

Personal Assistance For Independent Senior Citizens

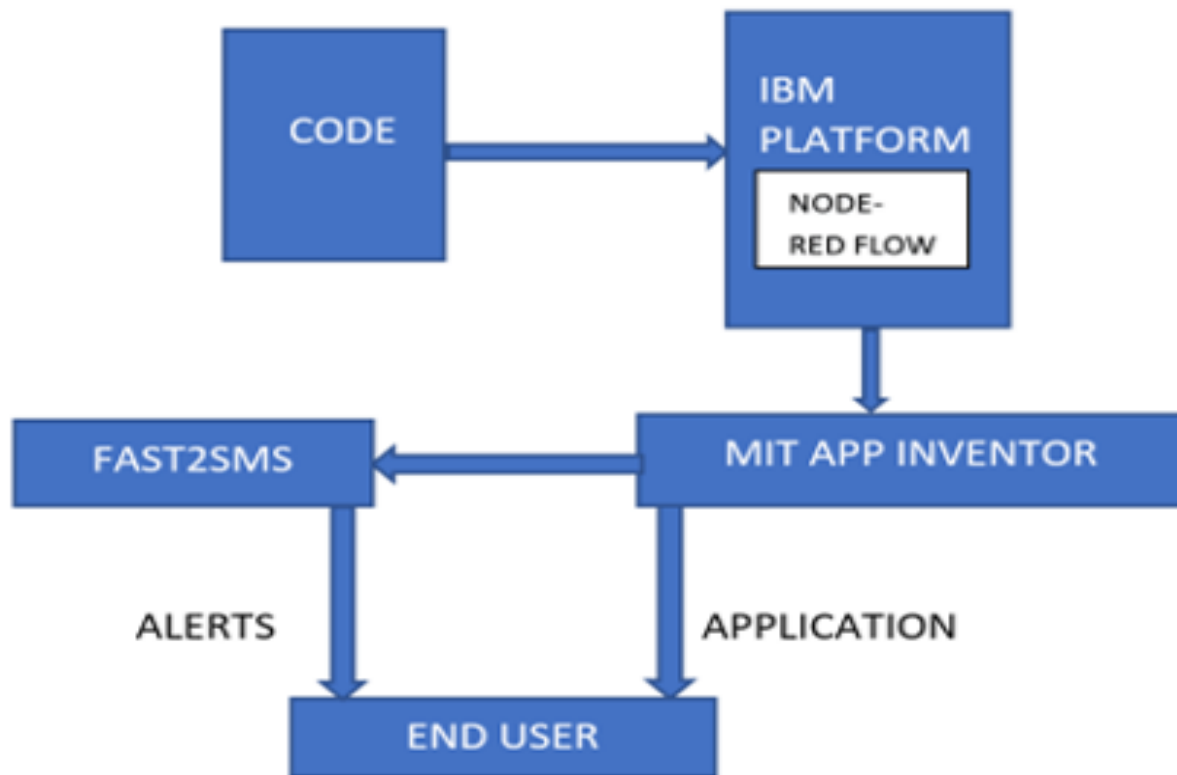
Overview: This project will guide you in developing a personal assistance for independent senior citizens using the IBM cloud services and MIT app inventor under IoT platform.

Purpose: To collect and store the information of the medicines of the user and thereby alerting the user to take medication from time to time without fail through an application.

Existing problem: Many old people forget about their medication either because of aging or memory related issues, which in turn affects their longterm health.

Proposed solution: Using the data we stored, we integrate the IBM NODE RED platform with MIT app inventor and Fast2Sms to send alerts to the user about the medicines.

Block diagram:



Hardware-Software required:

IBM: Acts as a platform for creating services like IoT,speech to text,text to speech etc., and software applications like NODE RED and also for storing and subscribing data.

NODE RED:Software available within the IBM platform and helps us in using the MIT app inventor and also it has the UI option to display the data and required commands in the python.Nodes act as the back-bone here.

