Take home interview - Store Monitoring

This will be a take-home interview that tests real-life problem-solving ability, ability to build something from scratch and handle complex algorithmic problems.

Problem statement

Loop monitors several restaurants in the US and needs to monitor if the store is online or not. All restaurants are supposed to be online during their business hours. Due to some unknown reasons, a store might go inactive for a few hours. Restaurant owners want to get a report of the how often this happened in the past.

We want to build backend APIs that will help restaurant owners achieve this goal.

We will provide the following data sources which contain all the data that is required to achieve this purpose.

Data sources

We will have 3 sources of data

1. We poll every store roughly every hour and have data about whether the store was active or not in a CSV. The CSV has 3 columns (`store_id, timestamp_utc, status`) where status is active or inactive. All timestamps are in **UTC**
    a. Data can be found in CSV format here
2. We have the business hours of all the stores - schema of this data is `store_id, dayOfWeek(0=Monday, 6=Sunday), start_time_local, end_time_local`
    a. These times are in the **local time zone**
    b. If data is missing for a store, assume it is open 24*7
    c. Data can be found in CSV format here
3. Timezone for the stores - schema is `store_id, timezone_str`
    a. If data is missing for a store, assume it is America/Chicago
    b. This is used so that data sources 1 and 2 can be compared against each other.
    c. Data can be found in CSV format here

System requirement

- Do not assume that this data is static and precompute the answers as this data will keep getting updated every hour.
- You need to store these CSVs into a relevant database and make API calls to get the data.

Data output requirement

We want to output a report to the user that has the following schema

`store_id, uptime_last_hour(in minutes), uptime_last_day(in hours), update_last_week(in hours), downtime_last_hour(in minutes), downtime_last_day(in hours), downtime_last_week(in hours)`

1. Uptime and downtime should only include observations within business hours.

2. You need to extrapolate uptime and downtime based on the periodic polls we have ingested, to the entire time interval.
    a. eg, business hours for a store are 9 AM to 12 PM on Monday
        i. we only have 2 observations for this store on a particular date (Monday) in our data at 10:14 AM and 11:15 AM
        ii. we need to fill the entire business hours interval with uptime and downtime from these 2 observations based on some sane interpolation logic

Note: The data we have given is a static data set, so you can hard code the current timestamp to be the max timestamp among all the observations in the first CSV.

API requirement

1. You need two APIs
    a. /trigger_report endpoint that will trigger report generation from the data provided (stored in DB)
        i. No input
        ii. Output - report_id (random string)
        iii. report_id will be used for polling the status of report completion
    b. /get_report endpoint that will return the status of the report or the csv
        i. Input - report_id
        ii. Output
            - if report generation is not complete, return “Running” as the output
            - if report generation is complete, return “Complete” along with the CSV file with the schema described above.

Considerations/Evaluation criteria

1. The code should be well structured, handling corner cases, with good type systems.
2. The functionality should be correct for trigger + poll architecture, database reads and CSV output.
3. The logic for computing the hours overlap and uptime/downtime should be well documented and easy to read/understand.
4. The code should be as optimized as people and run within a reasonable amount of time.

You can use any Python framework to build this.

Submission instructions

Send us the following in the same thread

1. Github link to the repo
2. Loom/any other screen-sharing video of a demo of the functionality

We will get back on the next steps.