**Please Read the Instructions**

**1. Before Deployment:**

**Step 1:** Replace the API keys with your own in project folder/backend/.env.

**2. If you want to run the project locally on your system:**

**Step 1:** Install these tools: VS Code, Ollama, Node.js, and Python.

**Step 2:** Open the project folder in VS Code. Open the terminal and navigate to the frontend folder. Run the command:

**npm install**

**npm start**

(Note: Before running the frontend, make sure you have replaced the backend URL in the .env file located in the frontend folder according to your local server settings.)

**Step 3:** Open another terminal in VS Code and navigate to the backend folder. Install the necessary packages using pip. Here's a list of imports you need:

**from flask import Flask, request, jsonify**

**from langchain\_community.document\_loaders import WebBaseLoader**

**from langchain.text\_splitter import RecursiveCharacterTextSplitter**

**from langchain\_community.embeddings import HuggingFaceEmbeddings**

**from langchain\_core.output\_parsers import JsonOutputParser**

**from langchain.schema import Document**

**from langchain\_qdrant import QdrantVectorStore**

**from groq import Groq**

**from dotenv import load\_dotenv**

**import ollama**

**import os**

**from flask\_cors import CORS**

**Step 4:** **Uncomment the following code:**

**# Step 3: Use Hugging Face model to generate a response based on the retrieved text**

**# response = ollama.chat(model="llama3.1", messages=[**

**# {**

**# "role": "user",**

**# "content": "Educate customers like a salesperson. Use simple bullet points like this: - Point one - Point two"**

**# },**

**# {**

**# "role": "Assistant",**

**# "content": retrieved\_text,**

**# }**

**# ])**

**# return response['message']['content'] if response else "There is no search!"**

**And comment out the following code:**

**# Groq model implementation**

**# response = client.chat.completions.create(**

**# messages=[**

**# {**

**# "role": "user",**

**# "content": "Educate customers like a salesperson, using clear sections and bullet points for easy readability."**

**# },**

**# {**

**# "role": "Salesman",**

**# "content": retrieved\_text,**

**# }**

**# ],**

**# model="llama3-8b-8192",**

**# )**

**# return response.choices[0].message.content if response else "There is no search!"**

**Step 5:** Run the command:

**python Rag.py**