Python(IDLE): Helps us in writing the code for sending values to the IBM IoT platform and also for calling back them into the python shell.

Fast2Sms: It helps us in sending the alerts/messages to multiple users at same instance of time.

MIT app inventor: This helps us in building the model apk's/applications to simulate the data from IBM platform and also for sending alerts through Fast2Sms.Blocks acta as a back-bone in this platform.

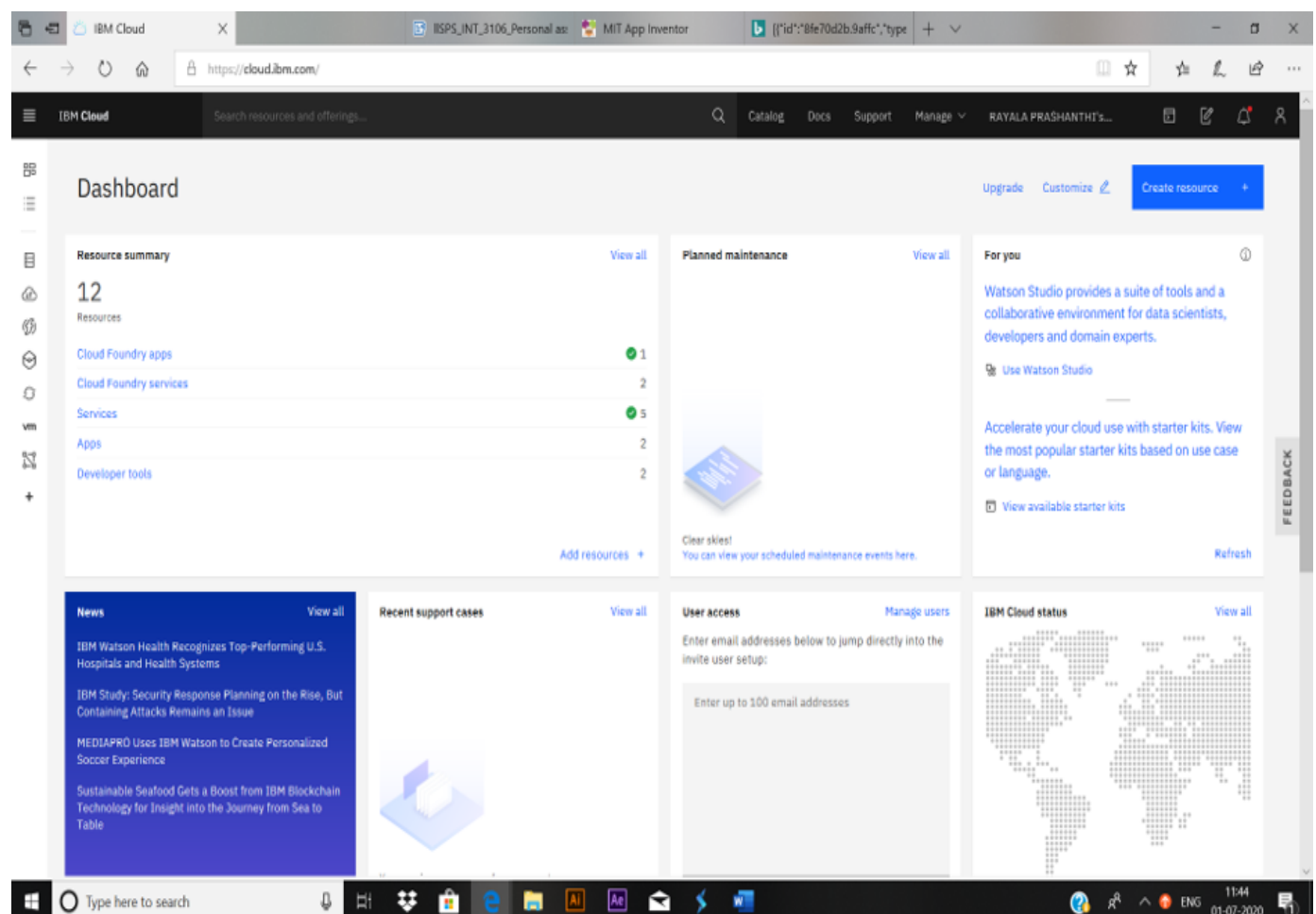
Application:It displays the data related to medication and alerts the user to take the medicine.

