**Task: Create a RAG-based Chatbot with Langchain, Django, and Streamlit using Open-Source LLM**

**Objective:**  
Develop a RAG-based chatbot that retrieves data from a local knowledge base (using Langchain for retrieval) and generates responses using an open-source LLM like GPT-J, GPT-Neo, or Bloom, all powered by Django Rest Framework and Streamlit for the frontend.

**Steps:**

1. **Set Up Django Backend (2-3 hours):**
   * Create a new Django project with Django Rest Framework.
   * Implement a simple API to send user input to the chatbot and get the response.
     + **API Endpoints:**
       - POST /api/chat/: Takes user input and returns the response from the chatbot.
   * Integrate Langchain with a vector database (such as FAISS or any other store for your documents).
     + Use Langchain’s VectorStore interface to fetch relevant information from your knowledge base (e.g., a set of pre-loaded documents or text).
   * Integrate an open-source LLM like GPT-J, GPT-Neo, or Bloom.
     + You can use libraries such as transformers from Hugging Face to load and interact with these models.
2. **Integrate Langchain (1 hour):**
   * Implement a custom Langchain retrieval pipeline that pulls in relevant documents from the local knowledge base based on user input.
   * Ensure the retrieved documents are used to augment the LLM’s response (i.e., provide context from the knowledge base).
3. **Integrate Open-Source LLM (1 hour):**
   * Use a framework like transformers to load an open-source model (e.g., GPT-J or GPT-Neo) and generate responses from it.
     + You can use Hugging Face’s pre-trained models for GPT-Neo or GPT-J.
   * Ensure that the input to the model is augmented with the retrieved knowledge from Langchain.
4. **Streamlit Frontend (1-2 hours):**
   * Set up Streamlit for the frontend.
     + Create a simple interface with a text input field for user queries.
     + Display the chat conversation, showing user input and bot response dynamically.
   * Call the Django API using Streamlit's built-in HTTP request functionality to send user queries and display responses.
5. **Testing and Debugging (1 hour):**
   * Test the chatbot by interacting with the UI to ensure that the chatbot retrieves context correctly and generates appropriate responses using the open-source LLM.
   * Debug any issues in the API, Langchain integration, LLM model, or the frontend interaction.

**Deliverables:**

* A Django backend API that connects with Langchain to retrieve data and generate responses using the open-source LLM.
* A working chatbot frontend in Streamlit.
* Basic error handling and logging.
* A working demo of the RAG chatbot with interaction flow.

**Notes:**

* Do not do hardcoding in your code
* Use design patterns for clean coding wherever applicable.