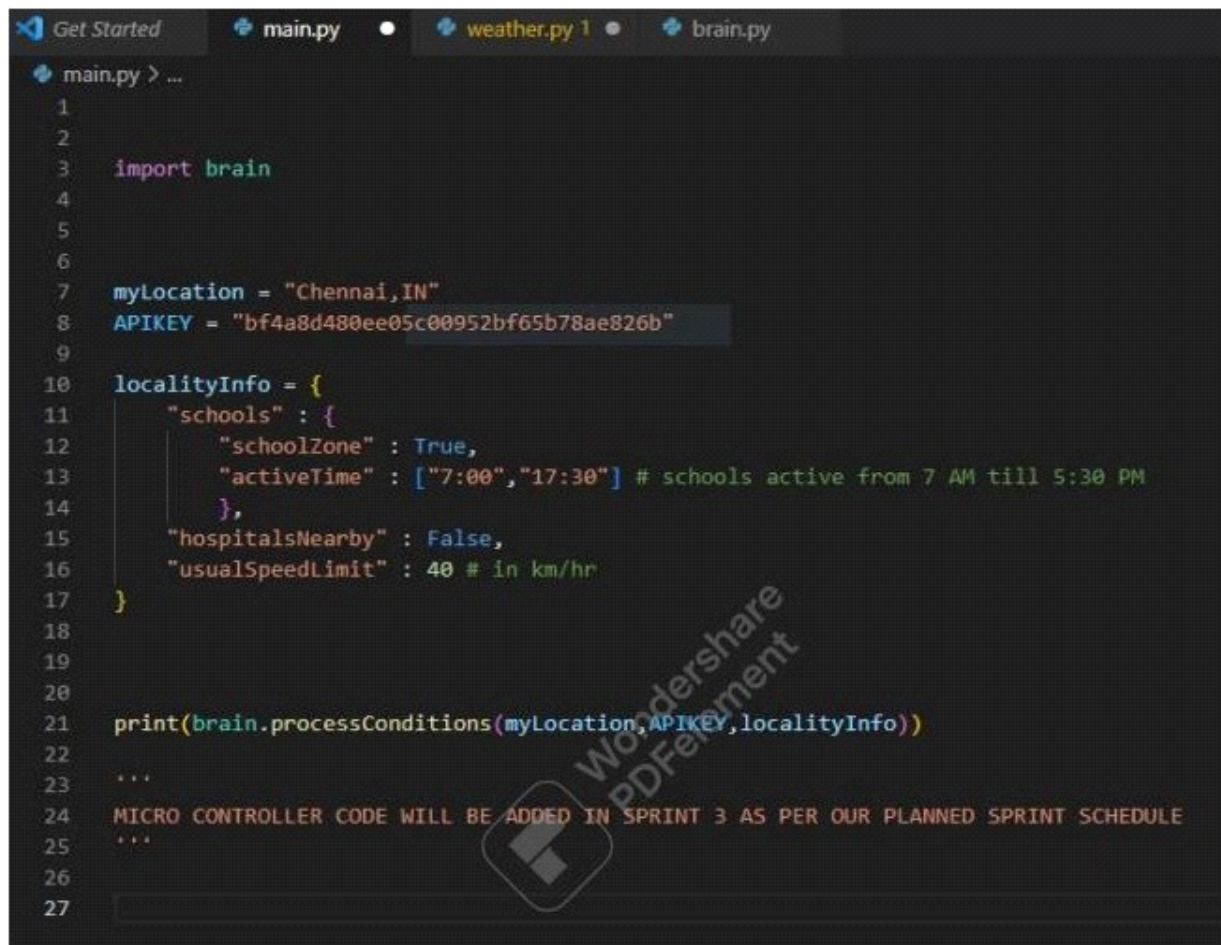


## Development Phase Sprint 4

Team ID : PNT2022TMID41853

Project Name: Signs with Smart Connectivity for Better Safety

Main.py



```
1
2
3 import brain
4
5
6
7 myLocation = "Chennai,IN"
8 APIKEY = "bf4a8d480ee05c00952bf65b78ae826b"
9
10 localityInfo = {
11     "schools" : {
12         "schoolZone" : True,
13         "activeTime" : ["7:00","17:30"] # schools active from 7 AM till 5:30 PM
14     },
15     "hospitalsNearby" : False,
16     "usualSpeedLimit" : 40 # in km/hr
17 }
18
19
20
21 print(brain.processConditions(myLocation,APIKEY,localityInfo))
22
23 ...
24 MICRO CONTROLLER CODE WILL BE ADDED IN SPRINT 3 AS PER OUR PLANNED SPRINT SCHEDULE
25 ...
26
27
```