EXPERIMENTAL INVESTIGATIONS AND THE PROJECT

IBM is a platform which is used to create several services and softwares.It helps us to store the data and retrieve it whenever required.Initially we should create IBM IoT service, speech to text service and text to speech services in the IBM cloud platform. Using the python code, we need to send the details of time and the medicine to the IBM IoT service by providing the required credentials like device type, device ID, token and the organisation.This data can be seen in the device status in the IoT platform.This data is in the form of JSON.We should also give the text to speech and speech to text python codes in IDLE to understand what actually are those services.Credentials of speech to text and text to speech can be accessed through the services created earlier.NODE-RED is a software inside the IBM platform, which uses nodes to define and run a specified task/process.Initially the data from IBM IoT platform is viewed in the debug section of the node red platform.This can be done using the "IBM IoT IN NODE" and the debug node.We can also create some ui dashboards in the node red to print back the data and alerts into the python codes using buttons.Here, we used the "ALERTS" and "MEDICINES" buttons to get the warning and data into the python code

respectively. Buttons will be displayed in the url/ui page and clicking on them and running the code helps us to visualize data and alerts in the python. Now, an http "get" request is given for visualizing the medicine data into an url. We can now get a json data of medicines in the url using http response node. For understanding, we can also use text to speech nodes to let the node red platform speak out tablet names to just simulate how basically the text to speech service can be used in node red and in general applications. Now, using this url and the blocks provided in the MIT app inventor, we can build an application to see the time and medicine data and also send alerts to the user for using tablets using Fast2SMS Service. Finally, the end user is able to take medicines in time without any fail because of the data available in the application and the alerts we send.

IBM CLOUD PLATFORM :



LIST OF SERVICES AND SOFTWARES USED IN IBM CLOUD :

Resource list

Name	Group	Location	Offering	Status	Tags
Filter by name or IP address... Filter by group or org... Filter... Filter... Filter... Filter...					
Devices (0)					
VPC infrastructure (0)					
Clusters (0)					
Cloud Foundry apps (1)					
Node RED DYYIH	prashanthir2001@gmail.com / dev	London	SDK for Node.js™	Started	-
Cloud Foundry services (2)					
Services (5)					
Continuous Delivery	Default	Dallas	Continuous Delivery	Active	-
Internet of Things Platform-6f	Default	London	Internet of Things Platform	Active	-
Speech to Text-g7	Default	London	Speech to Text	Active	-
Text to Speech-45	Default	London	Text to Speech	Active	-
node-red-dyyih-cloudant-1593500707654	Default	London	Cloudant	Active	-
Storage (0)					
Network (0)					
Cloud Foundry enterprise environments (0)					
Functions namespaces (0)					

IOT PLATFORM :

IBM Watson IoT Platform

Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By
123456	Connected	SAMSUNG	Device	Jun 29, 2020 5:38 PM		prashanthir2001@gmail.com

Items per page: 50 | 1-1 of 1 item

1 of 1 page

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes tabs for 'Service Details - IBM Cloud', 'IBM Watson IoT Platform', 'BSPS_JNT_3106_Personal as', 'MIT App Inventor', and a search bar. The main header displays the user's email 'prashanthir2001@gmail.com' and ID 'nu3zsh'. The dashboard is divided into sections: 'Browse', 'Action', 'Device Types', and 'Interfaces'. The 'All Devices' tab is selected, showing a table of devices. The device with ID 123456 is highlighted, showing its status as 'Connected', device type as 'SAMSUNG', and class ID as 'Device'. Below the table, there is a section for 'Recent Events' showing a stream of data for the device. The events are listed in a table with columns: Event, Value, Format, and Last Received. The events are all of type 'DHT11' and contain JSON data representing temperature and humidity readings.

