Smart Document Q&A API — Project Documentation

# 📌 Overview

This project showcases a GenAI-powered backend service that allows users to:  
1. Upload a text-based document  
2. Ask natural language questions  
3. Get contextual answers based on the uploaded document  
  
It demonstrates how to combine FastAPI with Azure OpenAI to create a lightweight and effective internal document assistant.

# 🧠 Tech Stack

- FastAPI: Backend framework used to create REST endpoints  
- Azure OpenAI: Powers the question-answering logic using a GPT model deployed on Azure  
- Swagger UI: Provides an interactive interface for API testing  
- .env Variables: Secure handling of API keys and deployment configs  
- Pydantic: Request validation for incoming question payloads

# ⚙️ Features

/upload Endpoint  
- Accepts `.txt` files via POST  
- Reads and stores the content for use in the next step  
  
/ask Endpoint  
- Accepts a question string  
- Sends the document content and user question as context to Azure OpenAI  
- Returns a concise and relevant answer generated by the LLM

# 🧪 Example Flow

1. User uploads `company\_policy.txt` using Swagger or Postman.  
2. They then ask: "What is the company's remote work policy?"  
3. The model analyzes the document and responds with a contextually accurate answer.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black and white striped background

AI-generated content may be incorrect.

# 🧱 Architecture Flow

User → FastAPI Upload API → Document Stored in Memory  
 ↓  
User → FastAPI Ask API → Prompt created with Context + Question  
 ↓  
Azure OpenAI (via API) → LLM Generates Answer  
 ↓  
Answer → Returned to User

# 🧑‍💼 Why It’s Valuable for Business

✅ Knowledge On-Demand: Enables quick access to internal document insights without manual searching  
✅ Scalable Use Cases: Legal docs, HR policies, sales playbooks, SOPs, and more  
✅ Easy Integration: Can be embedded into web apps, chatbots, or internal tools  
✅ Cloud-Ready: Can be containerized and deployed on Azure infrastructure

# 🚀 Next Steps (Enhancements You Can Propose)

- Add support for PDFs (e.g. PyMuPDF or pdfminer.six)  
- Upload document to Azure Blob instead of local memory  
- Extend to allow multi-document search and ranking  
- Add authentication for secure internal access  
- Build a simple frontend using React or Streamlit