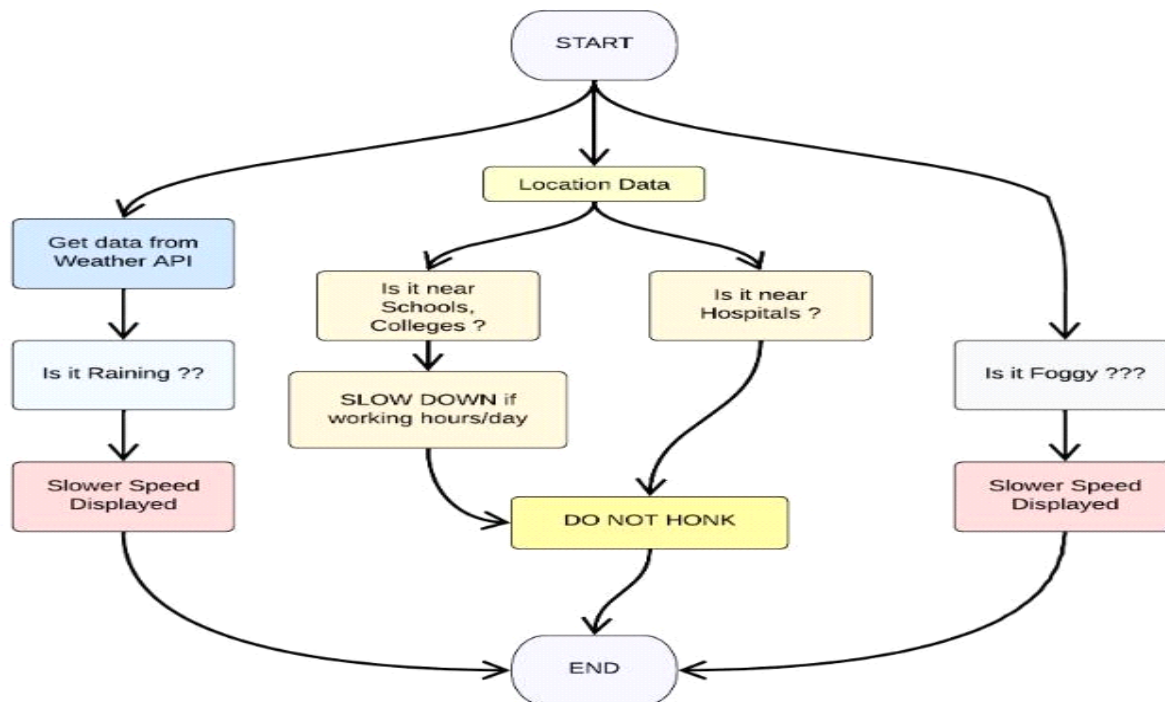
weather.py

```
Get Started | main.py | weather.py 1 | brain.py
weather.py > get
1  # Python code
2
3  import requests as reqs
4
5  def get(myLocation,APIKEY):
6
7      apiURL = f"https://api.openweathermap.org/data/2.5/weather?q={myLocation}&appid={APIKEY}"
8      responseJSON = (reqs.get(apiURL)).json()
9      returnObject = {
10         "temperature" : responseJSON['main']['temp'] - 273.15,
11         "weather" : [responseJSON['weather'][_]['main'].lower() for _ in range(len(responseJSON['weather']))],
12         "visibility" : responseJSON['visibility']/100, # visibility in percentage where 10km is 100% and 0km is
13     }
14     if("rain" in responseJSON):
15         returnObject["rain"] = [responseJSON["rain"][key] for key in responseJSON["rain"]]
16     return(returnObject)
17
```

brain.py

```
Get Started | main.py | weather.py | brain.py |
brain.py > ...
1
2 import weather
3 from datetime import datetime as dt
4
5 # IMPORT SECTION ENDS
6 # -----
7 # UTILITY LOGIC SECTION STARTS
8 def processConditions(myLocation,APIKEY,localityInfo):
9     weatherData = weather.get(myLocation,APIKEY)
10
11     finalSpeed = localityInfo["usualSpeedLimit"] if "rain" not in weatherData else localityInfo["usualSpeedLimit"]
12     finalSpeed = finalSpeed if weatherData["visibility"]>35 else finalSpeed/2
13
14     if(localityInfo["hospitalsNearby"]):
15         # hospital zone
16         doNotHonk = True
17     else:
18         if(localityInfo["schools"]["schoolZone"]==False):
19             # neither school nor hospital zone
20             doNotHonk = False
21         else:
22             # school zone
23             now = [dt.now().hour,dt.now().minute]
24             activeTime = [list(map(int,_.split(":"))) for _ in localityInfo["schools"]["activeTime"]]
25             doNotHonk = activeTime[0][0]<now[0]<activeTime[1][0] and activeTime[0][1]<now[1]<activeTime[1][1]
26
27     return({
28         "speed" : finalSpeed,
29         "doNotHonk" : doNotHonk
30     })
31
32 # UTILITY LOGIC SECTION ENDS
33
```

Code Flow:



output:

# Code Output

```

2022-11-06 21:38:33,452 wiotp.sdk.device.client.DeviceClient INFO
Connected successfully: d:epmoec:testDevice:device0
2022-11-06 21:38:33,452 wiotp.sdk.device.client.DeviceClient INFO
Disconnected from the IBM Watson IoT Platform
2022-11-06 21:38:33,452 wiotp.sdk.device.client.DeviceClient INFO
Closed connection to the IBM Watson IoT Platform
{'speed': 40, 'doNotHonk': False}
2022-11-06 21:38:35,631 wiotp.sdk.device.client.DeviceClient INFO
Connected successfully: d:epmoec:testDevice:device0
2022-11-06 21:38:35,631 wiotp.sdk.device.client.DeviceClient INFO
Disconnected from the IBM Watson IoT Platform
2022-11-06 21:38:35,631 wiotp.sdk.device.client.DeviceClient INFO
Closed connection to the IBM Watson IoT Platform
{'speed': 40, 'doNotHonk': False}

```

... repeats every 1 sec

output image:

