

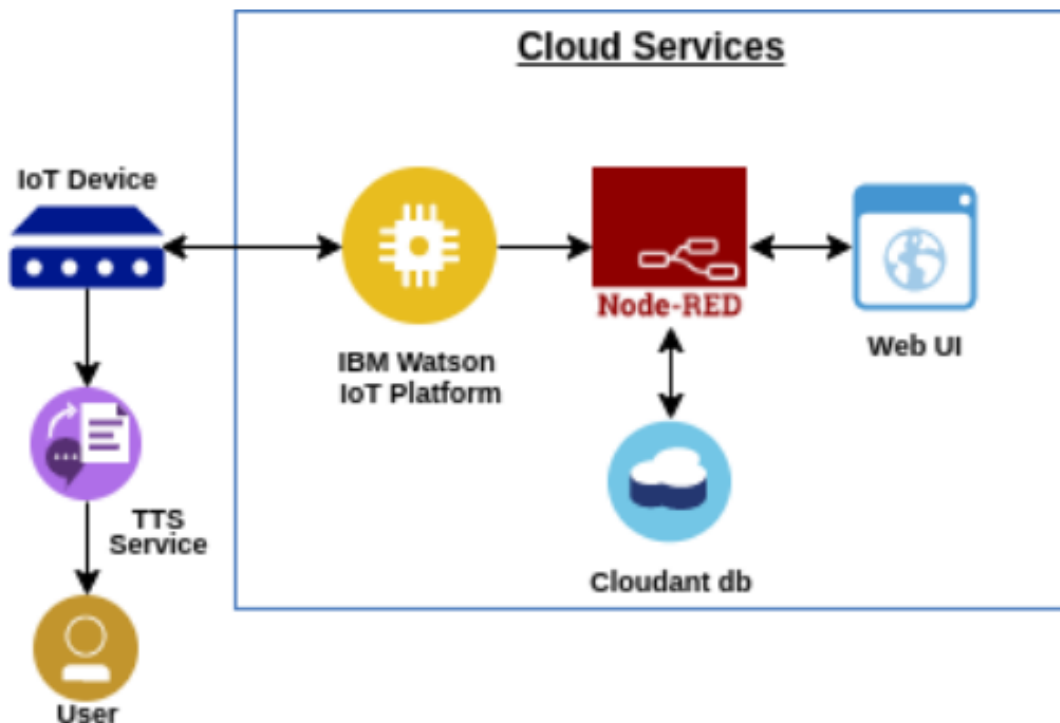
Final Deliverables

Team ID	PNT2022TMID21393
Project Name	Personal Assistance for Seniors Who Are Self-Reliant.
Team Members	Nishok Rajan P (917719D060) Karthikeyan B (917719D036) Saravanaprakash S (917719D081) Viswa Bharathi K (917719D112)

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.

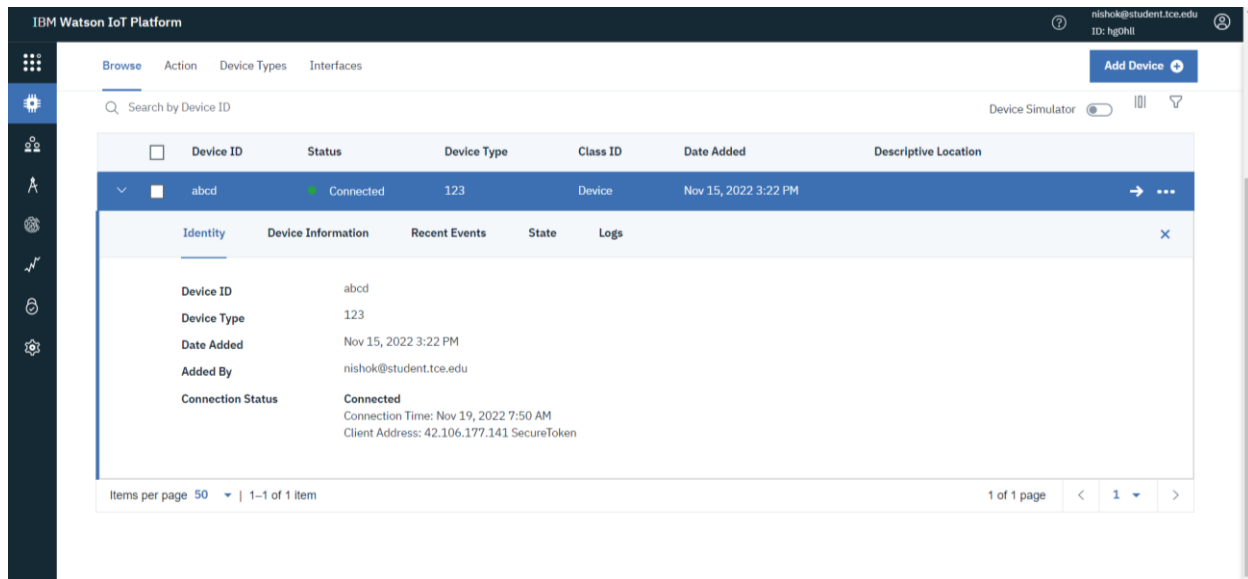
FLOW OF THE PROJECT:



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.

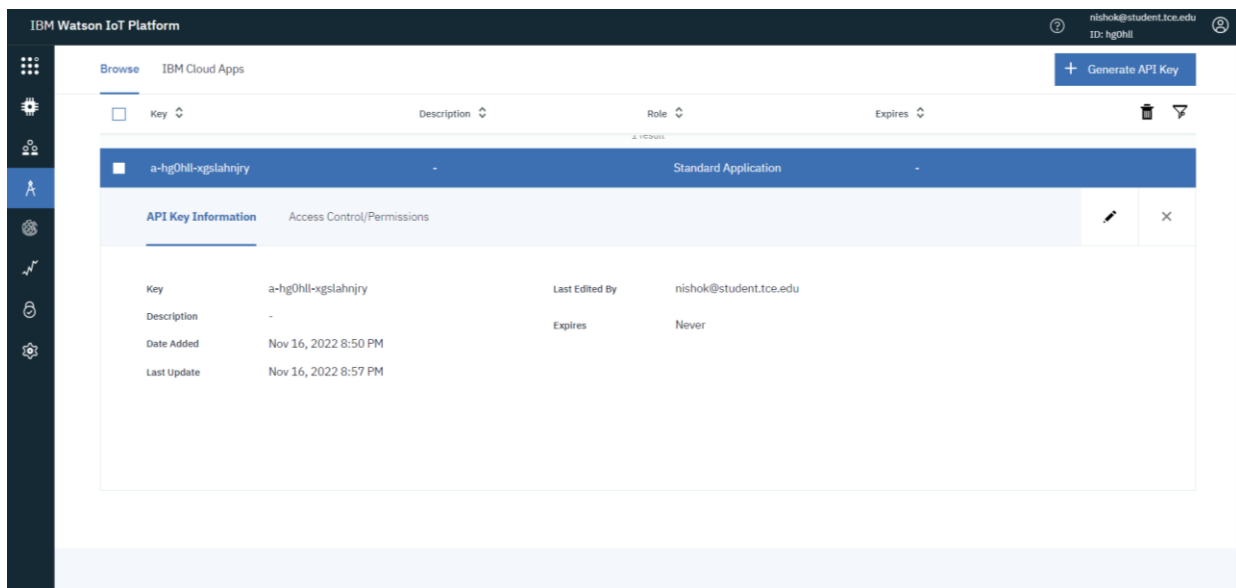


The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. A table lists devices, with one device 'abcd' highlighted. The device status is 'Connected', and it was added on 'Nov 15, 2022 3:22 PM'. A detailed view of the device is shown below the table, including its ID, type, date added, and connection status.

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

Device Information

- 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. A table lists API keys, with one key 'a-hg0hli-xgslahnjry' highlighted. The key is a 'Standard Application' and has no expiration date. A detailed view of the API key is shown below the table, including its key, description, date added, and last update.

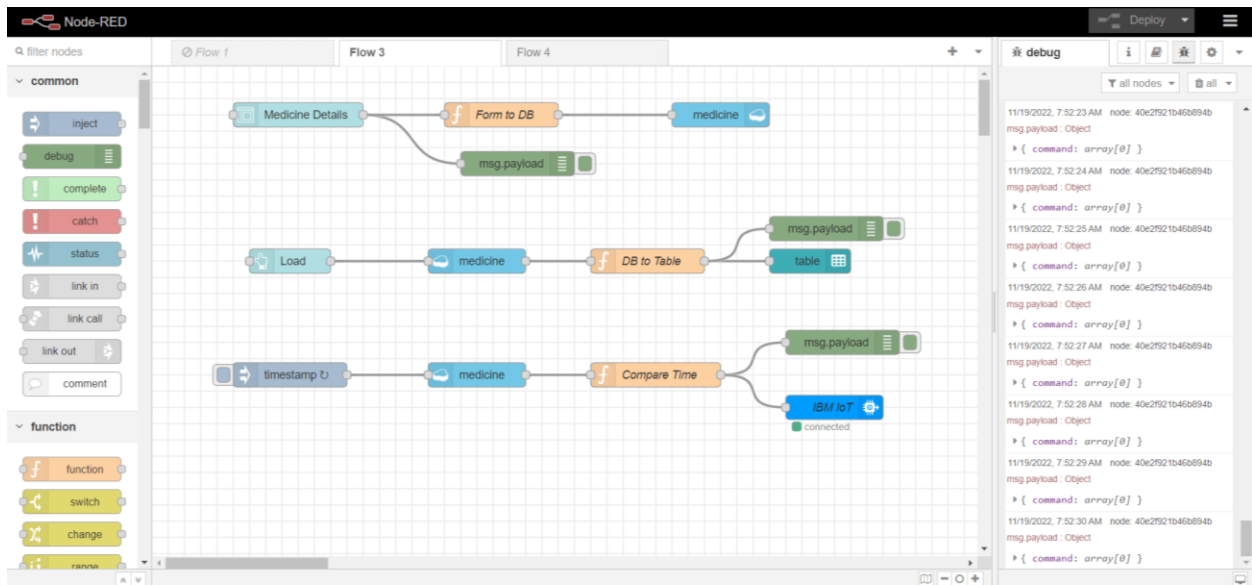
Key	Description	Role	Expires
a-hg0hli-xgslahnjry	-	Standard Application	-

API Key Information

- Key: a-hg0hli-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

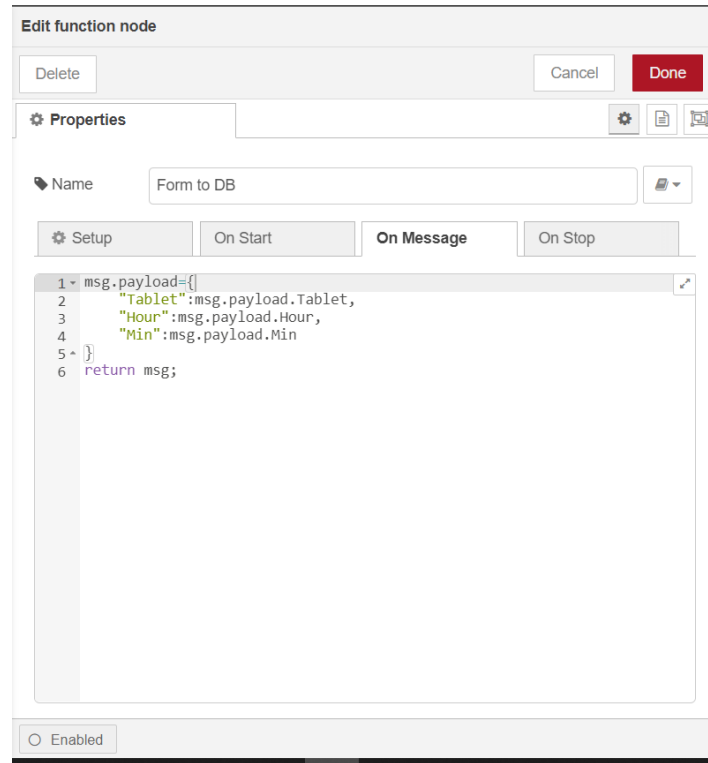
2. Function Node:

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

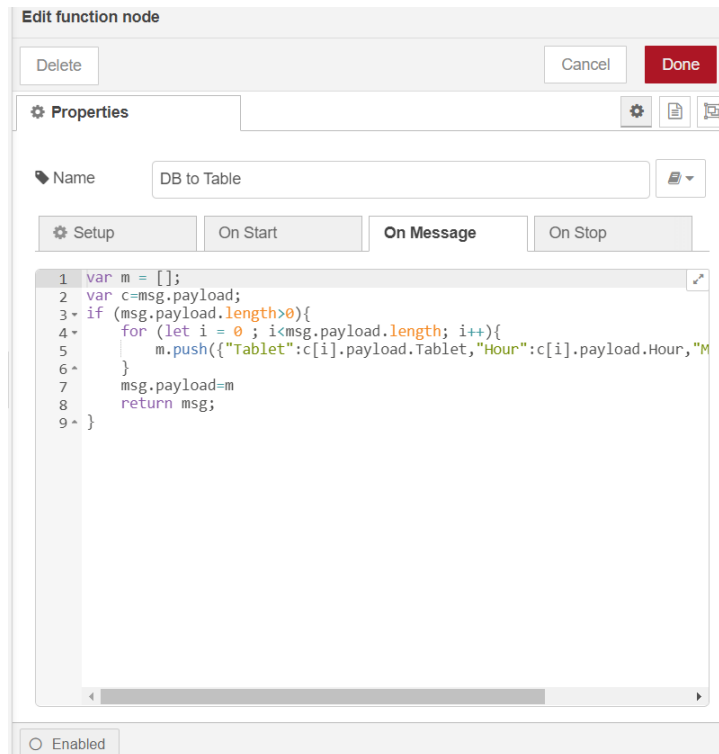
Function;

- Form to DB
- DB to Table
- Compare Time

2.1. Form to DB:



2.2.DB to Table:



2.3. Compare Time:

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.

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🔗

Service

node-red-zhhd-2022--cloudant-166635985€

▼

🗄️ Database

medicine

🔧 Operation

insert

▼

☐ Only store msg.payload object?

🔍 Name

Name

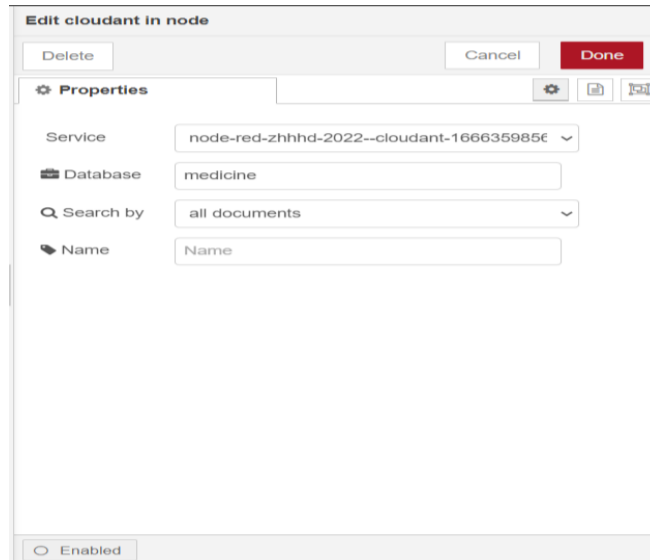
☐ Enabled

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:

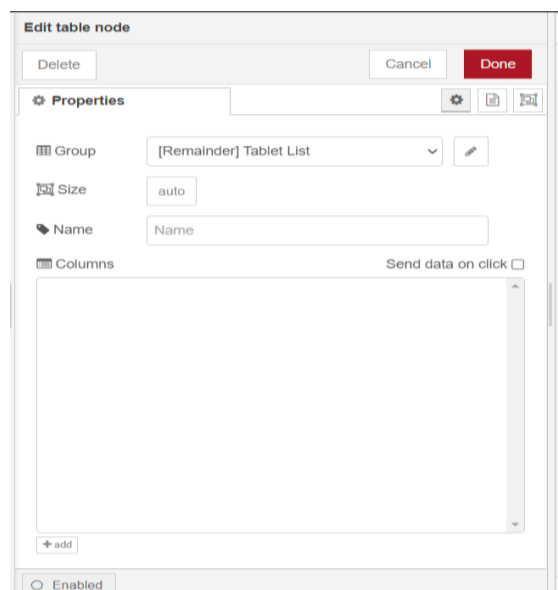
Drag “Cloudant in Node” from the storage nodes, which is use to retrieve the data that stored in the database. enter the credentials and name database you need to access.



The screenshot shows the 'Edit cloudant in node' configuration window. It has a title bar with 'Edit cloudant in node' and buttons for 'Delete', 'Cancel', and 'Done'. Below the title bar is a 'Properties' section with a search icon and a list of properties. The properties are: 'Service' (a dropdown menu showing 'node-red-zhhhd-2022--cloudant-1666359856'), 'Database' (a text input field containing 'medicine'), 'Search by' (a dropdown menu showing 'all documents'), and 'Name' (a text input field containing 'Name'). At the bottom of the window, there is a checkbox labeled 'Enabled' which is currently checked.

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. It has a title bar with 'Edit table node' and buttons for 'Delete', 'Cancel', and 'Done'. Below the title bar is a 'Properties' section with a search icon and a list of properties. The properties are: 'Group' (a dropdown menu showing '[Remainder] Tablet List'), 'Size' (a text input field containing 'auto'), 'Name' (a text input field containing 'Name'), and 'Columns' (a text input field containing 'Name'). There is also a checkbox labeled 'Send data on click' which is currently unchecked. At the bottom of the window, there is a checkbox labeled 'Enabled' which is currently checked.

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

Property	Value
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
Service	registered

☐ Enabled

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” from Common Nodes. Which is used to view the payloads.

Step 3:

Create Cloundant Database to save the incoming data.

↔

📊

🗄️

🔍

📋

👤

🔊

📖

🔒

Log Out

Databases

Database name ▾

Create Database

{ } JSON

📖

🔔

Your Databases

Name	Size	# of Docs	Partitioned	Actions
medicine	0.9 KB	7 📌	No	<div>🔍</div> <div>🔒</div> <div>🗑️</div>
noderedzhhd20221021	33.2 KB	4	No	<div>🔍</div> <div>🔒</div> <div>🗑️</div>
test	0 bytes	0	No	<div>🔍</div> <div>🔒</div> <div>🗑️</div>
test1	359 bytes	3 📌	No	<div>🔍</div> <div>🔒</div> <div>🗑️</div>

Showing 1–4 of 4 databases. Databases per page 20 ▾ < 1 >

↔

📊

🗄️

🔍

📋

👤

🔊

📖

🔒

Log Out

< medicine

⋮

All Documents +

Query

Permissions

Changes

Design Documents +

Document ID ▾

Options

{ } JSON

📖

🔔

☐

Table

Metadata

{ } JSON

🗄️

Create Document

	_id ▾	payload ▾	socketid ▾	topic ▾
☐	2f220b7493ebefedfda38f9e9bbd...	{ "Tablet": "Paracetamol", "Hour": ...	F4GrDmWgYeAih6CAAaf	{ "Tablet": "Paracetamol", "Hour": ...
☐	602ee33908ddad7a18dce906d9...	{ "Tablet": "Dolo 650", "Hour": 8, "...	F4GrDmWgYeAih6CAAaf	{ "Tablet": "Dolo 650", "Hour": 8, "...
☐	627f83547320b4ecda41f2a2f29...	{ "Tablet": "Deplatta", "Hour": 17, "...	DJFb8_3KByUj0AH9AAAb	{ "tab": "Deplatta", "Hour": 17, "Mi...
☐	9a5df1a280e43ed2b57d454d30f...	{ "Tablet": "Ativan", "Hour": 17, "Mi...	DJFb8_3KByUj0AH9AAAb	{ "tab": "Ativan", "Hour": 17, "Min":...
☐	a960d38e7e2f5eab2f5634f4fc86...	{ "Tablet": "Concor", "Hour": 17, "...	DJFb8_3KByUj0AH9AAAb	{ "tab": "Concor", "Hour": 17, "Min...
☐	f0dcafdd96eabf4afa776e2672f7...	{ "Tablet": "Cardace", "Hour": 17, "...	DJFb8_3KByUj0AH9AAAb	{ "tab": "Cardace", "Hour": 17, "Mi...
☐	f7bc3b6818fc517b36355d14eff9...	{ "Tablet": "Rosuvas", "Hour": 17, "...	DJFb8_3KByUj0AH9AAAb	{ "tab": "Rosuvas", "Hour": 17, "Mi...

Showing 4 of 5 columns. ☐ Show all columns.

Showing document 1 - 7. Documents per page: 20 ▾ < >

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

rl="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(rl)

# 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":"hg0h1l",           # 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="QGxbVq1lTgSFnn8_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)

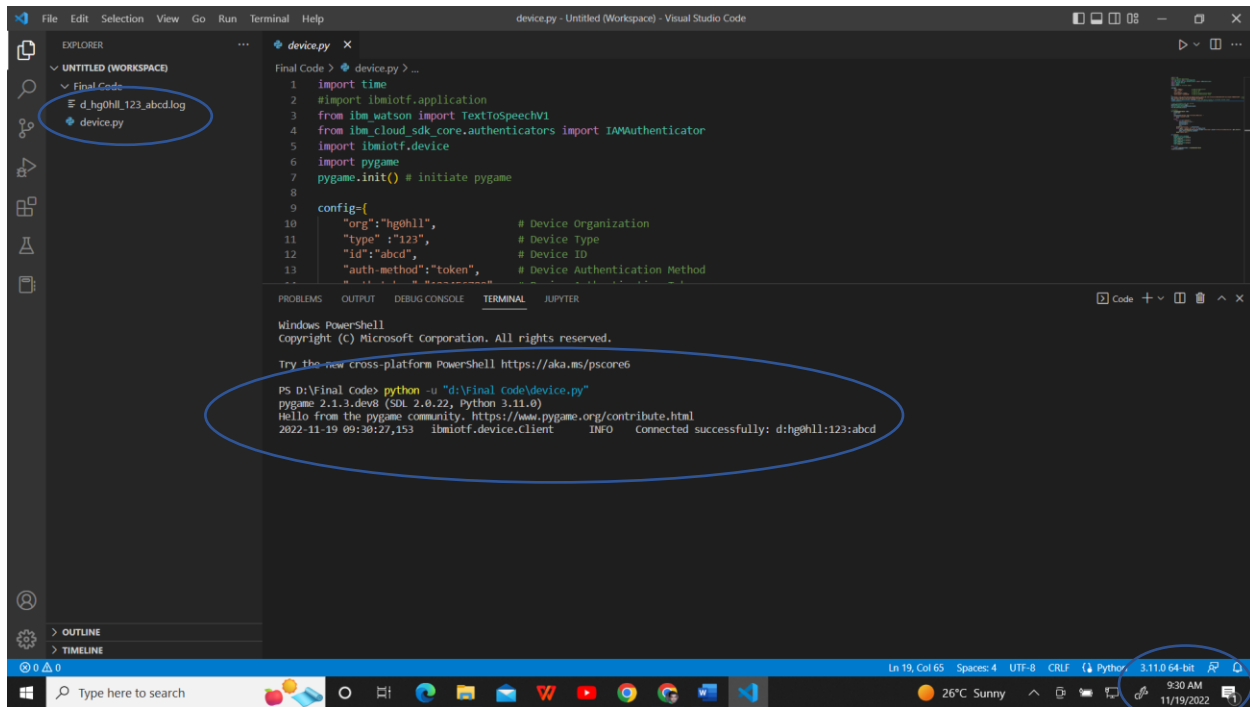
# callback and
def myCommandCallback (cmd):
    a=cmd.data
    c=1
    instruction="Please Take following Medicine. "
    if len(a["command"])==0:
        pass
    else:
        for i in a["command"]:
            instruction+=str(c)+". "
            instruction+=i
            instruction+=". "
            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()
    |
```

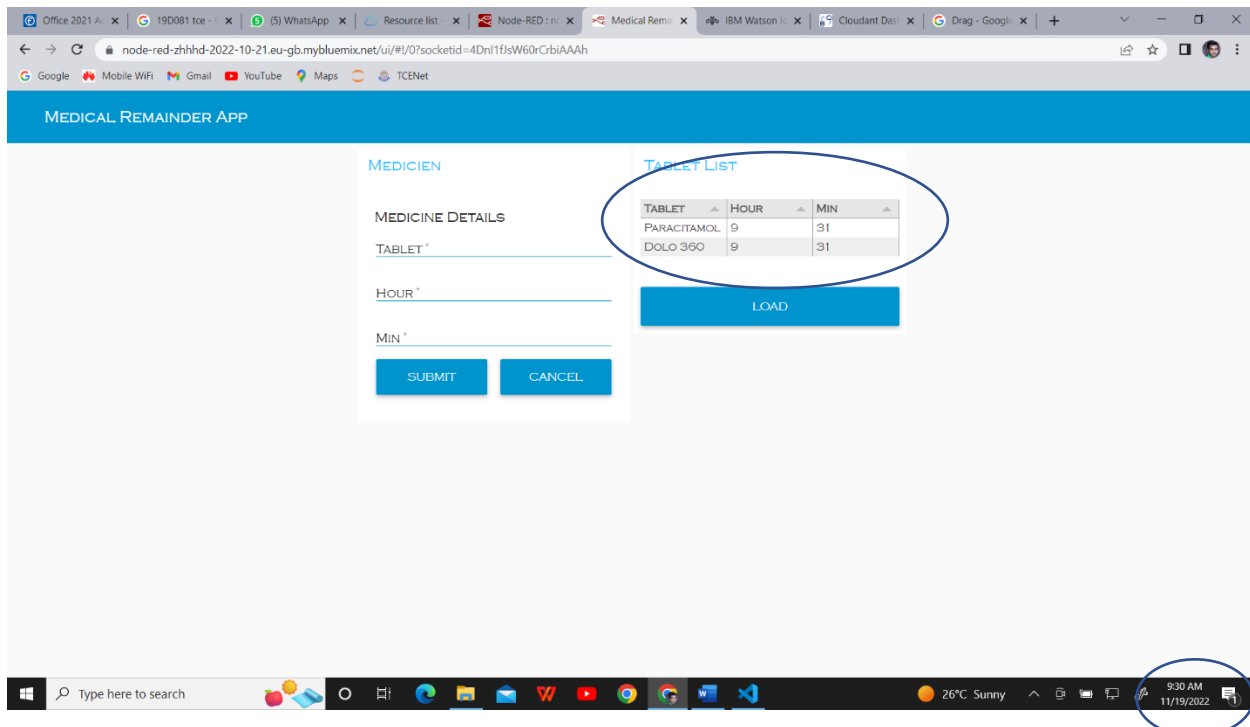
1. Connect with the device.

1. Connect with the device.

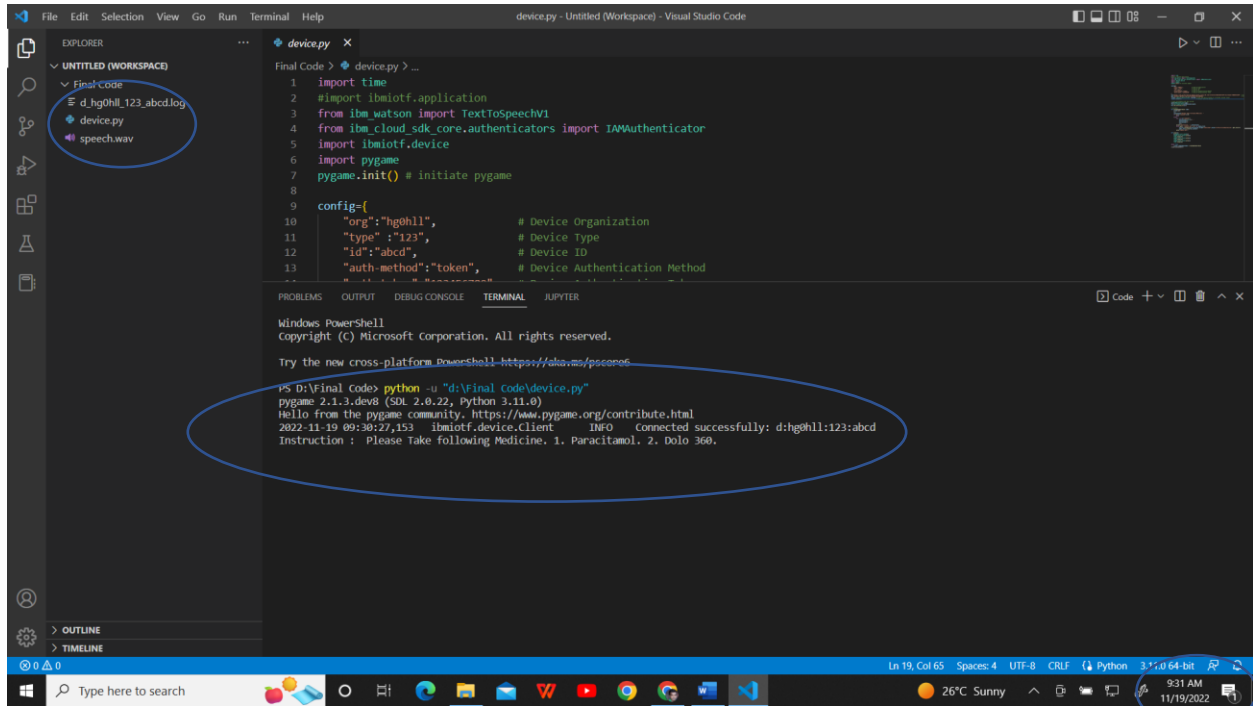


2. Load Tablet name and time in User UI

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



- 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.



The screenshot displays the Visual Studio Code interface. On the left, the Explorer pane shows a workspace with files: 'Final Code', 'd_hg0hl123_abcd.log', 'device.py', and 'speech.wav'. The 'device.py' file is open in the editor, showing Python code that imports necessary libraries and configures an IBM Watson Text-to-Speech client. The code includes a configuration dictionary with organization, type, ID, and authentication method. Below the editor, the Terminal pane shows the execution of the script. The output indicates a successful connection to the IBM Watson Text-to-Speech service and the generation of an audio file. The system clock in the bottom right corner shows 9:31 AM on 11/19/2022.

```
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": "hg0hl1",           # Device Organization
11     "type": "123",            # Device Type
12     "id": "abcd",             # Device ID
13     "auth-method": "token",    # Device Authentication Method
14 }
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS D:\Final Code> python -u "d:\Final Code\device.py"
pygame 2.1.3.dev8 (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:hg0hl1:123:abcd
Instruction : Please Take following Medicine. 1. Paracetamol. 2. Dolo 360.
```

CONCLUSION:

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