CO2 Sensor Application

Table of Contents

1. Prerequisite:

This application has been tested using below mention software/hardware.

Software

Software	Version
OS	Mac OS Mojave
Java	8 or above
Mongo DB	v4.0.3
RabbitMQ	3.7.12
Maven	Apache Maven 3.6.0
Application Server	Tomcat 9.0

Hardware:

Memory	8GB
Hard disk	50GB
Processor	2.6 GHz I7

2. Build and Deploy:

1. Download the code from github. It contains one project “co2-sensor” and three modules.
    - Co2-sensor-domain
    - Co2-sensor-server

- Co2-sensor-worker

2. Traverse to the location of pom file of co2-sensor and build the code using the following command maven command. “mvn install” or “mvn package”
3. Copy co2-sensor-server.war and co2-sensor-worker.war and deploy it in any application server. War files will be generated in their respective target folder.

3. Configuration:

Configuration is required for MongoDB and RabbitMQ. Default configuration is:

RabbitMQ:

Host: **localhost**, Port: **5672**

MongoDB:

mongoUrl: **mongodb://localhost:27017/local**

Mongo DB by default is configured to point to database named “local”. To create the database “local” use the following command

“use local”

To Change the configuration, following files needs to be modified.

RabbitMQ:

Co2-sensor-worker: src/main/resources/workerapplication.properties
Co2-sensor-server: src/main/resources/application.properties

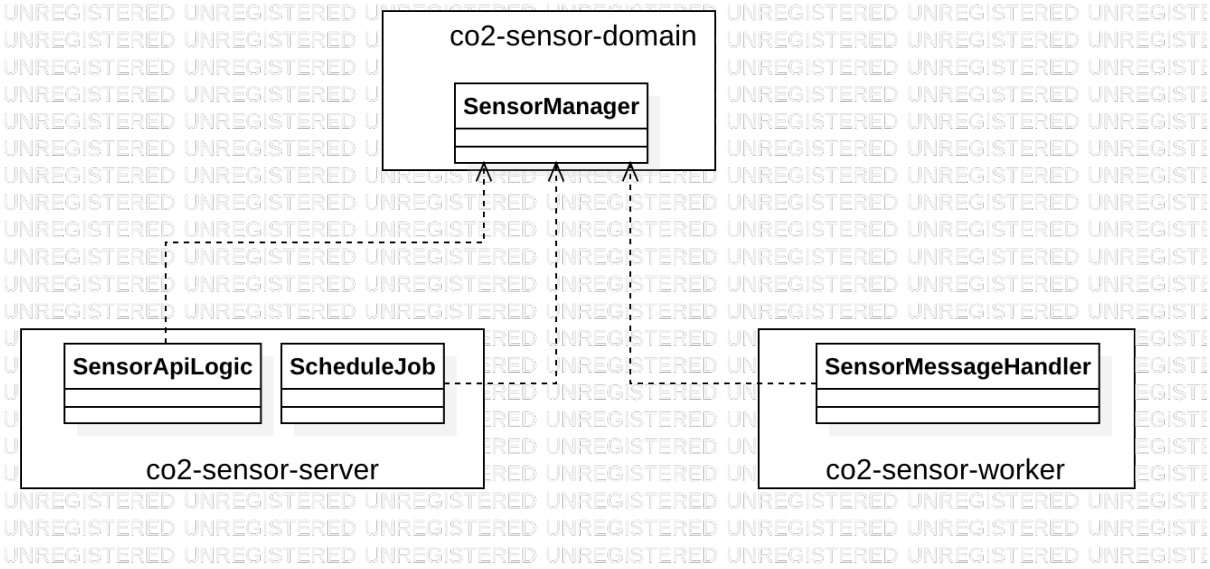
MongoDB:

