Team 1 Design

PREPARED FOR

IAS PROJECT - GROUP 4

IIIT-Hyderabad

PREPARED BY

TEAM 1

Akshay M 2020201023

Raman Shukla 2020201098

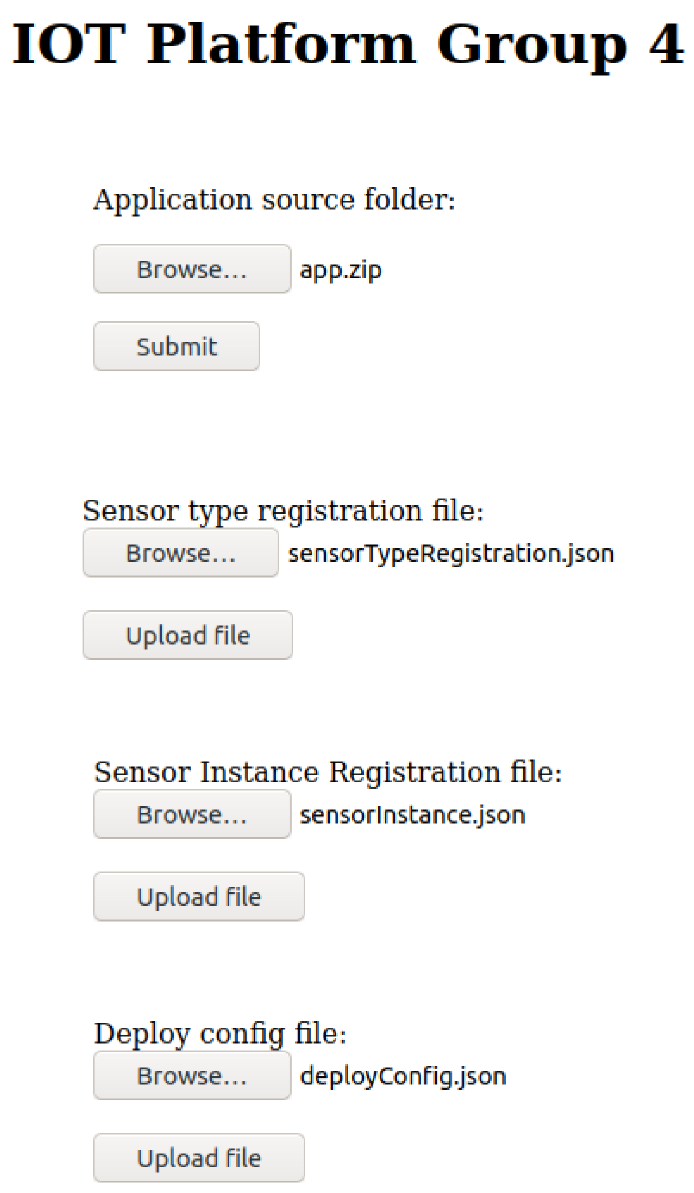
Varun Nambigari 2020201079

Our sub team worked on the following modules

1) Provide UI and Dashboard
2) Platform Manager
3) Validation
4) Scheduler
5) Action Service
6) Sample Application

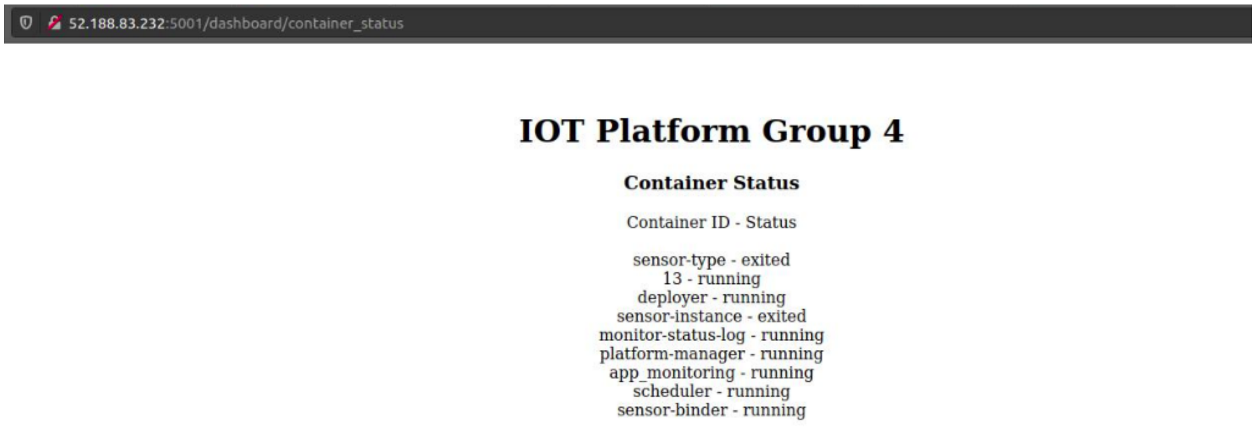
Platform Manager

UI is provided for the following operations(Platform Manager):



Dashboard

- The platform manager provides APIs for the dashboard.
- The dashboard shows the the following:
    - The logs from the running applications.
    - The status of various containers.
    - The logs of various containers.



