

SOURCE CODE:

```
from fastapi import FastAPI, UploadFile, File, Body, status
from pydantic import BaseModel
from PyPDF2 import PdfReader
from typing import List
import faiss
import numpy as np
import sqlite3
from datetime import datetime
import os

# CONFIG
EMBEDDING_PROVIDER = "sentence_transformers"
# options: "sentence_transformers", "huggingface", "openai"

# APP
app = FastAPI( title="Document Q&A RAG Service", version="1.0.0")

# VECTOR DB
EMBEDDING_DIM = 384
vector_index = faiss.IndexFlatL2(EMBEDDING_DIM)
stored_chunks: List[dict] = []

# METADATA DB
conn = sqlite3.connect("metadata.db", check_same_thread=False)
cursor = conn.cursor()
cursor.execute(""" CREATE TABLE IF NOT EXISTS documents ( id INTEGER PRIMARY KEY AUTOINCREMENT, filename TEXT, upload_time TEXT ) """)
conn.commit()
```

```
# EMBEDDING MODELS

if EMBEDDING_PROVIDER == "sentence_transformers":
    from sentence_transformers import SentenceTransformer
    embedding_model = SentenceTransformer("all-MiniLM-L6-v2")

if EMBEDDING_PROVIDER == "huggingface":
    from transformers import AutoTokenizer, AutoModel
    import torch
    tokenizer = AutoTokenizer.from_pretrained("sentence-transformers/all-MiniLM-L6-v2")
    hf_model = AutoModel.from_pretrained("sentence-transformers/all-MiniLM-L6-v2")

if EMBEDDING_PROVIDER == "openai":
    from openai import OpenAI
    client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

def get_embeddings(texts: List[str]) -> List[List[float]]:
    if EMBEDDING_PROVIDER == "sentence_transformers":
        return embedding_model.encode(texts).tolist()

    if EMBEDDING_PROVIDER == "huggingface":
        inputs = tokenizer(texts, padding=True, truncation=True, return_tensors="pt")
        with torch.no_grad():
            outputs = hf_model(**inputs)
            return outputs.last_hidden_state.mean(dim=1).tolist()

    if EMBEDDING_PROVIDER == "openai":
        response = client.embeddings.create( model="text-embedding-3-small", input=texts )
        return [item.embedding for item in response.data]
```

```
raise ValueError("Invalid embedding provider")

# HELPERS

def extract_text(file: UploadFile) -> str:
    if file.filename.endswith(".pdf"):

        reader = PdfReader(file.file)

        return " ".join(page.extract_text() for page in reader.pages)

    return file.file.read().decode("utf-8")

def chunk_text(text: str, size: int = 500) -> List[str]:
    words = text.split()

    return [" ".join(words[i:i + size]) for i in range(0, len(words), size)]


# SCHEMAS

class QueryRequest(BaseModel):
    question: str

class Source(BaseModel):
    filename: str
    content: str

class QueryResponse(BaseModel):
    question: str
    answer: str
    sources: List[Source]

# ENDPOINTS

@app.post("/upload", status_code=status.HTTP_201_CREATED)
async def upload_document(file: UploadFile = File(...)):
```

```
text = extract_text(file)

chunks = chunk_text(text)

embeddings = get_embeddings(chunks)

vector_index.add(np.array(embeddings))

for chunk in chunks:

    stored_chunks.append({

        "text": chunk,

        "filename": file.filename

    })

cursor.execute(

    "INSERT INTO documents (filename, upload_time) VALUES (?, ?)",

    (file.filename, datetime.now().isoformat())

)

conn.commit()

return {

    "message": "Document uploaded successfully",

    "filename": file.filename,

    "chunks_stored": len(chunks)

}

@app.post("/query", response_model=QueryResponse)

async def query_document(payload: QueryRequest = Body(...)):

    question_embedding = get_embeddings([payload.question])[0]

    _, indices = vector_index.search(np.array([question_embedding]), k=3)

    sources = []

    context = ""

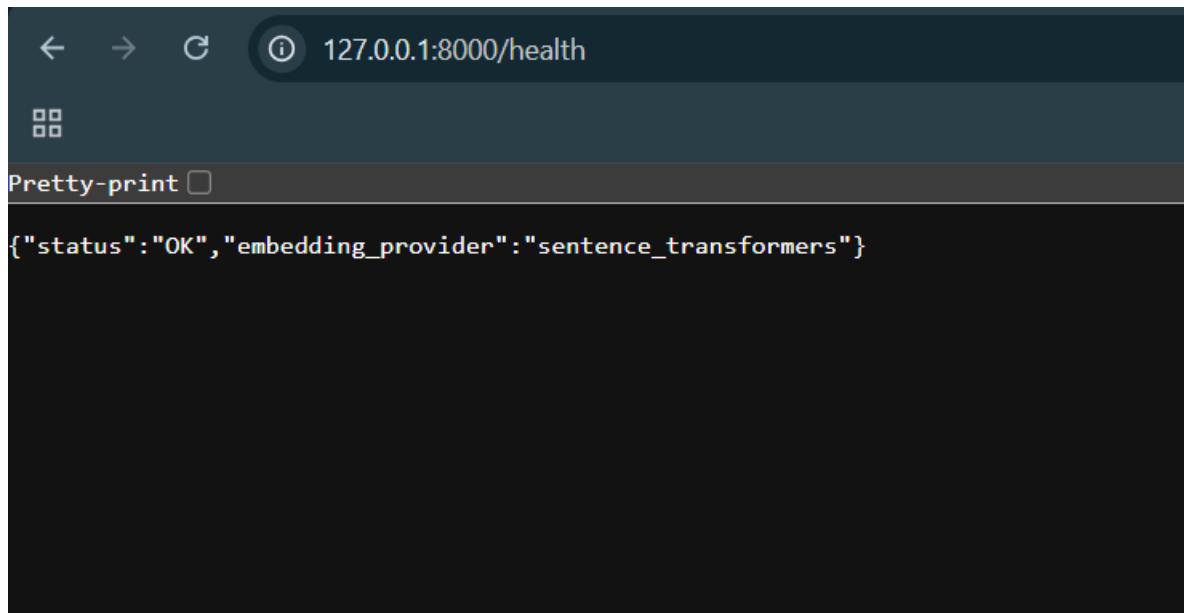
    for i in indices[0]:

        chunk = stored_chunks[i]

        context += chunk["text"] + " "
```

```
sources.append(  
    Source(  
        filename=chunk["filename"],  
        content=chunk["text"]  
    )  
)  
  
answer = f"Answer based on document context:\n{context}"  
  
return QueryResponse(  
    question=payload.question,  
    answer=answer,  
    sources=sources  
)  
  
  
@app.get("/health")  
def health():  
    return {  
        "status": "OK",  
        "embedding_provider": EMBEDDING_PROVIDER  
    }
```

