Store Monitoring Assignment

**API Overview**

* Trigger Report

Endpoint: GET /trigger\_report  
Description: Triggers the generation of a report for monitoring store uptime/downtime. The process runs in the background, and a unique report\_id is returned, which can be used to check the status of the report.  
Parameters: None

Response model:

{

"report\_id": "123e4567-e89b-12d3-a456-426614174000"

}

* Get Report Status

Endpoint: GET /get\_report/{report\_id}  
Description: Retrieves the status of a previously triggered report. If the report is completed, it returns "Completed", otherwise "Started" or "Not Found".  
Parameters:

report\_id (path parameter) - The unique identifier of the report to check its status.

Responses:

If the report is still being generated:

{

"report\_id": "123e4567-e89b-12d3-a456-426614174000",

"status": "Started"

}

If the report has been successfully generated:

{

"report\_id": "123e4567-e89b-12d3-a456-426614174000",

"status": "Completed"

}

If the report ID is invalid or not found:

{

"report\_id": "invalid-report-id",

"status": "Not Found"

}

**Additional Notes**

* The report generation runs asynchronously in the background using FastAPI's BackgroundTasks.
* The generated report is stored as a CSV file, named {report\_id}.csv, containing:
* store\_id
* uptime and downtime for the last hour, last day, and last week.
* The get\_report API does not return the CSV file directly; it only provides the status of the report.

How does the API’s work internally?

As the name suggests this API triggers a function named generate report which is a Background task and returns the report\_id .

In the background task there are 60 threads which are working parallelly upon the database and Computations of the up\_time and downtime from the filtered data from the database.

Initially based on the Time Zones from store\_timezones the start\_time\_local and end\_time\_local are converted to utc format and the required data is filtered with using store\_id and then the polling data is filtered by checking the poll times whether present inside the working time limits. Since the data is old the present time and date are hardcoded.The calculated data is then stored inside a .csv file

When the get\_report endpoint is used the status of the reported started/completed is returned.

It is better to convert the received data from polls to utc format and then append to the database so that further calculations are prevented especially during calculations of uptime and downtime and the same approach is used

Futher Improvements

* Handling and tracking thread process states efficientely
* Realtime progress tracking and transfer
* Have Mutex locks for CSV Files to prevent read-write issues
* Have a calculation for number of threads and use the proved number of threads for better and fast results