📘 Documentation: Medical AI Assistant with Gradio & Hugging Face

🔷 Overview

This application is a **medical AI assistant** built using **Gradio** for the UI and **Hugging Face Transformers** for natural language processing.
It uses the **IBM Granite 3.2B Instruct model** to:

- Predict possible medical conditions from symptoms.
- Suggest general treatment plans with

home remedies and guidelines.

⚠️ **Important Disclaimer**: This tool is **for informational purposes only** and must not replace professional medical advice. Users are advised to **consult a healthcare professional** for diagnosis and treatment.

🔷 Dependencies

Make sure the following Python packages are installed:

```
pip install gradio torch transformers
```

🔷 Code Breakdown

1. Model & Tokenizer Setup

```
model_name = "ibm-granite/granite-3.2-2b-
```

instruct" tokenizer = AutoTokenizer.from_pretrained(model_name) model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32, device_map="auto" if torch.cuda.is_available() else None)

- Loads the **Granite model** and tokenizer.
- Uses **GPU (float16)** if available, otherwise falls back to **CPU (float32)**.
- device_map="auto" automatically distributes the model across GPU(s).

If the tokenizer has no padding token, it sets:

tokenizer.pad_token = tokenizer.eos_token

2. Text Generation Function

```
def generate_response(prompt, max_length=1024): inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512) if torch.cuda.is_available(): inputs = {k: v.to(model.device) for k, v in inputs.items()} with torch.no_grad(): outputs = model.generate( **inputs, max_length=max_length, temperature=0.7, do_sample=True, pad_token_id=tokenizer.eos_token_id ) response = tokenizer.decode(outputs[0], skip_special_tokens=True) response = response.replace(prompt, "").strip() return response
```

- Tokenizes input prompt.

- Runs inference on GPU if available.
- Uses **temperature=0.7** (moderate randomness).
- Removes prompt from final output.

3. Disease Prediction

def disease_prediction(symptoms): prompt = f"""Based on the following symptoms, provide possible medical conditions and general medication suggestions. Always emphasize the importance of consulting a doctor for proper diagnosis. Symptoms: {symptoms} Possible conditions and recommendations: **IMPORTANT: This is for informational purposes only. Please consult a healthcare professional for proper diagnosis and treatment.**

Analysis:""" return generate_response(prompt, max_length=1200)

- Input: List of symptoms.