### **Personal Assistance For Independent Senior Citizens**

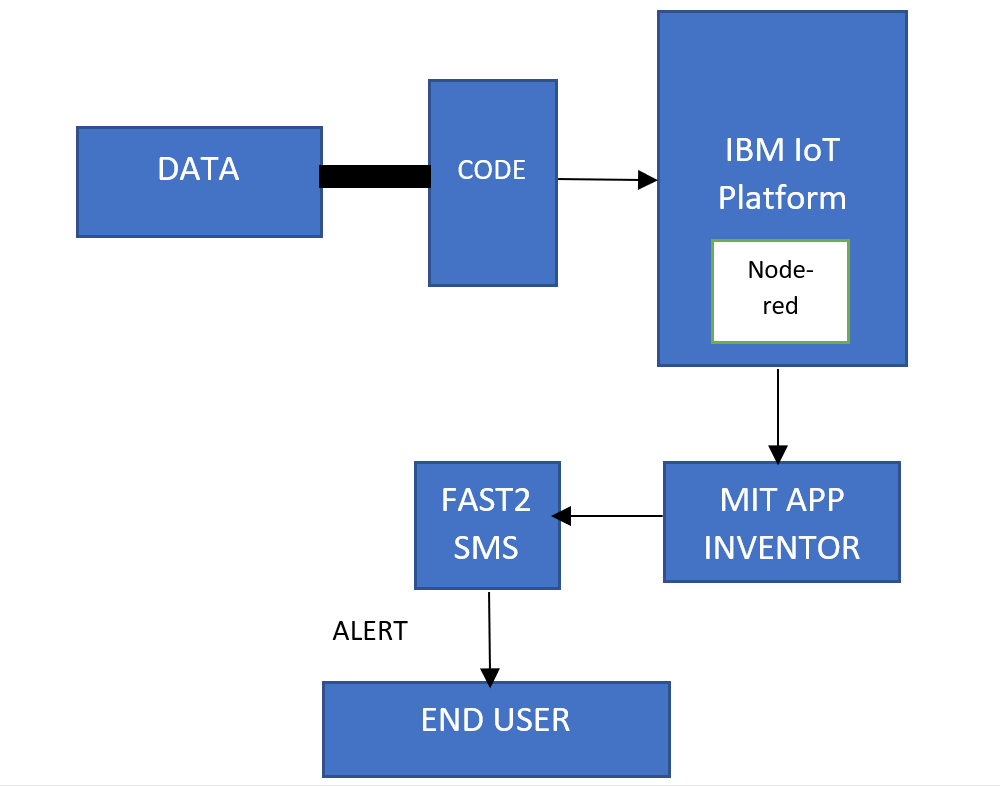
**Overview**: This project will guide you in developing the "Personal Assistance system for Independent Senior Citizens" using IBM Platform, FAST2SMS, MIT APP INVENTOR and Python IDLE.

**Purpose**: To give the data of medication along with the time and thereby alerting the user through FAST2SMS and mobile application.

**Existing Problem**: Many old people forget about taking medicine either due to aging or due to lack of care.

**Proposed Solution**: Using the data from user's reports, we integrate the node-red platform with MIT APP INVENTOR and FAST2SMS to send alerts to user about the medication and timing.

**Block diagram** :



**Hardware/Software required** :

**IBM** : acts as the platform for using services like IBM IoT, Text to Speech, Speech to text and softwares like node-red. It can also be used for storing, receiving and retrieving data.

**Node-red** : software available within the IBM platform which helps us in using MIT APP INVENTOR and also it has the UI option to display the data and required commands in the Python. Nodes act as the backbone.

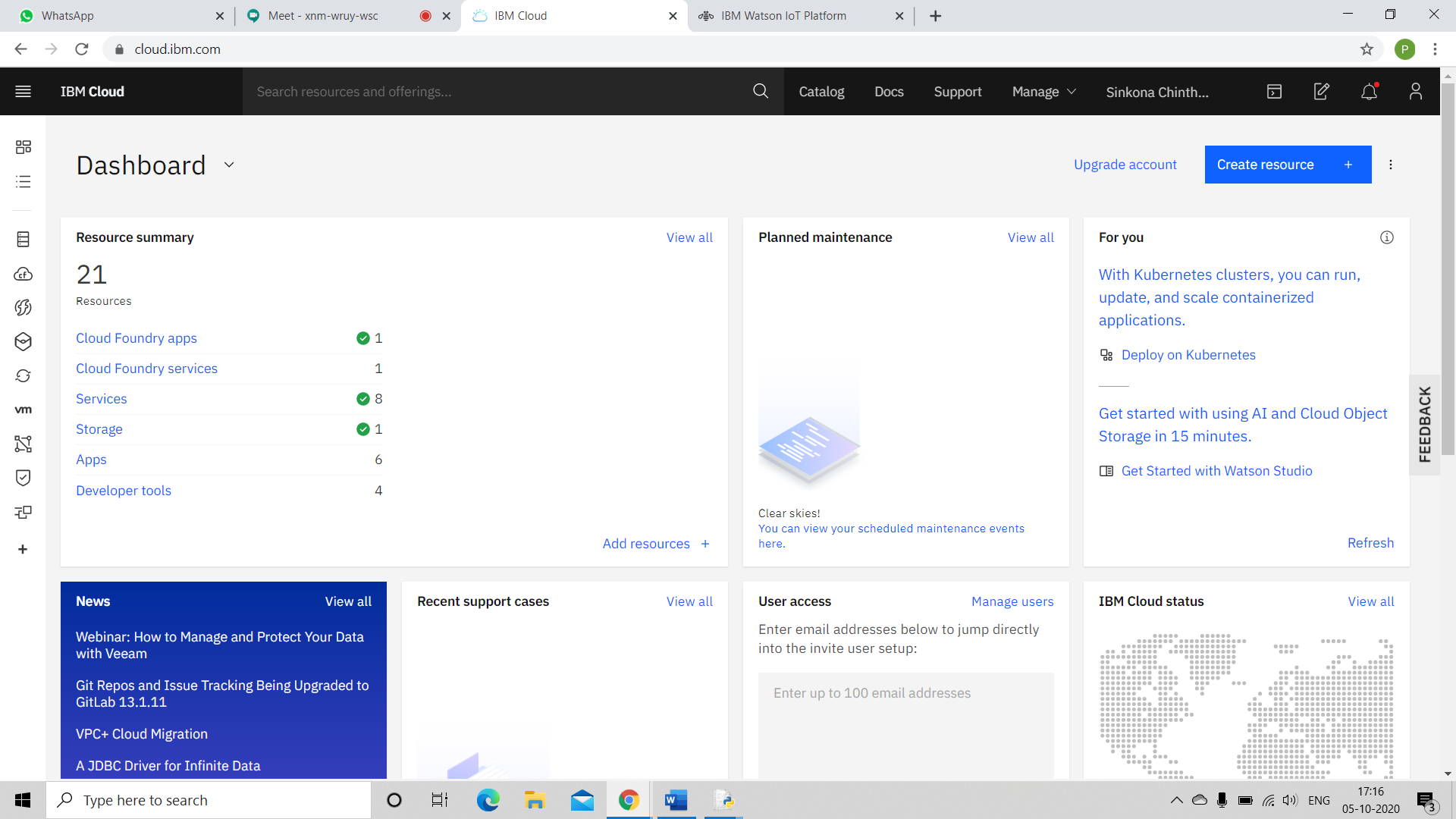
**Python(IDLE)** : helps in writing the code for Text to Speech, Speech to Text and for publishing data into IBM Cloud.

