Citizen AI: Intelligent Citizen Engagement Platform

Generative AI with IBM

Project Description:

Citizen AI uses the Granite model from Hugging Face to give quick, helpful answers about government services and civic issues. It tracks public sentiment and shows simple dashboards for officials to see feedback. This project will be deployed in Google Colab using Granite for easy, low-cost setup and reliable performance.

Pre-requisites:

1. Gradio Framework Knowledge: Gradio Documentation
2. IBM Granite Models (Hugging Face): IBM Granite models
3. Python Programming Proficiency: Python Documentation
4. Version Control with Git: Git Documentation
5. Google Collab's T4 GPU Knowledge: Google collab

Project Workflow:

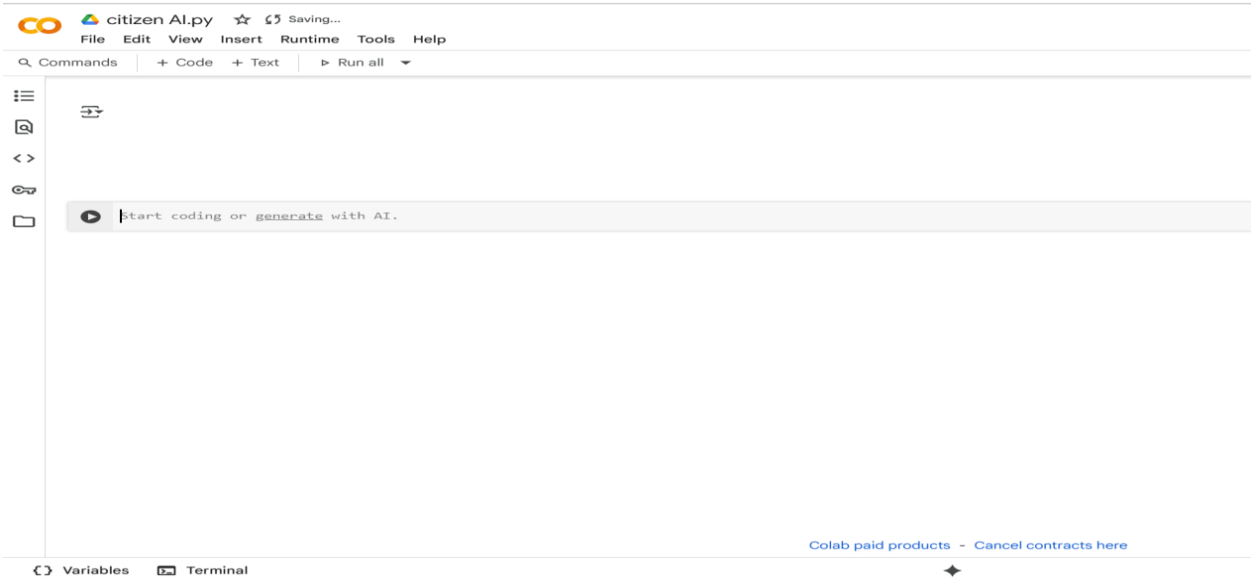
Activity-1: To write a code for Citizen AI in GoogleColab.

Activity-2: Choosing a IBM Granite Model From Hugging Face.