Event	Value	Format	Last Received
DHT11	{"8:00 ":"0lmat","9:30 ":"Metbd","11:15 ":"Aspiri..."}	json	a few seconds ago
DHT11	{"8:00 ":"0lmat","9:30 ":"Metbd","11:15 ":"Aspiri..."}	json	a few seconds ago
DHT11	{"8:00 ":"0lmat","9:30 ":"Metbd","11:15 ":"Aspiri..."}	json	a few seconds ago
DHT11	{"8:00 ":"0lmat","9:30 ":"Metbd","11:15 ":"Aspiri..."}	json	a few seconds ago

IBM SPEECH TO TEXT SERVICE :

The screenshot shows the IBM Cloud 'Speech to Text-g7' service page. The top navigation bar includes tabs for 'IBM Watson Service Pag', 'IBM Watson IoT Platform', 'BSPS_JNT_3106_Personal as', 'MIT App Inventor', and a search bar. The main header displays the user's email 'RAYALA PRASHANTH's...' and ID 'nu3zsh'. The service page is divided into sections: 'Manage', 'Getting started', 'Service credentials', 'Plan', and 'Connections'. The 'Service credentials' section is active, showing the API key and URL. The 'Plan' section shows the 'Lite' plan with an 'Upgrade' button. The 'Getting started' section includes links for 'Getting started tutorial' and 'API reference'.

Speech to Text-g7 Active Add tags

Plan
Lite
[Upgrade](#)

Credentials

API key:

URL:

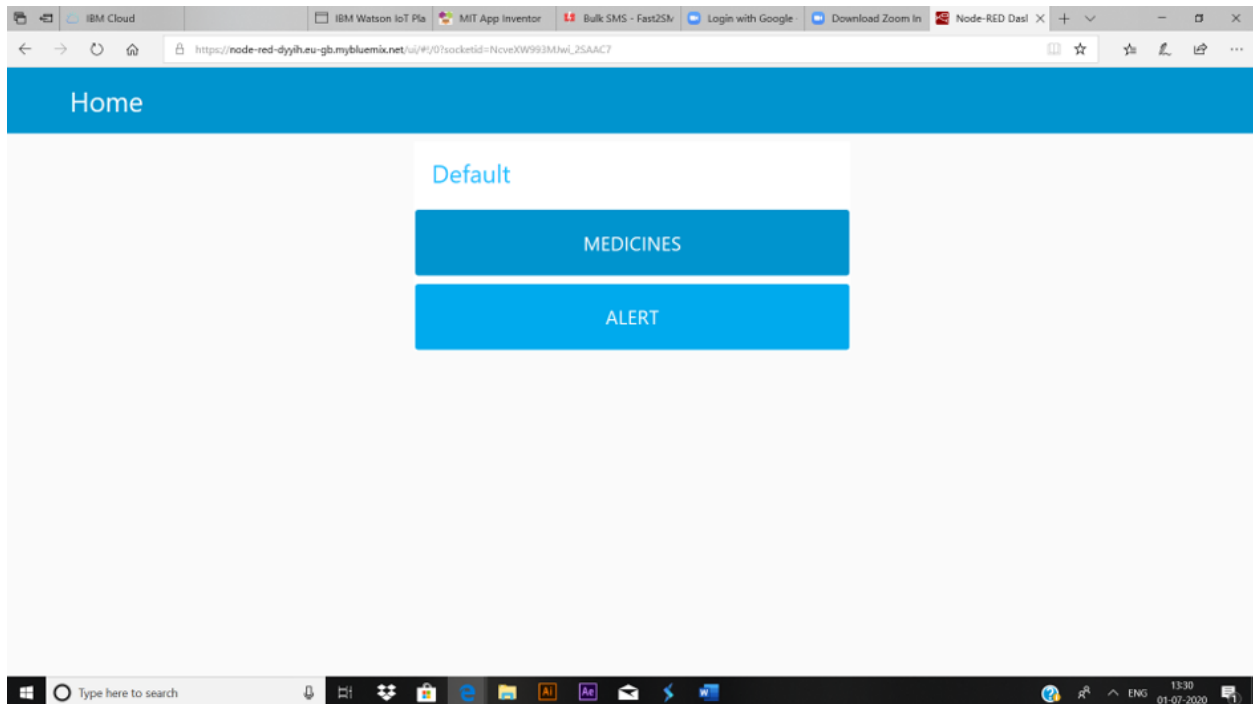
IBM TEXT TO SPEECH SERVICE :

