Sample Problem

The Application

Build an API that allows a user to enter a date time range and get back the price at which they would be charged to park for that time span

API Endpoints

- The application publishes an API with two endpoints that computes a price for an input datetime range.
- The API must respond to requests on port `5000`
- The API must respond using JSON

Rates

The first endpoint is `rates`.

This path takes a `PUT` where rate information can be updated by submitting a modified rates JSON. This submitted JSON overwrites the stored rates.

- A rate is comprised of a price, time range the rate is valid, and days of the week the rate applies to
- See the section `Sample JSON for testing` for the initial set of seeded rates. These rates are expected to be loaded into the database at application startup.

This path when requested with a `GET` returns the rates stored.

Price

The second endpoint is `price`. It allows the user to request the price for a requested time.

- It uses query parameters for requesting the price
- The user specifies input date/times as ISO-8601 with timezones
- The paramters are `start` and `end`.
- An example query is `?start=2015-07-01T07:00:00-05:00&end=2015-07-01T12:00:00-05:00`
- Response contains `price`
  - ```
    {
       "price": 5000
    }
    ```

Response Requirements

- User input can span more than one day, but the API mustn't return a valid price - it must return `"unavailable"`
- User input can span multiple rates, but the API mustn't return a valid price - it must return `"unavailable"`
- Rates will not span multiple days

Application startup

Rates are specified by a JSON file to be automatically loaded on start of the application. The format of this file is the same JSON structure that can be submitted to the `rates` endpoint. The values for these rates can be found in the `Sample JSON for testing` section below.

Other Requirements

- There must be documentation that one can follow to run the application
- API endpoints must be clearly documented
- Tests need to be in place
- We will be running automated requests against your submission. Please take care to follow the requirements laid out.
- Preferred languages are Python, Kotlin, Java, or Go to complete this
  - If you are planning on using a different language, please let the recruiting team know your language of choice

Submitting

- Zip up your submission and submit it via GreenHouse
  - You can achieve this with `git archive --format zip --output /full/path/to/zipfile.zip master` if using git
    ```

- Include any instructions on how to build, run, and test your application

Sample JSON for testing

```
{
    "rates": [
        {
            "days": "mon,tues,thurs",
            "times": "0900-2100",
            "tz": "America/Chicago",
            "price": 1500
        },
        {
            "days": "fri,sat,sun",
            "times": "0900-2100",
            "tz": "America/Chicago",
            "price": 2000
        },
        {
            "days": "wed",
            "times": "0600-1800",
            "tz": "America/Chicago",
            "price": 1750
        },
        {
            "days": "mon,wed,sat",
            "times": "0100-0500",
            "tz": "America/Chicago",
            "price": 1000
        },
        {
            "days": "sun,tues",
            "times": "0100-0700",
            "tz": "America/Chicago",
            "price": 925
        }
    ]
}
```

The timezones specified in the JSON file adhere to the 2017c version of the tz database. Assume that there could be other (non America/Chicago) timezones specified. For more information: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

Assume that rates in this file will never overlap

Sample result

Datetime ranges must be specified in ISO-8601 format. A rate must completely encapsulate a datetime range for it to be available.

- `2015-07-01T07:00:00-05:00` to `2015-07-01T12:00:00-05:00` must yield `{'price': 1750}`
- `2015-07-04T15:00:00+00:00` to `2015-07-04T20:00:00+00:00` must yield `{'price': 2000}`
- `2015-07-04T07:00:00+05:00` to `2015-07-04T20:00:00+05:00` must yield `"unavailable"`

Assessment

Members of our backend team will ensure your code meets the functional criteria and score the submission based on project design, language/framework best practices, code cleanliness and organization, ease of use, packaging, documentation, and testing practices.