Software Engineering Challenge

There are successful brands who have managed to stay relevant for more than a decade without innovating too much. Some of our colleagues have young children and we'd like to offer them a fresh perspective of the world of Pokemon.

We would like you to develop your own fun 'Pokedex' in the form of a REST API that returns Pokemon information. Don't worry, you'll be using existing public APIs to do the heavy lifting.

The API has two main endpoints:
1. Return basic Pokemon information.
2. Return basic Pokemon information but with a 'fun' translation of the Pokemon description.

Following are more detailed API requirements. Guidelines can be found on Page 3.

API Requirements
Endpoint 1 - Basic Pokemon Information
Given a Pokemon name, returns standard Pokemon description and additional information.
This endpoint should call the PokéAPI (https://pokeapi.co/).

Example endpoint:
```
/HTTP/GET /pokemon/<pokemon name>
```

Example call (using httpie):
```
http http://localhost:5000/pokemon/mewtwo
```

The API response should contain a minimum of:
- Pokemon's name
- Pokemon's standard description
- Pokemon's habitat
- Pokemon's `is_legendary` status

Example response:
```
{
  "name": "mewtwo",
  "description": "It was created by a scientist after years of horrific gene
splicing and DNA engineering experiments.",
  "habitat": "rare",
  "isLegendary": true
}
```

Endpoint 2 - Translated Pokemon Description
Given a Pokemon name, return translated Pokemon description and other basic information using the following rules:

1. If the Pokemon's habitat is `cave` or it's a legendary Pokemon then apply the Yoda translation.
2. For all other Pokemon, apply the Shakespeare translation.
3. If you can't translate the Pokemon's description (for whatever reason 😉) then use the standard description

This endpoint should call the PokéAPI (https://pokeapi.co/) and the FunTranslations API (https://funtranslations.com)

Example endpoint:
```
HTTP/GET /pokemon/translated/<pokemon name>
```

Example call (using httpie):
```
http http://localhost:5000/pokemon/translated/mewtwo
```

The API response should contain a minimum of:

- Pokemon name
- Translated Pokemon description
- Pokemon's habitat
- Pokemon's `is_legendary` status

Example response:
```
{
  "name": "mewtwo",
  "description": "Created by a scientist after years of horrific gene
splicing and dna engineering experiments, it was.",
  "habitat": "rare",
  "isLegendary": true
}
```

Guidelines

- Feel free to use any programming language, framework and library you want.
- Make it concise, readable and correct.
- If you would have made a different design decision for production, then comment or document it.
- We love high-value unit tests which test the right things!
- The task requirements are fairly trivial - we're more interested in your design decisions, code layout and approach (be prepared to explain this).
- Useful APIs:
    - PokéAPI: https://pokeapi.co/
        - hint: most of what you need is under the `pokemon-species` API.
        - hint: the Pokemon's description can be found under the `flavor_text` array. You can use any of the English descriptions.
    - Shakespeare translator: https://funtranslations.com/api/shakespeare
    - Yoda translator: https://funtranslations.com/api/yoda

Please describe in the README.md:

- How to run it (don't assume anything is already installed)
- Anything you'd do differently for a production API

Bonus points for:

- Dockerfile
- Include your git history

Have fun, take your time and when you are done please send a link to your public Github repo to your Talent Acquisition Partner!