**Overview**

Your manager needs a statistical summary of the incident data which is gathered every day. Not only will you need to develop the code to create the output but also set up the jobs to execute the process on a daily basis (4:00 EST).

The requirements for the outputs as well as the input file definition are outlined below.

**Requirements**

**Input**:

The input file represents a very simplified stream of incidents which occurred on a given day per asset. Each row represents an incident;

<Asset Name>, <Start Time>, <End Time>, <Severity>

Although the provided input file is small, the solution should be able to handle a source dataset well beyond the amount memory and hard disk space on your machine.

**Definitions**

- Asset Name: The name of the asset for which the incident occurred.

- Start Time: The date and time (accurate to seconds) when the incident began

- End Time: The date and time (accurate to seconds) when the incident ended

- Severity: A number between 1 and 3, indicating how severe the incident was.

**Safe Assumptions:**

- Start and End times are always on the same day, they won't roll over midnight.

- Only severity 1 incidents result in downtime of the application.

Example: here is a row for an incident

System1, 2019-05-01 13:05:20, 2019-05-01 13:11:14, 1

**Problem**:

Aggregate the data per asset:

- Find total incidents per asset for the day

- Find the total uptime per asset, expressed this as a % for the day:

(total\_seconds\_for\_day – down\_seconds\_for\_day = total\_up\_seconds\_for\_day)

- Assign a rating to each asset for the day based on how many incidents have occurred, taking the weight of the severity into consideration;

Severity 1 has a weight of 30

2 and 3 have a weight of 10

Rating = Sum(weight of all incidents for the day)

**Output**:

Your solution should produce a file called 'output.csv'.

file should be a comma separate file with this format:

<Asset Name>, <Total Incidents>, <Total Downtime>, <Rating>

**Constraints**:

1. Your solution should get this output file through an API call.
2. Conform to REST principles
3. Use maven, spring(boot), Java.

**Platform support**

● Tomcat 7 or 8 (if no springboot is used)

● JDK 8 or higher

**Time Limit**

*The solution is expect to be delivered within 48 hours of the assignment being received.*

**Suggestion**

1. Keep it simple. Part of the evaluation consists on being able to follow the requirements, and not over engineer the solution.
2. Be agile - slice your functionality in such a way if you do not complete anything – at least what you deliver has business value in terms of functionality. We are not looking for someone to do everything, focus is on someone to do something that creates impact, adheres to standards, clean code.
3. If you think the problem statement resources delivered as input is insufficient, be innovative. As long as conceptualization doesn’t change, it is ok.
4. A good way would be write a test that invokes your code and does an end to end functionality also
5. Code of Ethics – We promote freedom to make choices. Please adhere to code of ethics, try it on your own. We will have post evaluation to exercise with our engineers.