AI/ML – Pre-Employment Basic Assessment

Small Conversational AI Project:

Your task is to build a minimal conversational AI service using a small, self-hosted, instruction-tuned generative AI model that can run efficiently on a CPU. The chatbot must support multiple languages (at least English and one additional language of your choice) and be accessible via RESTful APIs. You can choose any lightweight, open-source model (e.g., Dolly, Alpaca, or a small variant of Llama 2 Chat/Vicuna) that meets these requirements.

Requirements

1. **Model Selection & Setup**
   - **Model Choice:** Select a lightweight generative model (ideally around 7B parameters or smaller) that has been fine-tuned for instruction following or conversational tasks.
   - **CPU Optimization:** Ensure that your model is configured to run on a CPU environment.
   - **Multi-Language Capability:** The model should be able to generate coherent responses in English and at least one additional language (e.g., Spanish, French, etc.).
2. **REST API Development**
   - **Framework:** Develop the API using a framework of your choice (e.g., Flask, FastAPI, or Django REST Framework).
   - **Endpoints:**
     - **POST /chat:** Accepts a JSON payload with at least:
       - conversation_id: Identifier for the conversation session.
       - messages: An array of message objects (each with a role (user/system) and content).
       - language (optional): The language code (e.g., "en" or "es").
     - **Response:** Returns the model's generative response as JSON.
   - **Error Handling:** Implement appropriate error responses for invalid inputs or internal failures.

3. **Generative Response Requirements**

    - **Instruction Following:** Pre-pend or incorporate a system prompt that defines the bot's role (e.g., "You are a helpful Saloon assistant to book appointment for hair cut, facial between 10AM to 8PM" or any context relevant to your test use-case).
    - **Multi-Language Output:** The bot should detect or be guided by the language parameter to produce responses in the specified language.
    - **Conversational Context:** Use the conversation history (provided in the messages array) to generate a context-aware response.

4. **Documentation & Testing**

    - **README:** Provide a README file that includes:
        - A brief explanation of your design decisions.
        - Instructions on how to set up and run your application (including model download/setup if applicable).
        - Examples of API requests (with Postman exported file).
    - **Tests:** Include simple unit tests or API tests to demonstrate your endpoints work as expected.

5. **Submission & Timeline**

    - Provide a Git repository link or a compressed archive with your complete code, documentation, and test cases.
    - A screen recording video demonstrating how it is changing behaviour based the system prompt.
    - This project needs to completed and submitted within 72 hours.

6. **Extra Functionalities (Optional – if achieved will give an edge)**

    - **State Management:** Implement session or conversation state management that preserves context between multiple API calls.
    - **Deployment:** Containerize your application using Docker and provide a Dockerfile with instructions on running the container.