**QA Bot with FastAPI and Streamlit**

**Project Overview**

This project implements a Question Answering (QA) bot that uses Cohere for natural language understanding and Pinecone for vector storage. Users can upload PDF documents, and the bot will answer questions based on the contents of those documents. The project is structured as a FastAPI backend for handling document processing, embeddings, and query logic, and a Streamlit frontend that allows user interaction through an intuitive interface.

**Key Features :**

- Upload a PDF and extract its content.

- Store document embeddings in Pinecone .

- Use Cohere to answer user queries based on the uploaded document.

- Serve multiple users concurrently through FastAPI's asynchronous capabilities.

- Containerized with Docker for easy deployment.

**Project Structure**

- backend/main.py : Contains the logic for document processing, embedding storage, and question answering.

- frontend/app.py : Implements the user interface using Streamlit.

- Dockerfile : Docker configuration to containerize both backend and frontend.

- docker-compose.yml : Manages the orchestration of both backend and frontend containers.

Setup Instructions

Prerequisites

- Python 3.9+

- Docker and Docker Compose installed on your machine.

- Accounts with Cohere and Pinecone for API keys.

**Installation**

1. Clone the repository:

bash

git clone <https://github.com/your-repo/qa-bot.git>

cd qa-bot

2. Add your API Keys :

- Get your Pinecone API key from [Pinecone.io](https://www.pinecone.io/).

- Get your Cohere API key from [Cohere.ai](https://cohere.ai/).

- Set these keys in your environment or directly in the code (e.g., replace `"YOUR\_PINECONE\_API\_KEY"` and `"YOUR\_COHERE\_API\_KEY"`).

**Backend and Frontend Dependencies**

If you want to run the project locally without Docker, follow these steps:

**Backend**

1. Navigate to the backend directory and install dependencies:

```bash

cd backend

pip install -r requirements.txt

```

2. Run the FastAPI server:

```bash

uvicorn main:app --reload

```

Frontend

1. Navigate to the frontend directory and install dependencies:

```bash

cd frontend

pip install -r requirements.txt

```

2. Run the Streamlit app:

```bash

streamlit run app.py

```

**Docker Setup**

To simplify deployment, you can run the entire project using Docker and Docker Compose.

1. Build and run the project using Docker Compose:

```bash

docker-compose up --build

```

2. Once the containers are up and running:

- Streamlit frontend will be available at: `http://localhost:8501`

- FastAPI backend will be running at: `http://localhost:8000`

**Usage**

Uploading a PDF Document

1. Open the Streamlit app at `http://localhost:8501`.

2. Upload a PDF document using the "Upload a PDF document" button.

3. The document is processed in the backend, and the text is split into chunks. Each chunk is embedded and stored in Pinecone for later retrieval.

**Asking Questions**

1. After the document is uploaded, enter a question in the input field.

2. Press the "Get Answer" button to submit the query.

3. The query is sent to the FastAPI backend, which retrieves the most relevant document chunks from Pinecone and uses Cohere to generate an answer.

**Endpoints**

- POST `/upload/` : Accepts a PDF file and processes it, storing embeddings in Pinecone.

- Request : Upload a file (`multipart/form-data`).

- Response : Success message upon successful document processing.

- POST `/query/` : Accepts a user question and retrieves an answer based on the document embeddings.

- Request : JSON body with a `question` string.

- Response : The answer generated by Cohere.

- GET `/health/` : Simple health check endpoint.

- Response : API health status.

**Technical Details**

**Backend**

- FastAPI : Powers the backend, offering APIs to process PDF documents, store embeddings in Pinecone, and query the document.

- Pinecone : Used for storing vector embeddings of document chunks for fast and efficient retrieval.

- Cohere : Responsible for generating human-like answers to user queries based on the context retrieved from Pinecone.

**Frontend**

- Streamlit : Provides an interactive web interface where users can upload documents and ask questions.

- The frontend interacts with the backend via HTTP POST requests using the `requests` library.

**Document Processing**

1. The uploaded PDF is split into text chunks (500 characters each) for accurate embedding generation.

2. Each chunk is embedded using Cohere's embedding model and stored in Pinecone with the chunk’s text as metadata.

**Query Processing**

1. When a user submits a question, the question is embedded using Cohere’s model.

2. The embedded query is sent to Pinecone, which returns the top `k` (e.g., 3) most relevant document chunks based on vector similarity.

3. The top chunks are then used to generate an answer via Cohere’s language model.

**Future Enhancements**

1. Scalability :

- Use cloud services (e.g., AWS, GCP) to deploy the FastAPI and Streamlit services.

- Scale the backend to handle larger documents and multiple users concurrently.

2. Optimized Querying :

- Implement more sophisticated document chunk ranking and filtering to ensure better contextual answers.

3. Improved User Interface :

- Add more user-friendly error handling and loading states to the Streamlit frontend.

**Conclusion**

This project demonstrates how to build a modular and scalable QA bot using state-of-the-art tools like FastAPI , Streamlit , Cohere , and Pinecone . The combination of these tools allows for efficient document processing, fast query retrieval, and accurate answer generation, all accessible through a clean web interface.