**FAST2SMS** : It helps us in sending the alerts/messages to multiple users at the same instant of time.

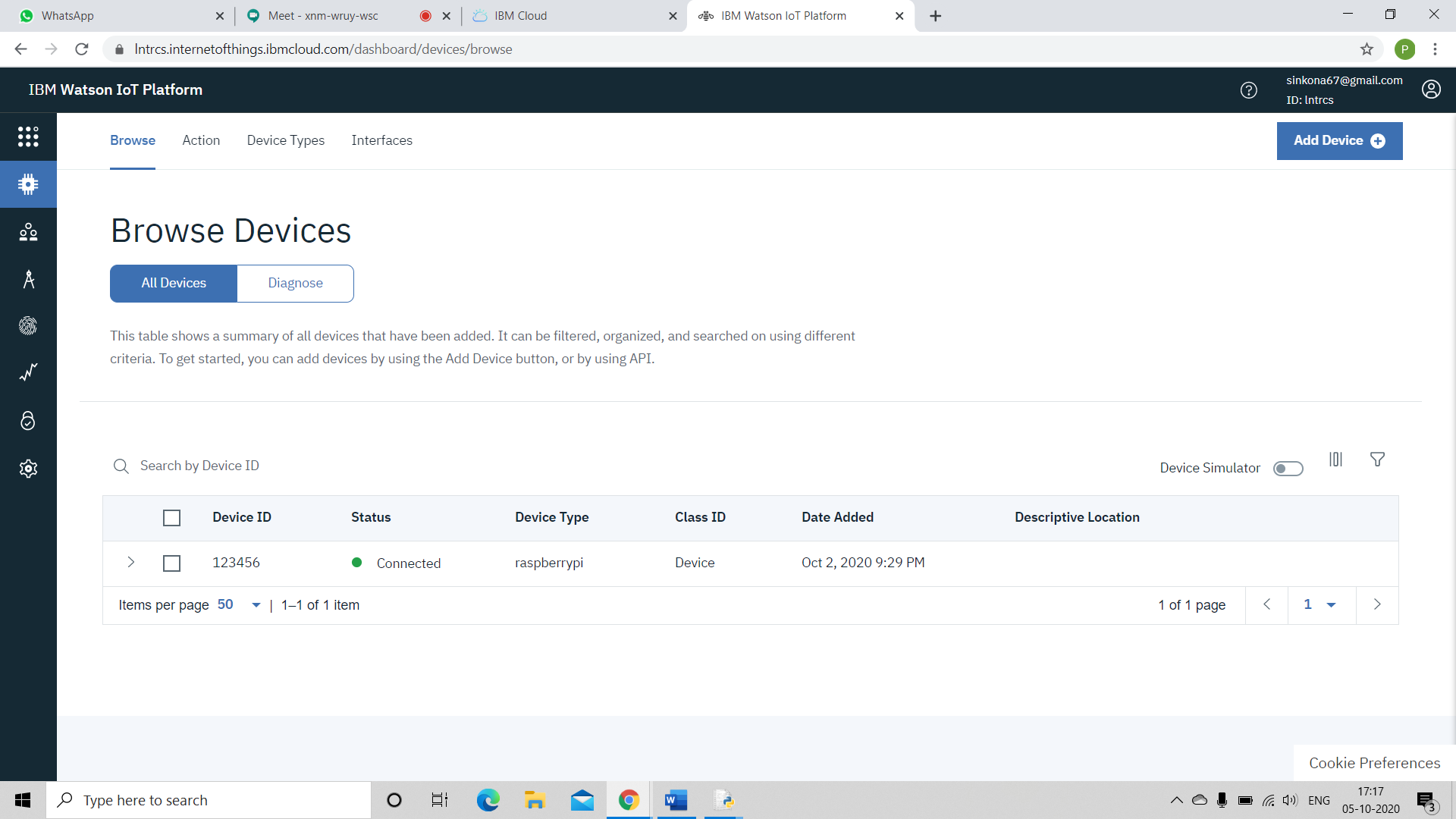
**MIT APP INVENTOR** : helps us in developing model applications/apk to simulate the data from IBM Platform and also for sending alerts through FAST2SMS. Blocks acts as the backbone.