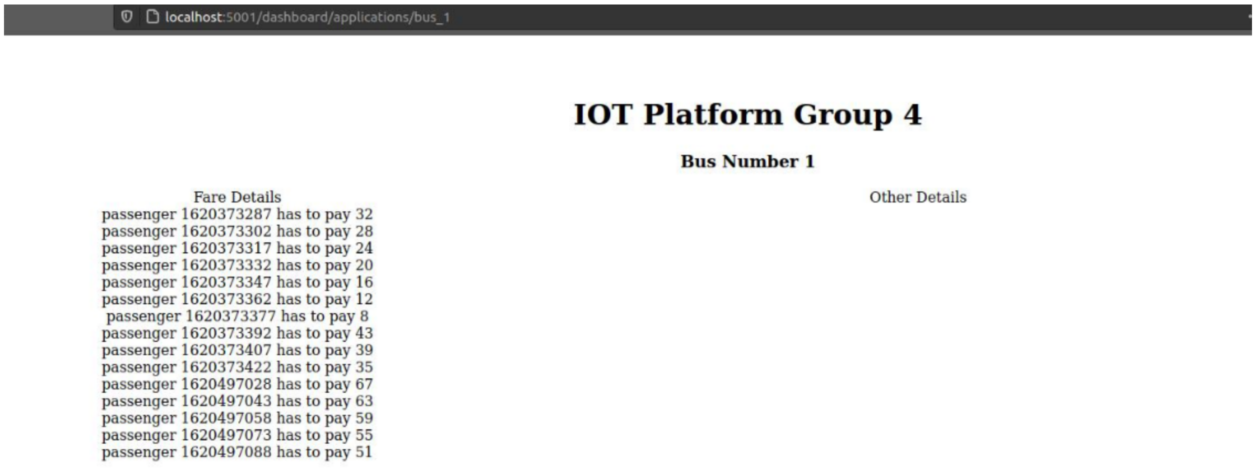
52.188.83.232:5001/dashboard/container_status

IOT Platform Group 4

Container Status

Container ID - Status

```
sensor-type - exited
13 - running
deployer - running
sensor-instance - exited
monitor-status-log - running
platform-manager - running
app_monitoring - running
scheduler - running
sensor-binder - running
```



localhost:5001/dashboard/applications/bus_1

IOT Platform Group 4

Bus Number 1

Fare Details

```
passenger 1620373287 has to pay 32
passenger 1620373302 has to pay 28
passenger 1620373317 has to pay 24
passenger 1620373332 has to pay 20
passenger 1620373347 has to pay 16
passenger 1620373362 has to pay 12
passenger 1620373377 has to pay 8
passenger 1620373392 has to pay 43
passenger 1620373407 has to pay 39
passenger 1620373422 has to pay 35
passenger 1620497028 has to pay 67
passenger 1620497043 has to pay 63
passenger 1620497058 has to pay 59
passenger 1620497073 has to pay 55
passenger 1620497088 has to pay 51
```

Other Details

Communication module:
UI - HTTP (client server)
Internal Services (Kafka)

Kafka Topic Names:

Producer	Topic Name	Consumer	Purpose
platform_manager	pm_to_sensor_type_reg	sensor_registration	Send sensorTypeRegistration.json contents
platform_manager	pm_to_sensor_ins_reg	sensor_registration	Send sensorInstance.json contents
platform_manager	pm_to_sensor_binder	sensor_binder	Send deployConfig.json contents
sensor_binder	sensor_binder_to_scheduler	scheduler	Send deployConfig.json contents
scheduler	scheduler_to_deployer	deployer	Send deployConfig.json contents

Validator Service

Has the following API:

validate_appConfig(path to app.zip)
validate_appzip()
validate_sensor_type()
validate_sensor_instance()
validate_deployConfig()

