GEN AI TOPIC: AI PPT CONTENT GENERATOR

TEAM:SENTRIO
TEAM MEMBERS:
1. NEHA SHRI V
2. SRI JANANI S
3. VIGNESH BALA R

CODE 1:
```
!pip install google-genai python-pptx duckduckgo-search
beautifulsoup4 requests
```

CODE 2:
```
import os
import json
import requests
from bs4 import BeautifulSoup
from google import genai
from pptx import Presentation
from pptx.util import Inches, Pt
from duckduckgo_search import DDGS

# --- 1. CONFIGURATION ---
# Use the Gemini 3 Flash model for the best balance of speed and logic
MODEL_ID = "gemini-3-flash-preview"
API_KEY = "AIzaSyBaYpw9iDhJlS6v3EvQjP4d4UaxzSt3hho"

def get_web_search_results(topic, num_results=3):
    print(f" Searching for: {topic}...")
    search_context = ""
    try:
        # Note: In latest versions, DDGS is sometimes imported as 'ddgs'
        with DDGS() as search_engine:
            results = list(search_engine.text(topic,
max_results=num_results))
            if not results:
                return "No search results found."

            for result in results:
                try:
                    response = requests.get(result['href'], timeout=8)
                    response.raise_for_status()
                    soup = BeautifulSoup(response.content, 'html.parser')
                    # Extract text only from paragraphs for cleaner data
                    paragraphs = soup.find_all('p')
```

```
                    page_text = "\n".join([p.get_text() for p in
paragraphs if len(p.get_text()) > 30])
                    search_context += f"Source:
{result['title']}\nContent: {page_text[:1200]}\n---\n"
                except:
                    # Fallback to snippet if scraping fails
                    search_context += f"Source:
{result['title']}\nSnippet: {result['body']}\n---\n"
        return search_context
    except Exception as e:
        print(f"⚠ Search error: {e}")
        return "Search failed."

def generate_presentation_content(topic, search_results):
    print(f"🤖 Generating content using {MODEL_ID}...")
    client = genai.Client(api_key=API_KEY)

    prompt = f"""
    Topic: {topic}
    Research Context: {search_results}

    Task: Create a 7-slide presentation structure.
    Output MUST be a single JSON object (no markdown formatting):
    {{
      "title_slide": {{"title": "Main Title", "subtitle": "Subtitle"}},
      "overview_slide": {{"title": "Overview", "points": ["P1", "P2",
"P3"]}},
      "key_point_slides": [
        {{"title": "Point 1", "points": ["Detail A", "Detail B", "Detail
C"]}},
        {{"title": "Point 2", "points": ["Detail A", "Detail B", "Detail
C"]}},
        {{"title": "Point 3", "points": ["Detail A", "Detail B", "Detail
C"]}},
        {{"title": "Point 4", "points": ["Detail A", "Detail B", "Detail
C"]}}
      ],
      "conclusion_slide": {{"title": "Conclusion", "points": ["Summary 1",
"Summary 2"]}}
    }}
    Constraint: Each 'key_point_slide' must have deep, descriptive bullet
points.
    """

    try:
```

```
        response = client.models.generate_content(
            model=MODEL_ID,
            contents=prompt
        )
        # Handle cases where model wraps JSON in code blocks
        json_str = response.text.strip().replace("```json",
"").replace("```", "")
        return json.loads(json_str)
    except Exception as e:
        print(f"❌ Content Generation Failed: {e}")
        return None

def create_pptx(content, topic):
    print("🛠️ Building PowerPoint...")
    prs = Presentation()

    # Title Slide
    slide = prs.slides.add_slide(prs.slide_layouts[0])
    slide.shapes.title.text = content["title_slide"]["title"]
    slide.placeholders[1].text = content["title_slide"]["subtitle"]

    # All Content Slides
    all_slides = [content["overview_slide"]] + content["key_point_slides"]
+ [content["conclusion_slide"]]

    for slide_data in all_slides:
        slide = prs.slides.add_slide(prs.slide_layouts[1])
        slide.shapes.title.text = slide_data["title"]
        tf = slide.placeholders[1].text_frame
        tf.word_wrap = True
        for point in slide_data["points"]:
            p = tf.add_paragraph()
            p.text = point
            p.level = 0
            p.font.size = Pt(16)

    filename = f"{topic.replace(' ', '_')}_Presentation.pptx"
    prs.save(filename)
    print(f"✅ Presentation created: {filename}")

def main():
    user_topic = input("Enter topic: ")
    if not user_topic: return

    data = get_web_search_results(user_topic)
```

```
    slides = generate_presentation_content(user_topic, data)

    if slides:
        create_pptx(slides, user_topic)
    else:
        print("Could not generate PPT content. Check your API key or
Topic.")

if __name__ == "__main__":
    main()
```

OUTPUT

Enter topic: global warming

global_warming_Presentation.pptx