AI Engineer Take-Home Assignment: Call Transcript Analysis

1. Introduction & Context

This assignment is designed to assess your practical AI engineering skills in a scenario that reflects the type of challenges we tackle here. Imagine that you need to build a new product to help businesses gain insights from their customer support calls. A core feature is the ability to automatically categorize the issues discussed in each call.

Your task is to build a program that analyzes a call center transcript and accurately identifies the topics discussed according to a dynamic, client-defined list of categories.

2. The Core Problem

You will write a program or script that takes a call transcript and a set of categories as input, and outputs a structured list of all issues identified in the call.
This assignment is broken into three parts, representing an evolution of the project's requirements. Your final submission should address all three parts in a single, cohesive solution.

Provided Assets

You will be working with two types of inputs:

1. **A sample transcript (transcript.txt):**

   Agent: Thank you for calling Tech Support, my name is Brenda. How can I help you today?
   Caller: Hi Brenda. I'm having a really frustrating issue. My internet connection has been incredibly slow for the past three hours. I can barely load a webpage.
   Agent: I'm sorry to hear that. I can definitely look into your network status. Can you please provide your account number?
   Caller: Yes, it's 123-456-789.
   Agent: Thank you. I'm pulling up your account... I see some reported slowness in your area, and we're working on it. I expect it to be resolved within the next hour.
   Caller: Okay, thanks for the update. While I have you on the line, I also wanted to ask about my last bill. It seems much higher than usual. I don't understand the new data usage charges.
   Agent: Of course. I can transfer you to our billing department to discuss the cost inquiry,

or I can pull up the details for you here if you'd like.
Caller: Just transfer me, that's fine.
Agent: Understood. One moment please.

2. **A categories definition file (categories.json):**

```
[
 {
   "value": "Network Issue",
   "description": "The caller is facing network issues",
   "subcategories": [
     {"value": "Slow Connection", "description": "The caller reported slow connection"},
     {"value": "Network Down", "description": "The caller reported their network is down"},
     {"value": "Other"}
   ]
 },
 {
   "value": "Billing",
   "description": "The caller made billing inquiries",
   "subcategories": [
     {"value": "Payment not processed", "description": "The caller reported an error while paying
for a service"},
     {
       "value": "Cost Inquiry",
       "description": "The caller is inquiring about costs for a particular service",
       "subcategories": [
         {"value": "DSL"},
         {"value": "5G"},
         {"value": "Fiber"}
       ]
     },
     {"value": "Other"}
   ]
 },
 {
   "value": "Account Management",
   "subcategories": [
       {"value": "Update Personal Info"},
       {"value": "Password Reset"}
   ]
 }
]
```

3. Assignment Requirements

Part 1: Single Category Classification (Static)

First, imagine the requirements are simple. The program only needs to find the *primary* issue in a transcript.

- **Task:** Write a function or script that analyzes the transcript and identifies the single most prominent category and its corresponding subcategory.
- **Categories:** For this part, assume a fixed, hard-coded list of categories:
  - **Network Issue:** (Subcategories: Slow Connection, Network Down, Other)
  - **Billing:** (Subcategories: Cost Inquiry, Payment Not Processed, Other)

Part 2: Multi-Issue Extraction

Callers often discuss more than one topic. Your program must be able to capture all of them.

- **Task:** Evolve your script to extract a **list** of all categories and their most specific subcategories mentioned in the transcript.
- **Example:** For the sample transcript provided, your program should identify both the "Network Issue" and the "Billing" issue.

Part 3: Dynamic and Nested Category Loading

Our clients want to manage their own categories without needing a developer to update the code. Your script must be flexible enough to handle this.

- **Task:** Adapt your script so that it no longer uses a hard-coded list. Instead, it must read the categories from the provided categories.json file at runtime. The script must be able to parse the **nested structure** of this file and handle varying levels of depth

.

4. Key Technical Guidelines & Constraints

- **Language:** You are free to use any programming language, though Python is preferred.
- **LLM Utilization:** You **must** use a Large Language Model (LLM) to perform the core analysis. The goal is not to build a traditional NLP model from scratch, but to demonstrate your ability to effectively use modern AI APIs.
- **Structured Output:** Your program's final output for a given transcript **must** be a structured format (e.g., a JSON object). Additionally, you must ensure that the subcategories fall under the correct categories.
- **Single API Call:** For efficiency and atomicity, it is **highly preferable** to design your prompt and system to solve the entire task (reading the transcript, using the dynamic categories, and generating the structured list) in a **single API call** to the LLM.

- **Error Handling:** Your program should gracefully handle cases where a transcript does not contain any of the listed categories. In such a scenario, it should output an empty list or simply just "Other".

5. Expected Output

For the sample transcript.txt and categories.json provided, the final output from your program should identify the most specific path for each issue found. The output should look exactly like this:

```
{
 "analysis_results": [
   ["Network Issue", "Slow Connection"],
   ["Billing", "Cost Inquiry", "DSL"]
 ]
}
```

Note: it is not necessary for the LLM's output to look like this, but you should be able to transform its output into this format.

6. What to Submit

Please provide your source code and include a README.md that contains:

- Instructions on how to set up and run your program.
- Any dependencies that need to be installed.
- A brief explanation of your approach, design choices, and how you engineered your prompt for the LLM.

7. Evaluation Criteria

We will be evaluating your submission based on the following:

- **Correctness:** Does the program successfully meet all requirements from Parts 1, 2, and 3? Does it produce the correct output for the sample data?
- **Code Quality:** Is the code clean, well-structured, commented, and easy to understand?
- **Design & Approach:** How effectively did you use the LLM? Did you make good design choices regarding prompt engineering and structured data extraction?

Good luck! We look forward to seeing your solution.