

**PROJECT DEVELOPMENT PHASE**  
**PROJECT DEVELOPMENT – DELIVERY OF SPRINT 1**

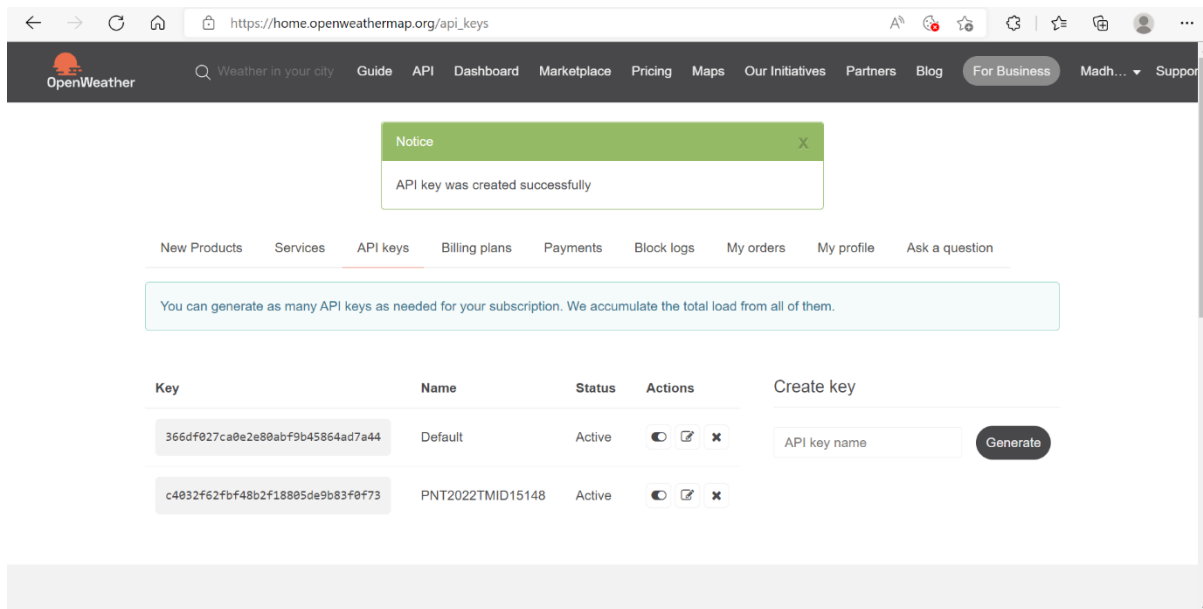
Date	4 November 2022
Team ID	PNT2022TMID15148
Project Name	Project – SIGNS WITH SMART CONNECTIVITY FOR BETTER ROAD SAFETY

**Project Development – Delivery of Sprint 1:**

**SIGNS WITH SMART CONNECTIVITY FOR BETTER ROAD SAFETY**

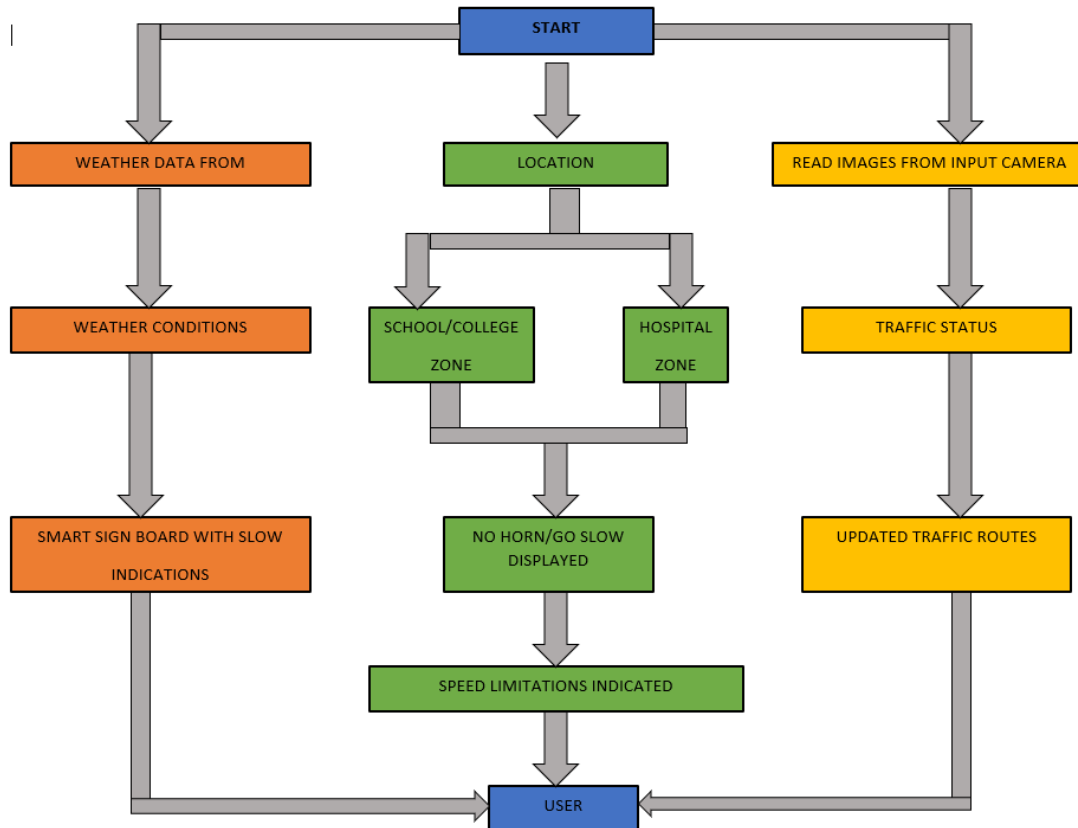
**Sprint Goals:**

- Create and set up accounts for numerous open APIs, such as the Open Weather API.



