**Project Development Phase**

**Project Development Delivery of Sprint 1**

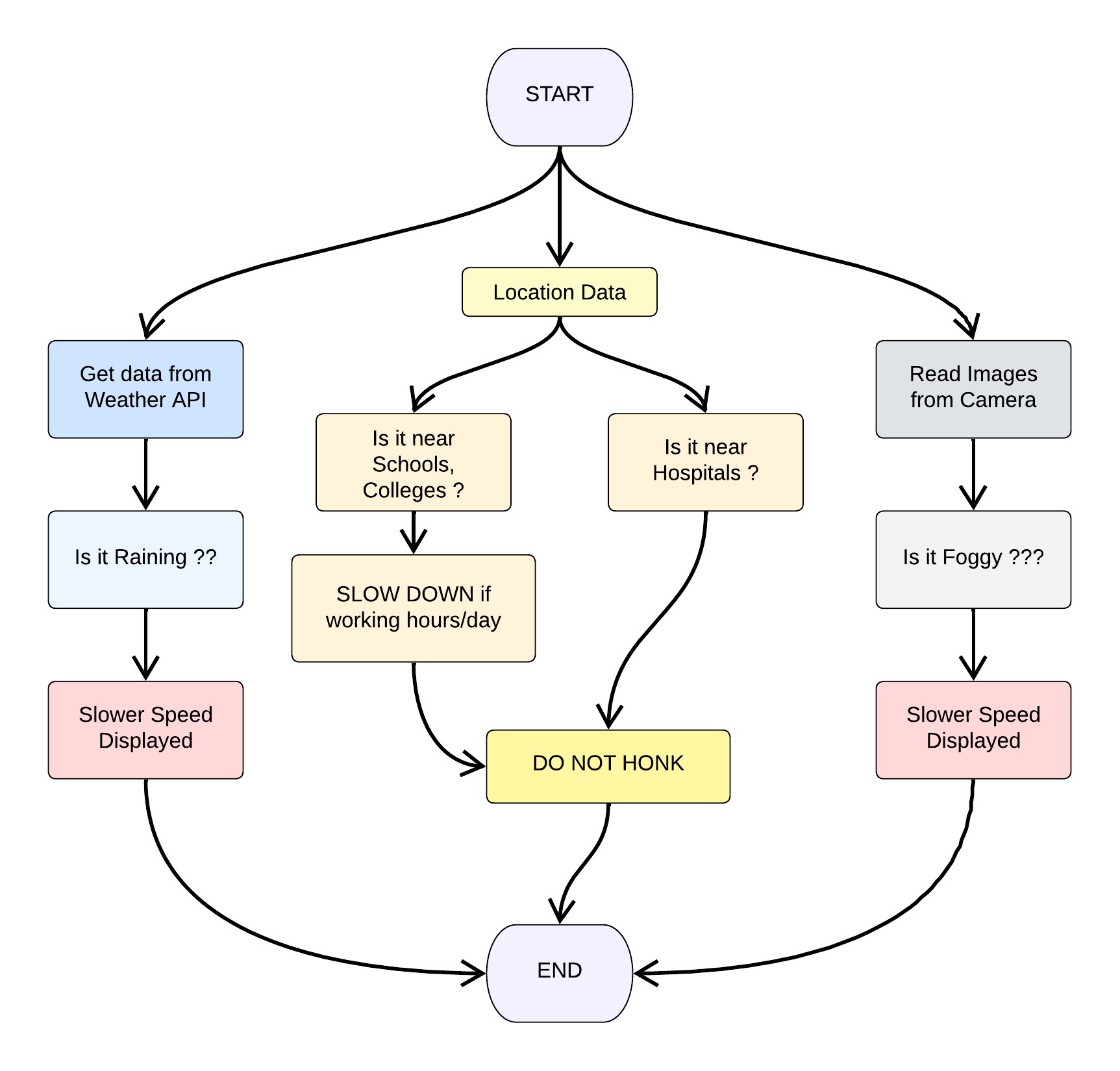
|  |  |
| --- | --- |
| Date | 28 October 2022 |
| Team ID | PNT2022TMID24784 |
| Project Name | Project - Signs with smart connectivity for Better road safety |
| Maximum Marks | 4 Marks |

**Signs with smart connectivity for Better road safety**

**Sprint Goals :**

1. Create and initialize accounts in various public APIs like OpenWeather API 2. Write a Python program that outputs results given the inputs like weather and location.

**Code Flow :**



**Program Code :**

# weather.py

# Python code importrequestsasreqs defget(myLocation,APIKEY):

apiURL=f"https://api.openweathermap.org/data/2.5/weather?q={myLocation}&appid=

{APIKEY}"

responseJSON= (reqs.get(apiURL)).json() returnObject= {

"temperature" : responseJSON['main']['temp'] -273.15,

"weather" : [responseJSON['weather'][\_]['main'].lower() for\_inrange(len(responseJSON['weather']))],

"visibility" : responseJSON['visibility']/100, # visibility in percentage where 10km is 100% and 0km is 0%

}

if("rain"inresponseJSON):

returnObject["rain"] = [responseJSON["rain"][key] forkeyinresponseJSON["rain"]] return(returnObject)

# info.py

This file is a utility function that returns only essential information to be displayed at the hardware side and abstracts all the unnecessary details. This is where the code flow logic is implemented.

# Python code

# IMPORT SECTION STARTS importweather fromdatetimeimportdatetimeasdt # IMPORT SECTION ENDS

# -----------------------------------------------

# UTILITY LOGIC SECTION STARTS defprocessConditions(myLocation,APIKEY,localityInfo):

weatherData=weather.get(myLocation,APIKEY) finalSpeed=localityInfo["usualSpeedLimit"] if"rain"notinweatherDataelselocalityInfo["usualSpeedLimit"]/2 finalSpeed=finalSpeedifweatherData["visibility"]>35elsefinalSpeed/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\_inlocalityInfo["schools"]["activeTime"]]

doNotHonk=activeTime[0][0]<=now[0]<=activeTime[1][0] andactiveTime[0][1]<=now[1]<=activeTime[1][1] return({

"speed" : finalSpeed,

"doNotHonk" : doNotHonk

})

# UTILITY LOGIC SECTION ENDS

# main.py

The code that runs in a forever loop in the microcontroller. This calls all the util functions from other python files and based on the return value transduces changes in the output hardware display.

# Python code

# IMPORT SECTION STARTS

importbrain

# IMPORT SECTION ENDS

# -----------------------------------------------

# USER INPUT SECTION STARTS myLocation="Chennai,IN"

APIKEY="9cd610e5fd400c74212074c7ace0d62c" 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

}

# USER INPUT SECTION ENDS

# -----------------------------------------------

# MICRO-CONTROLLER CODE STARTS

print(brain.processConditions(myLocation,APIKEY,localityInfo))

'''

MICRO CONTROLLER CODE WILL BE ADDED IN SPRINT 2 AS PER OUR PLANNED

SPRINT SCHEDULE

'''

# MICRO-CONTROLLER CODE ENDS

**Output :**

# Code Output

{'speed': 40, 'doNotHonk': False}