**Experimental investigations and the project** :

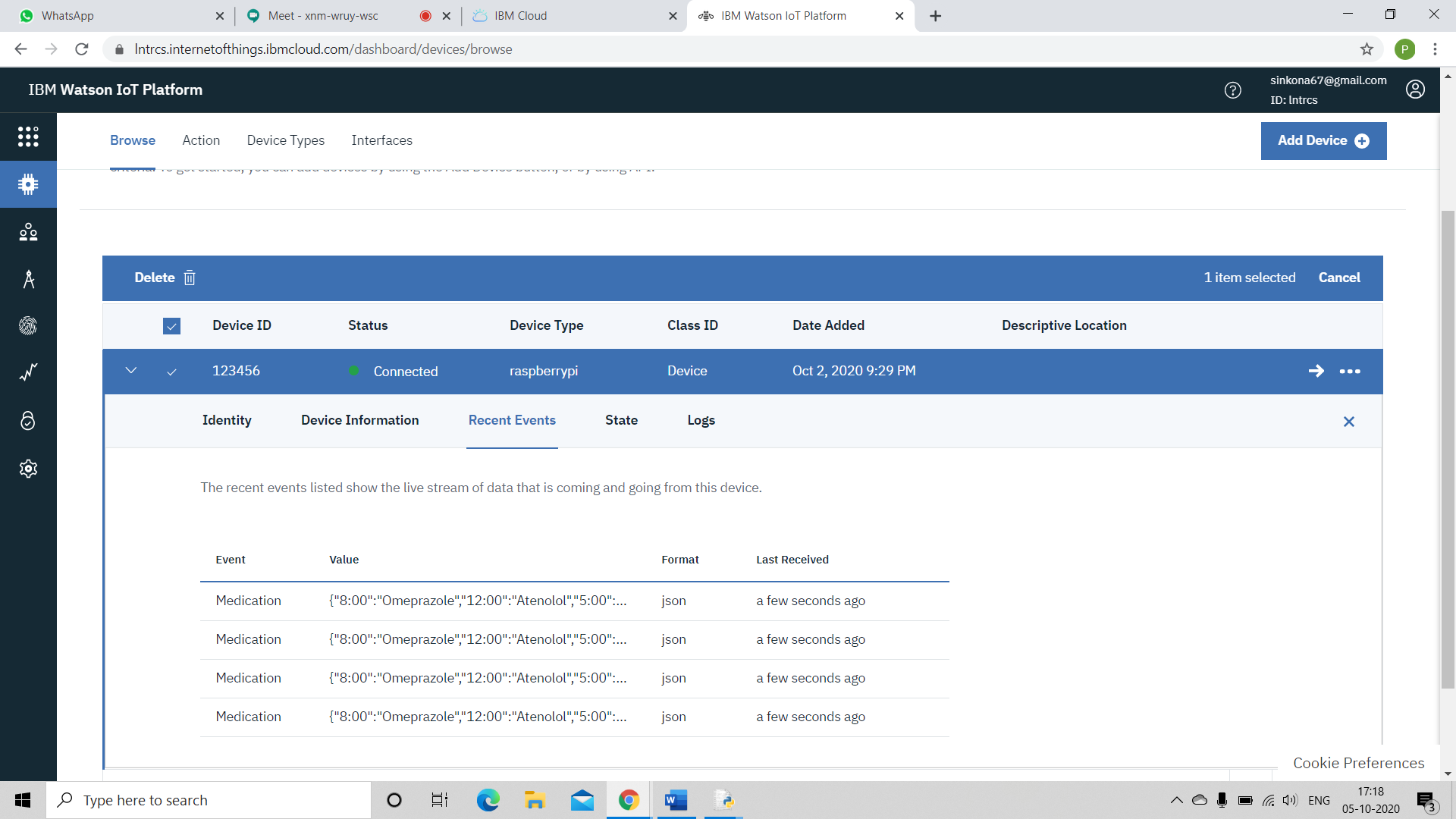
IBM is a platform which is used to create several services and softwares. It helps to store the data and retrieve whenever required. Initially, we use the python code and send medicine details to IBM IoT platform. IBM IoT service has it's own credentials to get connected with Python IDLE. IBM Cloud also provides us the software called node-red, which can be used for printing the values from IBM IoT service using IBM IoT nodes. Node-red also allows us to use dashboard nodes to create UI. We can also create http requests using node-red. MIT APP INVENTOR uses the response url and prints the required data in the application. MIT APP INVENTOR also helps us in alerting the user through FAST2SMS. Voice notifications regarding the medication will also be provided through the application, ensuring that the user takes medicine on time.

