```
!pip install transformers accelerate torch sentencepiece faiss-cpu beautifulsoup4 requests tqdm --quiet
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.6/23.6 MB 81.4 MB/s eta 0:00:00
```

```
import requests
from bs4 import BeautifulSoup
from tqdm import tqdm

import numpy as np
import faiss

from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline
from sentence_transformers import SentenceTransformer
```

```
# List of webpages to be scraped
urls = [
    "https://www.ideas-tih.org/",
    "https://www.ideas-tih.org/about-us",
    "https://www.ideas-tih.org/activities",
    "https://www.ideas-tih.org/s-projects-basic",
    "https://www.ideas-tih.org/events",
    "https://www.ideas-tih.org/career",
    "https://www.ideas-tih.org/media",
    "https://www.ideas-tih.org/internship",
    "https://www.ideas-tih.org/education",
    "https://www.ideas-tih.org/staff",
    "https://www.ideas-tih.org/staff",
    "https://www.ideas-tih.org/autumninternship2025",
    "https://www.isical.ac.in/about/about-institute",
    "https://www.isical.ac.in/content/timeline-0",
    "https://www.linkedin.com/company/ideastih/about/",
    "https://en.wikipedia.org/wiki/Indian_Statistical_Institute",
    "https://www.ideas-tih.org/initiatives"
]

def scrape_url(url):
    try:
        r = requests.get(url, timeout=10)
        soup = BeautifulSoup(r.text, "html.parser")
        text = soup.get_text(separator=" ", strip=True)
        print(f"✅ Scraped: {url} (chars: {len(text)})")
        return text
    except Exception as e:
        print(f"⚠️ Failed: {url} — {e}")
        return ""

corpus = "\n\n".join([scrape_url(u) for u in urls])
print("\n🧾 Total corpus size:", len(corpus), "characters")
```

```
✅ Scraped: https://www.ideas-tih.org/ (chars: 2441)
✅ Scraped: https://www.ideas-tih.org/about-us (chars: 1149)
✅ Scraped: https://www.ideas-tih.org/activities (chars: 1465)
✅ Scraped: https://www.ideas-tih.org/s-projects-basic (chars: 3569)
✅ Scraped: https://www.ideas-tih.org/events (chars: 2051)
✅ Scraped: https://www.ideas-tih.org/career (chars: 25)
✅ Scraped: https://www.ideas-tih.org/media (chars: 1189)
✅ Scraped: https://www.ideas-tih.org/internship (chars: 4278)
✅ Scraped: https://www.ideas-tih.org/education (chars: 1671)
✅ Scraped: https://www.ideas-tih.org/staff (chars: 2044)
✅ Scraped: https://www.ideas-tih.org/staff (chars: 2044)
✅ Scraped: https://www.ideas-tih.org/autumninternship2025 (chars: 4872)
⚠️ Failed: https://www.isical.ac.in/about/about-institute — HTTPSConnectionPool(host='www.isical.ac.in', port=443): Max retries exceeded w
⚠️ Failed: https://www.isical.ac.in/content/timeline-0 — HTTPSConnectionPool(host='www.isical.ac.in', port=443): Max retries exceeded with
✅ Scraped: https://www.linkedin.com/company/ideastih/about/ (chars: 0)
✅ Scraped: https://en.wikipedia.org/wiki/Indian_Statistical_Institute (chars: 91)
✅ Scraped: https://www.ideas-tih.org/initiatives (chars: 25)

🧾 Total corpus size: 26946 characters
```

```
def chunk_text(text, chunk_size=600):
    words = text.split()
    chunks = [" ".join(words[i:i+chunk_size]) for i in range(0, len(words), chunk_size)]
    return chunks

chunks = chunk_text(corpus)
print(f"📄 Total Chunks: {len(chunks)}")
```

```
📄 Total Chunks: 7
```

```
embed_model = SentenceTransformer("all-MiniLM-L6-v2")

print("🔍 Generating embeddings...")
embeddings = embed_model.encode(chunks, convert_to_numpy=True, show_progress_bar=True)

embeddings = np.array(embeddings).astype("float32")
print("✅ Embeddings shape:", embeddings.shape)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100% 349/349 [00:00<00:00, 5.36kB/s]
config_sentence_transformers.json: 100% 116/116 [00:00<00:00, 3.24kB/s]
README.md: 10.5k/? [00:00<00:00, 217kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 2.61kB/s]
config.json: 100% 612/612 [00:00<00:00, 18.0kB/s]
model.safetensors: 100% 90.9M/90.9M [00:01<00:00, 118MB/s]
tokenizer_config.json: 100% 350/350 [00:00<00:00, 19.4kB/s]
vocab.txt: 232k/? [00:00<00:00, 4.70MB/s]
tokenizer.json: 466k/? [00:00<00:00, 9.24MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 8.68kB/s]
config.json: 100% 190/190 [00:00<00:00, 14.8kB/s]
🔍 Generating embeddings...