Co2-sensor-domain: src/main/resources/domainapp.properties

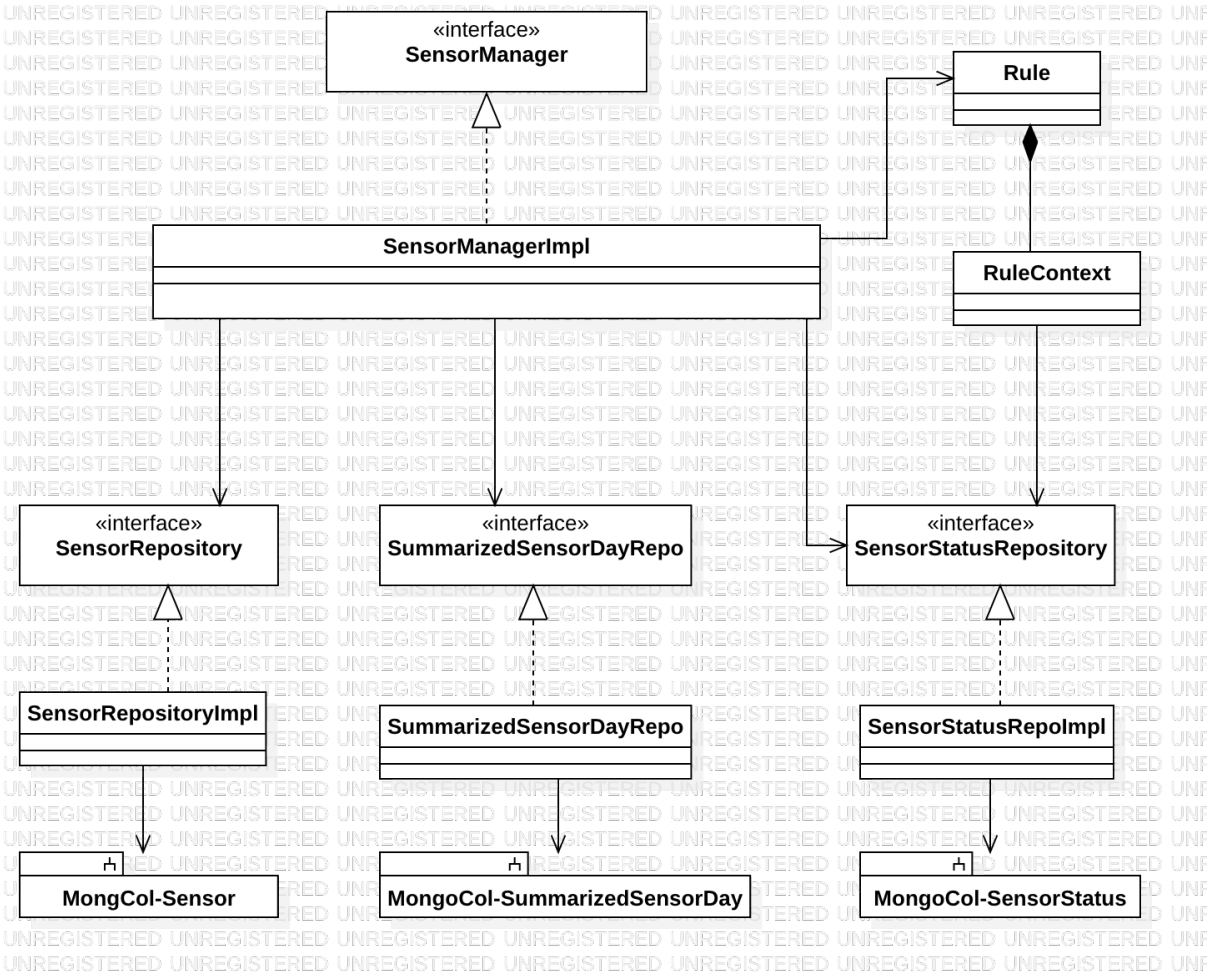
4. Design Diagram

Co2-sensor-domain: It is the library to be used by the other modules. It contains the core logic of the service. It interacts with DB and also contains the data model.
Co2-sensor-server: It contains the controller class for all the api’s. Output of this module is a war file and it needs to be deployed to test the application.
Co2-sensor-worker: It contains the handler class to listen to rabbit Queue. Output of this module is a war file and it needs to be deployed to test the application