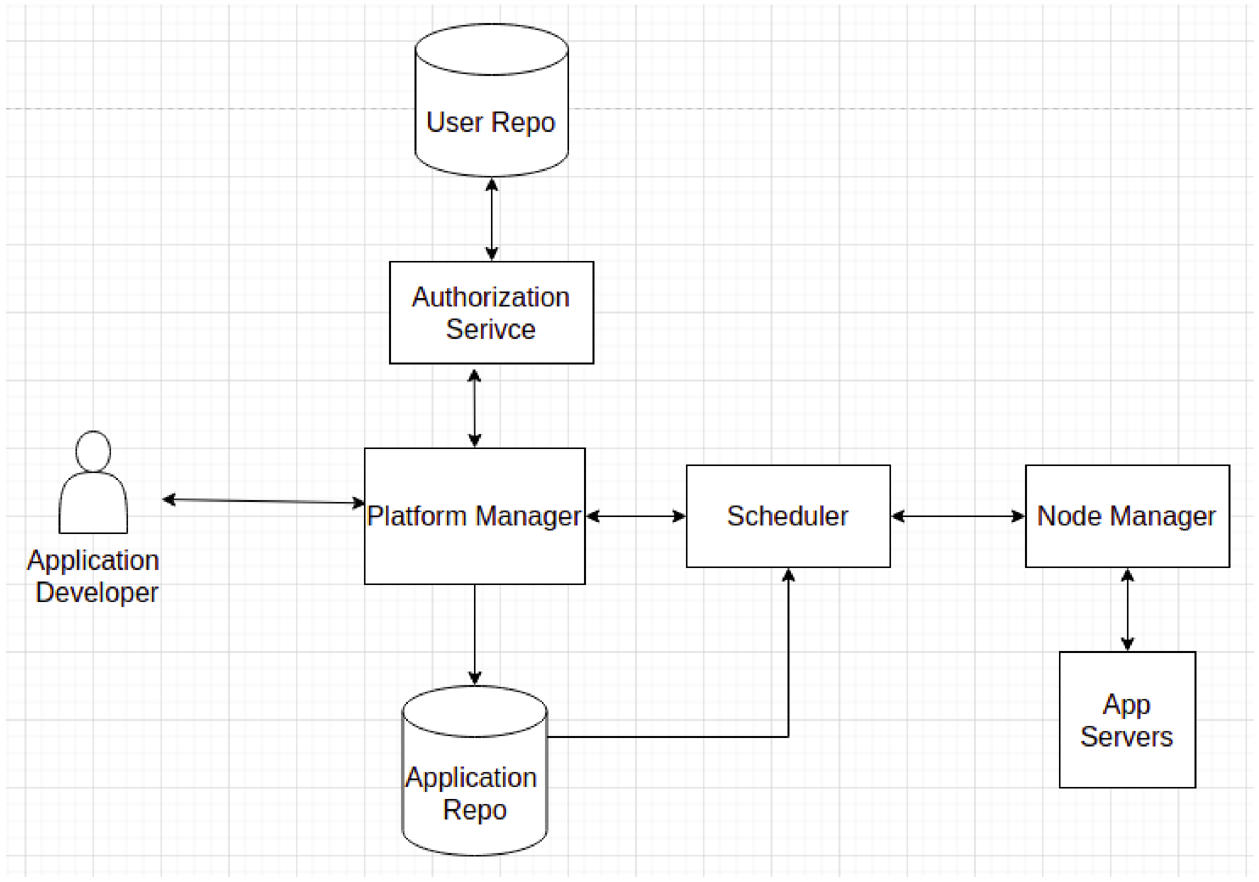
This service checks the folder structure of app.zip file.
Also checks the json format of the files given.

Application Code:
The platform provides a platform_lib.py file to the application developer to import in their code.
The platform_lib.py file provides the method to get the Kafka topic for the respective sensor from which the developer can consume the data.