The screenshot shows the IBM Cloud console for the 'Text to Speech-45' service. The page is titled 'Text to Speech-45' and is marked as 'Active'. On the left, there is a 'Manage' sidebar with links for 'Getting started', 'Service credentials', 'Plan', and 'Connections'. The main content area has a 'Start by viewing the tutorial' section with links to 'Getting started tutorial' and 'API reference'. Below this is the 'Credentials' section, which displays the 'API key' and 'URL'. The 'API key' is masked with dots, and the 'URL' is 'https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/aff6277e-aa23-4556-8ded-61d...'. On the right, there is a 'Plan' section showing the 'Lite' plan with an 'Upgrade' button. The bottom of the screen shows a Windows taskbar with the search bar and various application icons.

IBM NODE-RED :

The screenshot shows the IBM Node-RED interface. The main workspace displays a flow named 'Flow 1'. The flow starts with a 'Hello Node-RED' node, followed by a 'msg.payload' node. This is connected to an 'ALERT' node, which then connects to a 'MEDICINES' node. The 'MEDICINES' node is connected to a 'MEDICINE DETAILS' node, which is then connected to an 'http' node. The 'http' node is connected to an 'inject' node, which is then connected to a 'text to speech' node. Finally, the 'text to speech' node is connected to a 'play audio' node. The left sidebar shows a list of nodes, including 'delay', 'trigger', 'OpenWhisk', 'rbe', 'msg in', 'msg out', 'http in', 'http response', 'http request', 'websocket in', 'websocket out', 'tcp in', 'tcp out', 'tcp request', and 'udp in'. The right sidebar shows a 'debug' console with a log of messages, including timestamps and payloads. The bottom of the screen shows a Windows taskbar with the search bar and various application icons.

