Dashboard for equipment data

The requirement is to write a web-based tool to show a dashboard with summary about all the equipment in an organization. Equipment data is accessible via an api.

Dashboard Requirements

The dashboard should show the following pieces of information:

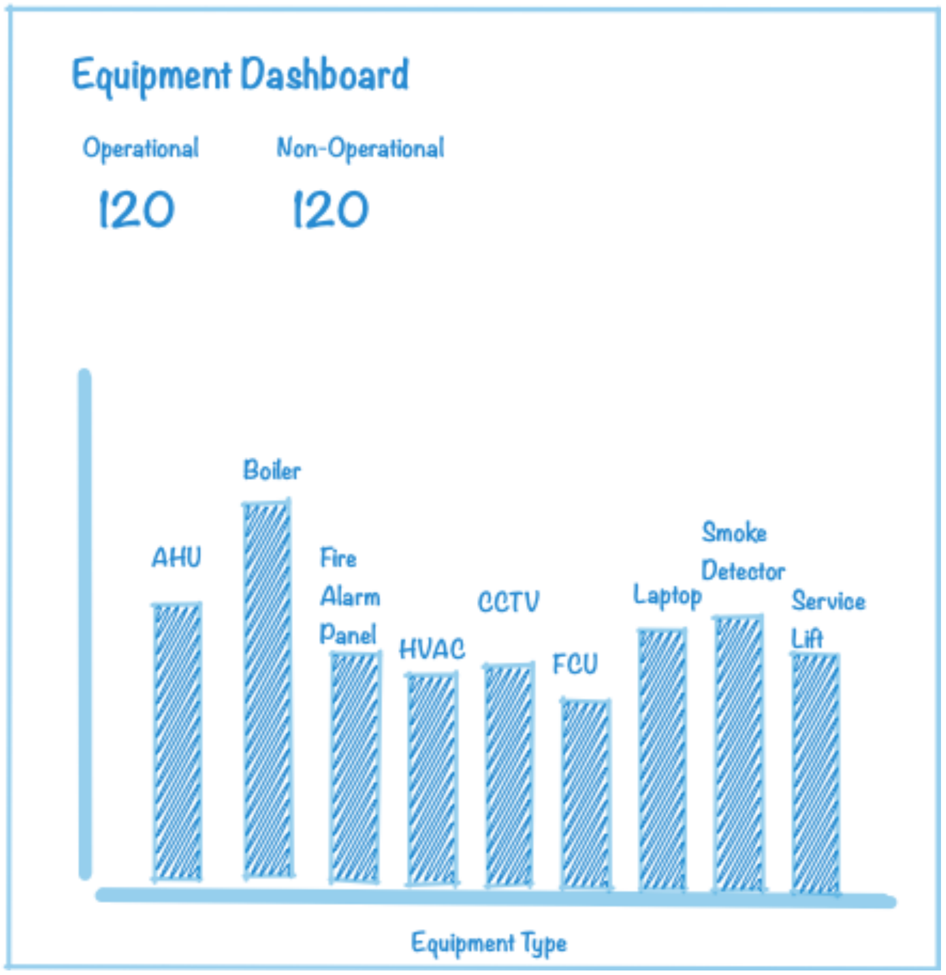
1. Number of operational equipment
2. Number of non-operational equipment
3. A column chart where each bar represents an equipment type and the height of the bar represents how many equipment of that type are present.

The dashboard should be viewable in a browser and should be served by some web server.

For example, we should be able to go to http://localhost:8080/ and see the dashboard loaded.

You can decide whether you want to query the api each time the page loads or if you want to cache the data locally.

Sample dashboard (doesn't have to look like this - just a guideline):



Retrieving Data

The following api can be used to retrieve all equipment data:

```
http://ivivaanywhere.ivivacloud.com/api/Asset/Asset/All?
apikey=SC:demo:64a9aa122143a5db&max=10&last=0
```

This is a simple HTTP GET request.

