Build and Deploy Your Own Custom Chatbot using Gradio + GROQ + Hugging Face

Objective:

By the end of this lab, students will:

- Learn how to use LLM APIs via HTTP (GROQ)
- Design and customize chatbot personalities and behavior
- Build and deploy a chatbot with a Gradio UI
- Publish their application on Hugging Face Spaces

Prerequisites:

- GitHub & Hugging Face account
- GROQ API key from https://console.groq.com
- Python basics, REST API familiarity

Step-by-Step Guide:

Step 1: Get Your GROQ API Key

1. Visit https://console.groq.com
2. Sign up and get an API key
3. Copy and keep it safe

Step 2: Create Project Files

app.py

Use the following **template code**, and **customize the SYSTEM_PROMPT** according to your chosen chatbot theme.

```
import gradio as gr
import os
import requests

# Load GROQ API key from environment (set it in Hugging Face secrets)
GROQ_API_KEY = os.environ.get("GROQ_API_KEY")

GROQ_API_URL = "https://api.groq.com/openai/v1/chat/completions"
MODEL_NAME = "llama3-8b-8192"  # Balanced and fast for Q&A bots

# 🔶 Customize this system prompt based on your bot's role
```

```
SYSTEM_PROMPT = """You are a friendly and helpful travel advisor.
You answer user questions about travel destinations, planning, and tips in a clear and engaging way."""

def query_groq(message, chat_history):
    headers = {
        "Authorization": f"Bearer {GROQ_API_KEY}",
        "Content-Type": "application/json"
    }

    messages = [{"role": "system", "content": SYSTEM_PROMPT}]
    for user, bot in chat_history:
        messages.append({"role": "user", "content": user})
        messages.append({"role": "assistant", "content": bot})
    messages.append({"role": "user", "content": message})

    response = requests.post(GROQ_, headers=headers, json={
        "model": MODEL_NAME,
        "messages": messages,
        "temperature": 0.7
    })

    if response.status_code == 200:
        reply = response.json()["choices"][0]["message"]["content"]
        return reply
    else:
        return f"Error {response.status_code}: {response.text}"

def respond(message, chat_history):
    bot_reply = query_groq(message, chat_history)
    chat_history.append((message, bot_reply))
    return "", chat_history

with gr.Blocks() as demo:
    gr.Markdown("## 🤖 Your Custom Chatbot (Powered by GROQ LLM)")
    chatbot = gr.Chatbot()
    msg = gr.Textbox(label="Ask a question")
    clear = gr.Button("Clear Chat")
    state = gr.State([])

    msg.submit(respond, [msg, state], [msg, chatbot])
    clear.click(lambda: ([], []), None, [chatbot, state])

demo.launch()
```

requirements.txt

```
gradio
requests
```

Step 3: Deploy on Hugging Face Spaces

1. Create a new **Gradio Space** on https://huggingface.co/spaces
2. Upload app.py and requirements.txt
3. Go to **Settings > Secrets** and add:
    - GROQ_API_KEY with your GROQ key

Step 4: Customize Your Chatbot

Every student must:

- Pick a unique chatbot theme:
    - Examples: Legal Advisor, Cooking Assistant, Fitness Coach, Programming Tutor, History Expert, Pet Care Helper, Language Translator, etc.
- Update the SYSTEM_PROMPT accordingly
- Give the chatbot a name and personality

Lab Task: Build Your Own AI Chatbot

Task Description:

Design, build, and deploy a chatbot based on a role or theme of your choice.

Task Checklist:

1. **[2 pts]** Choose a unique chatbot theme (not repeated among students).
2. **[6 pts]** Modify the system prompt to match your chatbot’s personality.
3. **[6 pts]** Deploy the chatbot to Hugging Face Spaces with correct functionality.
4. **[6 pts]** Add one UI improvement (e.g., dropdown to select mood/topic, slider for response length, etc.)
5. **[5 pts]** Submit:
    - Public Hugging Face Space link
    - Screenshot of chatbot working
    - Brief description (1 paragraph) about your bot's role

📤 **Submission:**

- Hugging Face Space link
- Screenshot of working chatbot
- 1–2 sentence description