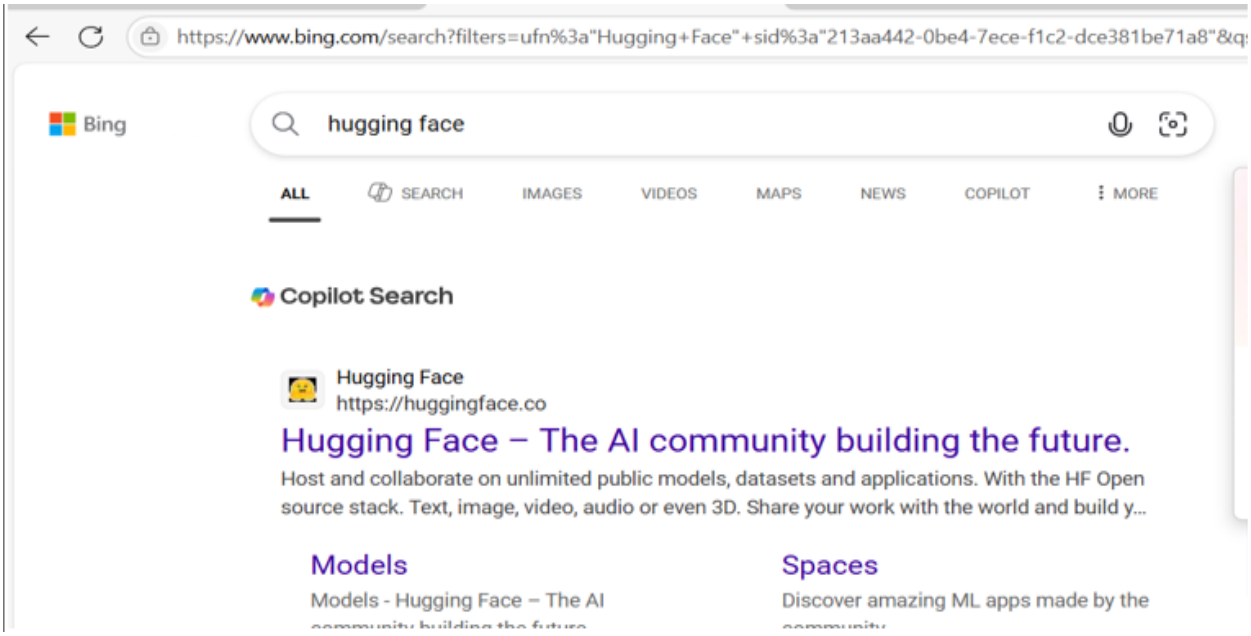
Activity-3: Running Application In Google Colab.

Activity-4: Upload your Project in Github.

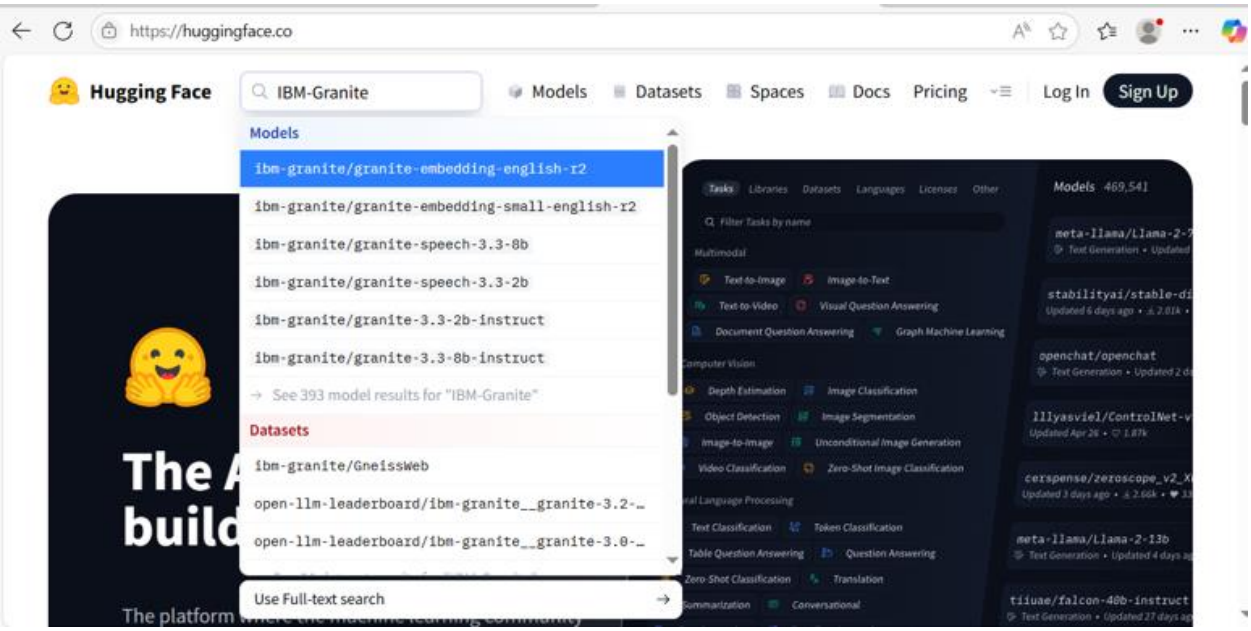
1.Open Google Colob and save it as HealthAI.ipynb and open a new notebook in Google Colob.



