**📦 2. Create Virtual Environment**

**Open a terminal inside the chatbot/ folder:**

**# Create venv**

**python -m venv venv**

**# Activate venv**

**# On Windows:**

**venv\Scripts\activate**

**# On Mac/Linux:**

**source venv/bin/activate**

**📚 3. Install Dependencies**

**Install everything from requirements.txt:**

**pip install -r requirements.txt**

**This will download Flask, pandas, torch, and sentence-transformers.**

**⚠️ First time it runs, sentence-transformers will also download the all-MiniLM-L6-v2 model automatically (~90 MB).**

**▶️ 4. Run Locally with Flask**

**Inside your virtual environment:**

**python app.py**

**You should see something like:**

**\* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)**

**🌐 5. Open in Browser**

**Go to:**

**👉** [**http://127.0.0.1:5000**](http://127.0.0.1:5000)

**You’ll see the chatbot UI.  
Type a question like:**

* ***“When are you open?”***
* ***“How can I change my order?”***

**You should get smart answers from the FAQ CSV 🎉**

**🐳 6. (Optional) Run with Docker**

**If you want to test the Dockerized version:**

**# Build image**

**docker build -t chatbot .**

**# Run container**

**docker run -p 5000:5000 chatbot**

**Then open the same URL:** [**http://127.0.0.1:5000**](http://127.0.0.1:5000)

**✅ Now you can test locally.  
Once it works, you’ll be ready to deploy it to Railway/Heroku/Render.**

**🏆 My Recommendation for You**

Since your goals are:

* **Portfolio project (GitHub)** → needs to look professional.
* **Free (or very low cost)**.
* **Runs in cloud (Render/Railway)**.
* **FAQ + fallback chat**.

👉 Use this stack:

* **FAQ Layer**: simple pandas lookup (what you already have).
* **Fallback Chat Layer**:
  + Start with **LiteLLM + Hugging Face Inference API** (free tier, no GPU needed).
  + Example model: mistralai/Mistral-7B-Instruct-v0.2 (HF hosted).
* **Deployment**: Containerize with **Docker** → deploy on **Render** (750 free hours/month).

That way:

* You don’t pay for OpenAI credits.
* You don’t need a GPU server.
* You can demo a live **customer service chatbot** online for recruiters.
* You can later swap HF model → GPT-4o mini if you want extra quality.

**🟢 Approach 1 (LiteLLM + HF Inference API)**

* **FAQ Layer** → pandas/embeddings (what you already built).
* **Fallback Chat** → call Hugging Face’s hosted models (e.g., Mistral-7B) through **LiteLLM**, which gives you an OpenAI-like API interface.
* **Deployment** → Docker + Render.

**🏆 My Verdict**

Since you’re making a **portfolio project** (where clarity, professionalism, and maintainability matter more than raw speed):

👉 **Approach 1 (LiteLLM + HF Inference API)** is the **better choice**.

Perfect 👍 since you already have a **working chatbot** with your own **FAQ dataset (faq\_data.csv)**, **Sentence Transformers for embeddings**, and **UI styling**, you’re in a very good place.

Now, the **next steps** depend on your goal:

**🚀 Next Steps for Approach 1 (Embedding-based FAQ Chatbot)**

**1. Make Retrieval More Efficient**

* Instead of keeping embeddings in memory, store them in a **vector database**:
  + Free & Open Source: **FAISS (Facebook AI Similarity Search)**
  + Alternatives: Milvus, Weaviate, ChromaDB
* This makes searching faster and scalable if you add more FAQs later.

**2. Improve Query Matching**

* Add a **semantic similarity threshold**:
  + If the best match score is below a threshold (say 0.65), respond with something like:  
    *“Sorry, I don’t know the answer to that yet.”*
* Prevents irrelevant answers when the model guesses.

**3. Optional: Add an Open-Source Chat Model for fallback**

* You can plug in a **small LLaMA-2-7B / Mistral** model as a fallback when FAQ doesn’t have a good match.
  + Example flow:
    1. User asks → Check FAISS → If high similarity → return FAQ answer.
    2. Else → Forward to LLaMA/Mistral (running locally or via Hugging Face Inference API).

**4. Deployment Options**

* **Containerize with Docker** (so it works anywhere).
* Deploy on **Render / Railway / Fly.io / Hugging Face Spaces** (free tiers available).
* If using FAISS, bundle your embeddings + CSV with the container.

**5. Enhance Frontend**

* Add features like:
  + **Typing animation** for chatbot replies
  + **Dark/light theme toggle**
  + **Chat history persistence** (localStorage or backend DB)
  + **Feedback button** (thumbs up/down → helps you improve FAQ answers later)

**6. Logging & Monitoring**

* Log all user queries + matched FAQ.
* This helps:
  + Spot missing questions
  + Continuously improve your FAQ dataset

✅ So, the **very next step** I’d recommend:  
👉 Integrate **FAISS** (or ChromaDB if you want Python-friendly) to store & query embeddings instead of searching with cosine similarity in plain lists. That’ll make your chatbot much more scalable & production-ready.