import gradio as gr

import torch

from transformers import AutoTokenizer, AutoModelForCausalLM

MODEL\_NAME = "ibm-granite/granite-3.2-2b-instruct"

# Load model and tokenizer

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

)

# Set pad token if missing

if tokenizer.pad\_token is None:

    tokenizer.pad\_token = tokenizer.eos\_token

# Response generation function

def generate\_response(prompt, max\_length=512):

    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)

        # Clean up prompt from output

        response = response[len(prompt):].strip() if response.startswith(prompt) else response.strip()

        return response

# Concept explanation

def concept\_explanation(concept):

    if not concept.strip():

        return "❗ Please enter a valid concept."

    prompt = f"Explain the concept of '{concept}' in detail with an example."

    return generate\_response(prompt, max\_length=800)

# Quiz generator

def quiz\_generator(concept):

    if not concept.strip():

        return "❗ Please enter a valid concept."

    prompt = (

        f"Generate 5 quiz questions about '{concept}' using different question types "

        f"(multiple choice, true/false, short answer). Provide the answers at the end."

    )

    return generate\_response(prompt, max\_length=1200)

# Gradio UI

with gr.Blocks() as app:

    gr.Markdown("# 📘 Educational AI Assistant\nLearn Concepts & Test Your Knowledge")

    with gr.Tabs():

        with gr.TabItem("📚 Concept Explanation"):

            concept\_input = gr.Textbox(label="💡 Enter a Concept", placeholder="e.g., Artificial Intelligence")

            explain\_btn = gr.Button("Explain Concept")

            explanation\_output = gr.Textbox(label="📝 Explanation", lines=10)

            explain\_btn.click(concept\_explanation, inputs=concept\_input, outputs=explanation\_output)

        with gr.TabItem("🧠 Quiz Generator"):

            quiz\_input = gr.Textbox(label="💡 Enter a Concept", placeholder="e.g., Blockchain")

            quiz\_btn = gr.Button("Generate Quiz")

            quiz\_output = gr.Textbox(label="📋 Quiz Questions & Answers", lines=12)

            quiz\_btn.click(quiz\_generator, inputs=quiz\_input, outputs=quiz\_output)

# Launch app

app.launch(share=True)