SCHOOL OF INNOVATION

AI PPT CONTENT GENERATOR

Team Name	Sentrio
Team ID	P-2023-27-EI-006

Team Leader Name	NEHA SHRI V	711723106070
Team Member 1	SRI JANANI S	711723106105
Team Member 2	VIGNESH BALA R	711723106121

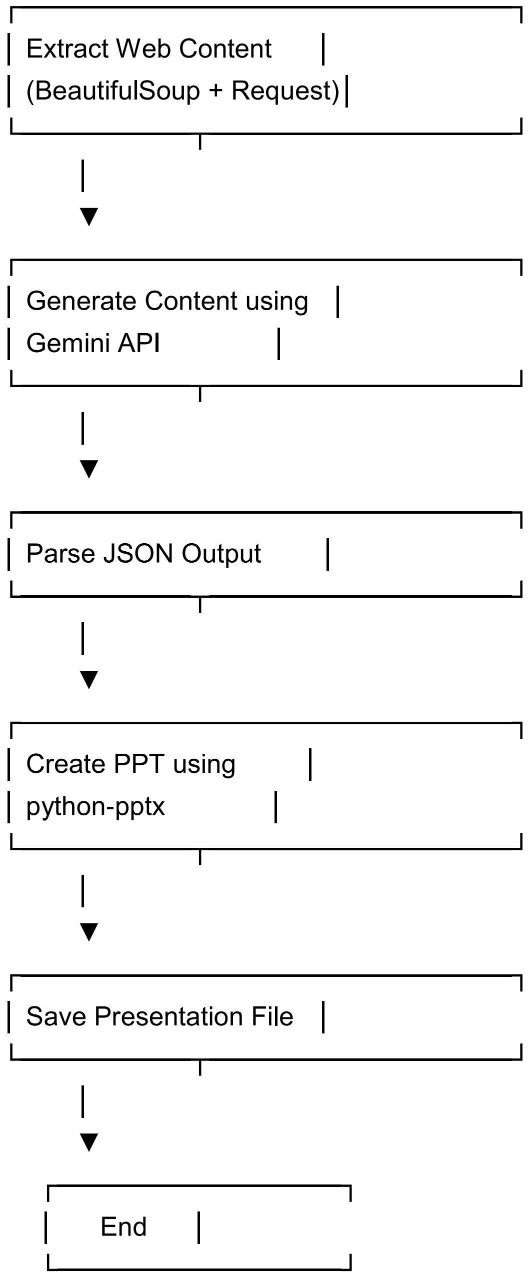
KiTE Department Name	ECE
SOI Vertical Name	EMBEDDED & IOT

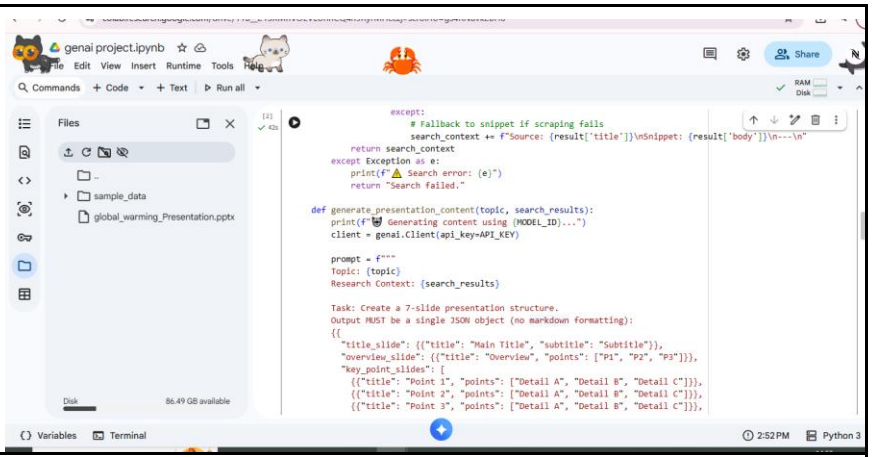
Mandatory Detail	Description
Introduction	The AI PPT Content Generator is a system that automatically creates PowerPoint presentations from a user-given topic. The project is developed using **Google Colab** and uses Generative AI to research, organize, and generate slide content. It reduces manual effort in preparing presentations.
Problem Statement	Creating presentations manually requires researching, summarizing, structuring slides, and designing them, which is time-consuming. Existing tools do not combine research and automatic slide generation. This project solves that by automating the entire process using AI.
Objectives	• Automate PPT content generation. • Use AI to generate structured slides. • Integrate web search for updated information. • Reduc
Generative AI Overview	Generative AI creates new content such as text, images, or code. In this project, it generates structured slide content using a Large Language Model (LLM).
GENAI Concepts Applied	• Large Language Models (LLMs) • Prompt Engineering • Text Generation • Retrieval-based context (web search)
Prompt Engineering	We designed a structured prompt instructing the model to: • Create 7 slides • Provide deep bullet points • Output in JSON format only
Text/Image/Code Generation	This project uses **Text Generation** as the primary generation type. The Gemini model generates structured slide content including titles and descriptive bullet points. No image generation is used in the current version. Code execution is used to automatically create PowerPoint files using the generated content**.**

Embeddings/Retrieval/ Fine-tuning	This project does not use embeddings or fine-tuning. However, it uses a simple retrieval mechanism through DuckDuckGo search. Relevant web content is fetched and passed as context to the AI model to improve the quality and accuracy of generated slides.
Libraries & Frameworks Used	The project is implemented using Python in Google Colab. **Python Libraries Used** • `google-genai` – To connect with Gemini API • `python-pptx` – To generate PowerPoint presentations • `duckduckgo-search` – To perform web searches • `beautifulsoup4` – To extract webpage content • `requests` – To fetch website data • `json` – To parse AI output
GenAI Frameworks	• Google Generative AI SDK • Gemini API No LangChain or Hugging Face frameworks were used.
Backend/Frontend Frameworks	• Backend: Python • Platform: Google Colab • No separate frontend framework is used.
Models &API Usage	The project uses Google's Gemini API for text generation. .
Pre-trained Models	• Gemini 3 Flash Preview
Hugging Face Models	No Hugging Face models were used in this project.
LLM APIs Used	• Google Gemini API
Local LLM Execution	No local LLM execution (such as Ollama) is used. The model runs through cloud API access.