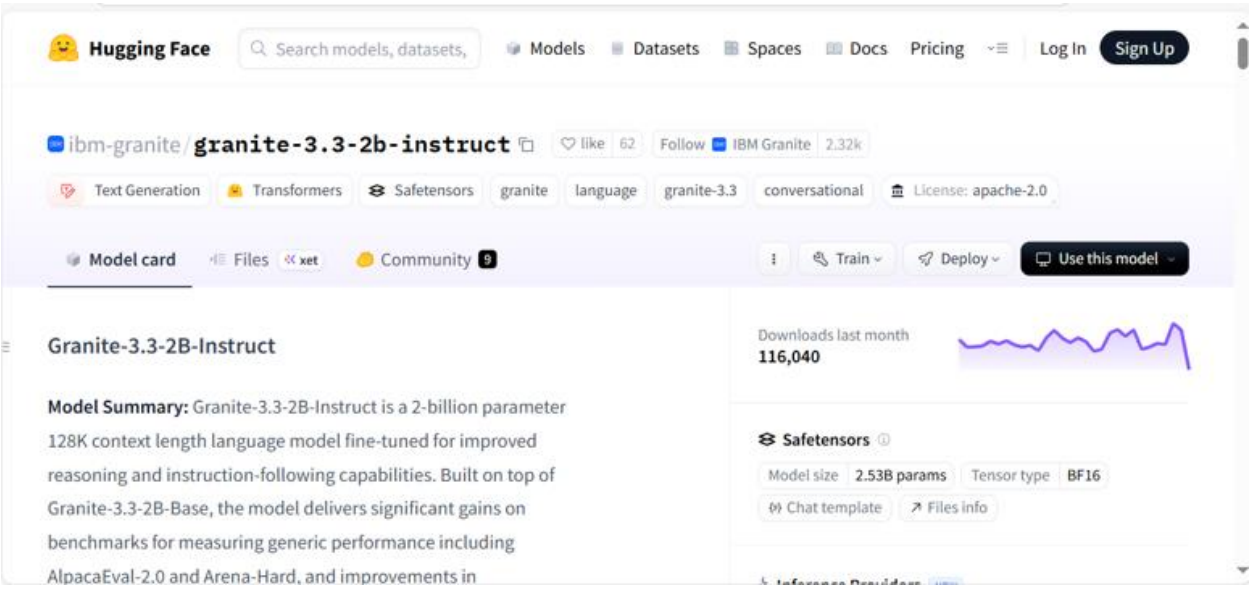
2.Search for “Hugging face” in any browser.



3.Then click on the first link (Hugging Face), then click on signup and create your own account in Hugging Face. Then search for “IBM-Granite models” and choose any model

