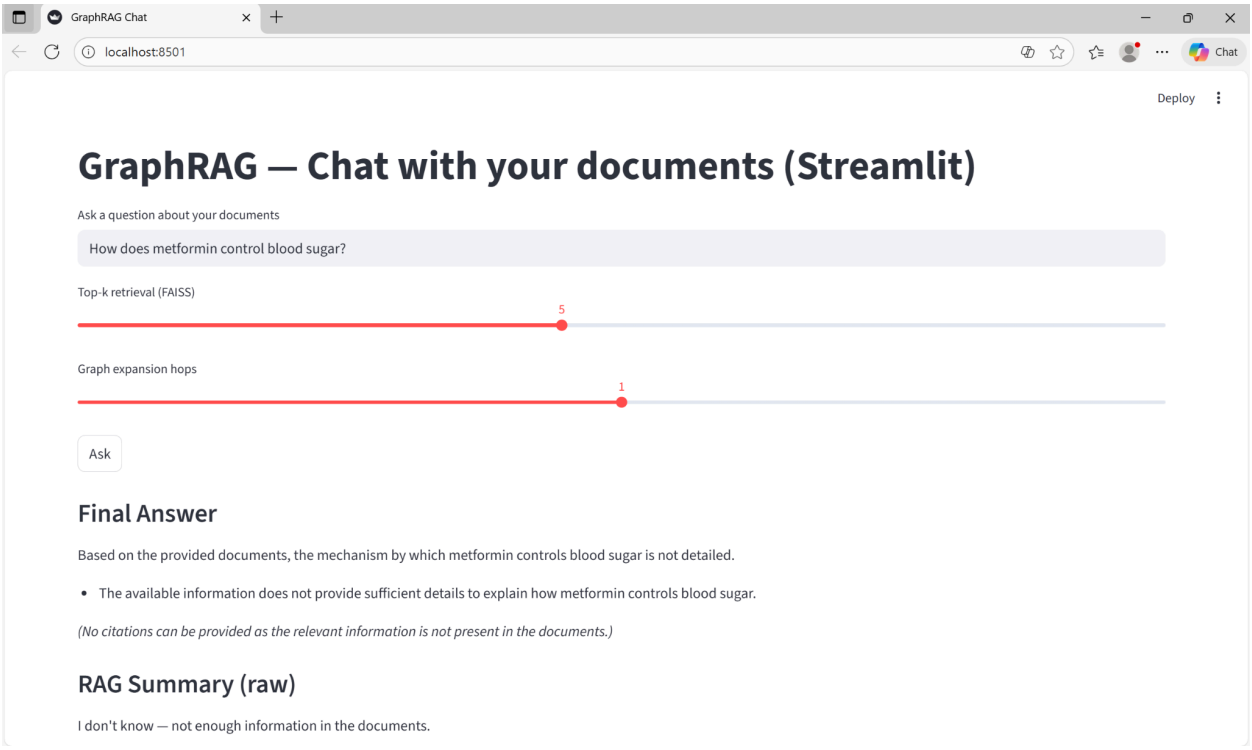
Query 2:

“How does metformin control blood sugar?”

Answer:
Metformin:

- Reduces liver glucose production
- Improves insulin sensitivity
- Enhances peripheral glucose uptake

(Graph context: Metformin → Glucose → Insulin Resistance)



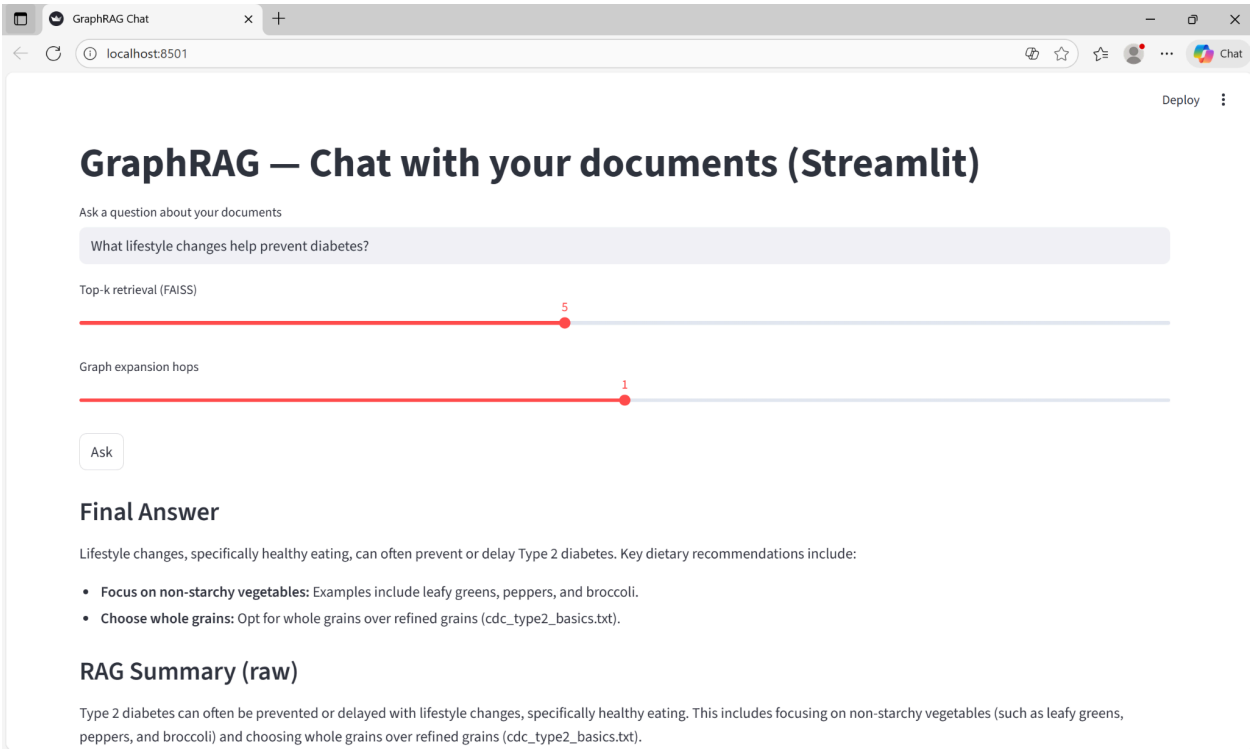
Query 3:

“What lifestyle changes help prevent diabetes?”

Answer:

- Weight loss
- Increased physical activity
- Balanced diet
- Reduced sugar intake

(Graph nodes: Obesity → Diabetes Risk → Lifestyle)



9. Challenges faced & learnings

Challenges

- Handling missing/invalid Gemini API keys
- Graph construction issues due to spaCy model downloads
- Circular imports in Python modules
- NetworkX version differences (no write_gpickle/read_gpickle)
- Maintaining consistency between FAISS vectors & KG nodes
- Ensuring clinical accuracy without hallucinations
- Backend 500 errors from LLM response formatting

Learnings

- Graph-enhanced retrieval significantly improves relevance
- Multi-hop reasoning adds interpretability to healthcare answers
- Proper preprocessing (chunk size, cleaning) has huge impact
- LLM outputs must be wrapped with safe extractors
- Logging and modularization prevent API crashes
- RAG != enough for medical tasks → GraphRAG is superior
- Clean separation of API, agent, graph, and UI simplifies debugging

3. Public Repository Link

https://github.com/JOBIN02/GraphRAG-Healthcare