**Version Management and Insights:***Input explanation:*  
  
1. Attached file (raw\_data.txt) contains data from April 1 to April 15.  
2. Data in file has one row corresponding to each entry.  
3. An example entry looks like this:  
  
**{"dd":"<did>","vn":"<vn>","ep":1617236100,"dt":{"tm":27.2,"hm":82.7,"pp":0.50,"wd":94,"ws":0.0,"sm":439.6,"st":284.3,"sc":636,"lt":0.0,"lw":0.0,"bl":7.9,"pv":0.0}}**

This format is called as json (Javascript Objection Notation).   
a) “dd” represents -> device imei  
b) “vn” represents -> version of raw data

*Note: same device can send data with different version but it is not mandatory.*

c) “ep” represents time in epoch -> you can learn more about epoch here  
<https://en.wikipedia.org/wiki/Epoch>  
ep tells us the time for which data belongs like an “ep” in example belongs to April1- 12:15 GMT, that means that corresponds to this time value.

d) “dt” represents data being sent from that device

e) All the keys inside dt field (tm,hm etc are sensor values)

We have to focus only on “tm” and “hm” values which are temperature(degree Celsius) and humidity (percentage)respectively.

*Output Explanation:*1. Following REST Apis to be exposed by code :

(for understanding REST APIs-<https://en.wikipedia.org/wiki/Representational_state_transfer>)

1. Get all versions

Endpoint: /get-version

Method: GET

Response Code: 200

Response Body:

{

“status”: “versions extracted successfully”,

“payload”: [“92.1.1.1,…..”92.1.1.2”]

}

1. GET all device ids(dd) – unique device id’s to be return

Endpoint: /get-dd

Method: GET

Response Code: 200

Response Body:

{

“status”: “Device ids extracted successfully”,

“payload”: [“8626450982”,…..”86264509876”]

}

1. GET versions on a specific date

Endpoint: /get-version-on-date/?date=12-04-2021

Method: GET

Response Code: 200

Response Body:

{

“status”: “versions extracted successfully for required date”,

“payload”: [“92.1.1.1,…..”92.1.1.2”]

}

1. GET number of days where exactly (n-1) versions reported. Where “n” is the total number of versions available in data

Endpoint: /get-days-1-less-n-data

Method: GET

Response Code: 200

{

“status”: “required number of days extracted”,

“payload”: [“days”:4]

}

1. GET highest temperature and humidity value for a particular day

Endpoint: /get-highest values /?date=12-04-2021

Method: GET

Response Code: 200

Response Body:

{

“status”: “ highest tm and hm values for given date extracted”,

“payload”: [“tm”:34,”hm”:98”]

}

1. GET number of data points of a particular version between start date and end date

Endpoint: /get-data-points /?start\_date=12-04-2021&&end\_date=15-04-2021&&vn=”92.1.1.1”

Method: GET

Response Code: 200

Response Body:

{

“status”: “number of data points for given version extracted”,

“payload”: [“datapoints”:50]

}

**Note: Proper error handling to be there.**

1. Convert a data point to a different version (only from list of available versions in API get –versions)

Endpoint: /convert-version

Method: POST

Response Code: 201

Request Body:

{ “dataepoch”: 1617236100,”dd”:<dd>,new\_vn:”92.1.1.3”}

Response Body:

{“status”: “Datapoint successfully updated with new version”}

1. Exchange tm and hm values for a particular data-point

Endpoint: /exchange-tm-hm

Method: POST

Response Code: 201

Request Body:

{ “dataepoch”: 1617236100,”dd”:<dd>}

Response Body:

{“status”: “tm and hm in data point are successfully exchanged”}

**Note: Both g) and h) to be tested by checking APIs (d) and (e)**

1. Delete a particular data point

Endpoint: /delete-data-point

Method: DELETE

Response Code: 202

Request Body:

{ “dataepoch”: 1617236100,”dd”:<dd>}

Response Body:

{“status”: “Datapoint deleted succesfully”}

*Solution Management*:  
  
1. Data can be read directly from file. You can also store it in a particular database of your choice (not required necessarily). Choices of Database can include – MongoDB, Cassandra DB, ElasticSearch etc.

2. If reading from file, filepath to be configurable in code. If reading from database, database connection string to be configurable in code i.e. should be placed in a separate configuration file and not to be made part of code.

3. Preferred languages include, Python,Golang. In case, not having proficiency in any of these, Java or c++ can be used.  
  
4. Detailed explanation of steps performed in solution to be provided.