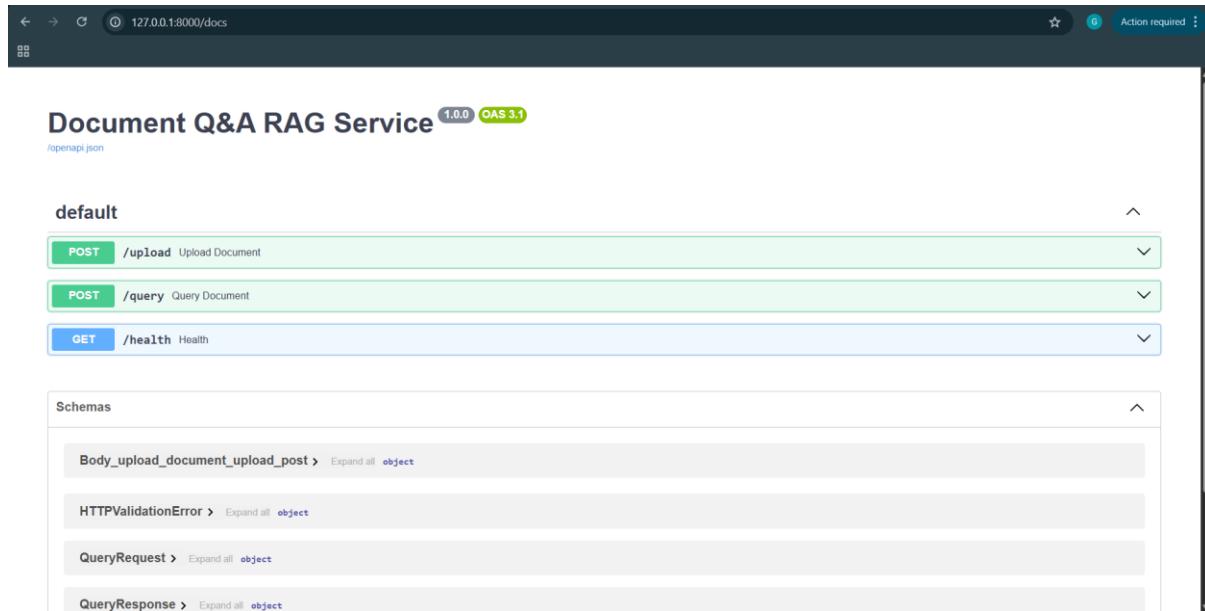
Outputs:



```
← → ⌂ ⓘ 127.0.0.1:8000/health
Pretty-print □

{"status": "OK", "embedding_provider": "sentence_transformers"}
```

