Currency Converter API

Overview

The Currency Converter API will assist with converting amounts from one currency to another. This document outlines the functional and non-functional requirements for this API.

> Expected implementation time is around 30 to 45 minutes.

Submission Instructions

- Please use only .NET Core 6.0 or above.
- Please use either Azure functions or Asp.NET core to author your API.
- Please upload your solution to your GitHub repository and email us the link.
- Please include a README.MD file with instructions on how to run your application. The expectation is that we should be able to run your application locally and test it out.
- Please feel free to use ChatGPT CoPilot or any assistant of your choice. We'll be doing a follow-up interview to discuss your solution, so please be prepared to explain your thought process and the decisions you made.

Functional Requirements

1. Currency Conversion Endpoint

- **Endpoint:** `GET /convert`
- **Description:** Converts a specified amount from a source currency to a target currency.
- **Query Parameters:**
  - `sourceCurrency` (string): The ISO 4217 code of the source currency.
  - `targetCurrency` (string): The ISO 4217 code of the target currency.
  - `amount` (decimal): The amount to convert from the source currency.
- **Response:** A JSON object with the following properties:
  - `exchangeRate` (decimal): The exchange rate used for conversion.
  - `convertedAmount` (decimal): The resulting amount in the target currency.

2. Exchange Rates

- The API should support conversions between the following currency pairs: `USD`, `INR`, `EUR`
- The API should extract the above exchange/conversion rates from a local file called `exchangeRates.json`, which would look something like this:

```
{
    "USD_TO_INR": 74.00,
    "INR_TO_USD": 0.013,
    "USD_TO_EUR": 0.85,
    "EUR_TO_USD": 1.18,
    "INR_TO_EUR": 0.011,
```

```
    "EUR_TO_INR": 88.00
}
```

- We should be able to override these exchange rates using environment variables. E.g. An env var `USD_TO_INR=81.00` should override the value of `USD_TO_INR` in the `exchangeRates.json` file.

Non-Functional Requirements

These non-functional requirements are intentionally a bit vague, ambiguous, and open to interpretation.

1. Unit Testing

- Write unit tests for the conversion logic. Tests must cover various scenarios, including edge cases. Use any testing framework & mocking libraries of your choice (ie. any 3rd party nuget package is fine).

2. Logging

- Implement logging for key events in the application, including successful conversions and errors. If you want to, you can even use a 3rd party logging library.

3. Error Handling

- Robust error handling for invalid input, and unsupported currencies. Try to avoid 5xx / server errors.
- Clear, descriptive error messages should be returned in the API response.

Bonus Requirements

- **Dynamic Configuration:** Allow updating exchange rates without restarting the application.