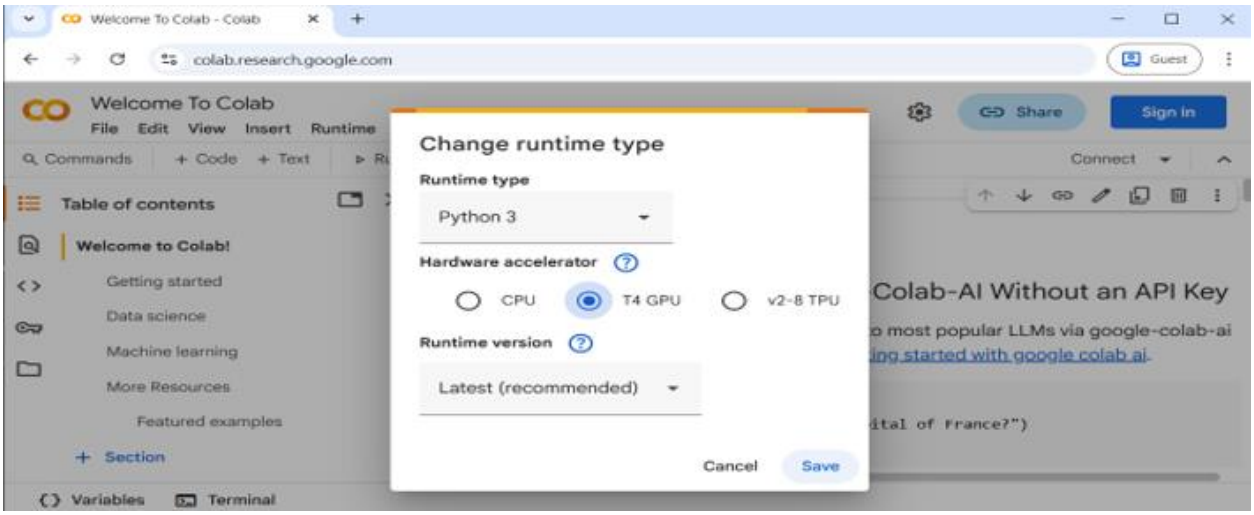


4.Here for this project we are using “granite-3.2-2b-instruct” which is compatible fast and light weight.