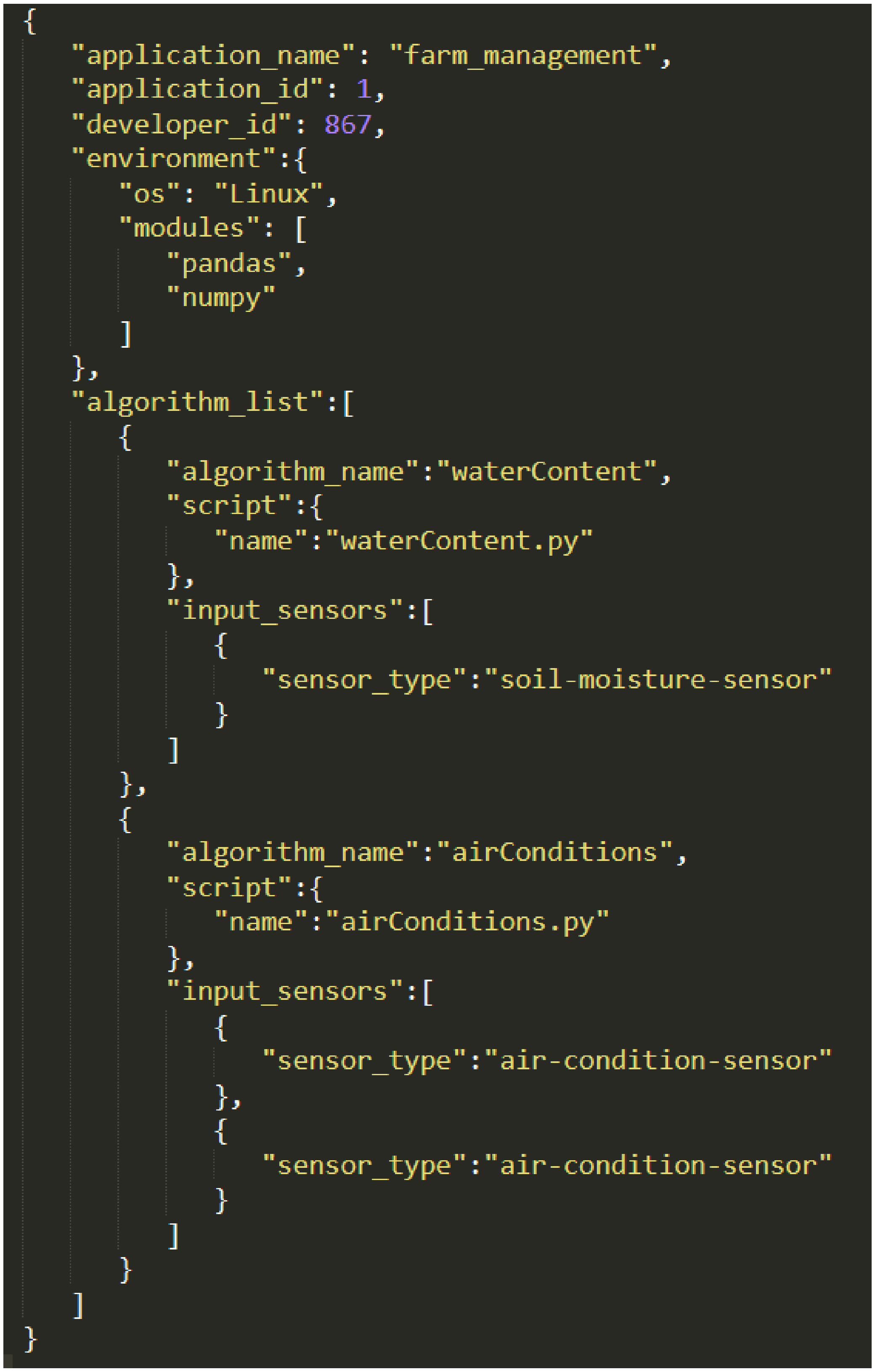
GetKafkaTopic:
The platform manager provides the api to the developer to fetch the kafka topic name given their application instance id and sensor index.

API in the Platform manager.
GetKafkaTopic(application instance id, sensor index)

This API is also used in the later part of the code where we need the kafka topic for controlling the sensor.