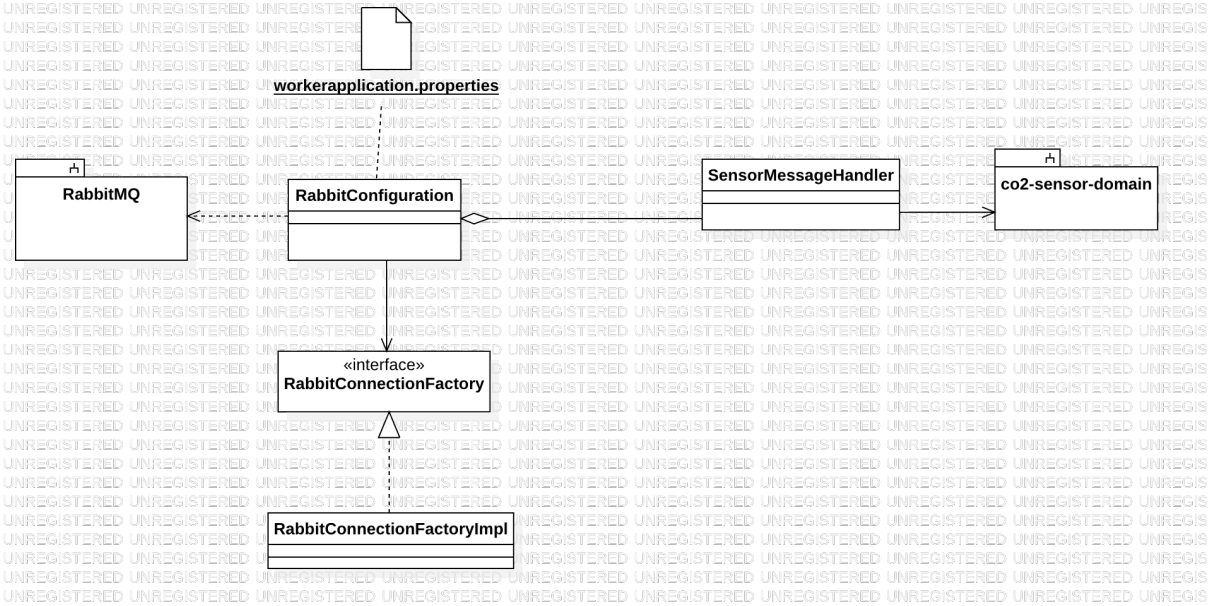
a. Module Dependency Diagram



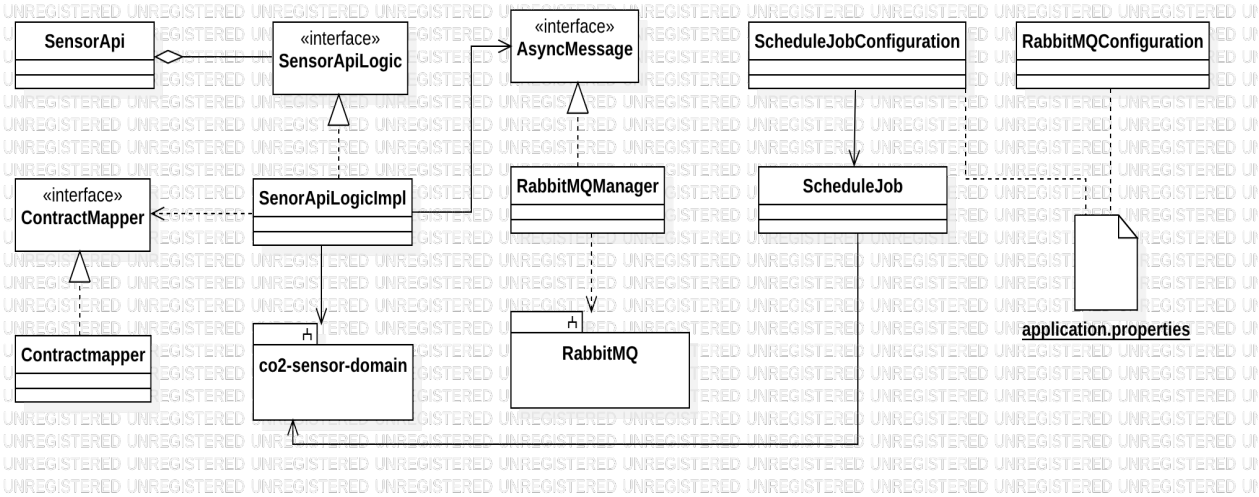
b. co2-sensor-domain Design Diagram



