Title: Building Chatbot used a RAG-Based Document QA System Using Gemini, Qdrant, and OpenAI

Here's a clear description of the RAG

(Retrieval-Augmented Generation) workflow based on your diagrams,

structured for technical implementation:

RAG System Workflow

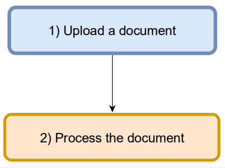
A 4-step pipeline that transforms documents into searchable knowledge and generates AI-powered answers:

1: Introduction

- Project Title: Document Question Answering with RAG Pipeline
- Key Technologies: Gemini (Embedding), Qdrant (Vector Search), OpenAI (Answer Generation)
- Goal: Upload a document and get accurate answers to user questions.

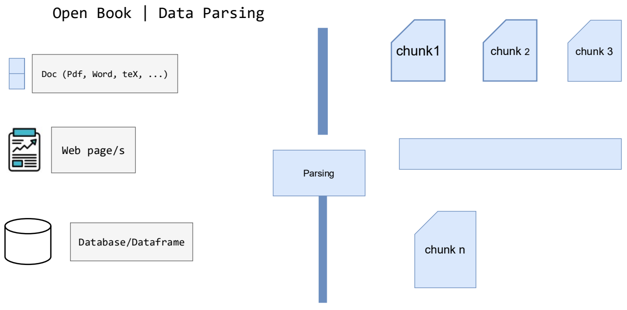
2: Problem Statement

- Users need a way to query documents in natural language.
- Manual searching is slow and inefficient.
- Objective: Automate document understanding and answering.



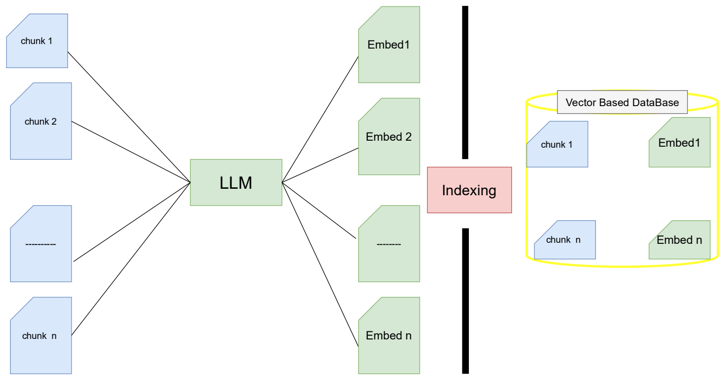
```
This step includes:

1. Extracting the text from the document

2. Chunking the text into documents
```

Open Book | Data Parsing



```
3. Indexing the documents into LanceDB
```

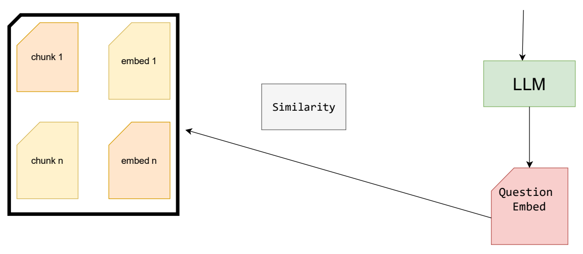
Open Book | Indexing



```
3) Search for similar documents
```

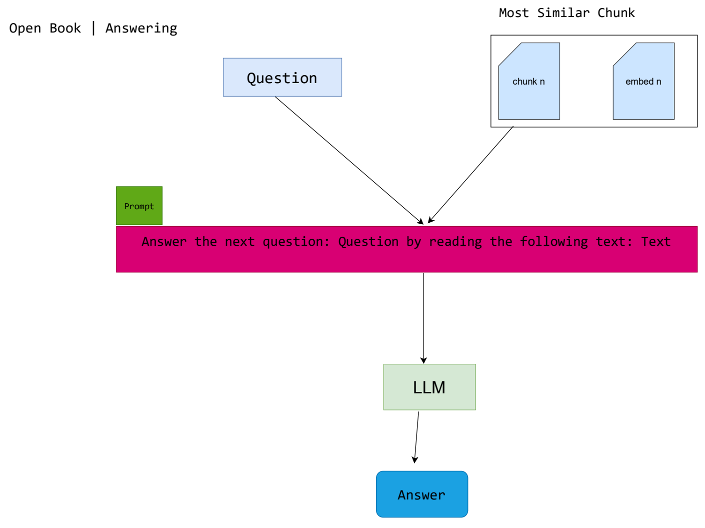
```
This step includes:

1. Convert query text to embeddings

2. Search for similar documents using the embeddings / or / keywords
```

Open Book | Asking | Semantic Search



4) Get the answer

1. Convert query text to embeddings
2. Search for similar documents using the embeddings / or / keywords
3. Create a prompt including the query and the similar documents
4. Pass the prompt to the LLM model to get the answer



✅ 1. System Requirements

Python 3.9+

- **FastAPI** (for API backend)
- **MongoDB** or any NoSQL DB (for storing metadata, projects, chunks)
- **Qdrant** (vector DB — self-hosted or cloud)
- **Ngrok** (for secure tunneling OpenAI call if local)
- **OpenAI API Key** (for answer generation)
- **Gemini Embedding API or local embedding service** (to convert text to vectors)

✅ 2. Python Package Requirements

```
fastapi==0.110.2
uvicorn[standard]==0.29.0  #    fastapi run as web server
python-multipart==0.0.9  # more file
python-dotenv==1.0.1
pydantic-settings==2.2.1
aiofiles==23.2.1
langchain==0.1.20
PyMuPDF==1.24.3
motor==3.4.0
pydantic-mongo==2.3.0
openai==1.35.13

qdrant-client==1.10.1
google-cloud-aiplatform>=1.44.0
google-generativeai>=0.4.1

# google-cloud-aiplatform google-auth requests
# pip install google-cloud-aiplatform
# google-generativeai>=0.3.0
configparser>=5.3.0
tenacity>=8.2.3
```