Batches: 100% 1/1 [00:00<00:00, 1.18it/s]
✅ Embeddings shape: (7, 384)
```

```
index = faiss.IndexFlatL2(embeddings.shape[1])
index.add(embeddings)

print("🎉 FAISS index built successfully!")
print("📦 Total vectors stored:", index.ntotal)
```

```
🎉 FAISS index built successfully!
📦 Total vectors stored: 7
```

```
import torch

model_name = "Qwen/Qwen2.5-1.5B-Instruct"

print("⏳ Loading Qwen2.5-1.5B (this may take 20–40 seconds)...")

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto"
)

gen = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    max_new_tokens=300,
    temperature=0.4
)

print("Qwen2.5-1.5B loaded successfully!")
```

```
⏳ Loading Qwen2.5-1.5B (this may take 20–40 seconds)...
tokenizer_config.json: ████ 7.30k/? [00:00<00:00, 172kB/s]
```

```
def retrieve(query, k=4):
    # Convert query → embedding → FAISS search
    q_emb = embed_model.encode([query], convert_to_numpy=True).astype("float32")
    distances, indices = index.search(q_emb, k)

    retrieved = []
    for i in indices[0]:
        if i < len(chunks):
            retrieved.append(chunks[i])
    return retrieved
```

```
Device set to use cuda:0
```

```
def build_prompt(query):
    retrieved_chunks = retrieve(query)
    context = "\n\n".join(retrieved_chunks)

    prompt = f"""
You are **IDEAS-TIH Assistant**, an AI designed to provide accurate, factual, and concise information
based strictly on the available documents and website content of IDEAS-TIH (Institute of Data Engineering, Analytics and Science

### Your Duties:
1. **Use ONLY the context provided below** to answer the user's question.
2. If relevant information is available, summarize it clearly and professionally.
3. If the context does not contain the required information, respond with:
   **"I could not find this information in the provided data."**
4. Do NOT make assumptions, invent details, or hallucinate.
5. Maintain a helpful, polite, and formal tone.
6. Include ONLY factual content from the context.

---

### CONTEXT EXTRACTED FROM IDEAS-TIH WEBSITE:
{context}

---

### USER QUESTION:
{query}

---

### ASSISTANT ANSWER (based strictly on the context above):
"""
    return prompt
```

```
def answer_query(query):
    prompt = build_prompt(query)
    full_generated_text = gen(prompt)[0]["generated_text"]

    answer_start_marker = "### ASSISTANT ANSWER (based strictly on the context above):"

    reply = ""
    if answer_start_marker in full_generated_text:
        reply = full_generated_text.split(answer_start_marker)[-1].strip()
    else:

        if full_generated_text.startswith(prompt):
            reply = full_generated_text[len(prompt):].strip()
        else:
            reply = full_generated_text.strip() # Last resort, return raw output

    return reply
```

```
print("\n🤖 IDEAS-TIH RAG Chatbot Ready!")
print("Type 'exit' to quit.\n")

while True:
    query = input("👤 You: ").strip()

    if query.lower() in ["exit", "quit"]:
        print("👋 Goodbye!")
        break

    try:
        reply = answer_query(query)
        print("🤖 Bot:", reply, "\n")
    except Exception as e:
```

```
        print("⚠️ Error:", e)
```

```
🤖 IDEAS-TIH RAG Chatbot Ready!
Type 'exit' to quit.

👤 You: What does IDEAS provide
🤖 Bot: IDEAS provides various certification programs, ongoing projects, events, career opportunities, and more. Specifically, they mentio

- **Certification Courses**: Offering skill development initiatives such as certification courses.
- **Ongoing Programs**: Including an upcoming "Autumn Internship 2025" focused on Data Science with additional emphasis on AI/ML and LLM.
- **Career Opportunities**: Providing career-related services and resources.
- **Events**: Hosting seminars, workshops, and webinars on topics like LLMs and agentic AI.
- **Job Positions**: Inviting applications for internships in Software Development and Data Analytics.
- **Collaborations**: Partnering with organizations like JMA for specific programs like "Business Analytics & Machine Learning for Profess
- **Media Coverage**: Sharing updates and news about their activities and achievements.

The organization aims to help users gain practical skills and mentorship in fields such as Data Science and Artificial Intelligence throug

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipython-input-3578533450.py in <cell line: 0>()
      3
      4 while True:
----> 5     query = input("👤 You: ").strip()
      6
      7     if query.lower() in ["exit", "quit"]:

                                    1 frames
/usr/local/lib/python3.12/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
   1217             except KeyboardInterrupt:
   1218                 # re-raise KeyboardInterrupt, to truncate traceback
-> 1219                 raise KeyboardInterrupt("Interrupted by user") from None
   1220             except Exception:
   1221                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```