c. Co2-sensor-worker Design Diagram



d. Co2-sensor-server Design Diagram



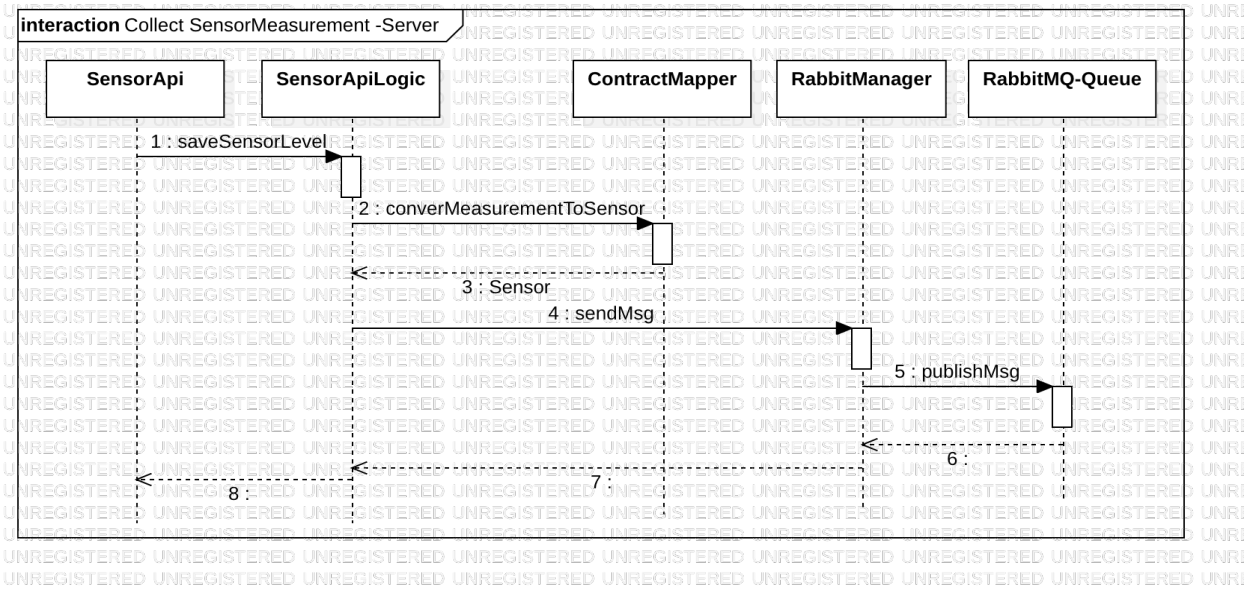
5. Api Design:

a. Collect Sensor Measurement

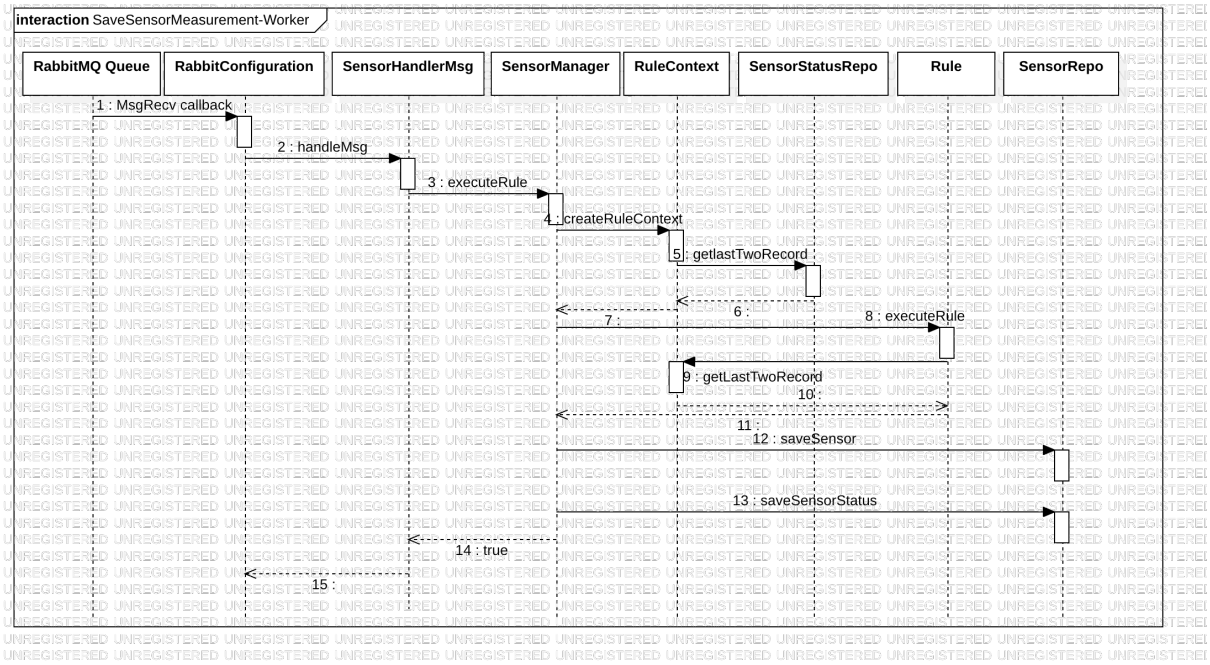
This api has two parts. When the request hit the co2-sensor-server it passes the request to RabbitMQ. Co2-sensor-worker listens to the rabbit queue and when the message arrives it picks the message execute the rule to determine the status of the sensor and stores it in sensor collection and sensorStatus collection.
Following collections are updated.

1. Sensor
2. SensorStatus:

Api sequence flow for co2-sensor-server:

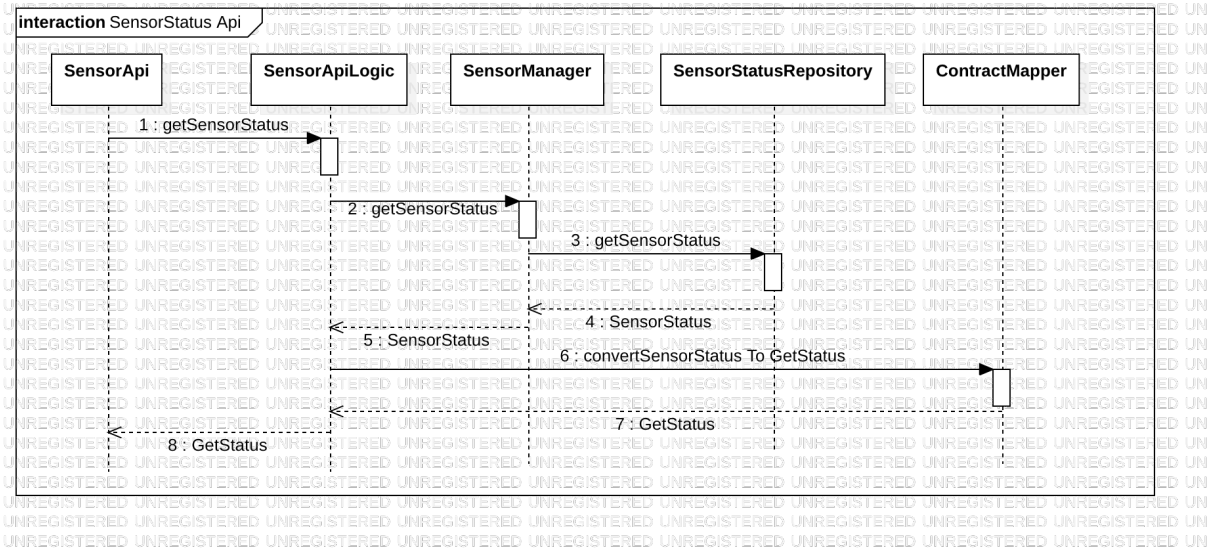


Sequence flow for co2-sensor-Worker:



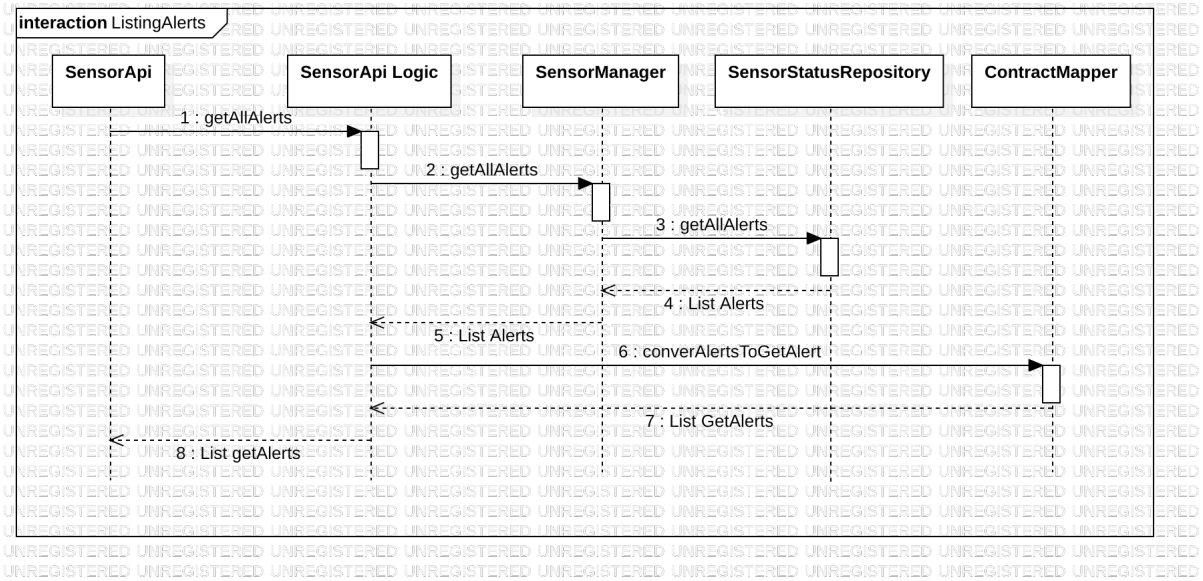
b. Sensor Status Api:

This api retrieves the status of the sensor from sensorStatus collection.



c. Listing Alerts

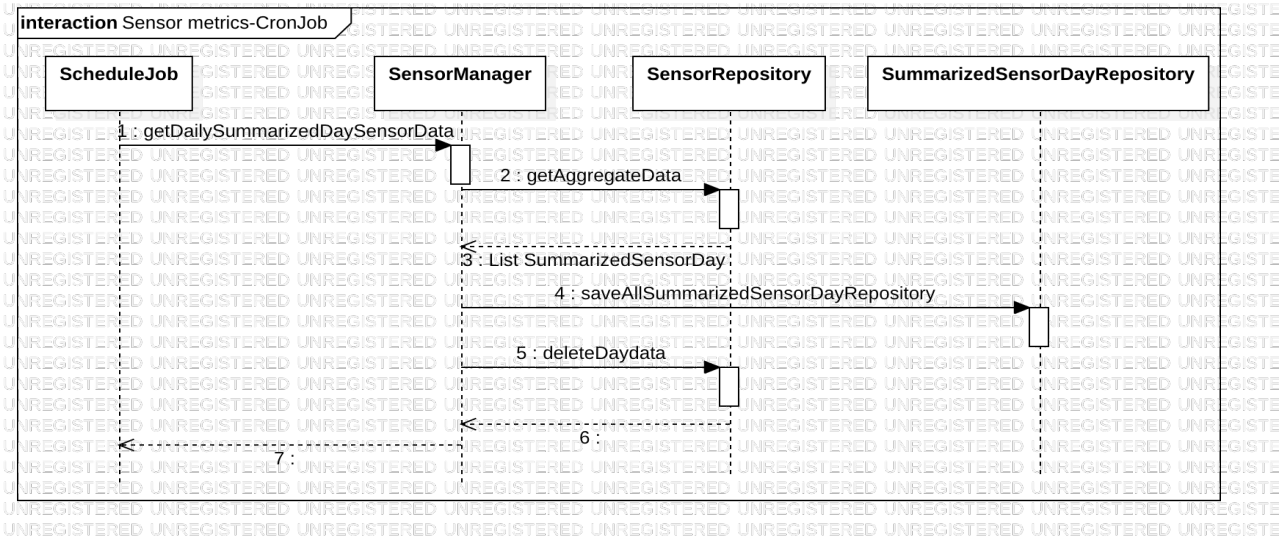
This api retrieves the all the alerts from sensorStatus collection



