

# How the RAG Chatbot Works

---

## Simple Explanation

Think of this chatbot like a smart librarian that:

1. **Reads your documents** and remembers everything
2. **Finds relevant parts** when you ask questions
3. **Only answers** based on what it read (no making stuff up!)

## The Process Step-by-Step

### 1. Auto Document Indexing

`./documents/` folder → Scan Files → Extract Text → Split into Chunks → Create Embeddings → Store in Database

#### What happens:

- System automatically scans the `./documents/` folder for PDF and TXT files
- Reads all the text from your files
- Detects new or updated files using file hashes
- Breaks text into small chunks (1000 characters each with 200 overlap)
- Converts each chunk into a "vector" (list of numbers that represents meaning)
- Stores these vectors in ChromaDB database

**Why chunks?** Large documents are split so the AI can find specific relevant sections instead of getting overwhelmed.

### 2. When You Ask a Question

Your Question → Convert to Vector → Search Database → Find Similar Chunks → Generate Answer





#### What happens:

- Your question gets converted to the same type of vector

- System searches for the most similar document chunks (top 3)
- Combines relevant chunks into context
- Sends context + question to Google Gemini
- Gemini generates answer ONLY from the provided context

### 3. Strict Answer Rules

The AI is instructed to:

-  Only use information from your documents
-  Cite which document the answer comes from
-  Say "I don't have information about that" if not in docs
-  Never make up or hallucinate information

## Technical Components

### Core Files

`rag_chatbot.py` - The brain of the system

- `RAGChatbot` class handles all the logic
- PDF text extraction
- Text chunking and embedding
- Vector search and response generation

`app.py` - The web interface

- Streamlit UI for uploading files
- Chat interface
- Progress indicators and status updates

### Key Technologies

**Google Gemini 2.5 Flash** - The AI that generates responses

- Fast and accurate
- Follows strict instructions to only use provided context

**ChromaDB** - Vector database

- Stores document chunks as vectors
- Enables fast similarity search

- Persistent storage (survives app restarts)

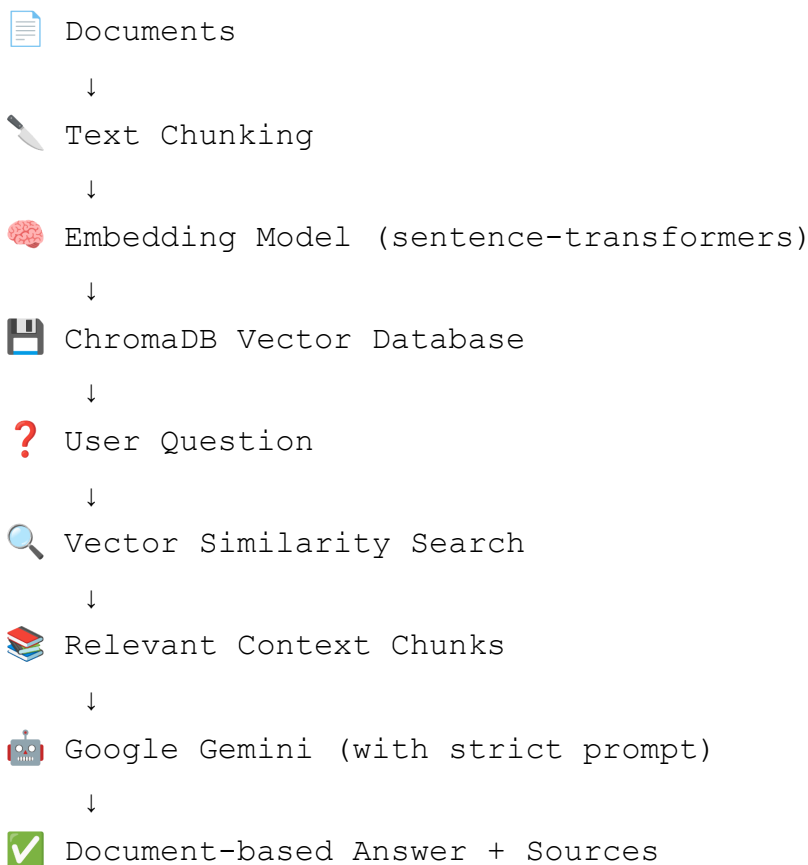
**sentence-transformers** - Creates embeddings

- Converts text to numerical vectors
- Model: `all-MiniLM-L6-v2` (fast, good quality, runs locally)

**Streamlit** - Web interface

- Easy file upload
- Chat interface
- Real-time updates

## Data Flow Diagram



## Why This Approach Works

1. **No Hallucination:** AI only sees your document content, nothing else
2. **Fast Retrieval:** Vector search finds relevant info in milliseconds
3. **Source Attribution:** Always shows which documents were used
4. **Persistent:** Your processed documents stay in the database
5. **Scalable:** Can handle multiple documents efficiently

# File Structure

```
|— app.py                # Streamlit web interface
|— rag_chatbot.py        # Core RAG logic
|— requirements.txt      # Python dependencies
|— .env                  # API keys (you need to set this)
|— README.md             # Setup instructions
|— HOW_IT_WORKS.md       # This explanation
└— chroma_db/            # Vector database (created automatically)
```

## Security & Privacy

- Documents are processed locally
- Only your question + relevant chunks sent to Google Gemini
- No full documents sent to external APIs
- Vector database stored locally on your machine