Delay Reason Classification using Maintenance Logs

1. Objective

- The objective of this exercise is to build a classifier flow that analyzes unstructured delivery, and maintenance logs, and automatically determines the **root cause** of delivery delays. The classifier must map free-text log entries into a set of predefined operational delay categories such as "Traffic", "Vehicle Issue", or "Weather".
- This helps logistics and transportation teams reduce the manual workload of log analysis, increase situational awareness, and accelerate operations reporting.

2. Problem Statement

- In logistics operations, drivers and maintenance teams often report incidents using free-text logs, e.g., “unexpected rainstorm at warehouse” or “vehicle failed mid-route.” Analyzing thousands of such reports manually to determine the **reason for delay** is inefficient and error prone.
- Your task is to build a simple classification pipeline using NLP techniques combined with a **refinement layer powered by Azure ChatOpenAI** to assign a proper category to each incident entry.

3. Inputs / Shared Artifacts

- Azure OpenAI Resource:
    - API key
    - Endpoint URL
    - Deployment name (to be set as environment variables).

- You are provided with the following **mock dataset** as input (you may hardcode it for testing or load from a CSV):

log_id	log_entry
1	"Driver reported heavy traffic on highway due to construction"
2	"Package not accepted, customer unavailable at given time"
3	"Vehicle engine failed during route, replacement dispatched"
4	"Unexpected rainstorm delayed loading at warehouse"
5	"Sorting label missing, required manual barcode scan"
6	"Driver took a wrong turn and had to reroute"
7	"No issue reported, arrived on time"
8	"Address was incorrect, customer unreachable"
9	"System glitch during check-in at loading dock"
10	"Road accident caused a long halt near delivery point"

4. Expected Outcome

Each log entry must be **classified into one of the following delay reason categories**:

```
[
  "Traffic",
  "Customer Issue",
  "Vehicle Issue",
  "Weather",
  "Sorting/Labeling Error",
  "Human Error",
  "Technical System Failure",
  "Other"
]
```

Example Input → Output:

Log Entry	Predicted Category
"Driver reported heavy traffic..."	Traffic
"Customer not available..."	Customer Issue
"Engine failed during route..."	Vehicle Issue
"Unexpected rainstorm..."	Weather
"Sorting label missing..."	Sorting/Labeling Error
"Wrong turn, rerouted..."	Human Error
"System glitch during check-in..."	Technical System Failure
"Address was incorrect..."	Customer Issue
"Road accident caused a halt..."	Traffic

5. Concepts Covered

- Azure OpenAI API usage
- Prompt Engineering
- NLP Task - Text Classification

6. Example: Implementation using Azure ChatOpenAI

```
import os
from openai import AzureOpenAI

# Step 1: Input Data
log_entries = [
    "Driver reported heavy traffic on highway due to construction",
    "Package not accepted, customer unavailable at given time",
    "Vehicle engine failed during route, replacement dispatched",
    "Unexpected rainstorm delayed loading at warehouse",
    "Sorting label missing, required manual barcode scan",
    "Driver took a wrong turn and had to reroute",
    "No issue reported, arrived on time",
    "Address was incorrect, customer unreachable",
    "System glitch during check-in at loading dock",
    "Road accident caused a long halt near delivery point"
]
```

```
# Step 2: Heuristic Pre-classifier
def initial_classify(text):
    keywords = {
        "traffic": "Traffic",
        "road accident": "Traffic",
        "customer": "Customer Issue",
        "unavailable": "Customer Issue",
        "engine": "Vehicle Issue",
        "vehicle": "Vehicle Issue",
        "rain": "Weather",
        "storm": "Weather",
        "label": "Sorting/Labeling Error",
        "barcode": "Sorting/Labeling Error",
        "wrong turn": "Human Error",
        "reroute": "Human Error",
        "system": "Technical System Failure",
        "glitch": "Technical System Failure"
    }
    for k, v in keywords.items():
        if k in text.lower():
            return v
    return "Other"

# Step 3: AzureOpenAI Setup
client = AzureOpenAI(
    api_version="2024-07-01-preview",
    azure_endpoint=os.environ["AZURE_OPENAI_ENDPOINT"],
    api_key=os.environ["AZURE_OPENAI_API_KEY"],
)

deployment_name = "gpt-4o-mini"

def refine_classification(text, initial_label):
    prompt = f"""
You are a logistics assistant. A log entry has been auto-categorized as
"{initial_label}". Please confirm or correct it by choosing one of the
following categories:

- Traffic
- Customer Issue
- Vehicle Issue
- Weather
- Sorting/Labeling Error
- Human Error
- Technical System Failure
- Other

Log Entry:
\"\"\"{text}\"\"\"

Return only the most appropriate category from the list.
"""
    response = client.chat.completions.create(
        model=deployment_name,
        messages=[{"role": "user", "content": prompt}],
```

```
        temperature=0,
    )
    return response.choices[0].message.content.strip()

# Step 4: Final Classification Pipeline
def classify_log(text):
    initial = initial_classify(text)
    final = refine_classification(text, initial)
    return {"log": text, "initial": initial, "final": final}

# Step 5: Example Execution
if __name__ == "__main__":
    for entry in log_entries:
        result = classify_log(entry)
        print(f"\nLog Entry: {result['log']}")
        print(f"Predicted Category: {result['final']}")
```

7. Final Submission

To complete this exercise, you should submit:

- A .py or .ipynb file containing the complete implementation.
- A list or table showing the **predicted category for each of the 10 sample log entries**.
- Short summary on how this approach differs from retrieval-based pipelines and where it fits best.