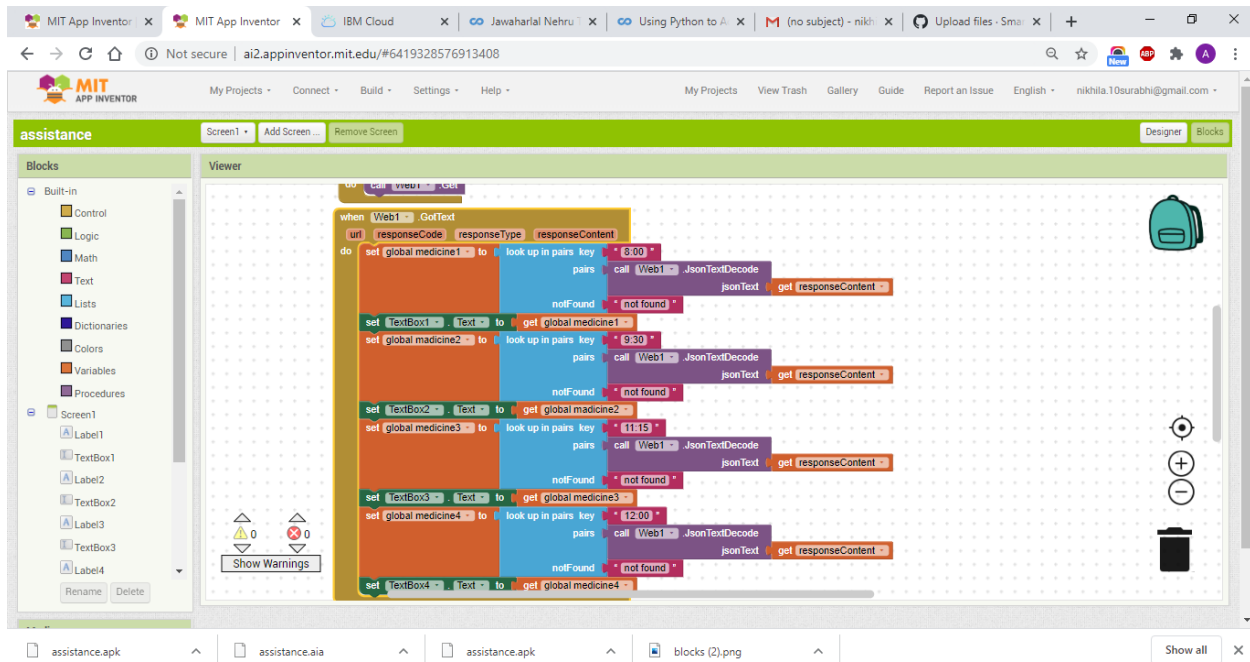
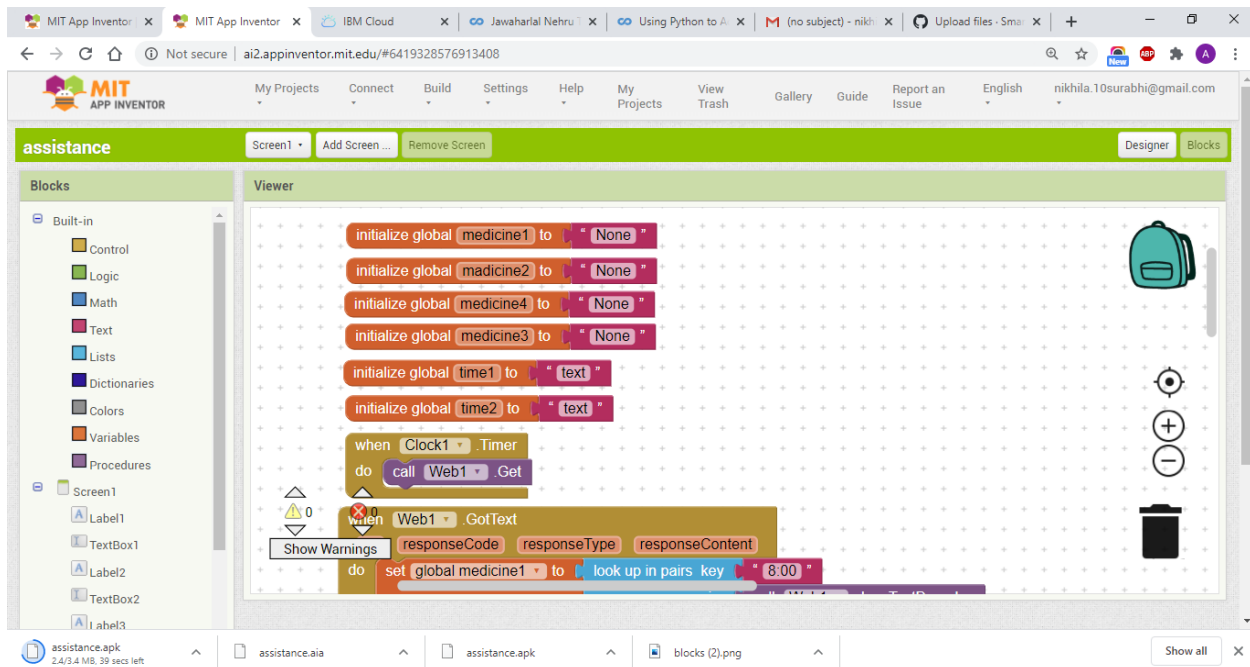
IBM NODE RED UI PLATFORM :



IBM NODE HTTP DATA WINDOW :



MIT APP INVENTOR :



MIT App Inventor x MIT App Inventor x IBM Cloud x Jawaharlal Nehru x Using Python to A x (no subject) - nikh x Upload files - Sma x + -

Not secure | ai2.appinventor.mit.edu/#6419328576913408

MIT APP INVENTOR My Projects Connect Build Settings Help My Projects View Trash Gallery Guide Report an Issue English nikhila.10surabhi@gmail.com

assistance Screen1 Add Screen Remove Screen Designer Blocks

Blocks

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - TextBox1
 - Label2
 - TextBox2
 - Label3
 - TextBox3
 - Label4

Rename Delete

Viewer

when Clock2.Timer

do

- set global time1 to call Clock2.Now
- set global time2 to join call Clock2.Hour instant get global time1
- call Clock2.Minute instant get global time1
- set TextBox5.Text to get global time2
- if get global time2 = 6:00 or get global time2 = 9:30 or get global time2 = 11:15 or get global time2 =
- then set Web2.Uri to https://www.fast2sms.com/dev/bulk?authorization= call Web2.Get

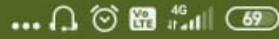
Show Warnings

0 0

blocks (2).png

Show all

11:48 PM



Screen1

8:00

Olmat

9:30

Metxl

11:15

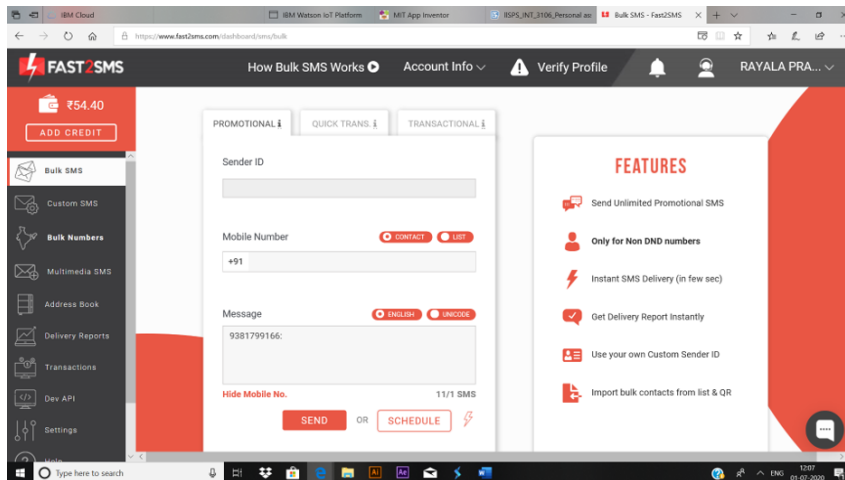
Aspirin

12:00

Metformin

23:48

FAST2SMS :



FLOW CHART :



RESULT :

Hence, using the data of the medicines provided, the timings of the medication will be provided to the user and alerts will be sent to the mobile using fast2sms. Thereby it ensures the safe medication and timely medication to the individual senior citizen.

ADVANTAGES :

- >> Ensures the proper health of the elderly people.
- >> It also helps the elderly users who have less memory power by creating alerts/reminders and makes them self dependent.

DIS-ADVANTAGES :

- >> Internet availability or app working may be a problem in some of the remote areas.
- >> Wrong feeding of the data affects the user's health.

APPLICATIONS :

- >> In old age homes
- >> For all the elderly people who either live individually or with their children.
- >> Also for people with little memory problems.

CONCLUSION :

The Personal Assistance system ensures the proper health of the individual senior citizens and also for the people who need to take medication (and are suffering with memory problems). Though there are some disadvantages with this system, it still has the ability to properly ensure proper care of senior citizens.

FUTURE SCOPE:

Elderly people play a very important role in the development of society. Though, they cannot contribute much financially, 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. Main need for any elderly citizen is "HEALTH". So, this "PERSONAL ASSISTANT FOR INDIVIDUAL SENIOR CITIZENS APPLICATION" has an ability of playing a major role in maintaining elderly people's health. Every person on this earth, have a right to live. So, by adapting this system in a larger scale ensures the proper development of ethical values in future generations and in increasing the life span of elderly people and thereby decreasing the mortality rate.

BIBLIOGRAPHY :

Some of the websites like cloud.ibm.com, www.fast2sms.com and appinventor.mit.edu were used.

Also the readily available "PYTHON IDLE" was used for the coding part in the project.

PROJECT DELIVERABLES :

WEB APPLICATION

PROJECT REPORT

PROJECT VIDEO

"RESPECT AND SERVE ELDER PEOPLE WITH ALL FACILITIES TO ENSURE YOUR ETHICS"

APPENDIX :

CODE FOR TEXT TO SPEECH :

```
from ibm_watson import TextToSpeechV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

from playsound import playsound


authenticator =
IAMAuthenticator('yci1-exDJ1UrPVWahNI478_0qq3G_9D5XUagkYUqjfCj')
text_to_speech = TextToSpeechV1(
    authenticator=authenticator
)


text_to_speech.set_service_url('https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/f4fc8363-62dc-4947-9293-9822fdb56590')


with open('medicine.mp3', 'wb') as audio_file:
    audio_file.write(
        text_to_speech.synthesize(
            'OImat ',
            voice='en-US_AllisonV3Voice',
            accept='audio/mp3'
        ).get_result().content)


audio_file.write(
    text_to_speech.synthesize(
        ' Met xl',
        voice='en-US_AllisonV3Voice',
        accept='audio/mp3'
    ).get_result().content)
audio_file.write(
    text_to_speech.synthesize(
```

```

        'Aspirin ',
        voice='en-US_AllisonV3Voice',
        accept='audio/mp3'
    ).get_result().content)
audio_file.write(
    text_to_speech.synthesize(
        'Metformin ',
        voice='en-US_AllisonV3Voice',
        accept='audio/mp3'
    ).get_result().content)
audio_file.write(
    text_to_speech.synthesize(
        'Linagliptin',
        voice='en-US_AllisonV3Voice',
        accept='audio/mp3'
    ).get_result().content)

playsound('medicine.mp3')

```

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('DhIN3LWVXzaNTbEZ1X2Shm12o12-bxUSNZgCQ3QbfVPI')
speech_to_text = SpeechToTextV1(
    authenticator=authenticator
)

```

```
speech_to_text.set_service_url('https://api.eu-gb.speech-to-text.watson.cloud.ibm.com/instances/d8ae2efd-eedd-4749-a50d-e7a35b49e52e')
```

```
with open(join(dirname(__file__), './.', 'medicine.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))
```

FINAL CODE:

```
import time  
import sys  
import ibmiotf.application  
import ibmiotf.device  
import random  
#Provide your IBM Watson Device Credentials  
organization = "nu3zxh"  
deviceType = "SAMSUNG"  
deviceId = "123456"  
authMethod = "token"  
authToken = "99121499"  
  
# Initialize GPIO  
  
def myCommandCallback(cmd):  
    if cmd.data:  
        print("Time to take medicine")#for commands  
  
try:
```



```

    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
"auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)#create client
    #.....

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()#connect client to platform

while True:

    data = { '8:00 ' : 'Olmat' , '9:30 ' : 'Metxl' , '11:15 ' : 'Aspirin' , '12:00 ' : 'Metformin' }
    #print (data)
    def myOnPublishCallback():
        print ('8:00 : Olmat' , '9:30 : Metxl' , '11:15 : Aspirin' , '12:00 : Metformin' )
        success = deviceCli.publishEvent("DHT11", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoT")
            time.sleep(2)

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()

```

THE END
