Team 1 Hackathon.

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

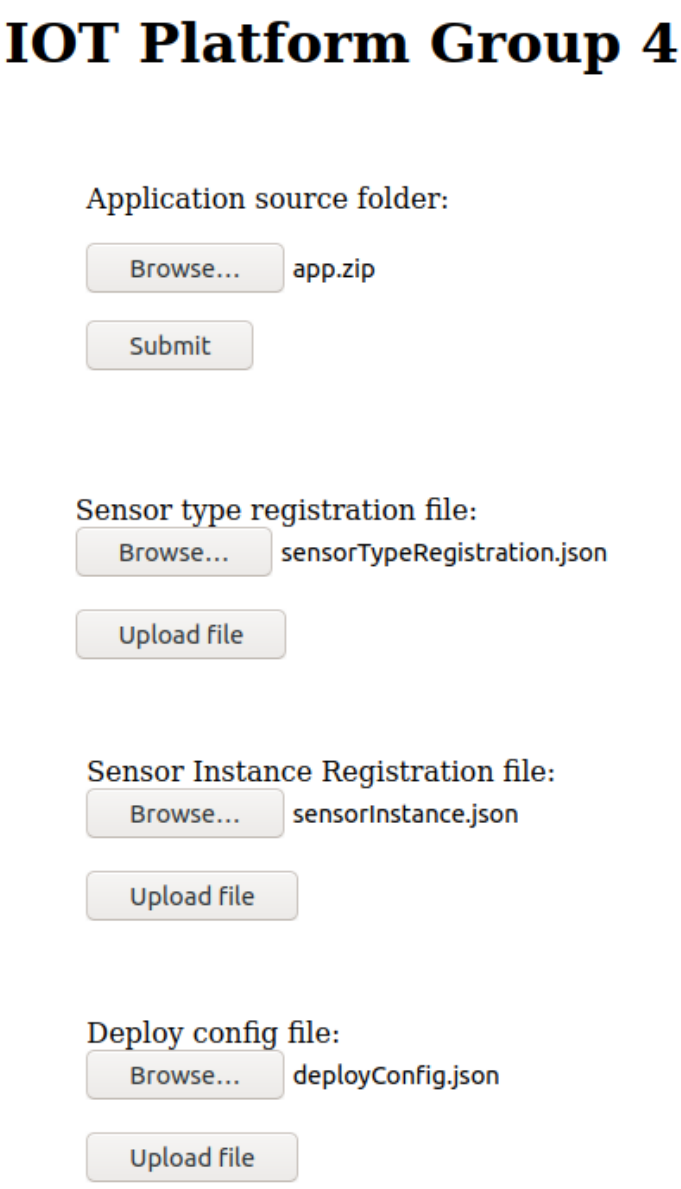
1) Provide UI
2) Platform Manager.
3) Validating config .
4) Scheduler
5) Action Service

Communication module :

UI - HTTP (client server)
Internal Services (Kafka)

Platform Manager

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



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()