With swagger



Document Q&A RAG Service 1.0.0 OAS 3.1

/openapi.json

default

- POST** /upload Upload Document
- POST** /query Query Document
- GET** /health Health

Schemas

- Body_upload_document_upload_post > Expand all object
- HTTPValidationError > Expand all object
- QueryRequest > Expand all object
- QueryResponse > Expand all object

POST /upload Upload Document

Parameters

No parameters

Request body required

file * required Choose File Resume-GUTTA_BHARATH-Valzo-Soft-Solutions.pdf

multipart/form-data

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/upload' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@Resume-GUTTA_BHARATH-Valzo-Soft-Solutions.pdf;type=application/pdf'
```

Request URL

```
http://127.0.0.1:8000/upload
```

Server response

Code Details

201 Response body

```
{
  "message": "Document uploaded successfully",
  "filename": "Resume-GUTTA_BHARATH-Valzo-Soft-Solutions.pdf",
  "chunks_stored": 1
}
```

Download

Response headers

```
content-length: 123
content-type: application/json
date: Wed,14 Jan 2026 17:00:54 GMT
server: unicorn
```

Responses

Code Description Links

201 Successful Response No links

Media type application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

422 Validation Error No links

Media type application/json

Here 201 means ok(successful).

After uploading the any file in the /upload. Then check the /query.

POST /query Query Document

Parameters

No parameters

Request body required

application/json

```
{
  "question": "What tells this about?"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/query' \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
    "question": "What tells this about?"
}'
```

Request URL