- Create a Python application that generates outputs based on inputs such as the local climate and geographic location.

**Data Flow:**



### Program Code:

✓ (./openweatherapi.py)

This file contains a utility function that uses OpenWeatherAPI to retrieve the weather. It only gives certain results necessary inputs for the API results.

```

import requests as reqs

def get(myLocation,APIKEY):
    apiURL = f"https://api.openweathermap.org/data/2.5/weather?q={myLocation}&appid={APIKEY}"
    responseJSON = (reqs.get(apiURL)).json()
    responseObject = {
        "temperature" : responseJSON['main']['temp'] - 273.15,
        "weather" : [responseJSON['weather'][_]['main'].lower() for _ in range(len(responseJSON['weather']))],
        "visibility" : responseJSON['visibility']/100,
    }
    if("rain" in responseJSON):

```

```

        responseObject["rain"] = [responseJSON["rain"][key] for key in responseJSON["rain"]]
    return(responseObject)

```

#### ✓ (./hardware.py)

This file is a utility function that abstracts all extraneous details and only returns the information that is necessary to be presented on the hardware side. The logic behind the code flow is put into action here.

```

import openweatherapi
from datetime import datetime as dt

def processConditions(myLocation,APIKEY,localityInfo):
    weatherData = weather.get(myLocation,APIKEY)

    finalSpeed = localityInfo["usualSpeedLimit"] if "rain" not in weatherData else localityInfo["usualSpeedLimit"]/2
    finalSpeed = finalSpeed if weatherData["visibility"]>35 else finalSpeed/2

    if(localityInfo["hospitalsNearby"]):
        # hospital zone
        doNotHonk = True
    else:
        if(localityInfo["schools"]["schoolZone"]==False):
            # neither school nor hospital zone
            doNotHonk = False
        else:
            # school zone
            now = [dt.now().hour,dt.now().minute]
            activeTime = [list(map(int,_split(":")) for _ in localityInfo["schools"]["activeTime"])]
            doNotHonk = activeTime[0][0]<=now[0]<=activeTime[1][0] and activeTime[0][1]<=now[1]<=activeTime[1][1]

    return({
        "speed" : finalSpeed,
        "doNotHonk" : doNotHonk
    })

```

#### ✓ (./mcon.py)

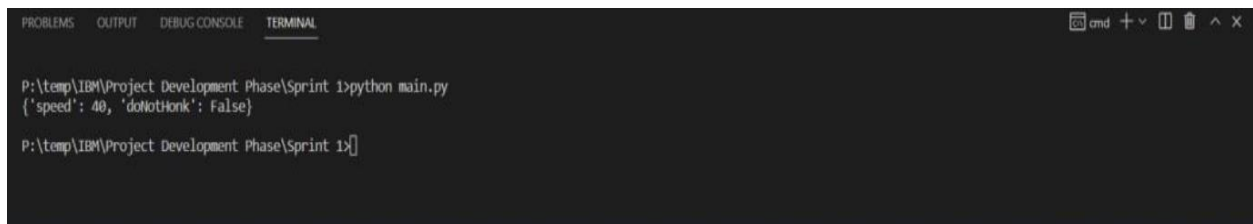
The programme that the microcontroller executes indefinitely. The output hardware display is changed dependent on the return result from each of the util functions called from other Python scripts.

```
import hardware

myLocation = "Chennai, IN"
APIKEY = "bf4a8d480ee05c00952bf65b78ae826b"

localityInfo = {
    "schools" : {
        "schoolZone" : True,
        "activeTime" : ["7:00", "17:30"] # schools active from 7 AM till 5:
30 PM
    },
    "hospitalsNearby" : False,
    "usualSpeedLimit" : 40 # in km/hr
}
print(brain.processConditions(myLocation, APIKEY, localityInfo))
```

### **Output:**

A screenshot of a terminal window with a dark background. The terminal shows the command 'python main.py' being executed in a directory 'P:\temp\IBM\Project Development Phase\Sprint 1'. The output of the script is a dictionary: {'speed': 40, 'doNotHonk': False}. The terminal window has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL', with 'TERMINAL' being the active tab. The window title bar shows 'cmd' and standard window controls.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL cmd + - [ ] ^ X

P:\temp\IBM\Project Development Phase\Sprint 1>python main.py
{'speed': 40, 'doNotHonk': False}

P:\temp\IBM\Project Development Phase\Sprint 1>
```