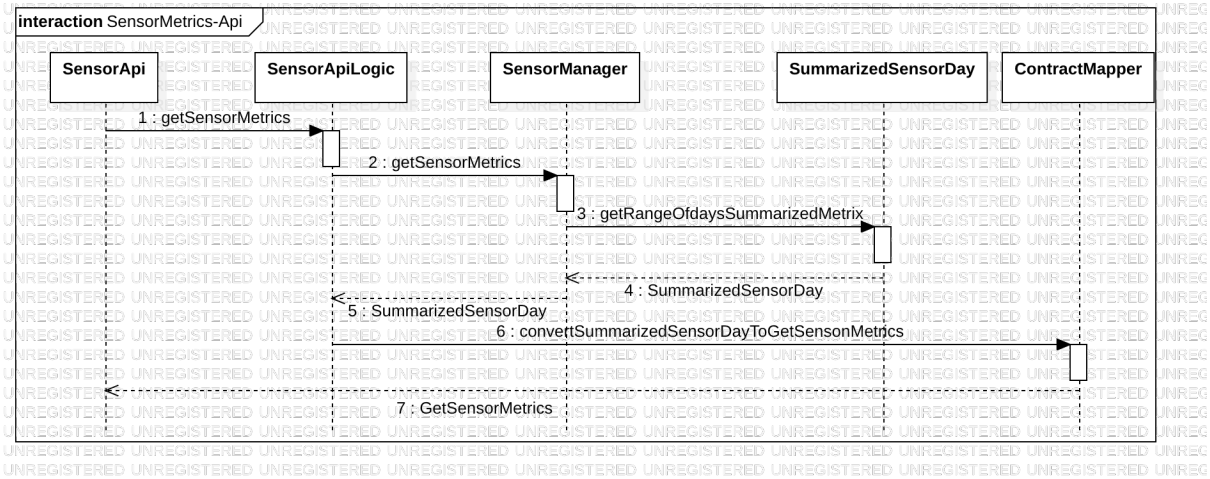
d. Sensor metrics

A Cron Job is configured to run every day at 00.00 am. This job aggregates previous day data from "**sensor"** collection and **bulk inserts** it in "**SummarizedSensorDay** "collection. It also a **bulk deletes** the collected data from "**Sensor**" collection.
Sensor metrics api aggregates one month data from "**SummarizedSensorDay**" collection return it back.

Cron Job Sequence:



Sensor metrics api flow:



6. Database Design:

MongoDb is selected to store data. There has been no evaluation done to select the MongoDB. It is selected as it was readily available. To select an optimized database performance evaluation is required.

MongoDB collections are:

Sensor:


```
{
        "_id":"qUBGECPhcvUM+LSzVqQCgA==",
        "_class":"com.assessment.co2.sensor.domain.model.Sensor",
        "sensorId":"I0yAyR43PpNjOEQkdJuFnA==",
        "level":2500,
        "recordingDateTime":"2019-08-02T19:11:47.000Z",
        "status":"ALERT",
        "isAlert":true
}
```


SensorStatus:


```
{       "_id":"I0yAyR43PpNjOEQkdJuFnA==",
        "_class":"com.assessment.co2.sensor.domain.model.SensorStatus",
        "status":"ALERT",
        "lastValue":1700,
        "lastMinusOneValue":3700,
        "alertStartTime":"2019-08-11T21:24:44.000Z",
        "alertEndTime":"2019-08-11T21:29:57.000Z",
        "alerts":[2500,2600,2700,2000,3700,3700]
}
```


SummarizedSensorDay


```
{
        "_id":"gk+IwKI7vy4aebmMB1xYhQ==",
        "_class":"com.assessment.co2.sensor.domain.model.SummarizedSensorDay",
        "sensorId":"I0yAyR43PpNjOEQkdJuFnA==",
        "date":"2019-08-01T18:30:00.000Z",
        "max":2400,
```

```
        "average":2160
}
```