

Health Ai Code

```
import gradio as gr

import torch

from transformers import AutoTokenizer, AutoModelForCausalLM


# Load model and tokenizer

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

)


if tokenizer.pad_token is None:

    tokenizer.pad_token = tokenizer.eos_token


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
```

```
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.\n\nSymptoms: {symptoms}\n\nPossible conditions and  
recommendations:\n\n**IMPORTANT: This is for informational purposes only. Please consult a  
healthcare professional for proper diagnosis and treatment.**\n\nAnalysis:"
```

```
    return generate_response(prompt, max_length=1200)
```

```
def treatment_plan(condition, age, gender, medical_history):
```

```
    prompt = f"Generate personalized treatment suggestions for the following patient information.  
Include home remedies and general medication guidelines.\n\nMedical Condition: {condition}\nAge:  
{age}\nGender: {gender}\nMedical History: {medical_history}\n\nPersonalized treatment plan  
including home remedies and medication guidelines:\n\n**IMPORTANT: This is for informational  
purposes only. Please consult a healthcare professional for proper treatment.**\n\nTreatment  
Plan:"
```

```
    return generate_response(prompt, max_length=1200)
```

```
# Create Gradio interface
```

```
with gr.Blocks() as app:
```

```
    gr.Markdown("# Medical AI Assistant")
```

```
gr.Markdown("**Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.**")
```

```
with gr.Tabs():
```

```
    with gr.TabItem("Disease Prediction"):
```

```
        with gr.Row():
```

```
            with gr.Column():
```

```
                symptoms_input = gr.Textbox(
```

```
                    label="Enter Symptoms",
```

```
                    placeholder="e.g., fever, headache, cough, fatigue...",
```

```
                    lines=4
```

```
                )
```

```
                predict_btn = gr.Button("Analyze Symptoms")
```

```
            with gr.Column():
```

```
                prediction_output = gr.Textbox(label="Possible Conditions & Recommendations",  
lines=20)
```

```
                predict_btn.click(disease_prediction, inputs=symptoms_input, outputs=prediction_output)
```

```
    with gr.TabItem("Treatment Plans"):
```

```
        with gr.Row():
```

```
            with gr.Column():
```

```
                condition_input = gr.Textbox(
```

```
                    label="Medical Condition",
```

```
                    placeholder="e.g., diabetes, hypertension, migraine...",
```

```
                    lines=2
```

```
                )
```

```
age_input = gr.Number(label="Age", value=30)

gender_input = gr.Dropdown(
    choices=["Male", "Female", "Other"],
    label="Gender",
    value="Male"
)

history_input = gr.Textbox(
    label="Medical History",
    placeholder="Previous conditions, allergies, medications or None",
    lines=3
)

plan_btn = gr.Button("Generate Treatment Plan")

with gr.Column():
    plan_output = gr.Textbox(label="Personalized Treatment Plan", lines=20)

plan_btn.click(treatment_plan, inputs=[condition_input, age_input, gender_input,
history_input], outputs=plan_output)

app.launch(share=True)
```