Argyle API Task (Node)

Welcome to the Argyle Technical Task!

Please use this assignment as a time to showcase your best software engineering skills as only the best tasks will be invited to the next stage.

This task has two main parts and completely optional third part. The first part requires writing a simple HTTP server that processes NLP requests, the second - deploying it using Docker. Don't worry if you've never used Docker before. It's not hard, use online resources to learn the basics and `docker-compose` to run it on your local machine. The optional part includes building mini web interface for the server and it’s up to you if you want to take it up.

Once done, send us a zip file with your source code, don't put it on Github or anywhere public!

Part A

Goal: Create an MVP of a system that allows searching for payments of a certain amount using natural language. We aim to build a modern solution enabling users to search payments via natural language input. A payment consists of an id and a value.

Sample user stories:

- As a user, when I enter the text "over fifty four" to the input field, I want see all payments with amount greater than $54
- As a user, when I enter the text "equal two thousand and forty five" to the input field, I want see all payments with amount equal to $2045
- As a user, when I enter the text "under three million one hundred thousand and ninety" to the input field, I want see all payments with amount lower than $3100090

- As a user, when I enter the text "asdasd" to the input field, I want see an error: "Incorrect input"
- As a user, when I enter the text "one one" to the input field, I want see an error: "Incorrect input"

Requirements:

- The max number should be 999 999 999.
- The minimum number is 0.
- We need to handle only `over`, `under` and `equal` phrases and numbers expressed in text
- Choose Node framework of your preference for the implementation

Part B

Write a Dockerfile that builds a self-contained docker image out of the service you wrote in part A. In addition to that, write a docker-compose configuration so that everything can run locally with a single command! It should include any databases, front-end deployments or other components that are necessary to run the application.

BONUS: Part C

Client web app:

- Basic setup: a simple React app.
- No need for detailed styling.
- Focus on functionality, ensuring it integrates well with the rest of the system.
- Should be runnable from `localhost`.

Solution Tips

- We suggest having some basic tests
- If anything is unclear, don't hesitate to ask us.
- This is quite open exercise for you to demonstrate your skills and knowledge, we do not have a scorecard for it, so just go with the flow
- Comments in code is a great way to explain your thinking process
- Assume we're dealing with high data volume, but it's not updated very frequent
- We'd like you to think of it as a production ready solution, so mind the performance, security and observability (it doesn't have to be perfect. Show us your way of thinking)

Good luck!