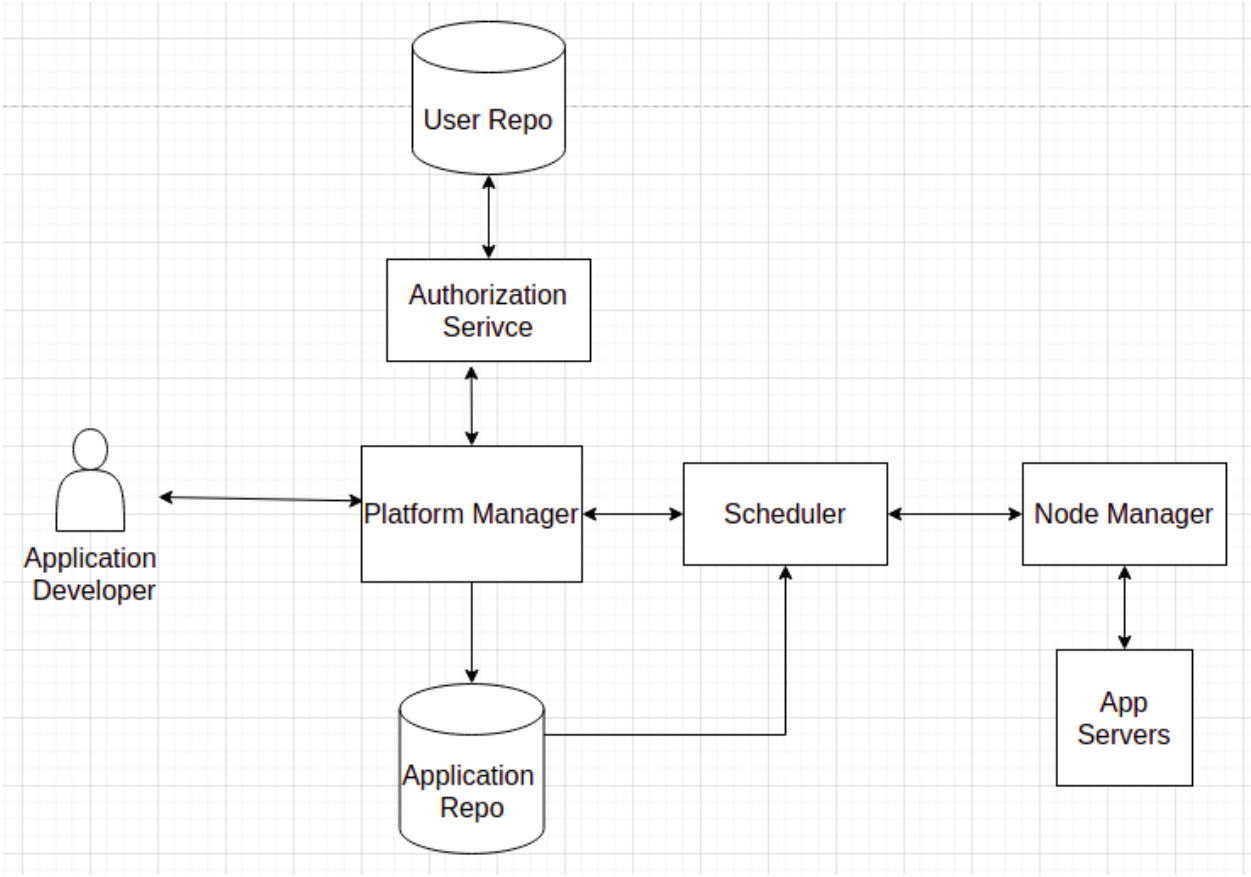
This service checks the folder structure of app.zip file.
Also checks the json format of the files given.

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.


User Repo
Authorization Serivce
Application Developer
Platform Manager
Scheduler
Node Manager
App Servers
Application Repo

appConfig.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":"soil-moisture-sensor",
         "company":"samsung",
         "sensor_data_structure":{
            "moisture":"float"
         },
         "control_functions":{
            "number_of_functions":1,
            "function_details":[
               {
                  "name":"switchOnSprinkler",
                  "number_of_parameters":1,
                  "params":[
                     {
                        "time":"int"
                     }
                  ]
               }
            ]
         }
      },
      {
         "sensor_type_name":"air-condition-sensor",
         "company":"lg",
         "sensor_data_structure":{
            "temperature":"int"
         },
         "control_functions":{
            "number_of_functions":0,
            "function_details":[

            ]
         }
      }
   ]
}
```

sensorInstanceRegistration.json


```
{
   "list_of_sensor_instances":[
      {
         "sensor_type":"soil-moisture-sensor",
         "ip":"x.x.x.x",
         "port":"x",
         "no_of_fields":1,
         "location":"Indore"
      },
      {
         "sensor_type":"soil-moisture-sensor",
         "ip":"x.x.x.x",
         "port":"x",
         "no_of_fields":1,
         "location":"hyderabad"
      },
      {
         "sensor_type":"air-condition-sensor",
         "ip":"x.x.x.x",
         "port":"x",
         "no_of_fields":1,
         "location":"value1"
      },
      {
         "sensor_type":"air-condition-sensor",
         "ip":"x.x.x.x",
         "port":"x",
         "no_of_fields":1,
         "location":"value1"
      },
      {
         "sensor_type":"air-condition-sensor",
         "ip":"x.x.x.x",
         "port":"x",
         "no_of_fields":1,
         "location":"value1"
      }
   ]
}
```

Scheduler:

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