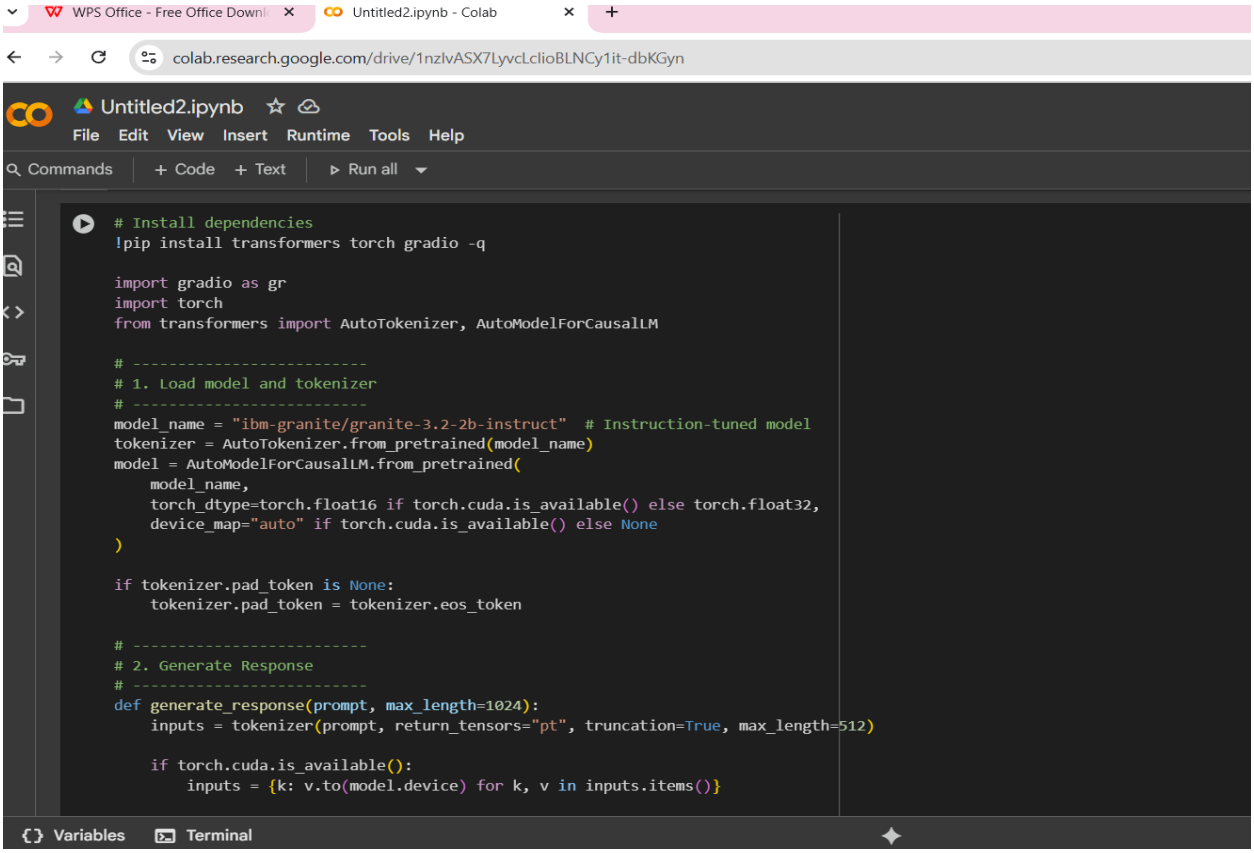


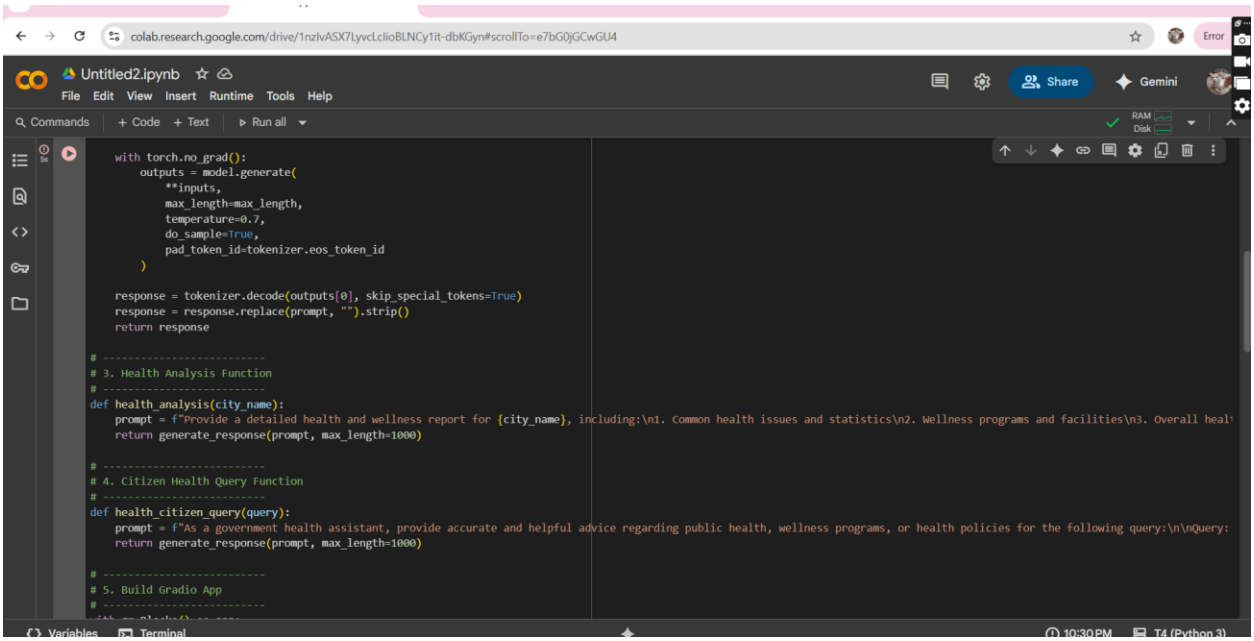
5.Now we will start building our project in Google collab.

6. Choose “T4 GPU” for Better Performance



6. Then start doing the code for “Citizen AI”.


WPS Office - Free Office Downl
Untitled2.ipynb - Colab
colab.research.google.com/drive/1nzlvASX7LyvcLclioBLNCy1it-dbKGyn
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Install dependencies
!pip install transformers torch gradio -q
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
# --------------------------
# 1. Load model and tokenizer
# --------------------------
model_name = "ibm-granite/granite-3.2-2b-instruct" # Instruction-tuned model
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
# --------------------------
# 2. Generate Response
# --------------------------
def generate_response(prompt, max_length=1024):
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()}
Variables Terminal



