Exchange Rate Service

Overview

You are tasked with building a Currency Exchange Rate Service that allows clients to fetch real-time exchange rates and perform currency conversions. The service should support caching for efficiency, and historical data retrieval.

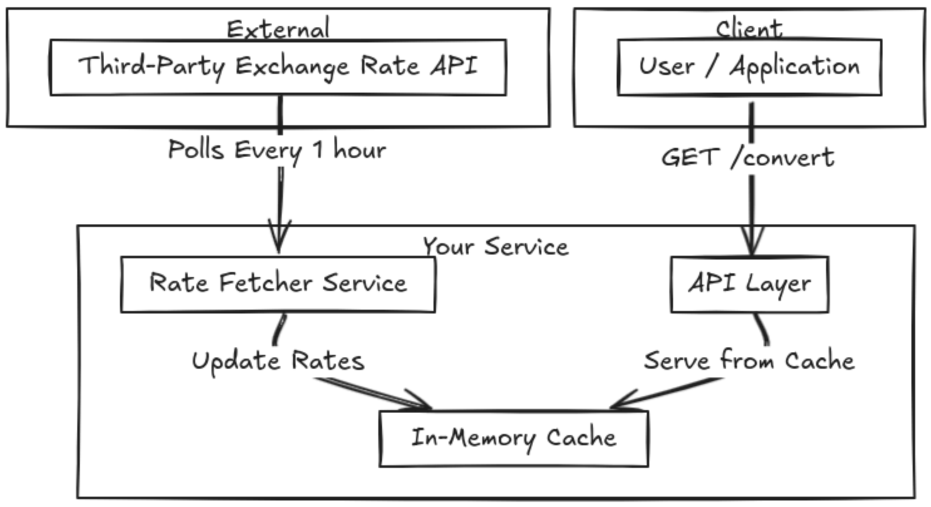
When fetching historical exchange rates, the system must enforce certain constraints to maintain performance and data integrity. The service should validate that the date is well-formed and logical (i.e., the date should be provided in a sane format). Additionally, the system should limit the maximum lookback date (e.g., a date older than 90 days from today should return an error).

Problem Statement

Your mission is to create a backend service that:

1. Ingests **foreign exchange rates** from a public API (suggestion: https://exchangerate.host/ or https://open.er-api.com/) or a mock data source.
2. Provides a **RESTful API** to:
    - Fetch the latest exchange rate for a currency pair.
    - Convert an amount from one currency to another.
    - Retrieve historical rates for a currency pair for a specific date range.
3. Refresh the current date's exchange rate every hour.
4. Can support a fixed set of currencies:
    - United States Dollar (USD)
    - Indian Rupee (INR)
    - Euro (EUR)
    - Japanese Yen (JPY)
    - British Pound Sterling (GBP)
5. You need to service the historic data only for the last 90 days. Anything beyond that, an error should be returned.

You can refer to the architecture diagram as seen below. This is just an indicative diagram, you may choose your own structure for better efficiency or performance:



Functional Requirements

API Endpoints

Request	Response	Description
`GET /convert?from=USD&to=INR&date=2025-01-01`	`{ "amount": 8312.50 }`	Convert the amount from one currency to another. The date field is **optional**. If not provided, it should consider the current date.

Bonus Points

- Implement **in-memory caching** for the latest exchange rates to reduce API calls.
- Handle **concurrent requests** gracefully, ensuring no data races occur during rate updates.
- Implement **graceful error handling** for third-party API failures.
- Write **unit tests** for key functions.

Expectations

- Code should be written in **Go** (or your strongest backend language if specified otherwise).
- Follow **clean architecture** principles (packages, separation of concerns).
- Dockerize your service for easy deployment (`Dockerfile` required).
- Provide a `README.md` with:
  - How to run the project.
  - How to test the endpoints (examples with `curl` or Postman).
  - Assumptions you've made.
- You will have to keep an eye for performance issues
- Think about scalability (horizontal as well as vertical)
- Write test cases to check for correctness
- Your architecture should comply with the read-heavy nature of this problem.

Submission

Once you are happy with your solution, please check in your repo to any open CVS like GitHub but as a private repo and share the link in the same email where you got the assignment. You can also share the repo privately with: arpit.ch@greedygame.com so that we can do a code review. **Please make sure that you do not commit this document as a part of your commit.**

If you need any help with the problem or otherwise, please respond back to the same thread where you got the problem from.

Bonus

1. If you can put in instrumentation using Prometheus and Grafana for your service, that will be an added benefit.
2. We always love to see a good and clean code commit history, instead of everything pushed into a single commit.
3. Try and use go-kit for this problem and see if you are able to structure things according to this framework.
4. Expand your Exchange Rate Service to support cryptocurrency conversions in addition to fiat currencies.
   a. The service should allow clients to query exchange rates and perform conversions between fiat currencies (like USD, EUR) and popular cryptocurrencies (BTC, ETH, USDT)

All the best!