[ ]
```
!pip install transformers torch gradio -q
```

[ ]
```
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

    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,
```

```
    return response

def disease_prediction(symptoms):
    prompt = f"Based on the following symptoms, provide possible medical conditions and general medication sugge
    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 re
    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

    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")
```

```
                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)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as sec
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 251kB/s]
vocab.json: 777k/? [00:00<00:00, 15.0MB/s]
merges.txt: 442k/? [00:00<00:00, 9.82MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 28.1MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.50kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 19.8kB/s]
config.json: 100% 786/786 [00:00<00:00, 71.4kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.01MB/s]
```

```
            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], o

app.launch(share=True)
```

```
vocab.json: 777k/? [00:00<00:00, 15.0MB/s]
merges.txt: 442k/? [00:00<00:00, 9.82MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 26.1MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.50kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 19.8kB/s]
config.json: 100% 786/786 [00:00<00:00, 71.4kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.01MB/s]
Fetching 2 files: 100% 2/2 [01:25<00:00, 85.55s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:25<00:00, 136MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 50.7MB/s]
Loading checkpoint shards: 100% 2/2 [00:19<00:00, 8.03s/it]
```