```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

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=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)
    response = response.replace(prompt, "").strip()
    return response

def concept_explanation(concept):
    prompt = f"Explain the concept of {concept} in detail with example."
    return generate_response(prompt, max_length=800)

def quiz_generator(concept):
    prompt = f"Generate 5 quiz questions about {concept} with different question types (multipl
    return generate_response(prompt, max_length=1200)

with gr.Blocks() as app:
    gr.Markdown("## 📘 Educational AI Assistant")
    with gr.Tabs():
        with gr.TabItem("Concept Explanation"):
            concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g., Machine Lear
            explain_btn = gr.Button("Explain")
            explanation_output = gr.Textbox(label="Explanation", lines=10)
            explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanation_ou
        with gr.TabItem("Quiz Generator"):
            quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., Machine Learning"
            quiz_btn = gr.Button("Generate Quiz Questions & Answers")
            quiz_output = gr.Textbox(label="Quiz Questions & Answers", lines=15)
            quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_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://hugging
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 da
  warnings.warn(
```

tokenizer_config.json: 8.88k/? [00:00<00:00, 198kB/s]

vocab.json: 777k/? [00:00<00:00, 7.65MB/s]

merges.txt: 442k/? [00:00<00:00, 10.7MB/s]

tokenizer.json: 3.48M/? [00:00<00:00, 56.9MB/s]

added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.45kB/s]

special_tokens_map.json: 100% 701/701 [00:00<00:00, 16.4kB/s]

config.json: 100% 786/786 [00:00<00:00, 21.3kB/s]

```
`torch_dtype` is deprecated! Use `dtype` instead!
```

model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.40MB/s]

Fetching 2 files: 100% 2/2 [01:20<00:00, 80.50s/it]

model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:00<00:00, 79.4MB/s]

model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:20<00:00, 110MB/s]

Loading checkpoint shards: 100% 2/2 [00:19<00:00, 8.20s/it]

generation_config.json: 100% 137/137 [00:00<00:00, 15.8kB/s]

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
```