how to create ai agent chatbot using langchain , crew ai and hugging Face to answer questions answer from my pdf notes with user interface

To build an **AI agent chatbot** using:

- □ **LangChain** — for managing documents and tools
- □ **CrewAI** — for agent collaboration
- □ **Hugging Face LLMs** — for local or hosted inference
- □ **PDF Q&A** — using FAISS and embeddings
- □ **User Interface** — with **Streamlit**

□ Project Overview

Goal: User uploads a PDF → Asks a question → AI agents (via CrewAI + LangChain + HF) answer based on the document.

□ Tools & Stack

Component	Tech Used
LLM	Hugging Face Transformers (e.g., `mistralai/Mistral-7B-Instruct-v0.2`)
Document Parsing	`LangChain` + `PyPDFLoader`
Vector Store	`FAISS` + `HuggingFaceEmbeddings`
Agents / Orchestration	`CrewAI`
UI	`Streamlit`

□ 1. Install Dependencies

pip install streamlit langchain faiss-cpu pypdf crewai transformers sentence-transformers langchain-community

🔹 2. Hugging Face Setup

We'll use a model like `mistralai/Mistral-7B-Instruct-v0.2`.

Install `transformers` and optionally `accelerate` if running locally.

pip install transformers accelerate

Hugging Face API Token (for hosted models)

export HUGGINGFACEHUB_API_TOKEN=your_hf_token

🔹 3. `app.py` — Full Working App

```
import os
import streamlit as st
from langchain_community.document_loaders import PyPDFLoader
from langchain_community.vectorstores import FAISS
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain.llms import HuggingFaceHub
from crewai import Agent, Task, Crew

# ----------------------------
# Load PDF and Embed
# ----------------------------
@st.cache_resource
def create_vectorstore(pdf_path):
    loader = PyPDFLoader(pdf_path)
    documents = loader.load()
    embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-
MiniLM-L6-v2")
    vectorstore = FAISS.from_documents(documents, embeddings)
    return vectorstore

# ----------------------------
# Define Search Tool
# ----------------------------
class PDFSearchTool:
    def __init__(self, vectorstore):
        self.vectorstore = vectorstore

    def run(self, query):
```

```
        results = self.vectorstore.similarity_search(query, k=3)
        return "\n\n".join([doc.page_content for doc in results])

# ----------------------------
# Set up CrewAI agents
# ----------------------------
def create_crew(vectorstore, question):
    tool = PDFSearchTool(vectorstore)

    llm = HuggingFaceHub(
        repo_id="mistralai/Mistral-7B-Instruct-v0.2",
        model_kwargs={"temperature": 0.6, "max_new_tokens": 512}
    )

    researcher = Agent(
        role="Researcher",
        goal="Find relevant context in the PDF",
        tools=[tool],
        llm=llm,
        verbose=True
    )

    answerer = Agent(
        role="Answer Generator",
        goal="Provide accurate and helpful answers using context",
        llm=llm,
        verbose=True
    )

    task = Task(
        description=f"Use the PDF content to answer: {question}",
        agent=answerer
    )

    return Crew(
        agents=[researcher, answerer],
        tasks=[task],
        verbose=True
    )

# ----------------------------
# Streamlit UI
# ----------------------------
st.set_page_config("PDF Chatbot (HuggingFace + CrewAI)")
st.title("🧠 Ask Your PDF (CrewAI + Hugging Face)")
```

```python
uploaded_file = st.file_uploader("Upload your PDF", type="pdf")

if uploaded_file:
    pdf_path = os.path.join("pdf_notes", uploaded_file.name)
    os.makedirs("pdf_notes", exist_ok=True)
    with open(pdf_path, "wb") as f:
        f.write(uploaded_file.read())

    question = st.text_input("❓ Ask a question from your notes:")

    if question:
        with st.spinner("Thinking..."):
            vs = create_vectorstore(pdf_path)
            crew = create_crew(vs, question)
            response = crew.run(question)

        st.markdown("### ✅ Answer:")
        st.write(response)
```

❓ 4. Run It

```
streamlit run app.py
```

Model Options from Hugging Face

You can try other Hugging Face models in `repo_id` like:

- `tiiuae/falcon-7b-instruct`
- `meta-llama/Llama-2-7b-chat-hf` (slow unless optimized)
- `google/flan-t5-xl` (small and fast)

✅ Final Notes

- Hugging Face `transformers` models work via **`HuggingFaceHub`** (hosted) or via local inference (using `pipeline`).
- For a **local model**, use `transformers.pipeline()` with `langchain.llms.HuggingFacePipeline`.