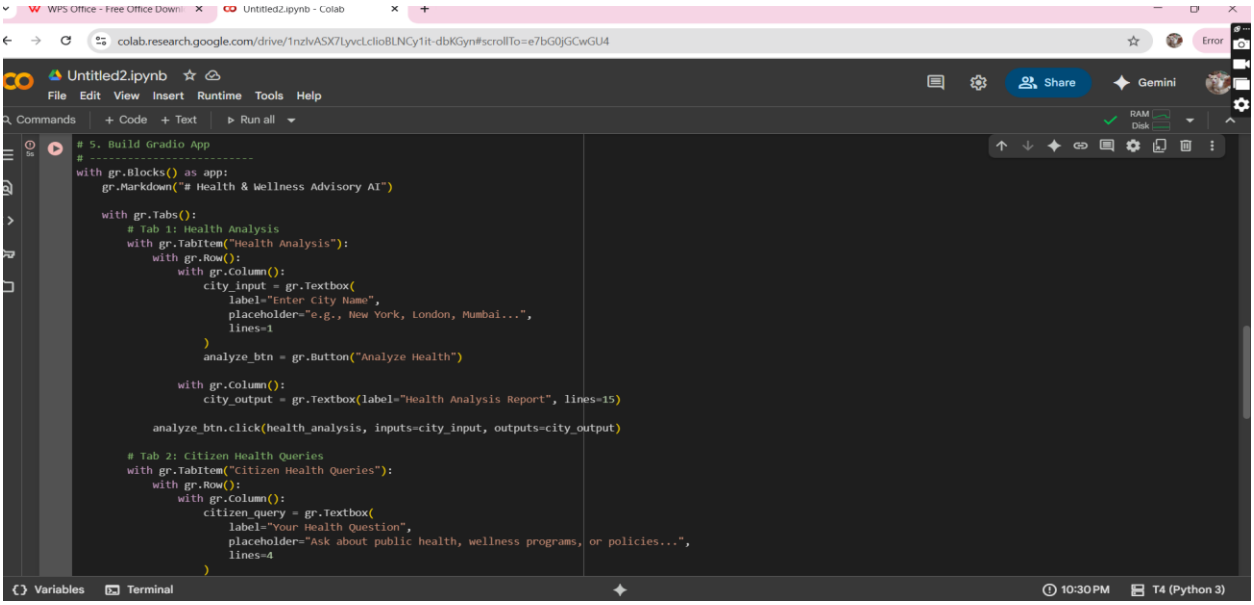
```
    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.replace(prompt, "").strip()
    return response

# -------------------------
# 3. Health Analysis Function
# -------------------------
def health_analysis(city_name):
    prompt = f"Provide a detailed health and wellness report for {city_name}, including:\n1. Common health issues and statistics\n2. Wellness programs and facilities\n3. Overall healt
    return generate_response(prompt, max_length=1000)

# -------------------------
# 4. Citizen Health Query Function
# -------------------------
def health_citizen_query(query):
    prompt = f"As a government health assistant, provide accurate and helpful advice regarding public health, wellness programs, or health policies for the following query:\n\nQuery:
    return generate_response(prompt, max_length=1000)

# -------------------------
# 5. Build Gradio App
# -------------------------
```



