RAG System Implementation Guide

Overview

This RAG system consists of two main components:

- `vector.py`: Handles document processing and vector database operations
- `llm.py`: Handles LLM response generation using Mistral API

Setup

1. Install Dependencies

```bash
pip install -r requirements.txt
```

2. Environment Variables

Create a `.env` file in your project root:

```
MISTRAL_API_KEY=your_mistral_api_key_here
```

3. Get Mistral API Key

- Sign up at https://console.mistral.ai/
- Create an API key
- Add it to your `.env` file

Implementation in main.py

Basic RAG Pipeline

python

```python
from vector import QdrantManager
from llm import MistralLLM

def rag_pipeline(query: str, collection_name: str = "docs"):
    """
    Complete RAG pipeline implementation

    Args:
        query: User's question
        collection_name: Qdrant collection name

    Returns:
        Generated response string
    """
    # Step 1: Initialize components
    vector_manager = QdrantManager(collection_name=collection_name)
    llm = MistralLLM()

    # Step 2: Retrieve relevant documents
    search_results = vector_manager.search(
        query=query,
        limit=5,  # Number of relevant chunks to retrieve
        score_threshold=0.3  # Minimum similarity score (optional)
    )

    # Step 3: Generate response using LLM
    response = llm.generate_response(query, search_results)

    return response

# Usage
response = rag_pipeline("What is the main topic of the document?")
print(response)
```

Document Processing (One-time Setup)

python

```python
from vector import QdrantManager

def process_and_store_documents(file_paths: list, collection_name: str = "docs"):
    """
    Process and store documents in vector database

    Args:
        file_paths: List of document file paths
        collection_name: Qdrant collection name
    """
    vector_manager = QdrantManager(collection_name=collection_name)

    for file_path in file_paths:
        print(f"Processing: {file_path}")

        # Extract text chunks from document
        chunks = vector_manager.process_file(file_path)

        if chunks:
            # Store in vector database
            success = vector_manager.store_documents(chunks)
            if success:
                print(f"Successfully stored {len(chunks)} chunks from {file_path}")
            else:
                print(f"Failed to store chunks from {file_path}")
        else:
            print(f"No content extracted from {file_path}")

# Usage - run this once to index your documents
document_files = [
    "path/to/document1.pdf",
    "path/to/document2.docx",
    "path/to/document3.txt"
]
process_and_store_documents(document_files)
```

Advanced Usage