IAS
Group 2

App model:

Assumption for app developer:

- Will write the application which contains the app code and will look for bindings.json file for sensor_ids. Initially it will be empty and will be populated with sensor_ids at runtime by the deployer
- Write a config.json file which will contain required types of sensors and number of sensors.

Config file structure :

```
{
  "app_name": "weather monitor",
  "app_description": "A cool app to know when it is going to be cool",
  "cmd": [
        "npm install",
        "npm run dev"
  ],
  "sensors": {
        "Temperature": 1,
        "Humidity": 0,
        "Luminosity": 1,
        "Power": 0,
        "Presence": 1,
        "Lamp": 2,
        "Buzzer": 0
  }
}
```

Schedule information structure:

```
{
  "start_date": "2023-04-19T18:30:00.000Z",
  "end_date": "2023-04-27T18:30:00.000Z",
  "timings": {
        "monday": {
        "start_hour": "",
        "end_hour": ""
        },
        "tuesday": {
        "start_hour": "",
        "end_hour": ""
```

```
        },
        "wednesday": {
        "start_hour": "",
        "end_hour": ""
        },
        "thursday": {
        "start_hour": "01:30:00",
        "end_hour": "03:30:00"
        },
        "friday": {
        "start_hour": "",
        "end_hour": ""
        },
        "saturday": {
        "start_hour": "",
        "end_hour": ""
        },
        "sunday": {
        "start_hour": "",
        "end_hour": ""
        }
  }
}
```

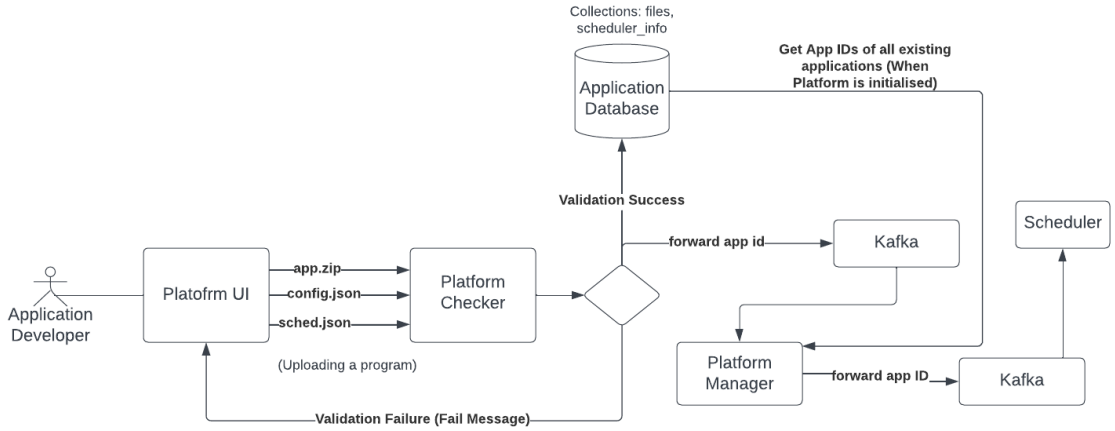

Sensor binding structure:


```
{
  "Buzzer": [],
  "Humidity": [],
  "Lamp": [
        "957e19bb-218b-4ce5-bd88-176bb30955bc",
        "7651d6f2-750c-403d-9916-15accb9f030e"
  ],
  "Luminosity": [
        "6467a9d0-aedf-484a-8362-dc089cd2c722"
  ],
  "Power": [],
  "Presence": [
        "3d5485bc-32e6-45f5-b146-9c8d2254467d"
  ],
  "Temperature": [
        "2794e7d8-fdba-4335-aa5b-f50bf135eaf4"
  ]
}
```

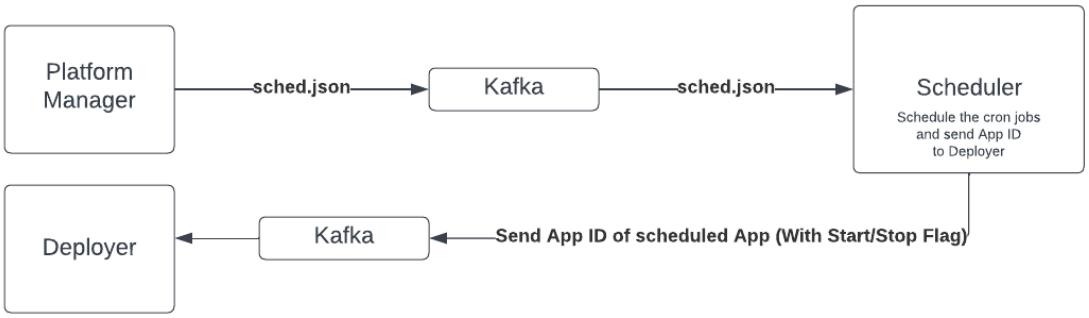
The app gets the sensor_id during the deployment phase as mentioned earlier. The sensor_ids will be fetched from the db and put by the deployer in the bindings.json file.

App journey:

1. The first step before uploading any application, is to visit the platform website and get educated about the config files to be uploaded and the available API functionalities for a list of given sensors.
2. The uploading process starts by the user registering as an application developer on the platform, after which they are provided by the app dev UI.
3. For a new application, the developer is supposed to upload the application code(in zip format), the config.json file and fill the scheduling information (which gets translated to a json format for further use). The config file is then cross-checked by the validator module, and if passed, the information is saved in the application DB. The platform manager is notified about this and the sched.json of the newly added application is sent to the scheduler.



4. On receiving the json file for scheduling, the json is parsed and the tasks are scheduled as cron jobs. The scheduler uses the python *schedule* module for this, which runs on a different thread. One thread is dedicated to the kafka for any incoming requests. While scheduling, the job is scheduled by sending the application ID to the deployer along with a start/stop flag.



5. On receiving the app ID from the scheduler via kafka, the deployer requests the platform manager for the code corresponding to the ID. An instance of the application is created and stored in the Deployer DB and the state of the instance is set to *init.* The flow is then transferred to the load balancer, who is responsible for monitoring the worker VMs, where the applications are supposed to run. The DB is also updated with a pending state. The applications are containerized and run on the worker VM and on completion, the status is updated to *completed*.