■■■ EduTutor AI – Gradio App Code

```
import gradio as gr
import torch
import random
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
)

# Ensure pad token exists
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token


# ■ Random concept list
example_concepts = [
    "Photosynthesis", "Newton's Laws", "Machine Learning", "Pythagorean Theorem",
    "Cloud Computing", "DNA Structure", "For Loops", "Ecosystems", "Electric Circuits"
]


# ■ Generate AI response
def generate_response(prompt, max_length=700, temperature=0.7):
    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=temperature,
            do_sample=True,
            pad_token_id=tokenizer.pad_token_id,
        )
    decoded = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return decoded.replace(prompt, "").strip()


# ■ Main Tutor Logic
def edu_tutor_response(concept, theme, temperature):
    if not concept.strip():
        return "■■ Please enter a topic or concept to learn about."

    prompts = {
        "■ Step-by-Step Explanation": f"You are EduTutor AI. Explain '{concept}' clearly in steps with e
        "■ Quick Revision Notes": f"You are EduTutor AI. Summarize '{concept}' in short bullet points fo
        "■ Visual Analogy": f"You are EduTutor AI. Explain '{concept}' with a simple visual idea or anal
        "■ Quiz Time (3 Questions)": f"You are EduTutor AI. Create 3 multiple-choice questions on '{conc
        "■ Quiz Feedback Coach": f"You are EduTutor AI. Give helpful feedback and hints to a student who
        "■ Real-Life Example": f"You are EduTutor AI. Connect '{concept}' to real-world examples or stor
        "■ Challenge Me": f"You are EduTutor AI. Ask a tricky or deeper question about '{concept}' and e
    }

    prompt = prompts.get(theme, f"Explain the concept '{concept}'.")
    return generate_response(prompt, temperature=temperature)


# ■ Random Concept Picker
def random_concept():
    return random.choice(example_concepts)
# ■ Gradio App UI
```

```
with gr.Blocks() as app:
    gr.Markdown("""
    # ■■■ **EduTutor AI** – Your Personalized Study Buddy
    Learn any topic *your way* – Visual, Quick, Quiz, Real-life, or even get Challenged!
    """)

    with gr.Row():
        concept_input = gr.Textbox(label="■ What topic do you want to learn?", placeholder="e.g., Photos

    with gr.Row():
        surprise_btn = gr.Button("■ Surprise Me with a Concept")
        surprise_btn.click(random_concept, outputs=concept_input)

    with gr.Row():
        theme_selector = gr.Dropdown(
            label="■ Choose your learning style",
            choices=[
                "■ Step-by-Step Explanation",
                "■ Quick Revision Notes",
                "■ Visual Analogy",
                "■ Quiz Time (3 Questions)",
                "■ Quiz Feedback Coach",
                "■ Real-Life Example",
                "■ Challenge Me"
            ],
            value="■ Step-by-Step Explanation"
        )

    with gr.Row():
        temperature_slider = gr.Slider(0.3, 1.2, value=0.7, label="■ AI Creativity Level", step=0.1)

    with gr.Row():
        generate_btn = gr.Button("■ Start Learning with EduTutor")

    output_box = gr.Markdown()

    generate_btn.click(
        edu_tutor_response,
        inputs=[concept_input, theme_selector, temperature_slider],
        outputs=output_box
    )

    gr.Markdown("---\n■ Made with love to help students everywhere succeed!")

app.launch(share=True)
```