**IBM CLOUD PLATFORM(DASHBOARD) :**

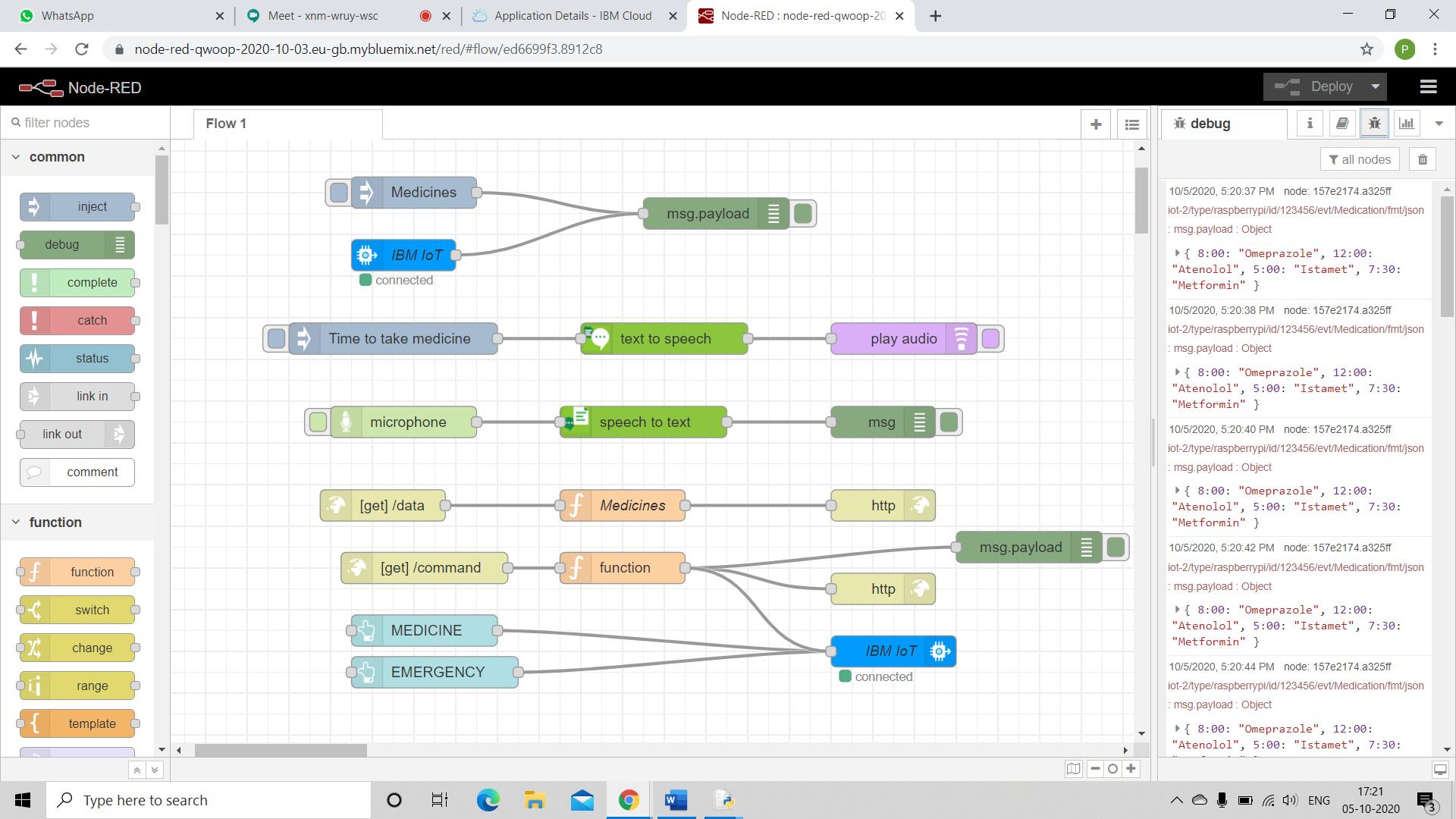
**IBM IoT PLATFORM :**



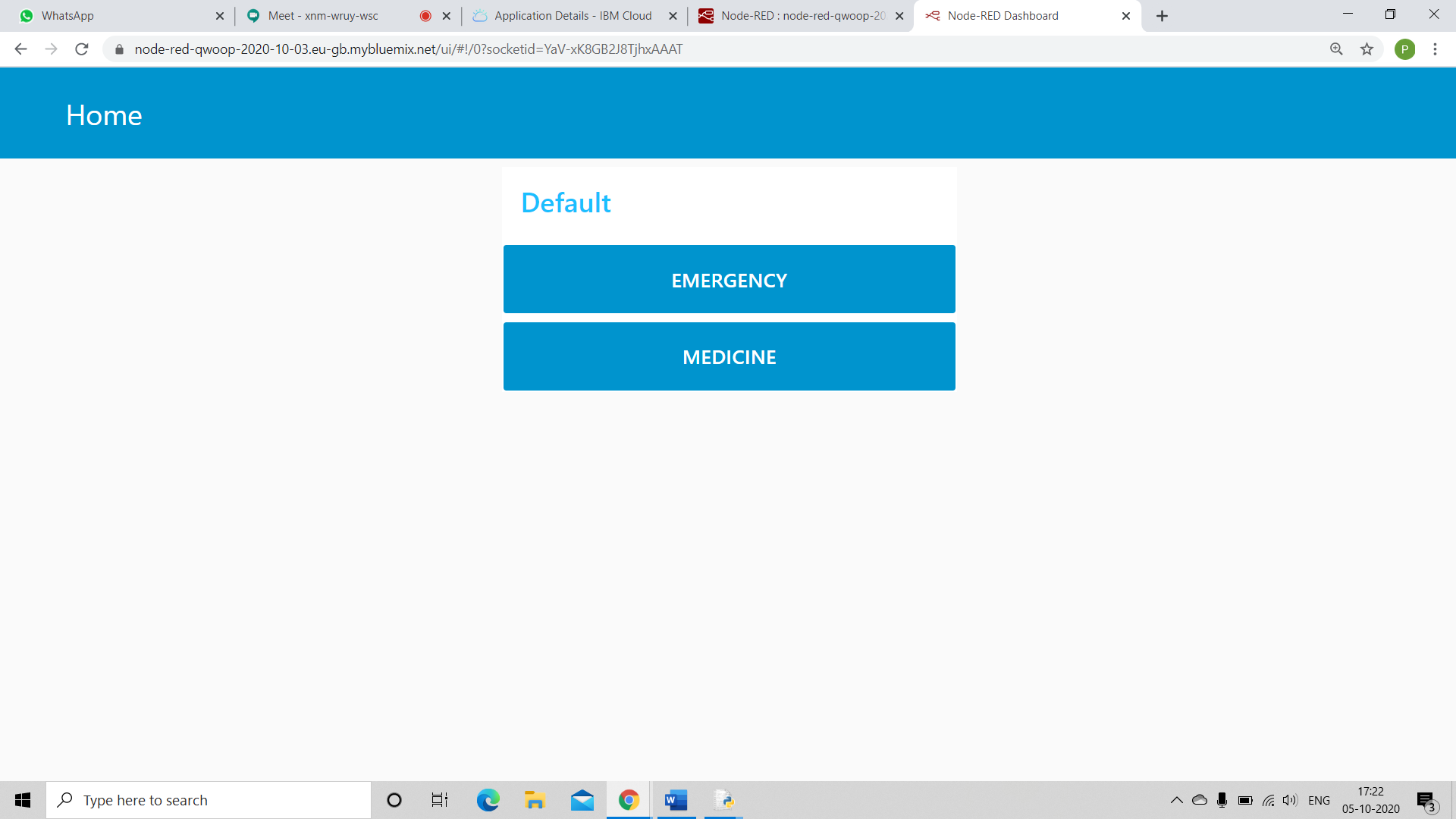
**IBM IoT(RECENT EVENTS) :**



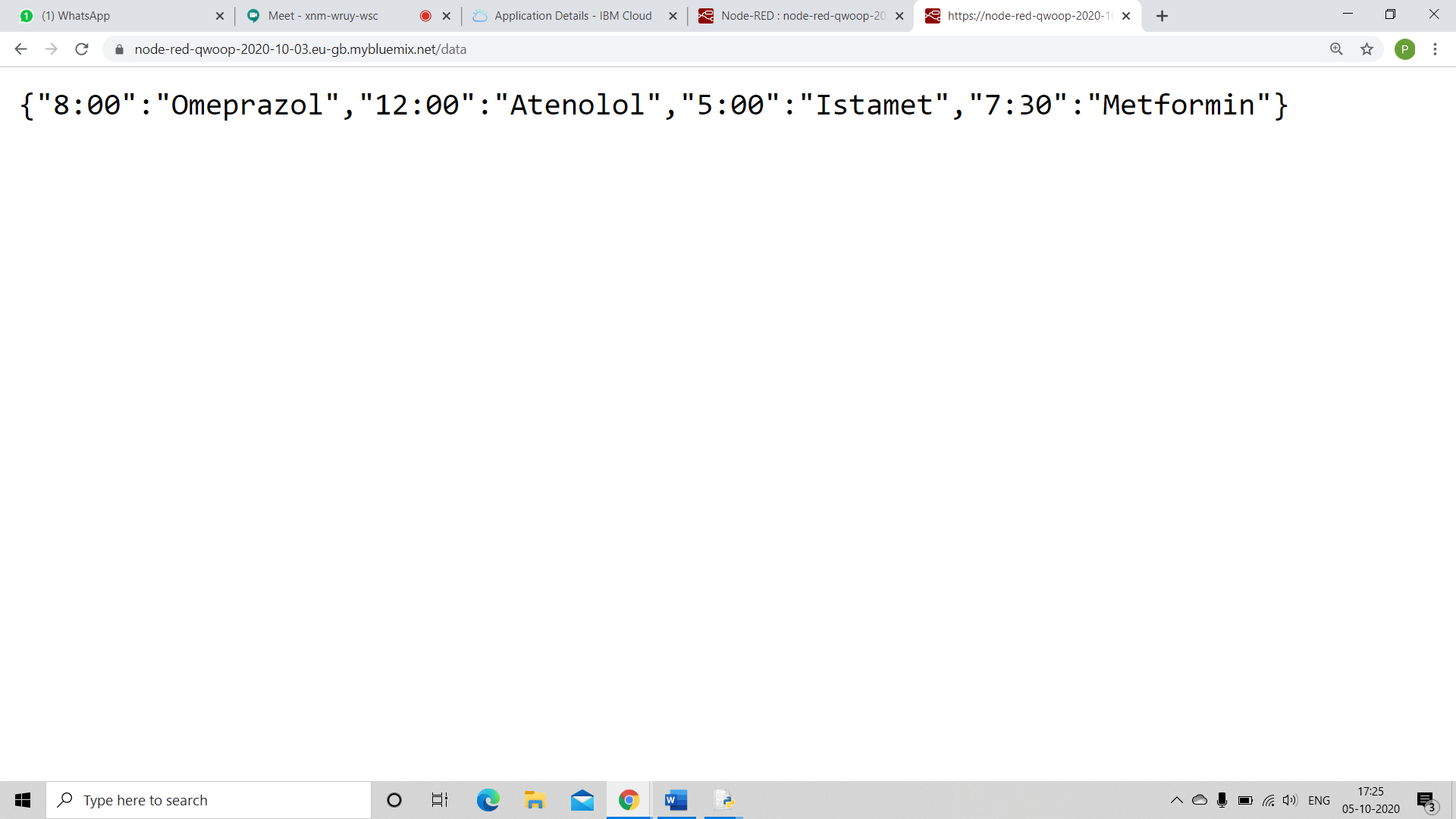
**NODE-RED PLATFORM :**

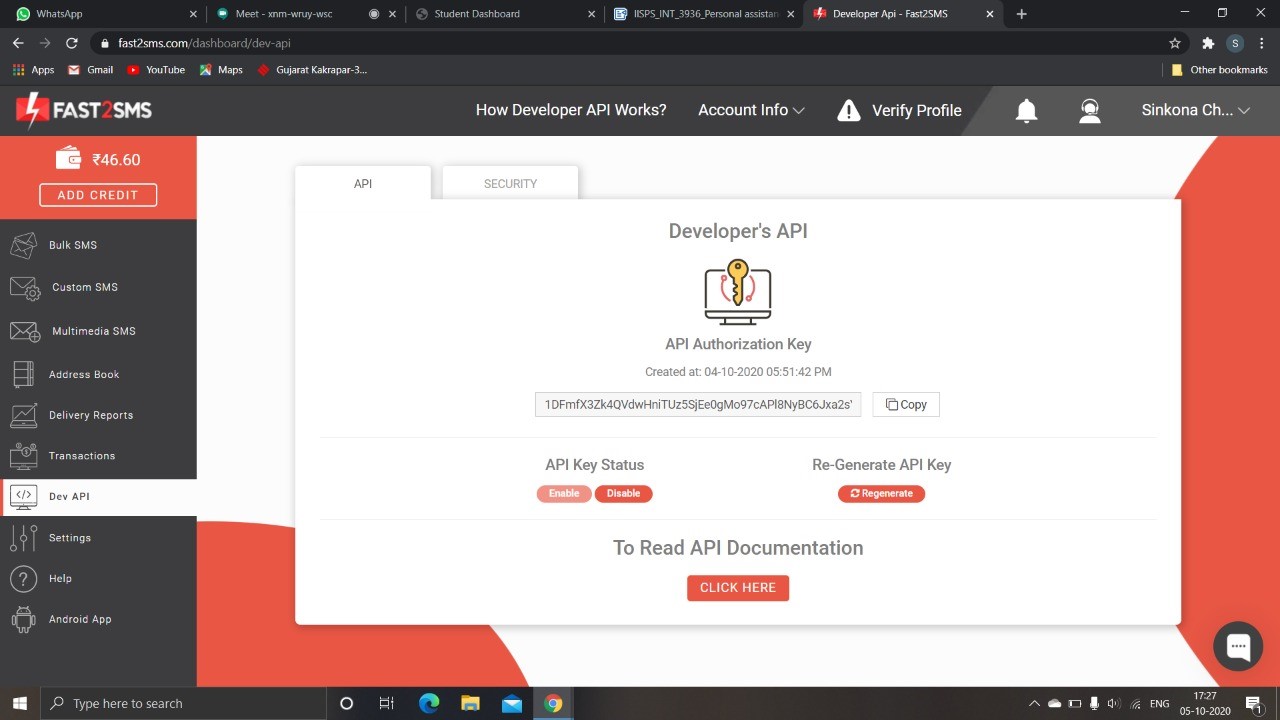


