Problem: LLM-Based Input Validator

Objective

Build a **small LLM-powered validation script in Python** that checks user input data and returns **strict structured output**.

No external validation libraries or helper tools may be used—the LLM must be the sole validator.

Problem Statement

You will build a script that validates a simple user profile object by delegating all validation logic to an LLM via prompt engineering.

The script must:

- Accept a file containing raw JSON input
- Ask the LLM to validate it based on high-level constraints
- Return only a strictly structured JSON response
- Be testable via automated evals

Input

A JSON file with the following fields:

```
{
  "name": string | null,
  "email": string | null,
  "age": number | null,
  "country": string | null,
  "phone": string | null
}
```

Output (must match schema exactly)

```
{
  "is_valid": boolean,
  "errors": string[],
```

```
  "warnings": string[]
}
```

Validation Rules

Errors

- `name` is required and must be non-empty
- `email` must be a valid email address
- `age` must be a positive number
- `country` must be a valid ISO-2 country code (e.g. `US`, `IN`)
- `phone number` must be present and in the E.164 format

Warnings

- `age` is below 18
- `name` is shorter than 3 characters
- `email` uses a disposable or temporary email address (eg: sasaled582@akixpres.com)
- country code in `phone` does not align with country

Important Rules

- **Do not use any validation library,** the objective is to engineer a prompt that makes sure an LLM can validate inputs and catch errors. The LLM should be the only validator
- **Do not put all validation rules in the prompt.** Instead, write prompts that express the constraint at a higher level and let the model infer the details. For example:
    - Bad
        - The phone number must not contain alphabets
        - The phone number must be exactly 10 digits
    - Good
        - The phone number must be in the E.164 format

      This avoids incomplete or contradictory rule lists and aligns better with real-world standards

- Do not infer or fabricate missing data
- Do not invent new fields or rules
- Missing fields should be ignored. Only apply validation rules on fields that are present in the input
- All messages must be grounded strictly in input values

- If multiple rules apply, report all of them

Examples

Example 1 — Invalid Input

input1.json

```
{
  "name": "",
  "email": "user@gmail",
  "age": 16,
  "country": "India",
  "phone": "99999"
}
```

Output

```
{
  "is_valid": false,
  "errors": [
    "name is required",
    "email is not a valid email address",
    "country must be a valid ISO-2 country code",
    "phone number is not in E.164 format"
  ],
  "warnings": [
    "age is below recommended minimum",
  ]
}
```

Example 2 — Valid Input

input2.json

```
{
  "name": "Aarav Patel",
  "email": "aarav.patel@gmail.com",
  "age": 24,
  "country": "IN",
  "phone": "+919876543210"
}
```

Output

```
{
  "is_valid": true,
  "errors": [],
  "warnings": []
}
```

Technical Requirements

1. **LLM-Based Validation**
    - Your script must rely **solely on an LLM** to perform all validations.
    - Do not use MCP tools, helper libraries, or hardcoded rules.
2. **Structured Output**

    Enforce the output schema strictly:

    ```
    {
      "is_valid": boolean,
      "errors": string[],
      "warnings": string[]
    }
    ```

    - Handle invalid or malformed LLM outputs gracefully (retry or fail cleanly).

3. **Automated Evals**
    - Use promptfoo to run evals:
        - https://www.promptfoo.dev/docs/getting-started
        - https://www.promptfoo.dev/docs/configuration/expected-outputs/
    - Following the docs, set up an eval test suite with test cases and expected outputs
        - You will be asked to run the eval during the interview

        - You will also be provided with hidden tests to test the output quality
    - Evals must check:
        - Schema correctness
        - No hallucinated rules or fields
        - Correct classification of valid vs invalid inputs

    - Evals must be runnable via a **single command**.

4. **Input / Output**
    - Accept a JSON file as input to the script.
    - Return JSON output following the schema above.

Example command:

```
python validate_user.py user.json
```

Where `user.json` is the input file containing a JSON object, for example:

```
{
  "name": "Aarav",
  "email": "aarav@gmail.com",
}
```

Tech Stack

- Python 3.10+
- LLM client (OpenAI, Anthropic, or equivalent)
- **Use Promptfoo for evals**:
    - https://www.promptfoo.dev/docs/getting-started
    - https://www.promptfoo.dev/docs/configuration/expected-outputs/

Deliverables

Your repository must include:

- Source code
- `.env.example`
- `README.md` with:

    - Setup instructions
    - How to run the script
    - How to run evals
- A script or command to run evals

Evaluation Criteria

You will be evaluated on:

- Output quality and schema discipline
- Prompt clarity
- Eval quality
- Ease of running the project
- Ability to communicate concepts and project structure