LoopAI Store Monitoring Task

Rishikeshav Ravichandran

rishi.r1804@gmail.com

Techstack:

- FastAPI
- PostgreSQL
- SQLAlchemy (ORM)

Packages:

- fastapi
- sqlalchemy (ORM)
- datetime (date and time handling)
- pytz (time conversion)
- csv, io (csv file handling)
- pydantic (schema and type validation)

Documentation for uptime downtime calculation

Assumptions made:

- If a store's business hours dont exist in the database, it is presumed to be open 24/7 i.e (00:00:00AM - 23:59:59PM)
- If a store's business hours exist but dont exist for a given day or days, it is presumed to be a closed on that day and wont be included in calculations
- If a store's timezone data doesn't exist in the database, it is presumed to be America/Chicago
- UTC time from Data Source 1 is converted to the store's local time and if within business hours, it is termed as valid in this documentation.
- If an entry for an hour within business hours does not exist, that hour is presumed to be downtime. Eg: 5PM - active, 7PM - active, then 5PM-6PM is counted as uptime and 6PM-7PM is counted as downtime.
- If the hour for the first entry for a day is not the same as the starting hour, it is presumed to be downtime till the first entry for that day.
- The same goes for the last entry; if the hour for the last entry for a day is not the same as the ending hour, it is presumed to also be downtime from the last entry till the ending hour for that day.

- #### For the last hour calculations:
  - The latest valid (within business hours) timestamp along with the 2nd latest valid timestamp are taken. If the 2nd latest timestamp does not exist for that day, the downtime is presumed to be 60 minutes.
  - If no valid data exists for that day, dowmtime is calculated as 60 minutes
  - If the status for both hours is active, then uptime is calculated as 60 minutes.
  - If the status for both hours is inactive, then downtime is calculated as 60 mins.
  - If the status for 2nd latest hour is inactive but latest is inactive, downtime is calculated as 60 mins.
  - As the entries are roughly every hour and not every hour on the dot:
    - If the minute of 2nd latest timestamp is greater than or equal to the minute of the latest hour and the status for 2nd latest timestamp is active but latest is inactive, uptime is calculated as `60 minutes - (minute of 2nd latest timestamp - minute of latest timestamp)` and downtime is calculated as `60 minutes - uptime` Eg: 13:10:30PM - active, 14:09:30PM - inactive, then Uptime = 59 minutes | Downtime = 1 minute.
  - **Edge Case:** Handled at the end to make sure the uptime + downtime for the hour is equal to 60 minutes
  - **Edge Case:** 2nd latest timestamp doesn't exist, as explained then downtime is taken as 60 minutes.
  - **Edge Case:** 2nd latest timestamp is not on the same day as latest timestamp, then it is taken as None (doesn't exist; previous edge case).
  - **Edge Case:** Store might not have business hours for a particular day or days, which we have assumed to be closed and not part of calculations. In that case, it is skipped.

- #### For the last day calculations:
  - Data within business hours of the store for that day is queried in order from earliest to latest and their necessary information is stored in a list.
  - The same logic for the hour calculations is applied to each 2 timestamps that are next to each other in the list (we queried in sorted fashion - earliest to latest) by passing their information into a calculation function; we will call them timestamp_1 and timestamp_2. Eg: The list is [1, 2, 3, 4, 5], first 1 and 2 are passed to the function then 2 and 3 then 3 and 4 and so on.
  - Again, if the hour for the first entry for a day is not the same as the starting hour, it is presumed to be downtime till the first entry for that day.
  - Again, the same goes for the last entry; if the hour for the last entry for a day is not the same as the ending hour, it is presumed to also be downtime

  from the last entry till the ending hour for that day.

  - If the hours of the 2 timestamps are the same:
    - If statuses of both timestamps are active or both are inactive: uptime and downtime stay the same
    - If status of timestamp_1 is inactive and status of timestamp_2 is active: uptime stays the same, `downtime += minute of timestamp_2`
    - If status of timestamp_1 is active and status of timestamp_2 is inactive: `uptime += minute of timestamp_2` , downtime stays the same
  - Else the logic for last hour calculation is iteratively applied to all the timestamps in the list, each passed in two by two.
  - **Edge Case:** Handled after each iteration to make sure the uptime + downtime for the hour is equal to 60 minutes
  - **Edge Case:** Store might not have business hours for a particular day or days, which we have assumed to be closed and not part of calculations. In that case, it is skipped.
  - **Edge Case:**: Two consecutive timestamps might have the same hour in timestamp, handled as explained above

- **For the last week calculation:**
  - The latest date is taken as a reference.
  - Data from the latest date to 1 week ago is queried in ordered fashion, earliest to latest
  - Valid data from the query is added to the list which is of length 7. Each index of the list will have: an array of valid timestamps along with their status, local start time, local_end_time. The 0th index will have data from day 0 (Monday), 1st index will have data from day 1 (Tuesday) and so on.
  - The same logic for the last day calculations are applied to each day of timestamps and summed to get the final uptime and downtime for that week
  - **Edge Case:** Handled after each iteration to make sure the uptime + downtime for the hour is equal to 60 minutes
  - **Edge Case:** Store might not have business hours for a particular day or days, which we have assumed to be closed and not part of calculations. In that case, it is skipped.
  - **Edge Case:**: Two consecutive timestamps might have the same hour in timestamp, handled as explained above