📘 AI Lead Collector Agent – Project Documentation

🧠 Overview

AI Lead Collector Agent is an intelligent, conversational lead collection system developed using Google ADK and Django. The system is designed to automate user interactions, collect essential user information with consent, and manage follow-up communication efficiently.

The agent is triggered automatically when a user submits a form, initiating a conversation with the user and collecting details that are stored in a structured format for future use.

🔄 System Flow

1. User Form Submission
    - The process begins when a user submits a web form on the frontend.
    - This action automatically initiates the AI agent via Google ADK.
    - A new conversation session is created and permanently stored in the SQLite database.
2. AI Agent Conversation
    - The agent starts with a friendly greeting and asks for consent to collect personal data.
    - If the user agrees, it asks a series of questions:
        - Name
        - Country
        - Interest
    - If the user declines, the agent records a minimal entry with a "no" status and gracefully ends the conversation.
3. Data Storage in CSV
    - Collected data is saved to a CSV file.
    - Each entry includes:
        - lead_id (unique ID per session)
        - name (if provided)
        - country (if provided)
        - interest (if provided)
        - status (either done or no)
        - timestamp (time when the entry was stored)
        - followup_status (initially set to no)

⏱️ Follow-Up System

🔁 Independent Follow-Up Logic (followup_trigger.py)

- A completely independent script called followup_trigger.py runs separately from the Django backend and the AI chat agent.

- It continuously monitors the CSV data for rows where:
  - status is no
  - followup_status is no
- If these conditions are true, it checks whether at least 1 minute has passed since the entry was saved (using the timestamp).
- If the time condition is met:
  - A follow-up message is triggered to re-engage the user.
  - The followup_status is then updated to prevent repeated messages.

This design ensures that follow-up actions do not interfere with Django’s main server or the AI agent's logic.

🗃️ Session & Message History

- All AI conversations and user sessions are stored in a SQLite database, ensuring:
  - Permanent storage of user interactions
  - Support for reloading and reviewing past sessions
  - Easy integration with Django admin tools if needed

🧰 Tech Stack

Component	Description
Google ADK	Agent development and conversation logic
Django	Web framework and backend logic
SQLite	Session and message history database
CSV	Lead data storage format
Python (followup_trigger.py)	Standalone scheduler for follow-ups

✅ Summary

This system provides a scalable, automated, and intelligent way to handle lead generation and engagement. By separating concerns—frontend interaction, conversational logic, and follow-up scheduling—it ensures high reliability and modularity.