```
http://127.0.0.1:8000/query
```

Server response

Code

Details

200

Response body

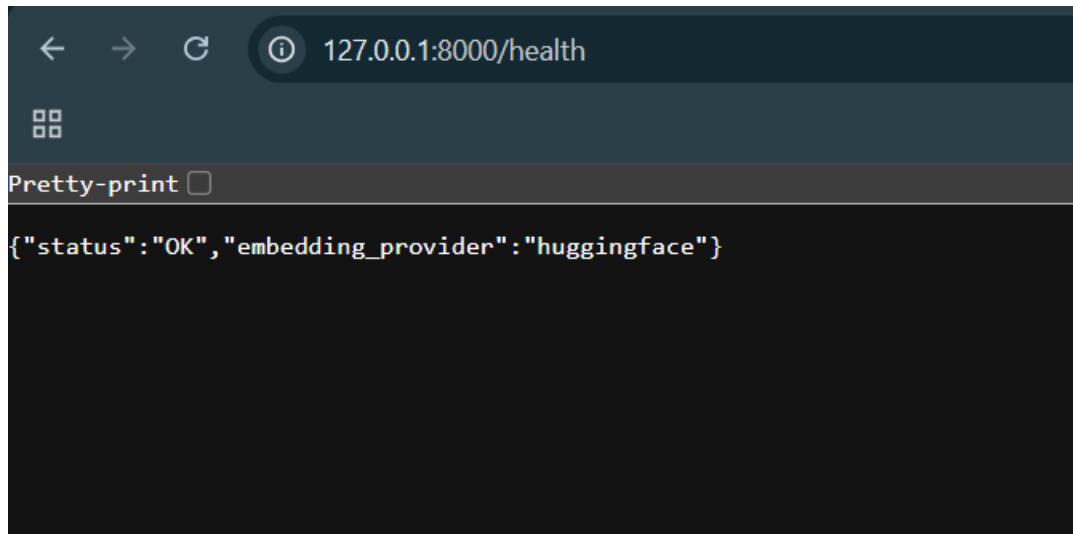
```
{
  "question": "What tells this about?",
  "answer": "Answer based on document context:\nGUTTA_BHARATH 2206030929cserv@gmail.com +91 9832553852 • Chittoor, Andhra Pradesh • https://www.linkedin.com/in/bharath-gutta-41880427b/ SUMMARY Highly motivated Computer Science Engineering graduate specializing in Data Science, Analytics, and Business Intelligence, with hands-on experience in data analysis, ETL processes, dashboard development, and predictive analytics. Strong ability to translate business requirements into analytical solutions, uncover trends and anomalies, and communicate insights effectively to stakeholders. Proficient in Python, SQL, Power BI, and cloud-based data platforms. Experienced in advanced analytics, machine learning, and data engineering. Eager to learn, innovate, and thrive in dynamic, ambiguous environments. EXPERIENCE AICTE-EduSkills (Virtual), Data Engineer Intern Jul 2023 - Oct 2023 • Built and maintained ETL pipelines using Python and SQL for structured datasets. • Performed data ingestion, transformation, and validation to ensure reliable data flow. • Gained exposure to cloud-based data workflows and analytics environments. • Assisted in documenting data processes, standards, and protocols. Elevate Labs, Data Analyst Intern Nov 2025 - Present • Perform data collection, cleaning, and exploratory data analysis to support analytical projects. • Developed interactive dashboards and reports using Power BI to communicate insights effectively. • Identify trends, anomalies, and risks in datasets and present findings to stakeholders. EDUCATION Kiran English Medium School, Tirupati SSL CGPA: 9.0 Sri Bhavishya Junior College, Vijayawada Intermediate CGPA: 8.0 KL University, Vijayawada B.Tech • Computer Science Engineering LICENSES & CERTIFICATIONS Sales Force certified AI Associate Sales Force • Issued Oct 2024 Certified Essentials Automation Professional Issued Jul 2024 SKILLS Python • SQL • EDA • Descriptive & Predictive Analytics • Front-end Development • ETL • Data Cleaning • Data Transformation • Power BI • Tableau • Dashboarding & Reporting HONORS & AWARDS Published Research Paper on 'Predictive Analysis of Stock Price Based on Various Trends and Advertising Impact' IIMKET Journal • GUTTA_BHARATH • Email: gutta_bharath2206030929cserv@gmail.com • LinkedIn: https://www.linkedin.com/in/gutta-bharath-41880427b/ SUMMARY Highly motivated Computer Science Engineering graduate specializing in Data Science, Analytics, and Business Intelligence, with hands-on experience in data analysis, ETL processes, dashboard development, and predictive analytics. Strong ability to translate business requirements into analytical solutions, uncover trends and anomalies, and communicate insights effectively to stakeholders. Experienced in Python, SQL, Power BI, and cloud-based data workflows, with a keen interest in advanced analytics, machine learning, and data engineering. Eager to learn, innovate, and thrive in dynamic, ambiguous environments. EXPERIENCE AICTE-EduSkills (Virtual), Data Engineer Intern Jul 2023 - Oct 2023 • Built and maintained ETL pipelines using Python and SQL for structured datasets. • Performed data ingestion, transformation, and validation to ensure reliable data flow. • Gained exposure to cloud-based data workflows and analytics environments. • Assisted in documenting data processes, standards, and protocols. Elevate Labs, Data Analyst Intern Nov 2025 - Present
```

If we interrupt b/w the /upload and /query. It throws logical error...

For HuggingFace

→ We have to change embedding_provider to huggingface

Make sure you installed torch. If not use this → python -m pip install torch



