!pip install gradio transformers

import gradio as gr

import transformers

# Load a pre-trained language model for recipe generation

model\_name = "gpt2" # Or try other models like "distilgpt2", "EleutherAI/gpt-neo-1.3B" (larger, more capable but requires more resources)

tokenizer = transformers.AutoTokenizer.from\_pretrained(model\_name)

model = transformers.AutoModelForCausalLM.from\_pretrained(model\_name)

def generate\_recipe(ingredients, cuisine, dish\_type, dietary\_restrictions):

"""Generates a recipe based on given ingredients, cuisine, dish type, and dietary restrictions."""

prompt = f"Write a recipe for a {dish\_type} {cuisine} dish. The main ingredients are: {ingredients}. "

if dietary\_restrictions:

prompt += f"It should be {dietary\_restrictions}. "

prompt += "Recipe:\n"

input\_ids = tokenizer.encode(prompt, return\_tensors="pt")

try:

output = model.generate(

input\_ids,

max\_length=500, # Adjust for desired recipe length

num\_return\_sequences=1, # Generate a single recipe

temperature=0.7, # Controls randomness (lower = more predictable)

num\_beams=5, # Improves quality, but slower. Remove if you have memory issues

no\_repeat\_ngram\_size=2, # Prevents repetition

early\_stopping=True

)

generated\_text = tokenizer.decode(output[0], skip\_special\_tokens=True)

recipe = generated\_text[len(prompt):].strip() # Remove the prompt from the output

return recipe

except Exception as e:

print(f"Error during recipe generation: {e}")

return "Sorry, I couldn't generate a recipe. Please try again with different inputs."

def create\_gradio\_interface():

"""Creates a Gradio interface for the recipe generator."""

with gr.Blocks() as interface:

# Add a Markdown section with colorful and fun text

gr.Markdown("""

<h1 style="color:#FF6347; text-align:center;">🌟 AI Recipe Blogger 🌟</h1>

<p style="font-size: 18px; color: #333; text-align: center;">

Welcome to the AI-powered recipe generator! 🍳🍽️<br>

Fill in the ingredients, choose your cuisine, and let the AI whip up something tasty for you! 😋

</p>

""")

with gr.Column():

# User input fields

ingredients\_input = gr.Textbox(label="Ingredients (comma separated)", placeholder="e.g., chicken, rice, vegetables")

cuisine\_input = gr.Dropdown(

["Italian", "Mexican", "Indian", "Chinese", "American", "French", "Japanese", "Thai", "Mediterranean", "Other"],

label="Cuisine",

value="Italian" # Default Cuisine

)

dish\_type\_input = gr.Dropdown(

["Appetizer", "Main Course", "Dessert", "Side Dish", "Soup", "Salad"],

label="Dish Type",

value="Main Course" # Default Dish Type

)

dietary\_restrictions\_input = gr.CheckboxGroup(

["Vegetarian", "Vegan", "Gluten-Free", "Dairy-Free", "Nut-Free"],

label="Dietary Restrictions"

)

generate\_button = gr.Button("Generate Recipe! 🌮")

recipe\_output = gr.Textbox(label="Recipe", lines=15)

# Displaying the colorful and stylish shareable link

gr.Markdown("""

<div style="font-size: 18px; font-weight: bold; color: #4CAF50; text-align: center; margin-top: 20px;">

🍴 <a href="{interface.share\_url}" target="\_blank" style="color: #FF6347; font-size: 20px; text-decoration: none;">Click here to Generate Your Recipe! 🌟</a> 🍴

</div>

""".format(interface=interface))

# Handle the button click

generate\_button.click(

fn=generate\_recipe,

inputs=[ingredients\_input, cuisine\_input, dish\_type\_input, dietary\_restrictions\_input],

outputs=recipe\_output

)

return interface

# Create and launch the Gradio interface

if \_\_name\_\_ == "\_\_main\_\_":

interface = create\_gradio\_interface()

interface.launch(share=True) # Shareable link for Colab

!pip install gradio transformers