**Graph generation code:**

# graph\_la.py

from langchain\_core.tools import tool

import pandas as pd

from graph\_code\_gen\_tool import generate\_graph\_image

from docling\_markdown import convert\_and\_save\_document

import boto3

import os

from dotenv import load\_dotenv

# Load AWS S3 credentials

load\_dotenv(r'C:\Users\Admin\Desktop\MS Data Architecture and Management\DAMG 7245 - Big Data Systems and Intelligence Analytics\Assignment 5 A\environment\access.env')

s3 = boto3.client(

's3',

aws\_access\_key\_id=os.getenv("AWS\_ACCESS\_KEY\_ID"),

aws\_secret\_access\_key=os.getenv("AWS\_SECRET\_ACCESS\_KEY"),

region\_name=os.getenv("AWS\_REGION")

)

BUCKET\_NAME = "bigdatasystems2"

PDF\_KEY = "Electricity2025.pdf"

@tool("generate\_graph\_code")

def generate\_graph\_code(query: str) -> dict:

"""

Given a user query, this tool extracts data from the Electricity2025 PDF,

generates a graph image using code from GPT, and returns the image path.

"""

# Step 1: Download the PDF

response = s3.get\_object(Bucket=BUCKET\_NAME, Key=PDF\_KEY)

pdf\_bytes = response['Body'].read()

# Step 2: Extract text from PDF

parsed\_text = convert\_and\_save\_document(pdf\_bytes)

# Step 3: Attempt to extract tabular data from text

import re

def extract\_tabular\_data(text):

lines = text.split("\n")

data = []

for line in lines:

# Matches lines like: 2020 123 456

if re.match(r"^\d{4}\s+", line):

parts = line.split()

if len(parts) >= 2:

data.append(parts)

df = pd.DataFrame(data)

return df

df = extract\_tabular\_data(parsed\_text)

if df.empty:

return {"error": "No structured data could be extracted from the report."}

# Step 4: Generate and run graph code

image\_path = generate\_graph\_image(query, df)

return {"image\_path": image\_path}

### **Final Integration Steps:**

1. **Add this new tool to langgraph\_llm.py:**

from graph\_la import generate\_graph\_code # ✅ Add this

tools = [

rag\_search\_filter,

web\_search,

generate\_graph\_code, # ✅ New tool added

final\_answer

]

2. Update your report\_builder.py to embed the image:

image\_path = output.get("image\_path", "")

image\_section = f"\n\nGRAPH VISUALIZATION\n-------------------\n![Graph]({image\_path})" if image\_path else ""

1. **Update final\_answer\_la.py to accept image\_path as an optional parameter** (if not already).
2. Conversion to pdf:

# report\_pdf\_exporter.py

import pdfkit

import os

from jinja2 import Template

def export\_report\_to\_pdf(output: dict, output\_pdf\_path: str = "final\_report.pdf") -> str:

research\_steps = output.get("research\_steps", "")

if isinstance(research\_steps, list):

research\_steps = "\n".join([f"- {r}" for r in research\_steps])

sources = output.get("sources", "")

if isinstance(sources, list):

sources = "\n".join([f"- {s}" for s in sources])

image\_path = output.get("image\_path", "")

image\_section = f"<h2>Graph Visualization</h2><img src='{image\_path}' width='600'/>" if image\_path else ""

# Use Jinja2 template for HTML layout

html\_template = Template("""

<html>

<head><title>Mini Manus Report</title></head>

<body style="font-family: Arial;">

<h1>INTRODUCTION</h1><p>{{ introduction }}</p>

<h2>RESEARCH STEPS</h2><pre>{{ research\_steps }}</pre>

<h2>REPORT</h2><p>{{ main\_body }}</p>

{{ image\_section|safe }}

<h2>CONCLUSION</h2><p>{{ conclusion }}</p>

<h2>SOURCES</h2><pre>{{ sources }}</pre>

</body>

</html>

""")

html\_content = html\_template.render(

introduction=output.get("introduction", ""),

research\_steps=research\_steps,

main\_body=output.get("main\_body", ""),

conclusion=output.get("conclusion", ""),

sources=sources,

image\_section=image\_section

)

# Convert HTML to PDF

pdfkit.from\_string(html\_content, output\_pdf\_path)

return output\_pdf\_path

### **How to Use It in Your Code**

Anywhere after calling build\_report(output) (e.g., in langgraph\_main.py), add:

from report\_pdf\_exporter import export\_report\_to\_pdf

# Generate and save the report

export\_path = export\_report\_to\_pdf(output=out["intermediate\_steps"][-1].tool\_input)

# Optional: return it from FastAPI or provide download link in Streamlit

In main.py (Streamlit frontend), add:

with open("final\_report.pdf", "rb") as file:

st.download\_button("Download PDF Report", file, file\_name="mini\_manus\_report.pdf")