```python
import os
from typing import List
from pydantic import BaseModel, ValidationError
from langchain_openai import AzureChatOpenAI
from langchain.prompts import PromptTemplate

# === ENVIRONMENT VARIABLES ===
AZURE_OPENAI_ENDPOINT = os.environ["AZURE_OPENAI_ENDPOINT"]
AZURE_OPENAI_API_KEY = os.environ["AZURE_OPENAI_API_KEY"]

# === Pydantic Model ===
class MeetingNotes(BaseModel):
    summary: str
    action_items: List[str]

# === Prompt Template (from assignment) ===
prompt_template = PromptTemplate(
    input_variables=["transcript"],
    template=(
        "You are a meeting assistant.\n"
        "1. Summarize the meeting transcript below in exactly two sentences.\n"
        "2. Then list all action items mentioned, each as a separate bullet beginning with a dash.\n"
        'Return the result strictly as JSON with keys "summary" and "action_items".\n\n'
        "Transcript:\n{transcript}"
    )
)

# === Azure OpenAI Model (use your proven working connection) ===
llm = AzureChatOpenAI(
    openai_api_key=AZURE_OPENAI_API_KEY,
    azure_endpoint=AZURE_OPENAI_ENDPOINT,
    openai_api_version="2025-01-01-preview",
    model="gpt-4o-mini",
    temperature=0
)

def extract_meeting_notes(transcript: str) -> dict:
    """Calls Azure OpenAI to extract a summary and action items from a transcript."""
    prompt_content = prompt_template.format(transcript=transcript)
    messages = [
        {"role": "system", "content": "You are a helpful meeting assistant."},
        {"role": "user", "content": prompt_content}
    ]
    # First call
    response = llm.invoke(messages)
    output_text = getattr(response, "content", None)
    if not output_text and hasattr(response, "choices"):
        output_text = response.choices[0].message.content

    # Try to parse
    try:
        import json
        data = json.loads(output_text)
        validated = MeetingNotes(**data)
        return validated.dict()
    except Exception:
        # Retry once with "Please output valid JSON only"
        retry_messages = [
            {"role": "system", "content": "You are a helpful meeting assistant. Please output valid JSON only."},
            {"role": "user", "content": prompt_content}
        ]
        response2 = llm.invoke(retry_messages)
        output_text2 = getattr(response2, "content", None)
        if not output_text2 and hasattr(response2, "choices"):
            output_text2 = response2.choices[0].message.content
        try:
            data = json.loads(output_text2)
            validated = MeetingNotes(**data)
            return validated.dict()
        except Exception as e:
            raise RuntimeError(f"Could not parse model output after retry: {e}\nLast output: {output_text2}")

# === Test with Sample Inputs from PDF ===
if __name__ == "__main__":
    SAMPLE_TRANSCRIPTS = [
        # Sample Input 1 from PDF
        """Host: LetвЂ™s kick off our marketing sync.
Emma: The social campaign draft is 80% done; IвЂ™ll share it today.
Frank: I spoke with the design teamвЂ”theyвЂ™ll deliver assets by Tuesday.
Emma: Once we have assets, IвЂ™ll schedule the ads for next week.
George: Reminder: submit your budget requests before end of day.
Host: Noted. IвЂ™ll send out the final budget spreadsheet.""",
        # Sample Input 2 from PDF
        """Alice: Welcome everyone. Today we need to finalize the Q3 roadmap.
Bob: IвЂ™ve emailed the updated feature listвЂ”please review by Friday.
Carol: IвЂ™ll set up the userвЂђtesting sessions next week.
Dan: LetвЂ™s push the new UI mockups to staging on Wednesday.
Alice: Great. Also, can someone compile the stakeholder feedback into a slide deck?
Bob: I can handle the slide deck by Monday.
Alice: Thanks, team. Meeting adjourned.""",
        # Sample Input 3 from PDF
        """Priya: WeвЂ™ve identified a bottleneck in the checkout API.
Carlos: IвЂ™ll benchmark the current response times by tomorrow.
Priya: After that, optimize the database indices.
Carlos: Sure, IвЂ™ll propose index changes by Thursday.
Sara: Also, draft a rollback plan in case of production issues.
Carlos: IвЂ™ll include that in my ticket.""",
        # Sample Input 4 from PDF
        """HR: We need to update the travelвЂђexpense policy.
Alex: IвЂ™ll draft the new policy doc and share with legal.
HR: Please include guidelines on per diem limits.
Legal: IвЂ™ll review and provide feedback within three days.
HR: Once finalized, schedule a companywide announcement.""",
        # Sample Input 5 from PDF
        """John: Sales numbers for April are inвЂ”up 12% over target.
Lia: IвЂ™ll distribute the detailed report to leadership this afternoon.
Mike: We need to train two new reps to handle increased volume.
Lia: IвЂ™ll coordinate recruiting with HR by end of week.
John: Excellent. LetвЂ™s aim to onboard them by May 15th."""
    ]

    for idx, transcript in enumerate(SAMPLE_TRANSCRIPTS, 1):
        print(f"\n=== Sample Input {idx} ===")
        try:
            notes = extract_meeting_notes(transcript)
            print("Summary:", notes["summary"])
            print("Action Items:")
            for item in notes["action_items"]:
                print(item)
        except Exception as e:
            print("Error parsing meeting notes:", e)
```