**NODE-RED UI WINDOW :**

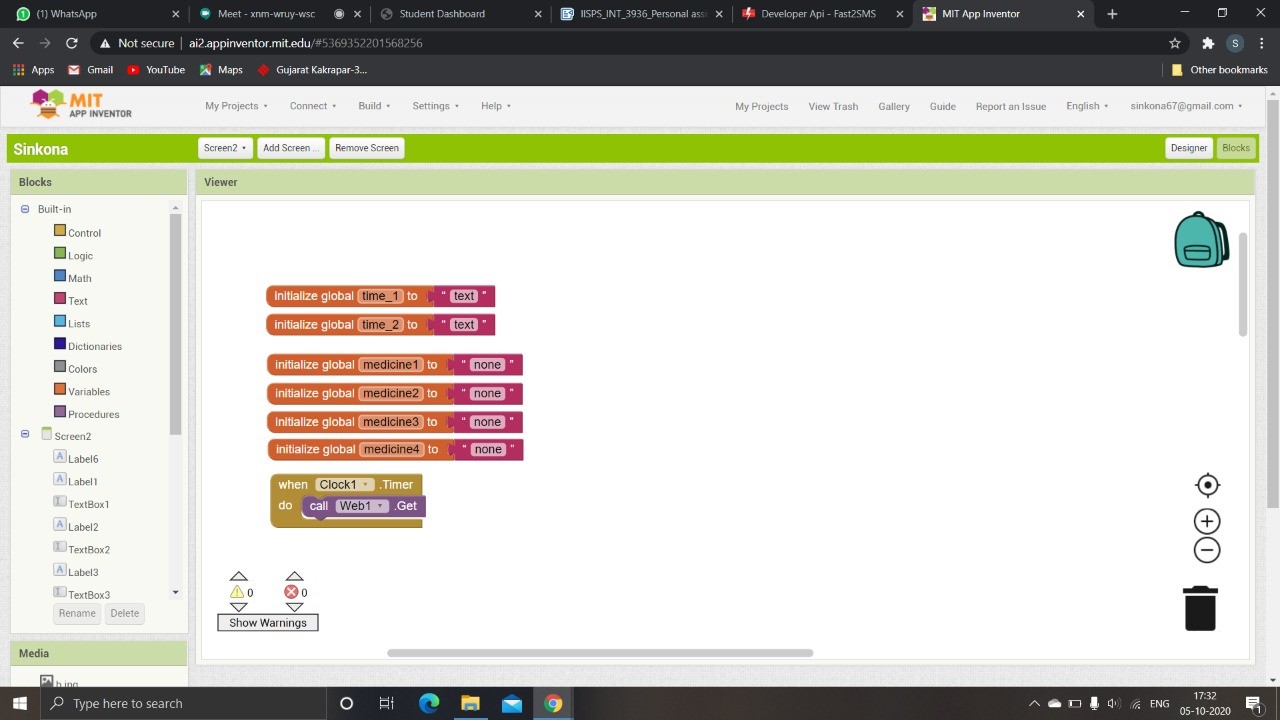


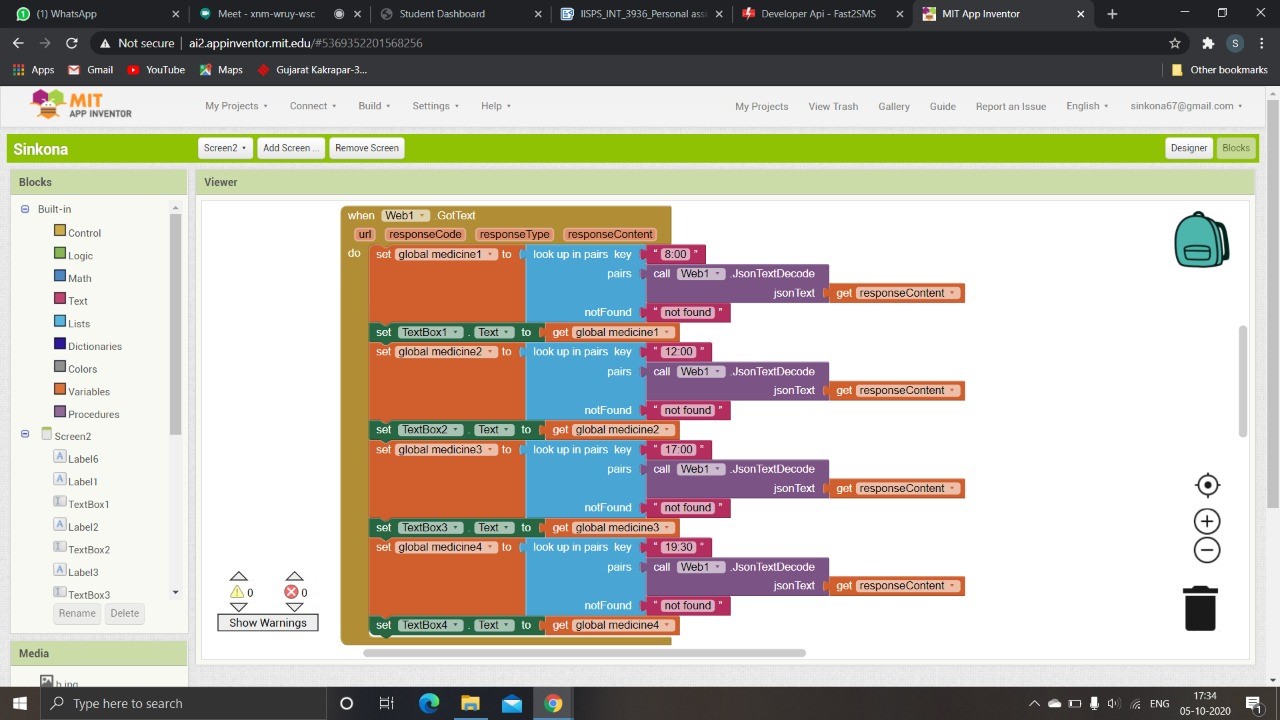
**NODE-RED DATA WINDOW :**

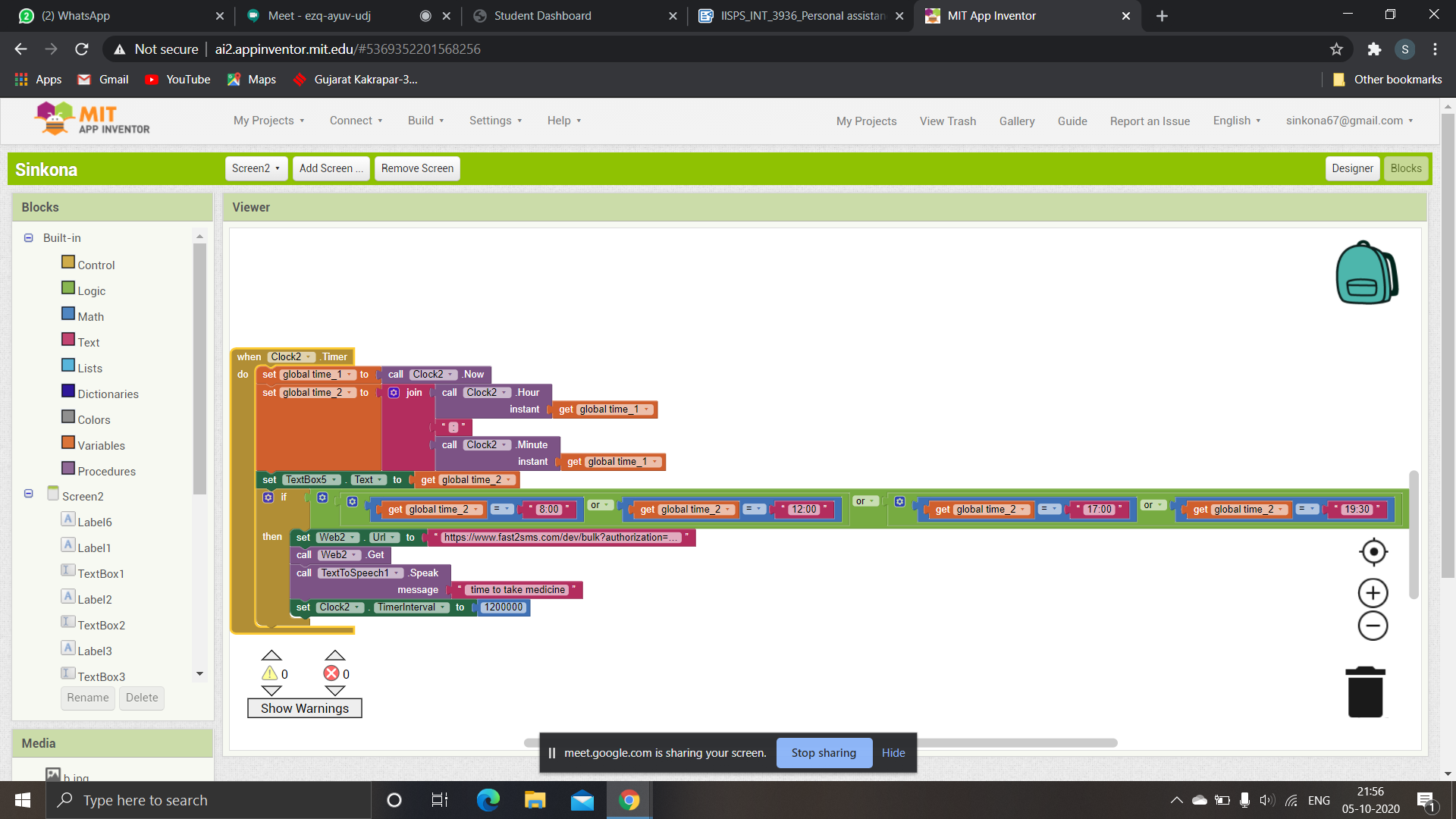
**FAST2SMS :**

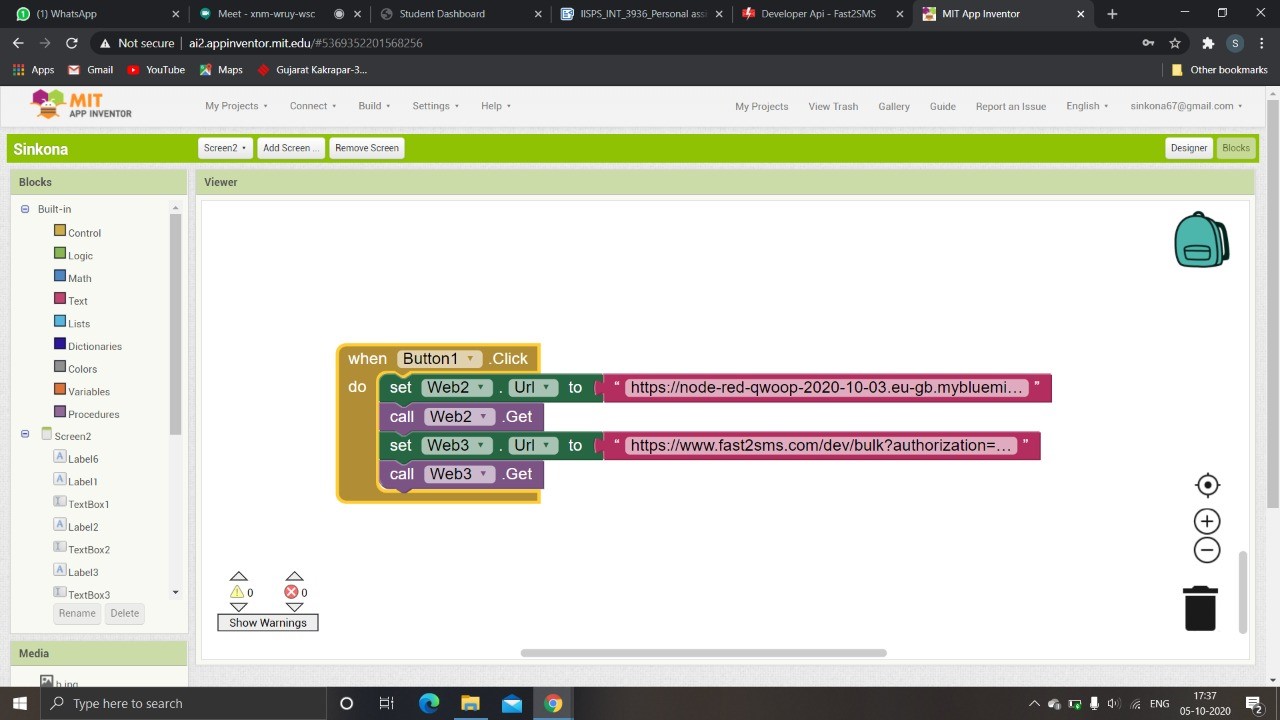


