Intern Assessment Task: Build a Simple Action Suggester API

Expected hours: (4-6 hours)

Goal:

This task assesses your ability to build a basic web service using Django and integrate it with an external AI (LLM) API. Your goal is to create a simple API that analyzes a short text message, understands its basic tone and intent using an LLM, and suggests relevant actions.

Project Overview:

You will build a Django backend service with a single API endpoint. When this endpoint receives a user's text message (e.g., "I want to order pizza"), it should:

- Call an external LLM API (like Google AI or OpenAI) to determine the message's tone (e.g., Happy, Urgent) and intent (e.g., Order Food, Ask Question).
- Based on the identified tone and intent, suggest 1-3 predefined actions (e.g., Order Food Online, Find Pizza Recipes).
- Log the request details (query, analysis, suggestions) to a PostgreSQL database.

Core Requirements:

*** Django Setup:**

* Create a standard Django project.
* Configure it to use a PostgreSQL database.
* Use Django REST Framework (DRF) to build the API endpoint.

*** API Endpoint:**

* Create one endpoint: POST /api/analyze/
* Request: Accepts JSON like { "query": "Your text message here" }.
* Response: Returns JSON like:

```
{
      "query": "User's message",
      "analysis": {
        "tone": "Identified Tone",
        "intent": "Identified Intent"
      },
      "suggested_actions": [
        {"action_code": "ACTION_1", "display_text": "Suggestion 1"},
        // ... more suggestions (up to 3)
      ]
}
```

*** LLM Integration:**
  * Choose an LLM provider (e.g., Google AI Studio's Gemini API - often has a free tier, or OpenAI's API).
  * Write Python code to call the LLM API with the user's query.
  * Prompt: Ask the LLM to identify the primary tone and intent of the query. Try to get a simple response.
  *** API Key Security:** Crucially, load the LLM API key from an environment variable (e.g., using a .env file and python-dotenv). Do not hardcode the key in your Python files.

*** Action Suggestion Logic:**
  * Define a small set of possible actions directly in your Python code (a dictionary is fine). Examples: ORDER_FOOD, FIND_RECIPE, ASK_HELP, SHARE_NEWS.
  * Write simple Python logic (e.g., if/elif/else based on tone and intent strings) to select 1-3 relevant actions from your predefined list for the API response.

*** Database Logging:**
  * Create a simple Django ORM model (e.g., QueryLog).
  * Store: the original query text, timestamp, identified tone, identified intent, and the list of suggested actions (you can store this list as JSON in a JSONField or as a simple string in a TextField).
  * Save a log entry to the database for each successful API call.

What We're Looking For:

* Functionality: Does the API endpoint work as described? Does it call the LLM, process the response, suggest actions, and log to the database?
* Code Clarity: Is your code reasonably well-organized and easy to understand?
* Following Instructions: Did you meet the core requirements, especially regarding API structure and key handling?
* Basic Error Handling: Does your code handle potential issues gracefully (e.g., if the LLM API call fails or returns unexpected data)? (Basic try-except blocks are sufficient).

Submission:

* Please provide a link to a Git repository (e.g., GitHub, GitLab) containing your complete Django project code.
* Include a requirements.txt file listing the necessary Python packages.
* Include a brief README.md file explaining:
  * How to set up the project (e.g., install requirements, run migrations).
  * Mention the api endpoints in README.md making sure can test using Postman.
  * Which environment variables are needed (especially for the LLM API key).
  * Which LLM provider you used.