The query string parameters are explained here:

Parameter	Description	Example
apikey	An API key used to access it. Use `SC:demo:64a9aa122143a5db` as the api key	`SC:demo:64a9aa122143a5db`
max	The maximum number of items to return each time the api is called. This should be less than 100	`10`
last	This specifies the last rowid that was received from the previous call. Used for paging. The first time you call the service, this should be `0`. On subsequent calls, you should pass the `__rowid__` value from the last item that was returned	`0`

The returned data will have `Content-Type:application/json`.

The expected response code is `200`

The body will have the following structure:

```
[
  {
    "AssetID": "SYN.SIN.AHU001",
    "AssetCategoryKey": "4",
    "Description": "Air Handling Unit for main hanger bay",
    "OperationalStatus": "Operational",
    "InstalledLocationKey": "141",
    "Make": "Mayekawa",
    "Model": "6HK",
    "SerialNumber": "4527-4726-987",
    "BarCode": "87475692",
    "InstalledDate": "20130401:000000",
    "CommissionedDate": "20130408:000000",
    "Ownership": "Owned",
    "AssetKey": "1",
    "ObjectKey": "1",
    "__key__": "1",
    "ObjectID": "SYN.SIN.AHU001",
    "InstalledLocationName": "Singapore.Changi Airport.T2 Departure.Operations
Office",
    "AssetCategoryID": "AHU",
    "__rowid__": "1"
  },
  {
    "AssetID": "SYN.SIN.AHU002",
    "AssetCategoryKey": "4",
    "Description": "Air Handling Unit",
    "OperationalStatus": "Operational",
      "InstalledLocationKey": "5",
      "Make": "Mayekawa",
      "Model": "6HK",
      "SerialNumber": "4527-4726-986",
      "BarCode": "87475693",
      "InstalledDate": "20130401:000000",
      "CommissionedDate": "20130408:000000",
      "Ownership": "Owned",
      "AssetKey": "2",
      "ObjectKey": "2",
      "__key__": "2",
      "ObjectID": "SYN.SIN.AHU002",
      "InstalledLocationName": "Singapore.Syneco Head Office.Floor 01",
      "AssetCategoryID": "AHU",
      "__rowid__": "2"
    }
  ]
```

The result is an array of equipment data. For each equipment, multiple properties are present (some have been removed from this example).

The main properties you need to use:

`AssetCategoryID` - this is the equipment type

`AssetID` - A unique id for the equipment

`__rowid__` - This needs to be passed as the `last` parameter to the api to make it retrieve the next set starting right after this asset.

`OperationalStatus` - This determines if the equipment is currently operational or not. Possible values are `Operational` and `Non-Operational`

So for example, the first time you call the api - you may call the following api :

`http://ivivaanywhere.ivivacloud.com/api/Asset/Asset/All?apikey=SC:ivivademo:8d756202d6159375&max=2&last=0`

This will return 2 items. The last item's `__rowid__` is 2 so then in the next api call, you will call:

`http://ivivaanywhere.ivivacloud.com/api/Asset/Asset/All?apikey=SC:ivivademo:8d756202d6159375&max=2&last=2`

You can keep calling the api and updating the `last` parameter each time to page through all the results.

It is recommended that you limit max to `100` or less.

There are about 250 items in total.

Technology Requirements

No specific platform is required to be used. However the result must be a tool that will run a web server which will serve up the dashboard html page.

Note that CORS is *not* enabled on the server. This is by design. This means any browser based code will not be able to directly talk to the API.

Performance Requirements

No specific performance requirements here - however reasonably measures are expected to be taken to make the system performant (for example, don't query the api separately for each equipment one by one)

Visual Requirements

No specific requirements except the ability to produce the required stats and chart. The look and feel is not the main criteria - however bonus points for creating something nice looking and user friendly :)

Final Delivery

The final delivery should be code with instructions on how to run it.

After it is run - we should be able to browse a web page which presents this dashboard