**BLOCKS IN MIT :**

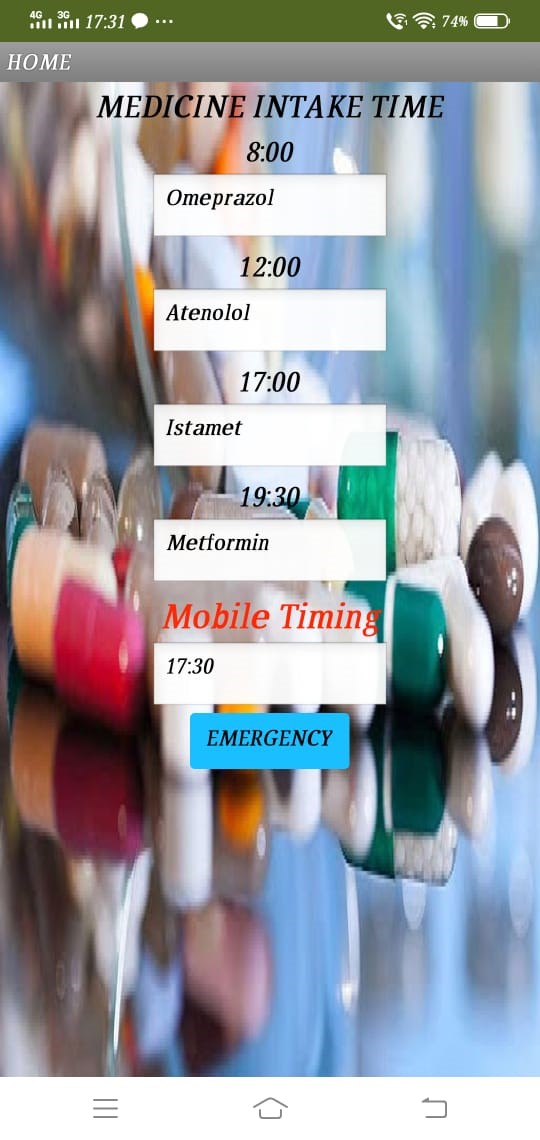








**MIT APP INVENTOR :**



|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | |
|  |  |  |

**FLOW CHART :**

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |

**RESULT :**

Hence, using the data of the medicines provided, timings of the medication will be reminded to the user and emergency notifications will be sent to the user's mobile through fast2sms. Thereby it ensures the safe and timely medication to the respective senior citizen.