Model /Version Details	• Model Name: gemini-3-flash-preview • Type: Pre-trained Large Language Model • Mode: API-based execution
Model Selection Justification	Gemini 3 Flash was selected because: • It provides fast responses. • It has good reasoning ability. • It is suitable for structured JSON output. • It balances performance and cost.
Trade-offs Considered	• Accuracy: High-quality structured content • Cost: API usage may incur cost • Latency: Fast response time • Privacy: Cloud-based processing
Dataset/ Input Description	The input to the system is: • A user-provided topic (e.g., Global Warming) The system collects relevant information from online sources dynamically.
Dataset Source	• DuckDuckGo web search • Publicly available websites
Preprocessing Steps	• Perform web search. • Fetch webpage content. • Extract paragraph text. • Remove short or irrelevant text. • Limit text length before sending to AI model.
System Architecture(Block Diagram)	User Input → Web Search

	→ Web Scraping → AI Content Generation → JSON Parsing → PowerPoint Creation → Output PPT File
Procedure / Methodology	1. User enters topic. 2. System performs web search. 3. Extracts relevant information. 4. Sends structured prompt to Gemini model. 5. Model returns slide structure in JSON format. 6. Python program converts JSON into PowerPoint slides. 7. PPT file is saved locally.
Step-by-Step Implementation Flow	• Install required libraries in Colab. • Configure Gemini API key. • Collect user input topic. • Retrieve web search results. • Generate presentation content using Gemini. • Parse JSON output. • Create PowerPoint file using python-pptx. • Save generated file.
Flowchart	┌──────────────┐ │ Start │ └──────┬───────┘ │ ▼ ┌──────────────────┐ │ Enter Topic (User) │ └────────┬─────────┘ │ ▼ ┌──────────────────────┐ │ Perform Web Search │ │ (DuckDuckGo Search) │ └──────────┬───────────┘ │ ▼

	 Extract Web Content (BeautifulSoup + Request) Generate Content using Gemini API Parse JSON Output Create PPT using python-pptx Save Presentation File End
Implementation Details	Main Functions: • `get_web_search_results()` – Retrieves search data • `generate_presentation_content()` – Calls Gemini API • `create_pptx()` – Builds PowerPoint file • `main()` – Controls workflow The system ensures structured JSON output before creating slides.

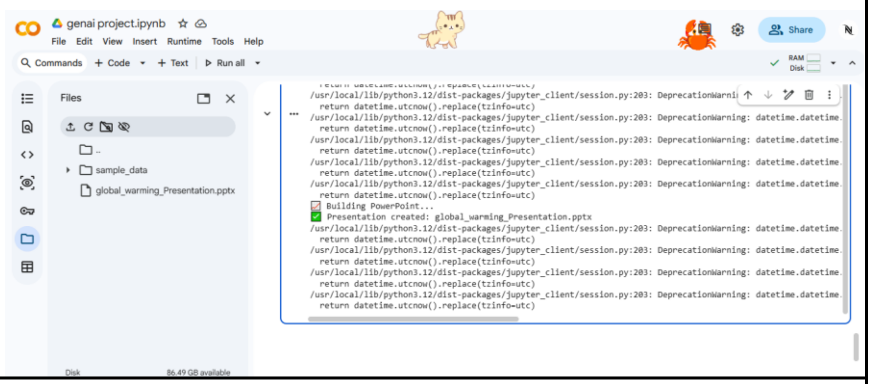
API Usage Screenshots	

```
                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"]}},
```

Results & Output Analysis	global warming Output: • 7-slide structured PowerPoint • Title slide • Overview slide • Four detailed key point slides • Conclusion slide The output meets the project objective of automatic PPT generation.
Screenshots/Visualizations	

```
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarni
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime
  return datetime.utcnow().replace(tzinfo=utc)
📈 Building PowerPoint...
✅ Presentation created: global_warming_Presentation.pptx
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: DeprecationWarning: datetime.datetime
  return datetime.utcnow().replace(tzinfo=utc)
```

Evaluation/Performance Analysis	• Fast response time • Structured slide generation • Accurate topic coverage • Minor dependency on internet and API limits
Challenges Faced	• Handling JSON formatting errors • Removing markdown formatting from AI response • Web scraping timeouts

	These were solved using exception handling and response cleaning.
Conclusion	The AI PPT Content Generator successfully automates presentation creation using Generative AI. It reduces manual effort and demonstrates practical integration of web search, AI text generation, and PPT automation.
Future Enhancements/ Scope	• Add automatic image generation • Add PPT theme customization • Create web application interface • Support multi-language input • Add voice-based topic input
References	• Google Gemini API Documentation • Python-PPTX Documentation • DuckDuckGo Search Library • BeautifulSoup Documentation
GitHub Link	
Additional Outputs	• Generated PowerPoint file • Google Colab notebook • Demo execution screenshots