Problem Description for Network and Platform Developer Position

- The internet of things is a system of interrelated computing devices, sensors that are provided with unique identifiers and the ability to transfer data over a network .One of the key challenges in building out the Internet of Things platform is ensuring that all those "things" are in fact able to communicate over the Internet. The number of IoT devices is massive—25 billion devices by 2020, according to one estimate—and any web based platform that supports that communication has to scale to handle the traffic. So the one of the important challenges that we have to tackle is to build a platform that would be able to scale as per the ongoing exponential growth of IoT devices communication.

- For this problem statement, you will build a Java based web-microservices API [Backend] for Car Tracker Sensor. To simulate the sensor readings you will be provided with a mocker service. This service will send will mock the sensor request and will send HTTP PUT and HTTP GET request to your API.

    1. Develop following REST (*Any two*) endpoints for ingestion from: [http://mocker.ennate.academy/]
        i. Load vehicle details in bulk via a **PUT** /vehicles endpoint.
        ii. If the vehicle with same VIN is already present, update the record in db.
        iii. Ingest readings from these vehicles via a **POST** /readings.

    2. Create (*Any two*) alerts with given priority when following rules are triggered
        i. Rule: engineRpm > readlineRpm, Priority: HIGH
        ii. Rule: fuelVolume < 10% of maxFuelVolume, Priority: MEDIUM
        iii. Rule: tire pressure of any tire < 32psi || > 36psi , Priority: LOW

    3. Develop REST end points for:
        i. Fetch details of all the vehicles like VIN, make, model, year etc.
        ii. Fetch HIGH alerts within last 2 hours for all the vehicles.
        iii. Ability to list a vehicle's all historical alerts.

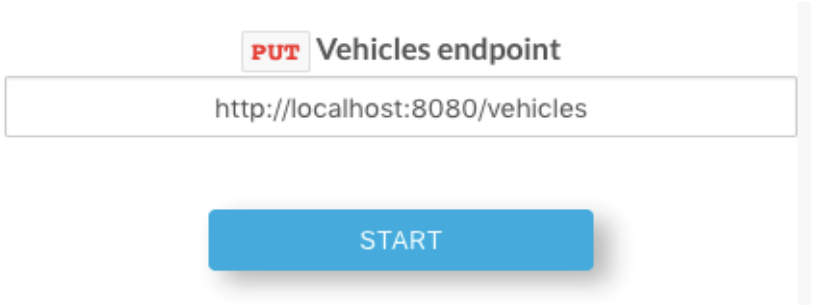
Input Format:

PUT Request Format

i. PUTs mock vehicle details (array of vehicle objects) every **1min**.

ii. Please put your local HTTP PUT end point URL in following input box and press start :

- Note : Make sure you are using [CORS support in Spring Framework]



iii. JSON :

Sample Vehicle List:

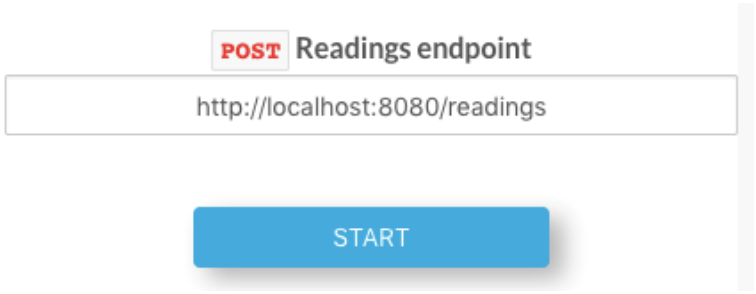

```
[
 {
    "vin": "1HGCR2F3XFA027534",
    "make": "HONDA",
    "model": "ACCORD",
    "year": 2015,
    "redlineRpm": 5500,
    "maxFuelVolume": 15,
    "lastServiceDate": "2017-05-25T17:31:25.268Z"
 },
 {
    "vin": "WP1AB29P63LA60179",
    "make": "PORSCHE",
    "model": "CAYENNE",
    "year": 2015,
    "redlineRpm": 8000,
    "maxFuelVolume": 18,
    "lastServiceDate": "2017-03-25T17:31:25.268Z"
 }
]
```

POST Request Format

i. POSTs mock vehicle readings every **3sec**.

ii. Please put your local HTTP POST end point URL in following input box and press start :

- Note : Make sure you are using [CORS support in Spring Framework]



iii. JSON :

Sample Reading:


```
{
   "vin": "1HGCR2F3XFA027534",
   "latitude": 41.803194,
   "longitude": -88.144406,
   "timestamp": "2017-05-25T17:31:25.268Z",
   "fuelVolume": 1.5,
   "speed": 85,
   "engineHp": 240,
   "checkEngineLightOn": false,
   "engineCoolantLow": true,
   "cruiseControlOn": true,
   "engineRpm": 6300,
   "tires": {
      "frontLeft": 34,
      "frontRight": 36,
      "rearLeft": 29,
      "rearRight": 34
   }
}
```

Submission Method :

- We use GitHub for version control. You will create a repository on GitHub with following nomenclature :
    - *<firstName_lastName>_IoTLab_project*
- Please share this repository's web URL with us.

Evaluation Criteria :

We generally specifically measure the quality of a project based on the following criteria:

- ***Readability*** : how easy is the source code for another experience developer to "read"- are functions well named, is there good commenting in the code base, is the code well formatted across different layers?

- ***Adaptability*** : how easy is it to make minor, common changes to the site? For example, how easy is it to add new fields to the database, add additional content or choices, or change the service layer logic?

- ***Maintainability*** : How easy is it to maintain the site on production, without outages, errors, or issues requiring a human to get involved?

- ***Performance*** : How fast is the application? Does it hang or time out? Do the requests load in a timely fashion? Does it scale when loads get high or when running CPU intensive functions?

- ***Accuracy*** : Is the microservice capturing and displaying sensor data accurately? Are there existing bugs in common features?