**ADVANTAGES :**

* Ensures the proper health of the elderly people .
* It also helps the elderly people with less memory power by alerting them with medicine intake notifications.

**DISADVANTAGES :**

* In remote areas due to network issues there would be problem in the working of application.

**APPLICATIONS :**

* In oldage homes.
* For all the elderly people who live individually or with the children.
* Also for the people with little memory power.

**CONCLUSION :**

The Personal Assistance system ensures the proper health of the individual senior citizen.

**FUTURE SCOPE :**

Elderly people play a very important role in the development of society. Though, they can not contribute functionally,they play a major role by teaching ethical values and morals to their children and grand children. So their well being will be present generation's well being. Hence we need to understand their needs and serve them properly. Basic need for any elderly citizen is **HEALTH.** So this **PERSONAL ASSISTANCE FOR INDIVIDUAL SENIOR CITIZENS APPLICATION** has an ability to improve and track the health condition. So by adapting this system in a large scale ensures inculcating ethical values in future generations. This also increases the life span of the elderly people and thereby decreasing the mortality rate.

**APPENDIX :**

**CODING PART :**

**CODE FOR TEXT TO SPEECH :**

from ibm\_watson import TextToSpeechV1

from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator

from playsound import playsound

import time

authenticator = IAMAuthenticator('A-byYc7kDPoZfScLm1O5OKLU3NLNwC9UoOEoYWPVkPq2')

text\_to\_speech = TextToSpeechV1(

authenticator=authenticator

)

text\_to\_speech.set\_service\_url('https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/e31acf7f-081c-4e28-93d8-7c422e71c988')

while True:

localtime=time.localtime()

result=time.strftime("%I:%M:%S %p",localtime)

print(result)

if(result=="08:00:00 AM"):

with open('medicine1.mp3', 'wb') as audio\_file:

audio\_file.write(

text\_to\_speech.synthesize(

'Time to take medicine Omeprazole',

voice='en-US\_AllisonV3Voice',

accept='audio/mp3'

).get\_result().content)

playsound('medicine1.mp3')

if(result=="12:00:00 PM":

with open('medicine2.mp3', 'wb') as audio\_file:

audio\_file.write(

text\_to\_speech.synthesize(

'Time to take medicine Atenolol',

voice='en-US\_AllisonV3Voice',

accept='audio/mp3'

).get\_result().content)

playsound('medicine2.mp3')

if(result=="05:00:00 PM"):

with open('medicine3.mp3', 'wb') as audio\_file:

audio\_file.write(

text\_to\_speech.synthesize(

'Time to take medicine Istamet',

voice='en-US\_AllisonV3Voice',

accept='audio/mp3'

).get\_result().content)

playsound('medicine3.mp3')

if(result=="07:30:00 PM"):

with open('medicine4.mp3', 'wb') as audio\_file:

audio\_file.write(

text\_to\_speech.synthesize(

'Time to take medicine Metformin',

voice='en-US\_AllisonV3Voice',

accept='audio/mp3'

).get\_result().content)

playsound('medicine4.mp3')

time.sleep(1)

**CODE FOR SPEECH TO TEXT :**

import json

from os.path import join, dirname

from ibm\_watson import SpeechToTextV1

from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator

authenticator = IAMAuthenticator('4qiTUyt\_9JtAUQ-9td87CcliqXjd1eFqR8ZT6hndzcPJ')

speech\_to\_text = SpeechToTextV1(

authenticator=authenticator

)

speech\_to\_text.set\_service\_url('https://api.eu-gb.speech-to-text.watson.cloud.ibm.com/instances/bab69765-c8fb-4722-8d28-a749e0e8cf85')

with open(join(dirname(\_\_file\_\_), './.', 'medicine1.mp3'),

'rb') as audio\_file:

speech\_recognition\_results = speech\_to\_text.recognize(

audio=audio\_file,

content\_type='audio/mp3',

).get\_result()

print(json.dumps(speech\_recognition\_results, indent=2))

with open(join(dirname(\_\_file\_\_), './.', 'medicine2.mp3'),

'rb') as audio\_file:

speech\_recognition\_results = speech\_to\_text.recognize(

audio=audio\_file,

content\_type='audio/mp3',

).get\_result()

print(json.dumps(speech\_recognition\_results, indent=2))

with open(join(dirname(\_\_file\_\_), './.', 'medicine3.mp3'),

'rb') as audio\_file:

speech\_recognition\_results = speech\_to\_text.recognize(

audio=audio\_file,

content\_type='audio/mp3',

).get\_result()

print(json.dumps(speech\_recognition\_results, indent=2))

with open(join(dirname(\_\_file\_\_), './.', 'medicine4.mp3'),

'rb') as audio\_file:

speech\_recognition\_results = speech\_to\_text.recognize(

audio=audio\_file,

content\_type='audio/mp3',

).get\_result()

print(json.dumps(speech\_recognition\_results, indent=2))

**CODE FOR INTEGRATING IBM DEVICE :**

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

#Provide your IBM Watson Device Credentials

organization = "lntrcs"

deviceType = "raspberrypi"

deviceId = "123456"

authMethod = "token"

authToken = "12345678"

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data)

i = cmd.data['command']

if i=='MEDICINE':

print("Time to take medicine")

elif i=='EMERGENCY':

print("Emergency! Need your help!!")

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Caught exception connecting device: %s" % str(e))

sys.exit()

#Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times

deviceCli.connect()

while True:

data = { '8:00' : 'Omeprazole', '12:00' : 'Atenolol', '5:00' : 'Istamet', '7:30': 'Metformin'} #print (data)

def myOnPublishCallback():

print ( '8:00 : Omeprazole', '12:00 : Atenolol', '5:00 : Istamet', '7:30 : Metformin')

success = deviceCli.publishEvent("Medication", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

print("Not connected to IoTF")

time.sleep(2)

deviceCli.commandCallback = myCommandCallback

#Disconnect the device and application from the cloud

deviceCli.disconnect()