```
127.0.0.1:8000/health
Pretty-print □
{"status": "OK", "embedding_provider": "huggingface"}
```

OpenAI

Make sure you whether installed → python -m pip install openai

For checking the version

```
PS C:\Users\gutta\Downloads\FastAPI> python -c "import openai; print(openai.__version__)"
2.15.0
PS C:\Users\gutta\Downloads\FastAPI>
```

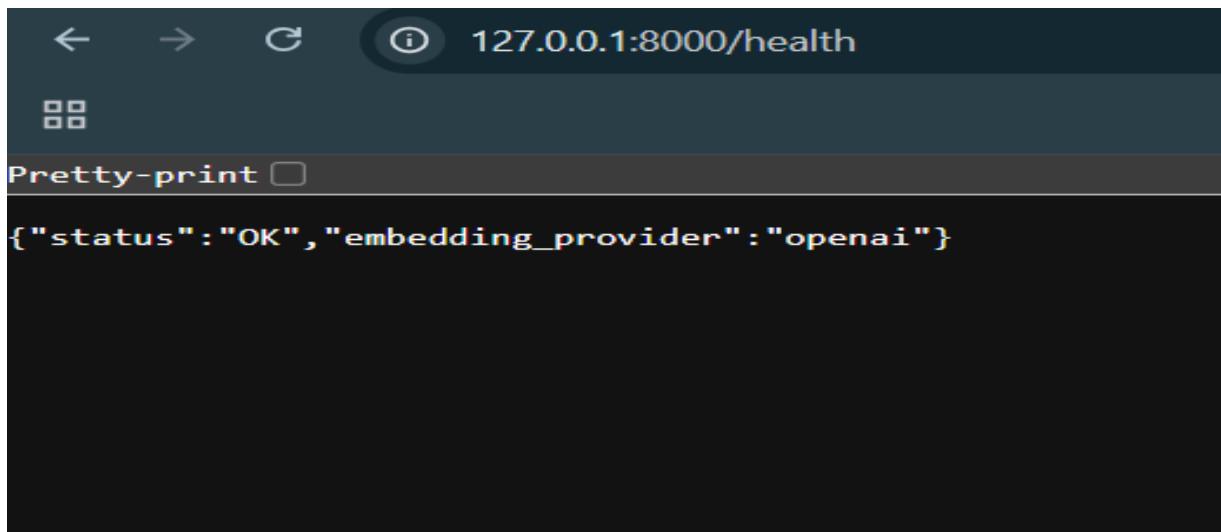
After

```
PS C:\Users\gutta\Downloads\FastAPI> setx OPENAI_API_KEY "your_openai_api_key_here"
SUCCESS: Specified value was saved.
PS C:\Users\gutta\Downloads\FastAPI>
FastAPI > 🐍 main.py
```

API keys

In Powershell

```
PS C:\Users\gutta> $env:OPENAI_API_KEY="skxxxxxxxxxxxxxxxxxxxxxx"
PS C:\Users\gutta> python -c "import os; print(os.getenv('OPENAI_API_KEY'))"
skxxxxxxxxxxxxxxxxxxxxxx
PS C:\Users\gutta>
```



A screenshot of a web browser window displaying a JSON response. The address bar shows the URL `127.0.0.1:8000/health`. Below the address bar, there are navigation icons (back, forward, search) and a refresh button. A "Pretty-print" checkbox is checked. The main content area displays the following JSON object:

```
{"status": "OK", "embedding_provider": "openai"}
```