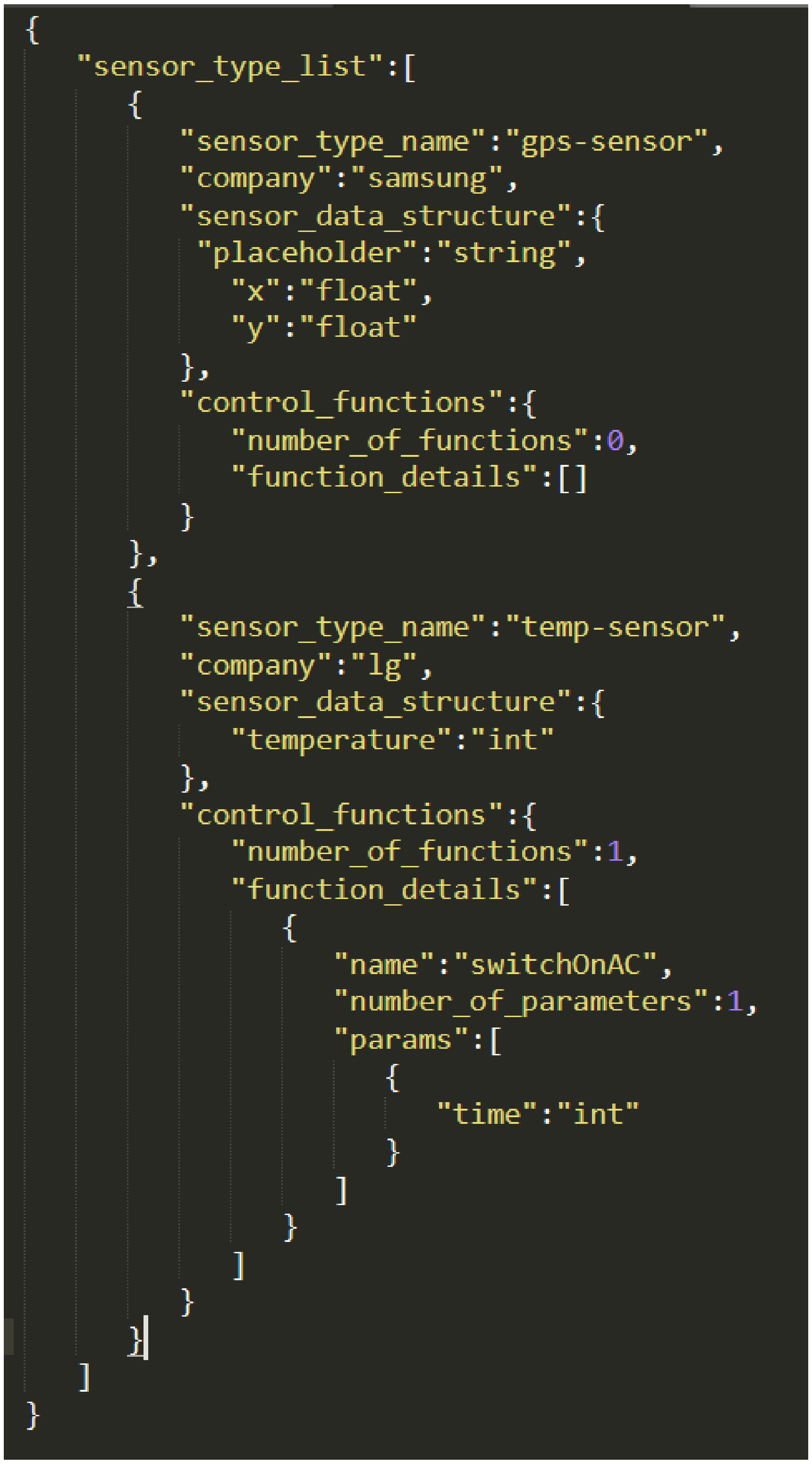


appConfig.json



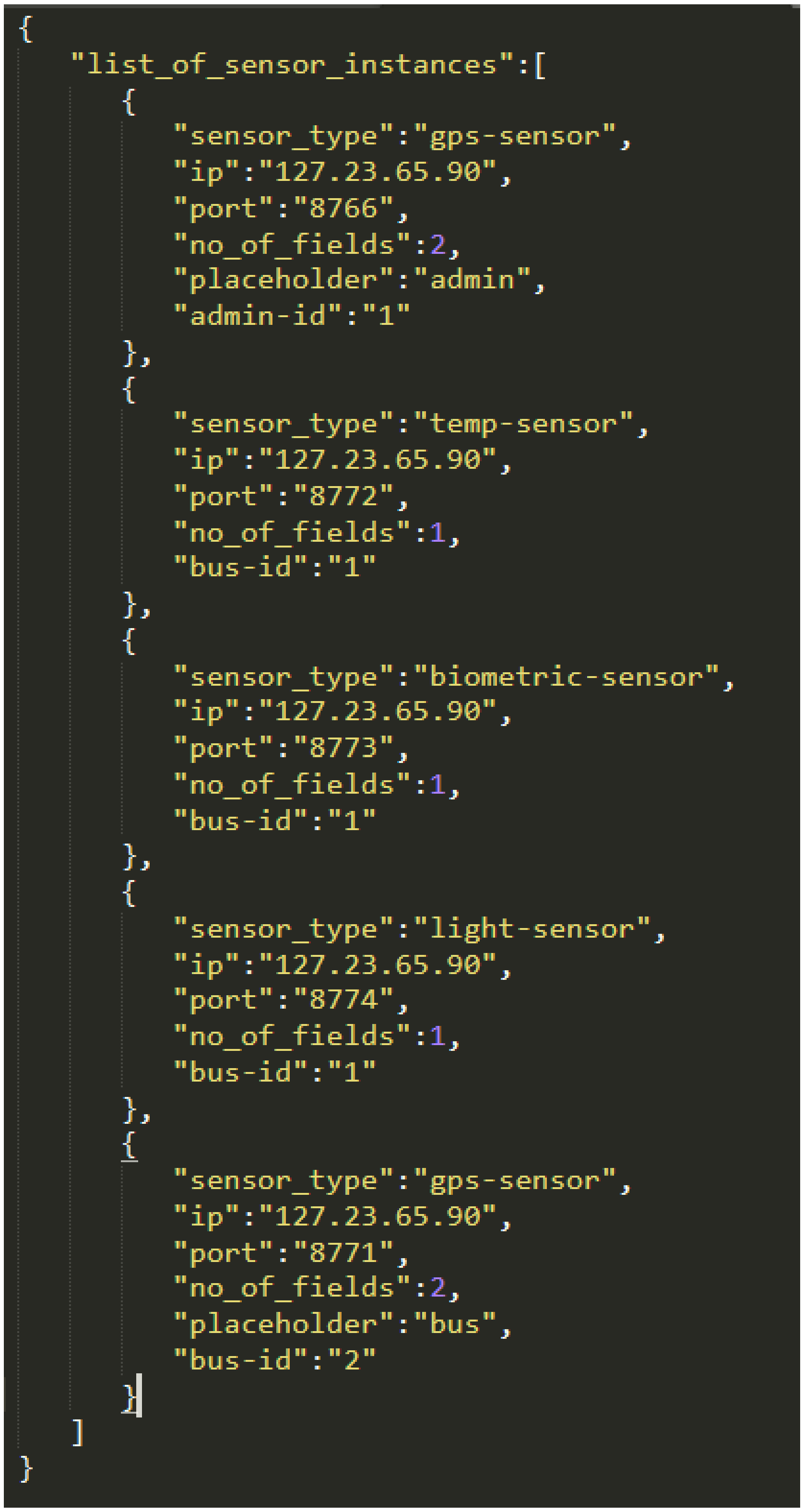
```json
{
    "application_name": "farm_management",
    "application_id": 1,
    "developer_id": 867,
    "environment":{
        "os": "Linux",
        "modules": [
            "pandas",
            "numpy"
        ]
    },
    "algorithm_list":[
        {
            "algorithm_name":"waterContent",
            "script":{
                "name":"waterContent.py"
            },
            "input_sensors":[
                {
                    "sensor_type":"soil-moisture-sensor"
                }
            ]
        },
        {
            "algorithm_name":"airConditions",
            "script":{
                "name":"airConditions.py"
            },
            "input_sensors":[
                {
                    "sensor_type":"air-condition-sensor"
                },
                {
                    "sensor_type":"air-condition-sensor"
                }
            ]
        }
    ]
}
```

