

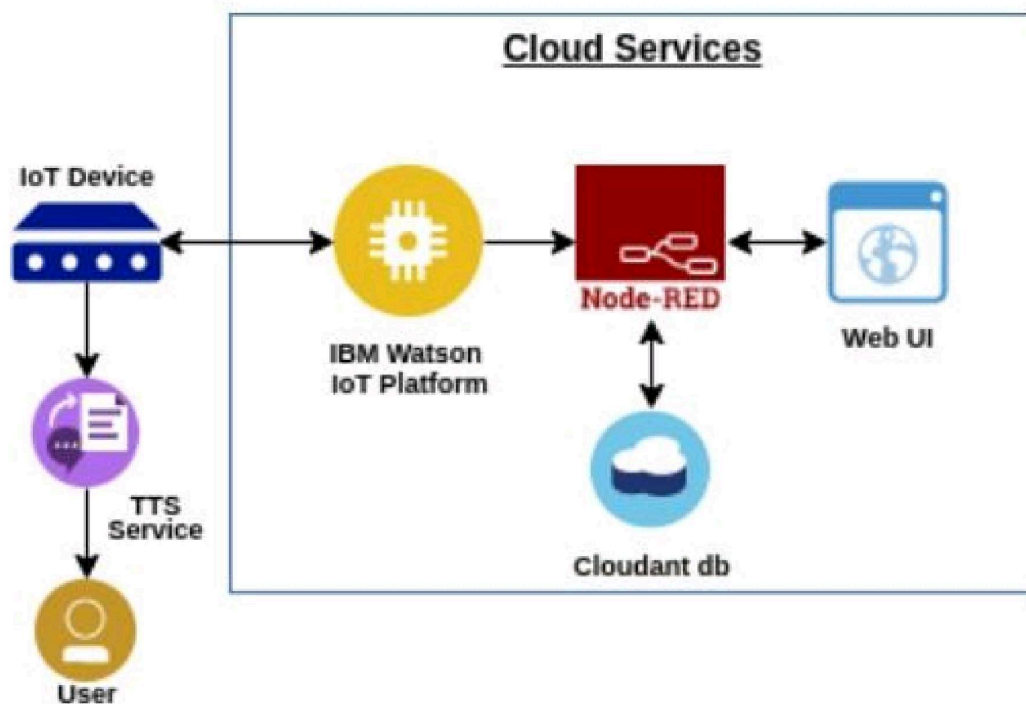
## Final Deliverables

Team ID	PNT2022TMID52021
Project Name	Project - Personal Assistance for Seniors who Are Self Reliant

### OBJECTIVE:

- Sometimes elderly people forget to take their medicine at the correct time.
- They also forget which medicine He / She should take at that particular time.
- And it is difficult for doctors/caretakers to monitor the patients around the clock. To avoid this problem, this medicine reminder system is developed.
- An app is built for the user (caretaker) which enables him to set the desired time and medicine. These details will be stored in the IBM Cloudant DB.
- If the medicine time arrives the web application will send the medicine name to the IoT Device through the IBM IoT platform.
- The device will receive the medicine name and notify the user with voice commands.

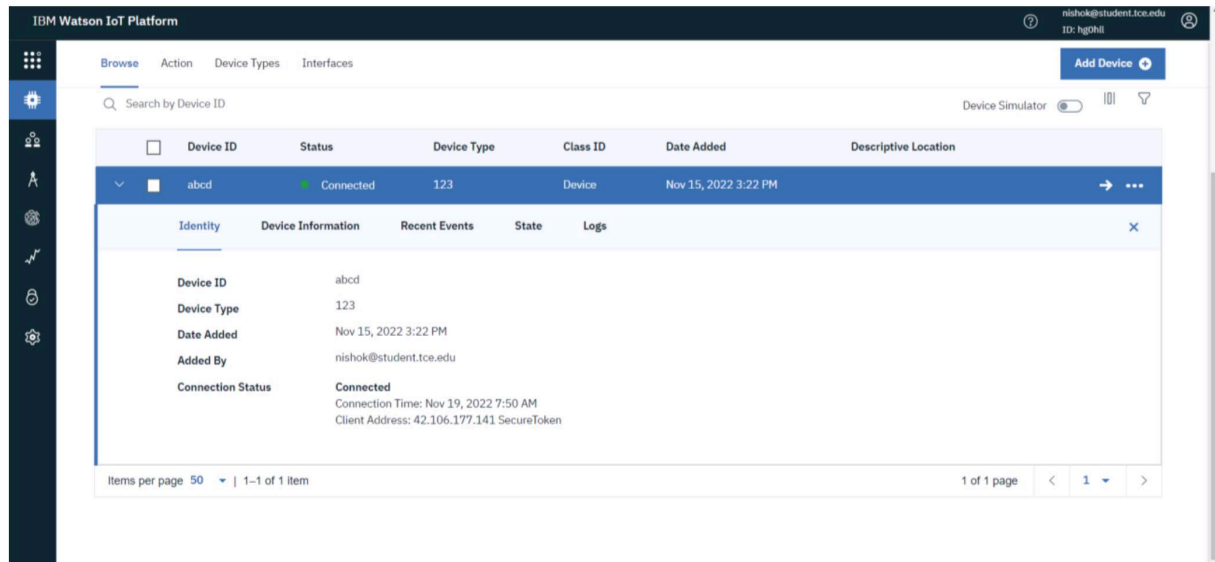
### FLOW OF THE PROJECT:



## WAYS ACHIEVES THE PROJECT FLOW:

### Step1:

Create an IBM Watson Device and note down the credentials, after that create a App “Standard App” and node down the API key and Token.

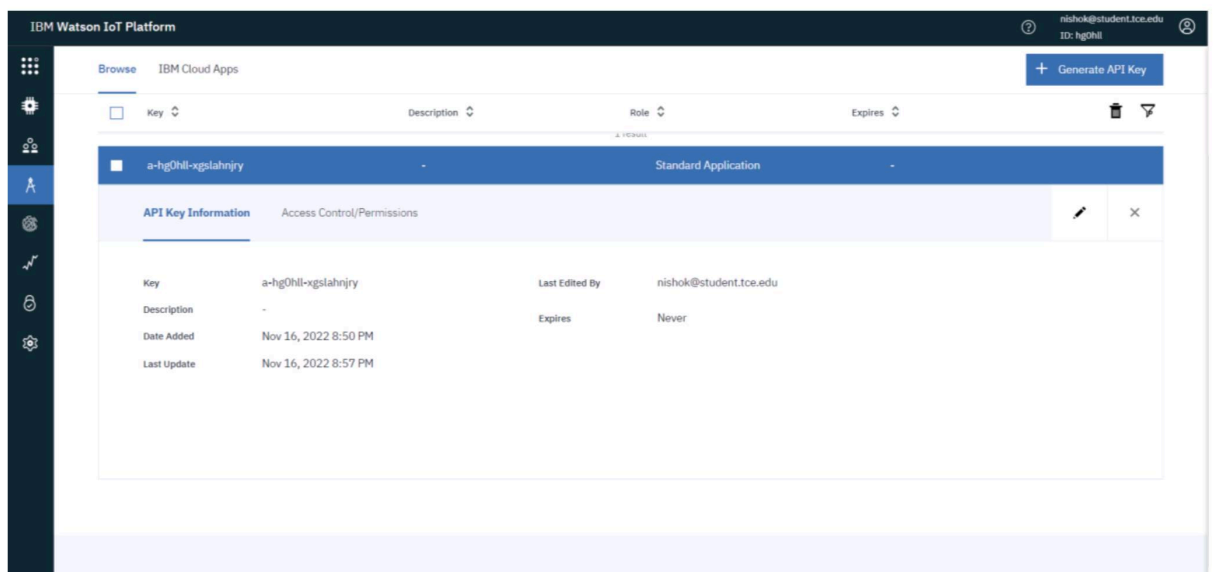


The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. The main content area displays a table with columns: Device ID, Status, Device Type, Class ID, Date Added, and Descriptive Location. A device with ID 'abcd' is listed as 'Connected' with Device Type '123'. Below the table, a detailed view for the selected device is shown, including fields for Device ID, Device Type, Date Added, Added By, and Connection Status. The connection status is 'Connected' with details: Connection Time: Nov 19, 2022 7:50 AM and Client Address: 42.106.177.141 SecureToken.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
abcd	Connected	123	Device	Nov 15, 2022 3:22 PM	

**Device Details:**

- Device ID: abcd
- Device Type: 123
- Date Added: Nov 15, 2022 3:22 PM
- Added By: nishok@student.tce.edu
- Connection Status: **Connected**  
Connection Time: Nov 19, 2022 7:50 AM  
Client Address: 42.106.177.141 SecureToken



The screenshot shows the IBM Watson IoT Platform interface for managing API keys. The top navigation bar includes 'Browse', 'IBM Cloud Apps', and a 'Generate API Key' button. The main content area displays a table with columns: Key, Description, Role, and Expires. An API key with ID 'a-hg0Hll-xgslahnjry' is listed as a 'Standard Application'. Below the table, a detailed view for the selected API key is shown, including fields for Key, Description, Date Added, Last Update, Last Edited By, and Expires.

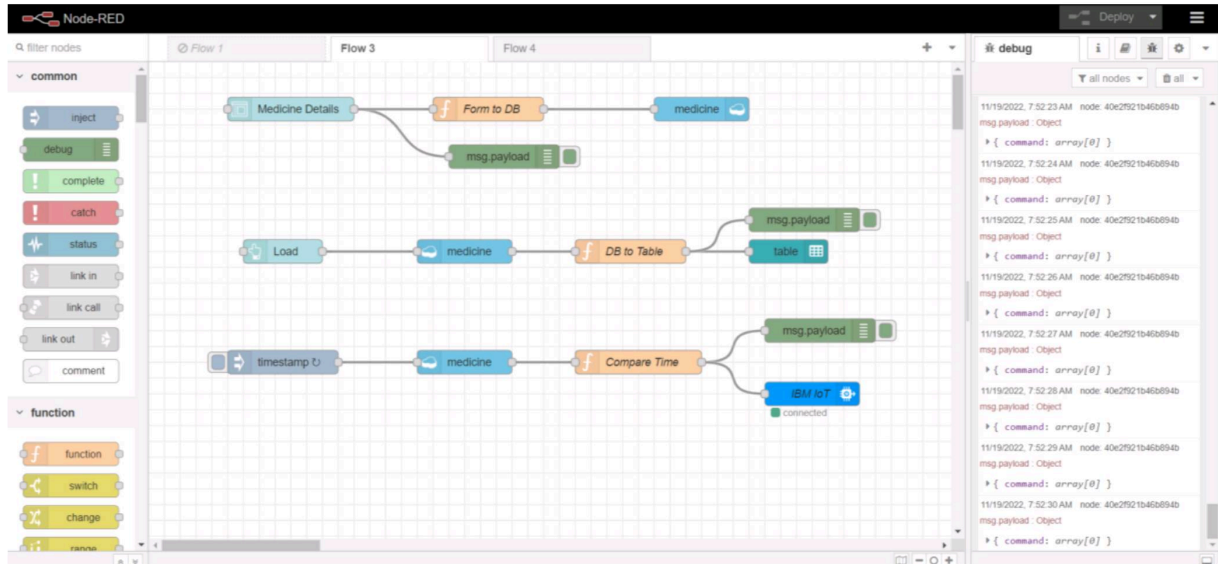
Key	Description	Role	Expires
a-hg0Hll-xgslahnjry	-	Standard Application	-

**API Key Information:**

- Key: a-hg0Hll-xgslahnjry
- Description: -
- Date Added: Nov 16, 2022 8:50 PM
- Last Update: Nov 16, 2022 8:57 PM
- Last Edited By: nishok@student.tce.edu
- Expires: Never

## Step 2:

Go to node red flow editor and create nodes for the project.



Nodes we use for this project:

1. Form
2. Function
3. Cloudant out
4. Button
5. Cloudant In
6. Table UI
7. IBM IoT Out
8. Inject Node
9. Debug Node

### 1. Form Node:

Drag “Form node” from dashboard nodes and create the required fields and name the node.

**Edit form node**

Delete Cancel Done

**Properties**

Group: [Remainder] Medicien

Size: auto

Label: Medicine Details

Label	Name	Type	Required	UIRows	Remove
Tablet	Tablet	Text	<input checked="" type="checkbox"/>		
Hour	Hour	Number	<input checked="" type="checkbox"/>		
Min	Min	Number	<input checked="" type="checkbox"/>		

+ element

Buttons: submit cancel

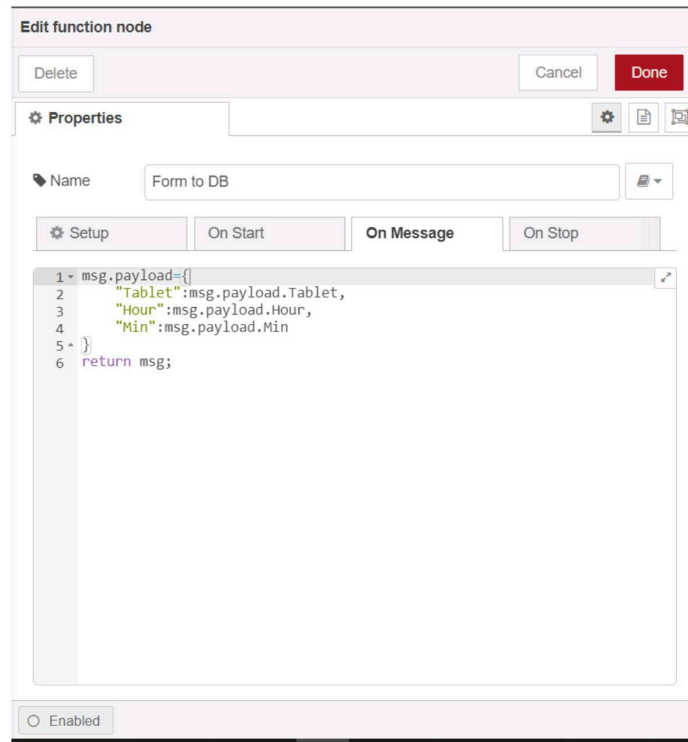
### 2. Function Node:

We created three different function nodes for three different functions. Drag “Function Node” below the function nodes.

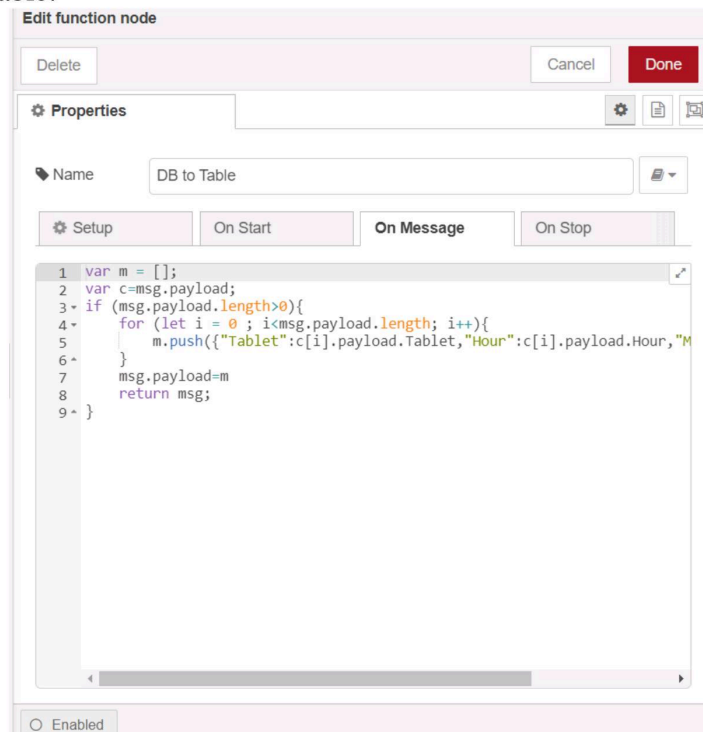
Function;

- Form to DB
- DB to Table
- Compare Time

## 2.1. Form to DB:



## 2.2.DB to Table:



### 2.3. Compare Time:

**Edit function node**

Delete Cancel Done

**Properties**

Name Compare Time

Setup On Start On Message On Stop

```
1 var date1 = new Date();
2 var hour=date1.getUTCHours();
3 var min=date1.getUTCMinutes();
4 var sec=date1.getUTCSeconds();
5 hour=hour+5
6 min=min+30
7 if(min>60){
8   hour=hour+1
9   min=min-60
10 }
11
12 var c=msg.payload;
13 var m = [];
14 for (let i = 0 ; i<msg.payload.length; i++){
15   if(hour===c[i].payload.Hour){
16     if(min===c[i].payload.Min){
17       if(sec===0){
18         m.push(c[i].payload.Tablet);
19       }
20     }
21   }
22 }
23 msg.payload={"command":m};
24 return msg
```

Enabled

### 3. Cloudant Out Node:

Drag “Cloudant Out Node” from storage nodes, the required credentials and in the database give any name that you need to create. This is use to store data in a database.

**Edit cloudant out node**

Delete Cancel Done

**Properties**

Service node-red-zhhd-2022--cloudant-166635985f

Database medicine

Operation insert

☐ Only store msg.payload object?

Name Name

Enabled

4. Button Node:

Drag “Button node” from dashboard nodes and name the node as “Load”. Button is use to trigger the process of loading the data that stored in the database to table.

5. Cloudant In Node:

Drag “Cloudant in Node” from the storage nodes, which is use to retrieve the data that stored in the database. enter the credentials and name database you need to access.

The screenshot shows the 'Edit cloudant in node' configuration window. At the top, there are buttons for 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' section with a gear icon and three sub-sections: 'Service' with a dropdown menu showing 'node-red-zhhd-2022--cloudant-1666359856', 'Database' with a text input field containing 'medicine', and 'Search by' with a dropdown menu showing 'all documents'. There is also a 'Name' field with the text 'Name'. At the bottom left, there is a checkbox labeled 'Enabled' which is currently unchecked.

6. Table UI Node:

Drag “Table UI Node” from Dashboard Nodes and name the table. The table is use to see the loaded medicine in the database by the user it seen in the user IU dashboard.

The screenshot shows the 'Edit table node' configuration window. At the top, there are buttons for 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' section with a gear icon and four sub-sections: 'Group' with a dropdown menu showing '[Remainder] Tablet List', 'Size' with a text input field containing 'auto', 'Name' with a text input field containing 'Name', and 'Columns' with a large empty text area. There is also a checkbox labeled 'Send data on click' which is currently unchecked. At the bottom left, there is a checkbox labeled 'Enabled' which is currently unchecked.

7. IBM IoT Out Node:

Drag “IBM IoT Out Node” from the Output Nodes, which is used to send the name of the medicine that need to take now to the device.

Enter the following details,

- a) IBM IoT App API Key
- b) IBM IoT App Token
- c) IBM IoT Device Type
- d) IBM IoT Device ID
- e) Output Type as Command
- f) Command Type as cmd
- g) Format as json
- h) Data as data

The screenshot shows the 'Edit ibmiot out node' configuration window. The window has a title bar 'Edit ibmiot out node' and buttons for 'Delete', 'Cancel', and 'Done'. Below the title bar is a 'Properties' section with various configuration fields. The fields are: Authentication (API Key), API Key (medicine), Output Type (Device Command), Device Type (123), Device Id (abcd), Command Type (cmd), Format (json), Data (data), QoS (0), Name (IBM IoT), and Service (registered). At the bottom, there is a checkbox labeled 'Enabled' which is currently unchecked.

8. Inject node:

Drag “Inject Node” from Common nodes. Which is used for inject the time stamp every second for retrieve the data from database and compare the data and the current time.

9. Debug node:

Drag “Debug Node” rom Common Nodes. Which is used to view the payloads.





#### Step 4:

Write a python script to connect with IBM IoT device and receiving the medicine name and convert the text to speech using the IBM Text To Speech and Play it using the pygame module of python.

##### 1. Library Used in code:

- a. time
- b. ibm\_watson TextToSpeech
- c. ibm\_cloud\_sdk\_core.authenticators
- d. ibmiotf.device
- e. pygame

##### 2. Code for Connect with IBM Watson IoT device and retrieve data.

```
import ibmiotf.device

config={
    "org":"hg0hl1",           # Device Organization
    "type": "123",            # Device Type
    "id":"abcd",              # Device ID
    "auth-method":"token",    # Device Authentication Method
    "auth-token":"123456789"  # Device Authentication Token
}
client= ibmiotf.device.Client (config) # Save the device Config in a Variable called client
client.connect()                    # Connect with the Device
client.disconnect()                # Disconnect the Device


# callback from device
def myCommandCallback (cmd):
    a=cmd.data
    if len(a["command"])==0:
        pass
    else:
        print(a["command"])

# publish Event to device
def pub (data):
    client.publishEvent (event="status", msgFormat="json",data=data, qos=0)
    print("Published data Successfully: %s",data)

while True:
    s=random.randint(0,100)
    h=random.randint(0,100)
    t=random.randint(0,100)
    data={"sm":s,"hum":h,"temp":t}
    pub(data)
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect()
|
```

### 3. Code For Speech to Text Convention:

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

r1="https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/8e5bc662-02f5-4cc3-b2a3-27086673e789" # TextToSpeech URL Link
api="QGxbVq1lTgSFNn8_7wpT1kGVYIKCHG8NLfHnC1BBXNwj" # TextToSpeech API Key

# Load TextToSpeech API Key and URL
auth=IAMAuthenticator(api)
tts=TextToSpeechV1(authenticator=auth)
tts.set_service_url(url)

# Text To Speech Convention
instruction="Hi Every One."
with open("./speech.wav","wb") as audio_file:
    res=tts.synthesize(instruction,accept="audio/mp3",voice='en-US_AllisonV3Voice').get_result()
    audio_file.write(res.content)
```

### 4. Code for Play audio file 3 time with time interval 20 sec:

```
import pygame
import time

pygame.init() # initiate pygame

p=pygame.mixer.Sound("Speech.wav") # Load audio file

pygame.mixer.Sound.play(p)
time.sleep(20)
pygame.mixer.Sound.play(p)
time.sleep(20)
pygame.mixer.Sound.play(p)
time.sleep(20)
```

## Step 5:

### Complete code for Project:

```
import time
#import ibmiotf.application
from ibm_watson import TextToSpeechV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
import ibmiotf.device
import pygame
pygame.init() # initiate pygame

config={
    "org":"hg0h11",          # Device Organization
    "type": "123",           # Device Type
    "id":"abcd",             # Device ID
    "auth-method":"token",   # Device Authentication Method
    "auth-token":"123456789" # Device Authentication Token
}
url="https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/8e5bc662-02f5-4cc3-b2a3-27086673e789" # TextToSpeech URL Link
api="QGxbVq1LTgSFN8_7wpT1kGVYIKCHG8NLFHnC1BBXNwj" # TextToSpeech API Key
client= ibmiotf.device.Client (config) # Save the device Config in a Variable called client
client.connect() # Connect with the device

# Load TextToSpeech API Key and URL
auth=IAMAuthenticator(api)
tts=TextToSpeechV1(authenticator=auth)
tts.set_service_url(url)

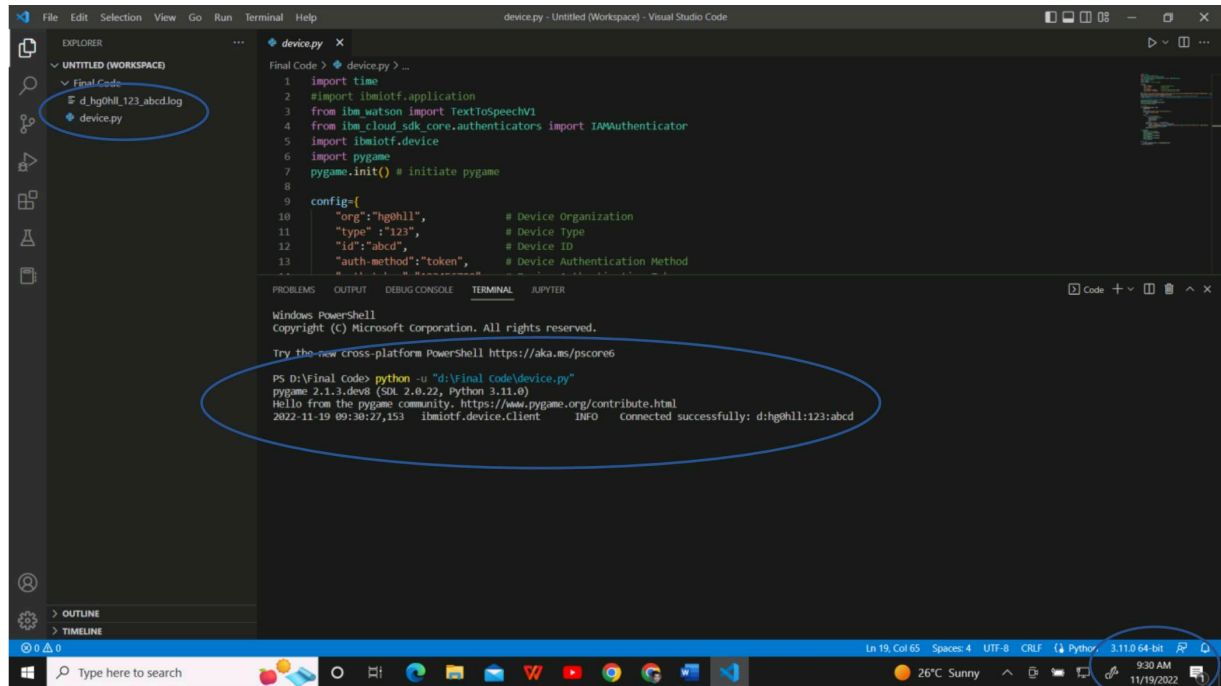
# callback and
def myCommandCallback (cmd):
    a=cmd.data
    c=1
    instruction="Please Take following Medicine. "
    if len(a["command"])==0:
        pass
    else:
        for i in a["command"]:
            instruction+=str(c)+". "
            instruction+=i
            instruction+=". "
            c+=1
        print("Instruction : ",instruction)
        with open("./speech.wav","wb") as audio_file:
            res=tts.synthesize(instruction,accept="audio/mp3",voice='en-US_AllisonExpressive').get_result()
            audio_file.write(res.content)
        play("speech.wav")

def play(a):
    p=pygame.mixer.Sound(a)
    pygame.mixer.Sound.play(p)
    time.sleep(20)
    pygame.mixer.Sound.play(p)
    time.sleep(20)
    pygame.mixer.Sound.play(p)
    time.sleep(20)

while True:
    client.commandCallback = myCommandCallback
client.disconnect()
|
```

## RESULT:

### 1. Connect with the device.



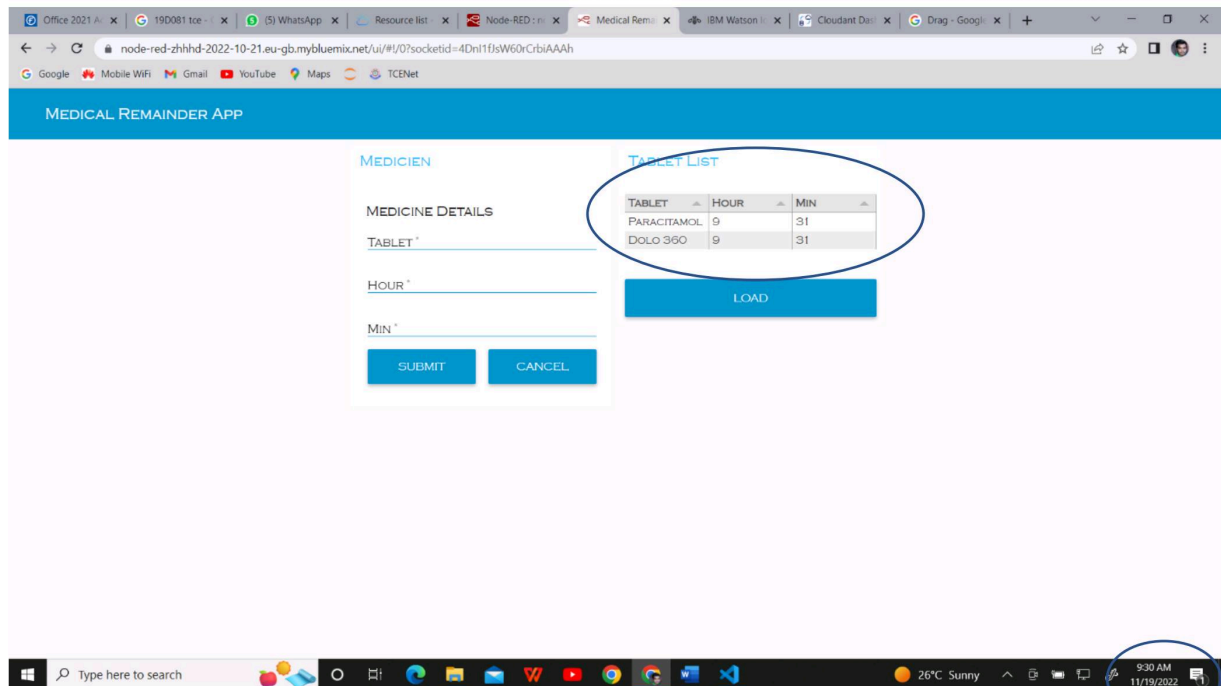
```
Final Code > device.py > ...
1 import time
2 #import ibmiotf.application
3 from ibm_watson import TextToSpeechV1
4 from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
5 import ibmiotf.device
6 import pygame
7 pygame.init() # initiate pygame
8
9 config={
10     "org":"hgohll",          # Device Organization
11     "type":"123",           # Device Type
12     "id":"abcd",           # Device ID
13     "auth-method":"token",  # Device Authentication Method
14 }
15
16 # Create an instance of the TextToSpeechV1 class
17 tts = TextToSpeechV1(authenticator=IAMAuthenticator(''), service_url='https://gateway.watsonplatform.net/text-to-speech/api/v1')
18
19 # Create an instance of the Device class
20 device = ibmiotf.device.Device(config, tts)
21
22 # Connect to the device
23 device.connect()
24
25 # Start the device
26 device.start()
```

Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
Try the new cross-platform PowerShell <https://aka.ms/pscore6>

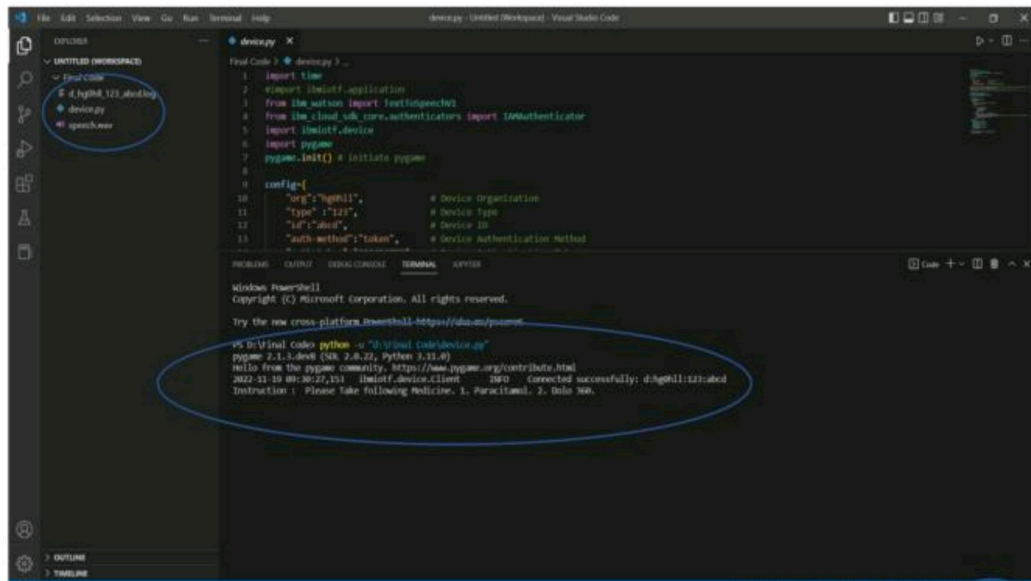
PS D:\Final code> python -u "d:\Final code\device.py"  
pygame 2.1.3.dev0 (SDL 2.0.22, Python 3.11.0)  
Hello from the pygame community. https://www.pygame.org/contribute.html  
2022-11-19 09:30:27,153 ibmiotf.device.client INFO connected successfully: d:hgohll:123:abcd

### 2. Load Tablet name and time in User UI

**We load Two tablet Paracetamol and Dolo 650 set remainder and time 9.31 Am**



3. At 9.31 Am data saved in the DB is received and the audio file for instructions is generated and voice command was given to the user.



```
1 import time
2 import ibmiotf.application
3 from ibm_mqtt import mqttclient
4 from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
5 import ibmiotf.device
6 import pygama
7 pygama.init() # initiate pygama
8
9 config={
10     "org": "hgbl123", # Device Organization
11     "type": "123", # Device Type
12     "id": "abc", # Device ID
13     "auth-method": "token", # Device Authentication Method
14 }
15
16 # Create a new device
17 device = ibmiotf.device.Device(config)
18
19 # Create a new application
20 app = ibmiotf.application.Application(device)
21
22 # Create a new MQTT client
23 client = mqttclient.MQTTClient(device)
24
25 # Create a new Pygama instance
26 pygama = pygama.Pygama(device, app, client)
```

Windows PowerShell  
Copyright (c) Microsoft Corporation. All rights reserved.  
Try the new cross-platform PowerShell <https://aka.ms/powershell>

PS C:\> python -u "D:\Visual Code\device.py"  
pygama 2.1.1-dev (SR 2.0.22, Python 3.11.0)  
Hello from the pygama community. <https://www.pygama.org/contribute.html>  
2022-11-15 09:30:27.151 - ibmiotf.device:client - INFO - Connected successfully: 4.hgbl123:abc  
Instruction : Please Take following Medicine. 1. Paracetamol. 2. Dolo 665.

**GIT HUB :** <https://github.com/IBM-EPBL/IBM-Project-43906-1660720526>

## CONCLUSION:

The Objects are achieved and the data flow is constructed as per the project flow mentioned in the Smartintenz Guided project.