ADIIVA Take-Home Assignment

Overview

Your task is to **prototype a text-based AI companion** that:

- Accepts a text prompt (simulating a 4-8 year old child's question).
- Filters unsafe or inappropriate content before sending it to an LLM.
- Returns either a safe child-friendly answer or a safe fallback message.
- Is packaged so it can be run via Docker with a single command.
- The service should expose a **simple REST API** so it can be extended into a real application.

Part 1 - Functional Prototype

Build a **Python-based service** that:

1. **Accepts text input** (via CLI or HTTP endpoint).
2. **Filters unsafe content** using either:
    - A simple keyword list you define, **or**
    - A moderation API (e.g., OpenAI's moderation endpoint).
3. **Generates a child-friendly response** from an LLM (open-source or commercial)
4. **Returns a safe fallback response** if the input is flagged as unsafe.
5. **Logs** the interaction to the console, file or DB, including whether the prompt was flagged as unsafe.
6. **Handles unsafe input gracefully** by returning a safe fallback response (e.g., "Let's talk about something else!").

Part 2 - REST API

- Implement a lightweight API (using FastAPI or Flask).
- Provide at least one endpoint:
    - `POST /ask` → takes a JSON body `{ "question": "..." }` and returns a JSON response `{ "response": "...", "safe": true/false }`.
- Ensure the service runs inside Docker and can be tested with `curl` or Postman.

Part 3 - Dockerized Deployment

- Include a `Dockerfile` that installs all dependencies.
- The application should run with **one command** after cloning (document it in the README).
- No platform-specific dependencies — it should run on Windows, macOS, and Linux.

Part 4 – Observability & Monitoring

Add **basic observability features** so the service can be monitored for uptime and health:

- Implement a **`GET /health` endpoint** that returns `{ "status": "ok" }` when the service is running.
- Expose at least **two simple metrics**
- Provide these metrics in a way that monitoring tools could scrape them, or log them in a structured format.
- Document in the README how you would monitor this service in production

Part 5 - Documentation & Examples

Your **README.md** should include:

- How to build and run the Docker container.
- How the safety filtering works (definition of “unsafe” content).
- The LLM you chose and why.
- **2–3 example inputs/outputs**, including at least one unsafe prompt.

Deliverables

- Python code implementing safe response logic.
- REST API exposing `/ask`.
- `Dockerfile` for reproducible setup.
- README with run instructions, safety approach, and examples.

Evaluation Criteria

- **Code Quality** – Clean, modular, easy to follow.
- **Safety Filtering** – Effectiveness and clarity of logic.
- **Prompt Engineering** – Ensures child-appropriate, clear language.
- **Ease of Deployment** – Minimal friction to run via Docker.