sensorTypeRegistration.json



```
{
    "sensor_type_list":[
        {
            "sensor_type_name":"gps-sensor",
            "company":"samsung",
            "sensor_data_structure":{
             "placeholder":"string",
                "x":"float",
                "y":"float"
            },
            "control_functions":{
                "number_of_functions":0,
                "function_details":[]
            }
        },
        {
            "sensor_type_name":"temp-sensor",
            "company":"lg",
            "sensor_data_structure":{
                "temperature":"int"
            },
            "control_functions":{
                "number_of_functions":1,
                "function_details":[
                    {
                        "name":"switchOnAC",
                        "number_of_parameters":1,
                        "params":[
                            {
                                "time":"int"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

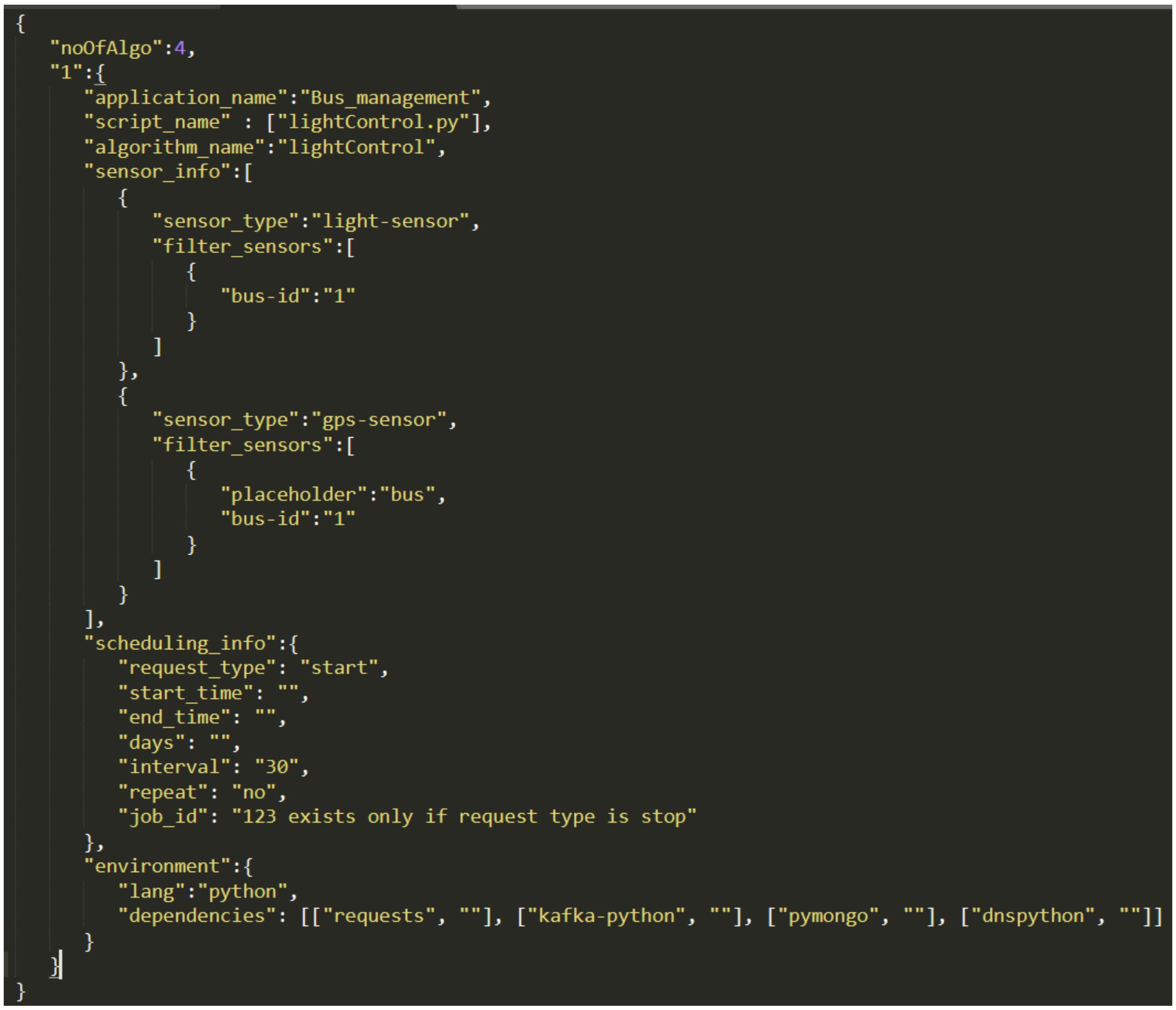
sensorInstanceRegistration.json



```json
{
    "list_of_sensor_instances":[
        {
            "sensor_type":"gps-sensor",
            "ip":"127.23.65.90",
            "port":"8766",
            "no_of_fields":2,
            "placeholder":"admin",
            "admin-id":"1"
        },
        {
            "sensor_type":"temp-sensor",
            "ip":"127.23.65.90",
            "port":"8772",
            "no_of_fields":1,
            "bus-id":"1"
        },
        {
            "sensor_type":"biometric-sensor",
            "ip":"127.23.65.90",
            "port":"8773",
            "no_of_fields":1,
            "bus-id":"1"
        },
        {
            "sensor_type":"light-sensor",
            "ip":"127.23.65.90",
            "port":"8774",
            "no_of_fields":1,
            "bus-id":"1"
        },
        {
            "sensor_type":"gps-sensor",
            "ip":"127.23.65.90",
            "port":"8771",
            "no_of_fields":2,
            "placeholder":"bus",
            "bus-id":"2"
        }
    ]
}
```

Scheduler:

The Scheduler receives **deployConfig.json** from the Sensor-Binding module which has the following format:



```
{
   "noOfAlgo":4,
   "1":{
      "application_name":"Bus_management",
      "script_name" : ["lightControl.py"],
      "algorithm_name":"lightControl",
      "sensor_info":[
         {
            "sensor_type":"light-sensor",
            "filter_sensors":[
               {
                  "bus-id":"1"
               }
            ]
         },
         {
            "sensor_type":"gps-sensor",
            "filter_sensors":[
               {
                  "placeholder":"bus",
                  "bus-id":"1"
               }
            ]
         }
      ],
      "scheduling_info":{
         "request_type": "start",
         "start_time": "",
         "end_time": "",
         "days": "",
         "interval": "30",
         "repeat": "no",
         "job_id": "123 exists only if request type is stop"
      },
      "environment":{
         "lang":"python",
         "dependencies": [["requests", ""], ["kafka-python", ""], ["pymongo", ""], ["dnspython", ""]]
      }
   }
}
```

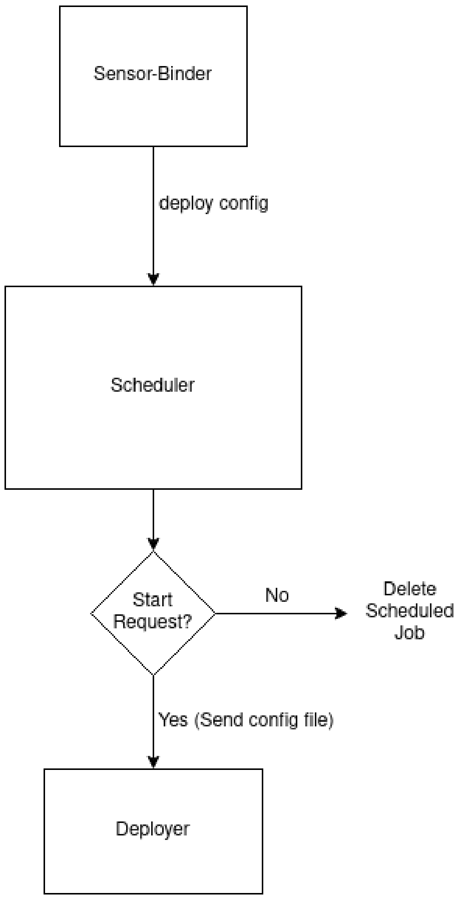
Instance ID Is added by the Sensor-Binding module, which is unique to an algorithm-sensor binding, and can be later used to stop the corresponding algorithm.

The different scheduling options provided by Scheduler are:

- request_type: Start or Stop. If it's a stop request, none of the other options need to be filled in. It is a mandatory field.
- start_time: Start time of the algorithm-sensor instance. 24- hour format should be used. It is a mandatory field for 'start' request.
- end_time: End time of the algorithm-sensor instance. 24- hour format should be used. It is an optional field.

- day: The day of the week on which algo. The values can be ‘monday’, ‘tuesday’, ‘wednesday’, ‘thursday’, ‘friday’, ‘saturday’, or ‘sunday’. It is an optional field.
- interval: This value should be set if the algorithm-sensor instance must be deployed every x hours. If it is set, then ‘repeat’ should not be ‘yes’. It is an optional field.
- repeat: Can be yes/no. If ‘yes’ and day is given, deploy it every week on that day. If day is not present, deploy it everyday. It is an optional field with default as ‘no’.

At the time of deployment, the scheduler sends the deployConfig file to the deployer. If the end time is present in the config file then the config file, with request_type set as ‘stop’, will be sent to the deployer at end time.



Sensor Controller:

API provided by this service

- **setSensorData(instance id, sensor index)**
  Returns the kafka topic

This service is used to set a value to a particular sensor or to control the sensor.
This is implemented using one more kafka topic. That is each sensor have two types of topics one for data inputs and other if it exists for the control functions.

The platform_libfile.py handles the api call to the platform manager whose job is to fetch the kafka topics related to our query.
We get the kafka topics then we will make a producer and add a signal according to the requirement.

Sample Application:

The application can be used by IIIT-H to facilitate transportation (using buses) needed for COVID vaccination drive.

The various sensors present in the buses are:

- **GPS**: Sends placeholder-id & co-ordinates. Each bus is installed with one such sensor, one sensor is at IIIT-H campus and each police barricade has one such sensor.
- **Biometric**: Sends placeholder-id & person-id(it will be unique on place-id level).
- **Temperature**: Sends current temperature of the place
- **Light Sensors**: Sends Lux Level of the place

The various controllers present in the buses are:

- Lights
- Air Condition
- Buzzer

The different use cases of the application are:

- Whenever someone boards a bus, he/she will do biometric check-in. Fare is calculated based on distance multiplied by a fixed rate & is displayed on the dashboard for that bus application instance so that guard can collect that amount.
- When the bus is not empty, if the temperature is more than a threshold air condition is switched on, or if lighting is lower than a lux level then lights are switched on.
- If more than two (>=3) buses come in a circle of given radius, except one, buzzer command to all buses except onet. Run this service after every 2 minutes.

- Whenever a bus comes closer to a barricade by a threshold distance, an algorithm is triggered which sends an email to the administration mentioning the unique identifier of the bus.

Our Implementation:

- We are making 4 files for the use cases 1,2 and 4.
- For every instance of the buses, these 4 files will also be instantiated.
- The 5th file is a common file which will only be instantiated once for all the buses.
- Thus the 4 files are the local application files and the 5th file is the global application file.
- The local files are Fare Calculation, Light Control,Temperature Control and the Barricade alert.
- The global file is the radius buzzer when more than 2 buses are close enough to be in a boundary of a threshold radius.