```
{
  "application_id": 12,
  "instance_id": 1,
  "script_name": "hello.py",
  "algorithm_name": "waterContent",
  "sensor_info": [
    {
      "sensor_type": "soil-moisture-sensor",
      "filter_sensors": [
        {
          "location": "hyderabad"
        }
      ]
    }
  ],
  "scheduling_info": {
    "request_type": "start",
    "start_time": "",
    "end_time": "16:25",
    "day": "",
    "interval": "3",
    "repeat": ""
  },
  "environment": {
    "lang": "python",
    "dependencies": [
      ["flask", ""],
      ["pandas", "1.2.3"]
    ]
  }
}
```

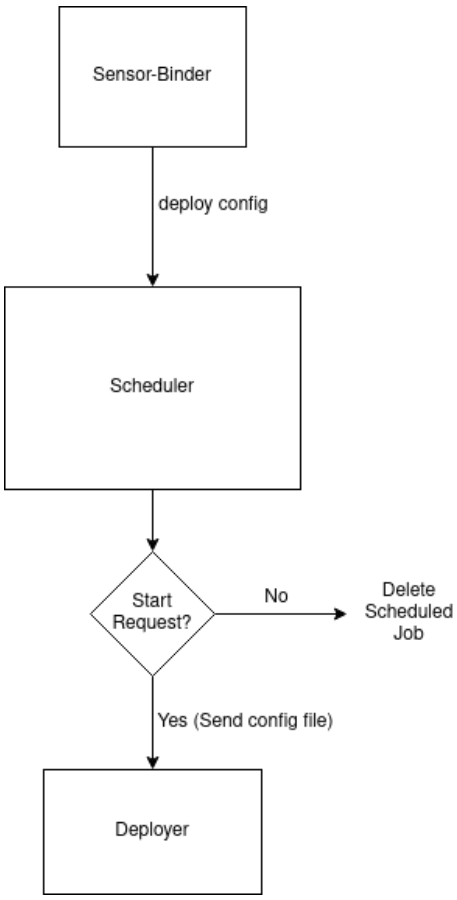
Instance ID Is added by the Sensor-Binding module, which is unique to an algorithm-sensor binding, and can be later used to stop the corresponding algorithm.

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

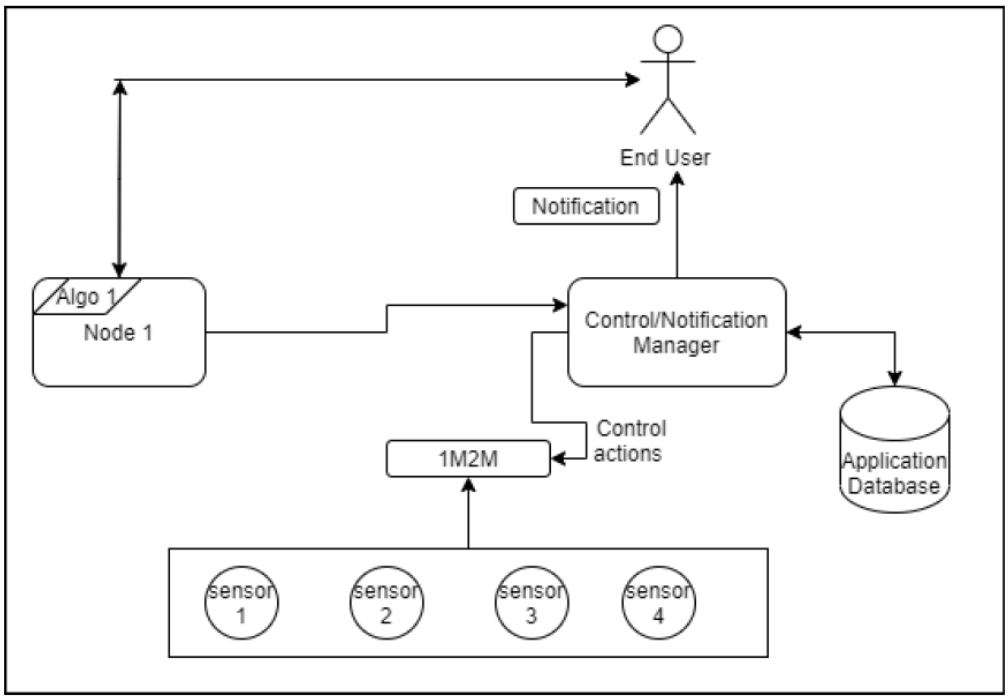
- **setSensorData(instance id, sensor index)**
  Returns the kafka topic

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 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.

Future Work

Notification Manager:



Api provided by this service

- getNotification(target=email,params=[email])

Notifying the output to the user of the application i.e it will trigger notification to the user about status of any event that is requested by the user.
b) If any action needs to be performed based on the output(notification), control manager identifies the corresponding algo from the application repo and schedules the corresponding action (might need to interact with sensors).
c) Maintain an application repository to store action algorithms.
d) Also processes action request from server and notify the user about it.