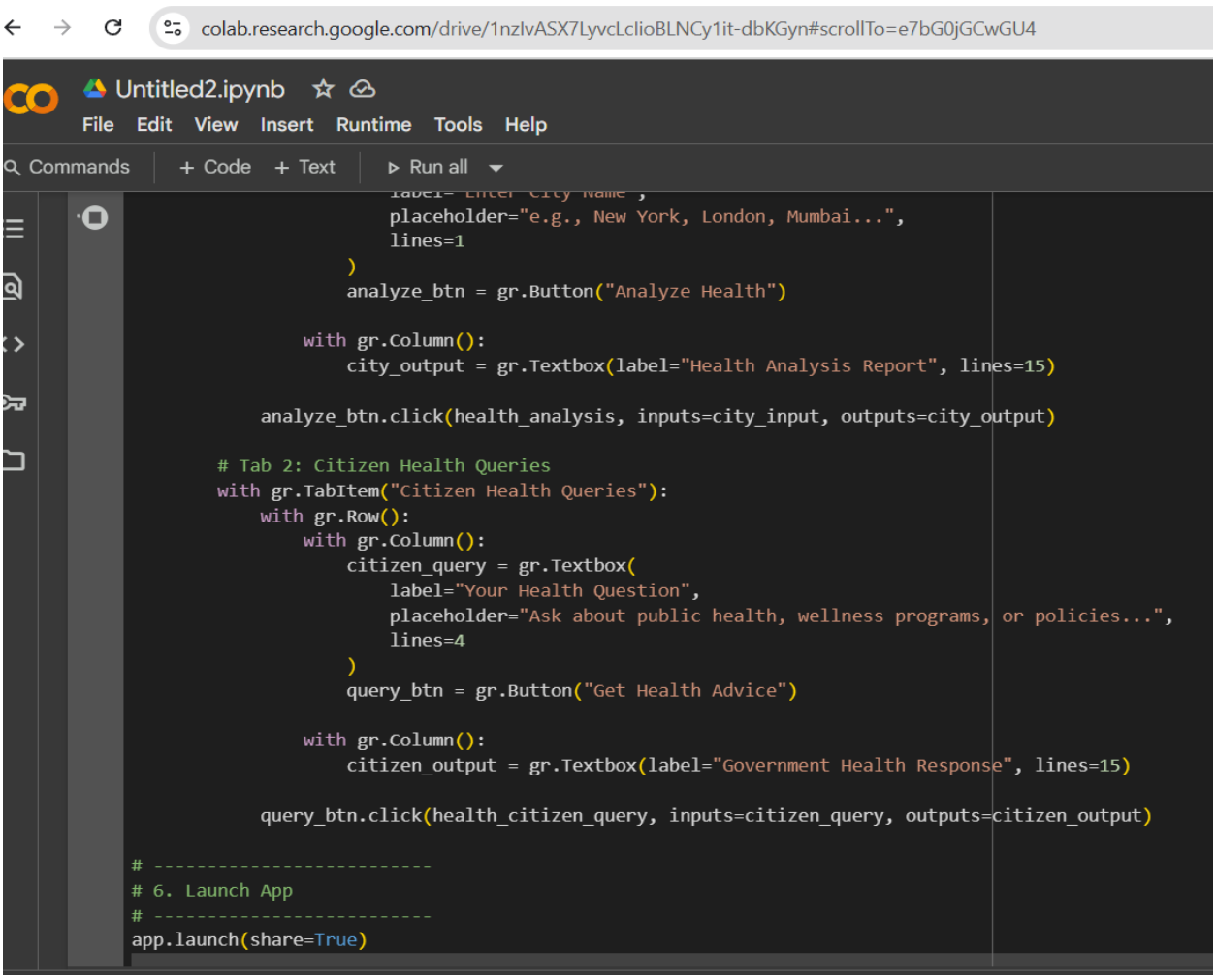
```
# 5. Build Gradio App
# -------------------------
with gr.Blocks() as app:
    gr.Markdown("# Health & Wellness Advisory AI")

    with gr.Tabs():
        # Tab 1: Health Analysis
        with gr.TabItem("Health Analysis"):
            with gr.Row():
                with gr.Column():
                    city_input = gr.Textbox(
                        label="Enter City Name",
                        placeholder="e.g., New York, London, Mumbai...",
                        lines=1
                    )
                    analyze_btn = gr.Button("Analyze Health")

                with gr.Column():
                    city_output = gr.Textbox(label="Health Analysis Report", lines=15)

            analyze_btn.click(health_analysis, inputs=city_input, outputs=city_output)

        # Tab 2: Citizen Health Queries
        with gr.TabItem("Citizen Health Queries"):
            with gr.Row():
                with gr.Column():
                    citizen_query = gr.Textbox(
                        label="Your Health Question",
                        placeholder="Ask about public health, wellness programs, or policies...",
                        lines=4
                    )
```


colab.research.google.com/drive/1nzIvASX7LyvcLcIioBLNCy1it-dbKGyn#scrollTo=e7bG0jGCwGU4
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
placeholder="e.g., New York, London, Mumbai...",
lines=1
)
analyze_btn = gr.Button("Analyze Health")
with gr.Column():
city_output = gr.Textbox(label="Health Analysis Report", lines=15)
analyze_btn.click(health_analysis, inputs=city_input, outputs=city_output)
# Tab 2: Citizen Health Queries
with gr.TabItem("Citizen Health Queries"):
with gr.Row():
with gr.Column():
citizen_query = gr.Textbox(
label="Your Health Question",
placeholder="Ask about public health, wellness programs, or policies...",
lines=4
)
query_btn = gr.Button("Get Health Advice")
with gr.Column():
citizen_output = gr.Textbox(label="Government Health Response", lines=15)
query_btn.click(health_citizen_query, inputs=citizen_query, outputs=citizen_output)
# --------------------------
# 6. Launch App
# --------------------------
app.launch(share=True)

8.OUTPUT WITH A SIMPLE USER INTERFACE.