In [1]:

```python
import gradio as gr
import requests
from datetime import datetime
import pandas as pd
import time

# Your existing OpenRouter API setup
OPENROUTER_API_KEY = "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"  # replace wit
OPENROUTER_API_URL = "https://openrouter.ai/api/v1/chat/completions"

SYSTEM_INSTRUCTION = """
You are a professional health assistant specialized in hair care and skin care.
Your task is to:
1. Identify the most likely common cause of the problem.
2. Suggest natural and safe home remedies that can be easily done at home.
3. Recommend mild over-the-counter medicines or topical treatments.
4. Clearly explain when and why the user should visit a doctor.
5. Use simple language and avoid medical jargon.
6. Never prescribe strong medication, injections, or controlled drugs.
7. If symptoms are severe, worsening, or unusual, immediately advise the user to seek medical attention.
Keep the response under 120 words.
"""

# Function to get AI solution (same as your code)
def get_ai_solution(problem_type, problem_description):
    payload = {
        "model": "meta-llama/llama-3.3-8b-instruct:free",
        "messages": [
            {"role": "system", "content": SYSTEM_INSTRUCTION},
            {"role": "user", "content": f"The user has a {problem_type} problem.\nDescription: {problem_descrip
        ]
    }

    headers = {
        "Authorization": f"Bearer {OPENROUTER_API_KEY}",
        "Content-Type": "application/json"
    }

    try:
        response = requests.post(OPENROUTER_API_URL, headers=headers, json=payload, timeout=20)
        response.raise_for_status()
        result = response.json()
        time.sleep(1)
        return result['choices'][0]['message']['content'].strip()
    except Exception as e:
        return f" OpenRouter API Error: {str(e)}"

# Function to save user data
def save_data(name, age, phone, location, problem_type, problem_description, ai_response):
    data = {
        "Timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "Name": name,
        "Age": age,
        "Phone": phone,
        "Location": location,
        "Problem_Type": problem_type,
        "Problem_Description": problem_description,
        "AI_Response": ai_response
    }
    df = pd.DataFrame([data])
    df.to_csv("health_chatbot_data.csv", mode='a', header=not pd.io.common.file_exists("health_chatbot_data.csv

# Function to handle Gradio input
def chatbot_interface(name, age, phone, location, problem_type, problem_description):
    ai_response = get_ai_solution(problem_type, problem_description)
    save_data(name, age, phone, location, problem_type, problem_description, ai_response)
    return ai_response

# Gradio inputs
with gr.Blocks() as demo:
    gr.Markdown("##  AI Health Assistant Chatbot")

    with gr.Row():
        name = gr.Textbox(label="Your Name")
        age = gr.Number(label="Your Age")

    with gr.Row():
        phone = gr.Textbox(label="Phone Number")
        location = gr.Textbox(label="Location")

    problem_type = gr.Radio(["Skin", "Hair"], label="Problem Type")
    problem_description = gr.Textbox(label="Describe your problem", lines=4)
```

```
    submit_btn = gr.Button("Get AI Recommendation")
    output = gr.Textbox(label="AI Suggestion", lines=6)

    submit_btn.click(chatbot_interface,
                     inputs=[name, age, phone, location, problem_type, problem_description],
                     outputs=output)

demo.launch()
```

```
* Running on local URL:  http://127.0.0.1:7860
* To create a public link, set `share=True` in `launch()`.
```

Out[1]:

In [ ]: