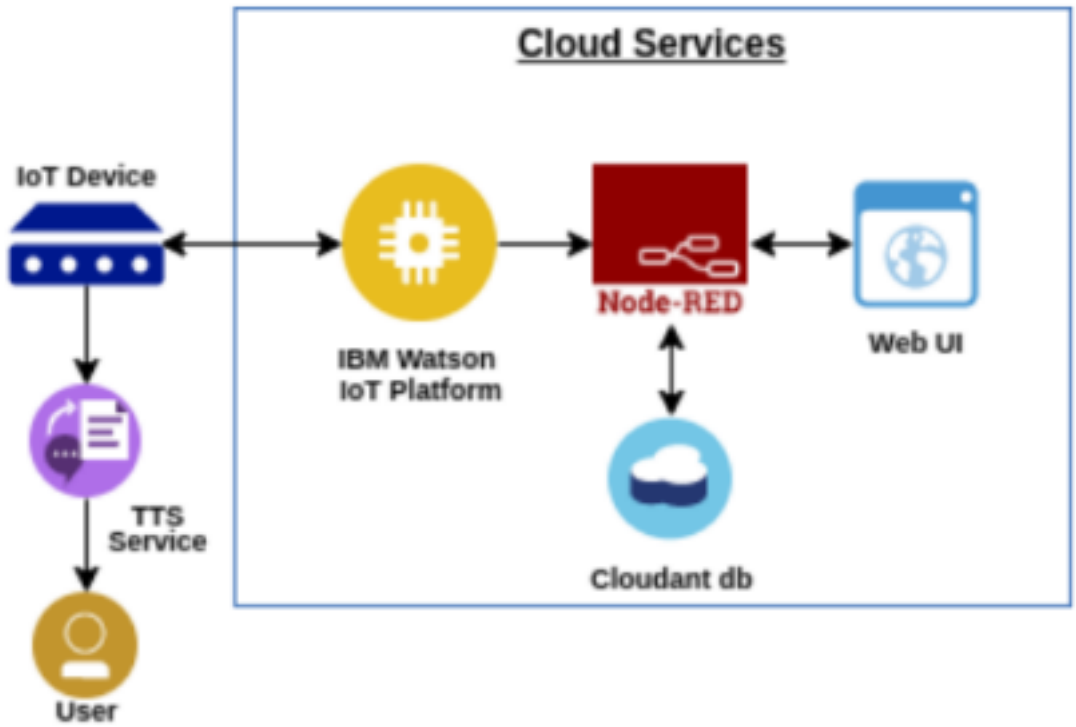
Final Deliverables

Team ID	**PNT2022TMID01850**
Project title	**Personal Assistance for seniors who are self-reliant**
Date	**14 November 2022**

OBJECTIVE:

- Sometimes elderly people forget to take their medicine at the correct time.
- They also forget which medicine . He / She should take it 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 an App "Standard App" and node down the API key and Token.

Device Drilldown - 12345

- Device Credentials
- Connection Information
- Recent Events
- State
- Device Information
- Metadata
- Diagnostics
- Connection Logs
- Device Actions

Device Credentials

You registered your device to the organization. Add these credentials to the device to connect it to the platform. After the device is connected, you can navigate to view connection and event details.

Organization ID	6xl5xp
Device Type	NodeMCU
Device ID	12345
Authentication Method	use-token-auth
Authentication Token	1234567890

Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.

Find out how to add these credentials to your device

IBM Watson IoT Platform

dharanigkd02@gmail.com
ID: 6xl5xp

← Back

Device Drilldown - 12345

- Device Credentials
- Connection Information
- Recent Events
- State
- Device Information
- Metadata
- Diagnostics
- Connection Logs
- Device Actions

Device Type	NodeMCU
Device ID	12345
Authentication Method	use-token-auth
Authentication Token	1234567890

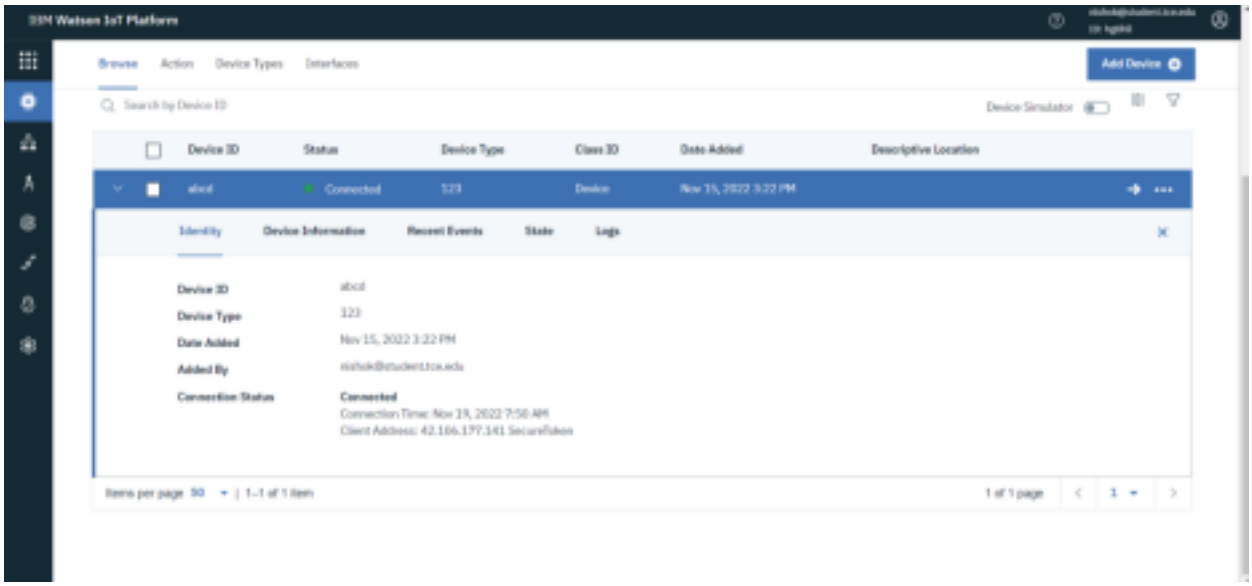
Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.

