## Project: Mayer Electric - Product Search POC

### 1. utils/config.py

```python

import os

def get\_openai\_config():

return {

"api\_key": os.getenv("AZURE\_OPENAI\_KEY"),

"api\_version": "2023-05-15",

"azure\_endpoint": os.getenv("AZURE\_OPENAI\_ENDPOINT")

}

```

### 2. ingestion/ingest\_data.py

```python

import json

import pandas as pd

def load\_json\_to\_dataframe(path):

with open(path, 'r') as f:

data = json.load(f)

return pd.DataFrame(data)

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

df = load\_json\_to\_dataframe("../data/mayer\_llm\_friendly\_catalog\_300\_rows.json")

print(df.head())

```

### 3. utils/chunking.py

```python

def chunk\_product\_record(record):

return f"""

Product Name: {record['ProductName']}

Brand: {record['Brand']}

Category: {record['Category']} > {record['SubCategory']}

Use Case: {record['UseCase']}

Description: {record['Description']}

Specifications: {record['Specifications']}

Tags: {record['Tags']}

"""

```

### 4. embedding/generate\_embeddings.py

```python

from openai import AzureOpenAI

from utils.config import get\_openai\_config

from utils.chunking import chunk\_product\_record

import pandas as pd

client = AzureOpenAI(\*\*get\_openai\_config())

def embed\_text(text):

response = client.embeddings.create(

input=[text],

model="text-embedding-ada-002"

)

return response.data[0].embedding

def generate\_embeddings(df):

return [embed\_text(chunk\_product\_record(row)) for \_, row in df.iterrows()]

```

### 5. vector\_store/cosmosdb\_utils.py

```python

from azure.cosmos import CosmosClient

import os

def get\_cosmos\_client():

return CosmosClient(

url=os.getenv("COSMOS\_URL"),

credential=os.getenv("COSMOS\_KEY")

)

def insert\_document(client, db\_name, container\_name, doc):

container = client.get\_database\_client(db\_name).get\_container\_client(container\_name)

container.upsert\_item(doc)

```

### 6. classification/root\_classification.py

```python

from openai import AzureOpenAI

from utils.config import get\_openai\_config

client = AzureOpenAI(\*\*get\_openai\_config())

def classify\_root\_category(query):

prompt = f"What root category does the following query belong to? Query: {query}"

response = client.chat.completions.create(

model="gpt-35-turbo",

messages=[{"role": "user", "content": prompt}]

)

return response.choices[0].message.content.strip()

```

### 7. retrieval/semantic\_search.py

```python

# Placeholder - customize based on Cosmos DB vector search implementation

def search\_vector\_store(embedding, top\_k=5):

# Return top K most similar products from vector store

return [{"ProductName": "LED Flood Light", "Score": 0.91}]

```

### 8. api/app.py

```python

from fastapi import FastAPI, Request

from classification.root\_classification import classify\_root\_category

from embedding.generate\_embeddings import embed\_text

from retrieval.semantic\_search import search\_vector\_store

app = FastAPI()

@app.post("/search")

async def search(request: Request):

body = await request.json()

query = body["query"]

category = classify\_root\_category(query)

embedding = embed\_text(query)

results = search\_vector\_store(embedding)

return {"category": category, "results": results}

```

### 9. requirements.txt

```txt

openai

azure-cosmos

fastapi

uvicorn

pandas

```

---

Let me know when you're ready and I’ll walk you through environment setup and execution (incl. `.env` setup).

Data set summary:

Here’s a **summary of the dataset**:

**📊 Dataset Overview**

* **Total SKUs (ProductIDs)**: **500**
* **Unique Categories**: **10**
* **Unique Subcategories**: **30**

**📂 Categories**

* Cables and Wires
* HVAC
* Wiring Devices
* Data and Communication
* Industrial Automation
* Conduit and Fittings
* Safety and Security
* Lighting
* Tools and Testing
* Electrical Distribution

**📂 Subcategories**

* PVC Conduits
* Voltage Testers
* Motion Sensors
* THHN Wires
* Wall Plates
* Switches
* Duct Heaters
* EMT Fittings
* Relays
* Ventilation Fans
* Panel Boards
* LED Floodlights
* Drill Bits
* Circuit Breakers
* Low Voltage Cables
* Patch Panels
* Receptacles
* Load Centers
* Thermostats
* Armored Cables
* Fiber Optics
* PLCs
* Smoke Detectors
* Exit Signs
* Ethernet Cables
* Crimping Tools
* Wall Pack Lights
* Flexible Conduits
* High Bay Lights
* Contactors

Would you like me to also:  
✅ Provide counts of products per category and subcategory?  
✅ Visualize this data (e.g., bar chart or pie chart)?  
✅ Extract **top brands** and **common tags** used?

Let me know how you'd like to proceed! 🚀

**You said:**

explain in suppose categorie cables and wire how many sub catories with subcateogaries name like that I need summary

**ChatGPT said:**

Here’s a **detailed summary** for each category, including its **subcategories and the count of SKUs per subcategory**:

**📊 Category-wise Summary**

**Data and Communication**

* Subcategories:
  + Ethernet Cables: 15 SKUs
  + Fiber Optics: 23 SKUs
  + Patch Panels: 18 SKUs

**Electrical Distribution**

* Subcategories:
  + Load Centers: 19 SKUs
  + Panel Boards: 14 SKUs
  + Circuit Breakers: 15 SKUs

**Industrial Automation**

* Subcategories:
  + Contactors: 15 SKUs
  + Relays: 12 SKUs
  + PLCs: 18 SKUs

**Safety and Security**

* Subcategories:
  + Smoke Detectors: 15 SKUs
  + Exit Signs: 22 SKUs
  + Motion Sensors: 14 SKUs

**Cables and Wires**

* Subcategories:
  + Armored Cables: 22 SKUs
  + Low Voltage Cables: 14 SKUs
  + THHN Wires: 19 SKUs

**Wiring Devices**

* Subcategories:
  + Wall Plates: 16 SKUs
  + Switches: 9 SKUs
  + Receptacles: 20 SKUs

**Tools and Testing**

* Subcategories:
  + Crimping Tools: 18 SKUs
  + Drill Bits: 15 SKUs
  + Voltage Testers: 19 SKUs

**HVAC**

* Subcategories:
  + Duct Heaters: 20 SKUs
  + Ventilation Fans: 14 SKUs
  + Thermostats: 19 SKUs

**Conduit and Fittings**

* Subcategories:
  + PVC Conduits: 17 SKUs
  + EMT Fittings: 17 SKUs
  + Flexible Conduits: 15 SKUs

**Lighting**

* Subcategories:
  + Wall Pack Lights: 13 SKUs
  + High Bay Lights: 18 SKUs
  + LED Floodlights: 15 SKUs

Would you like me to:  
✅ **Visualize** this data (e.g., bar or pie charts)?  
✅ **Summarize the total SKUs per category**?  
Let me know how you’d like to proceed! 🚀

Code for this:

import json

from collections import defaultdict

# Load the dataset

with open('your\_dataset.json', 'r') as f:

data = json.load(f)

# Create a summary dictionary

category\_summary = defaultdict(lambda: defaultdict(int))

# Populate the summary

for record in data:

category = record['Category']

subcategory = record['SubCategory']

category\_summary[category][subcategory] += 1

# Print the summary

for category, subcats in category\_summary.items():

print(f"Category: {category}")

for subcat, count in subcats.items():

print(f" Subcategory: {subcat}, SKUs: {count}")

print("-" \* 40)