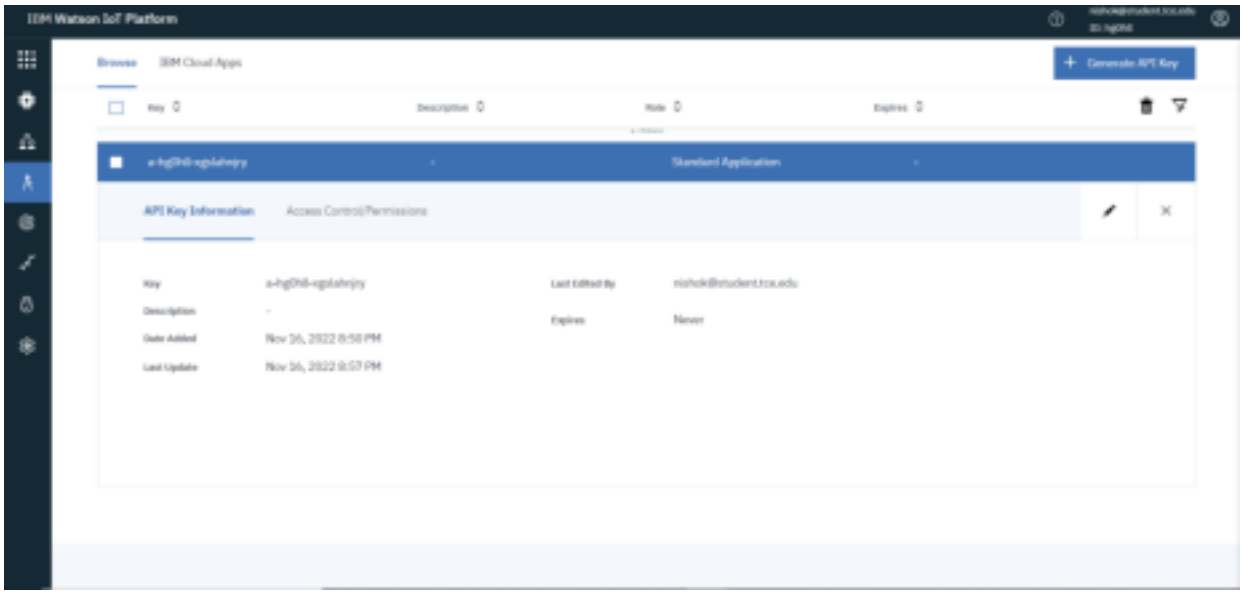
Find out how to add these credentials to your device

Connection Information

Basic connection information about this device.

Device ID	12345
Device Type	NodeMCU





Step 2:

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

Resource list /

node-red-fysyl-2022--cloudant-1667109493143 Active Add tags

Details Actions...

Manage
Service credentials
Plan
Connections

Overview Capacity Docs

Launch Dashboard

Deployment details

CRN	crn:v1:bluemix:public:cloudantnosqldb:au-syd:a/9154bf298d244d8c918065a7b7ae513f:1fdf6581-7567-4252-8858-61365168e092::
Location	Sydney
External endpoint	https://e9dbfd87-aa0d-4b2f-86ac-d38f62c6ee14-bluemix.cloudant.com
External endpoint (preferred)	https://e9dbfd87-aa0d-4b2f-86ac-d38f62c6ee14-bluemix.cloudantnosqldb.appdomain.cloud
Authentication methods	IBM Cloud IAM and Cloudant credentials Migrate to IAM Only
Activity Tracker event types	Management Save
Disk encryption	Yes. Automatically generated disk encryption key.

Resource list / App details /

Node RED FYSYL 2022-10-30 Add tags

Actions...

Details

App URL	You must deploy your app first
Source	Download code
Resource group	Default
Deployment target	You must deploy your app first
Created	30/10/2022

Services

Cloudant

Open dashboard Documentation API reference

Credentials

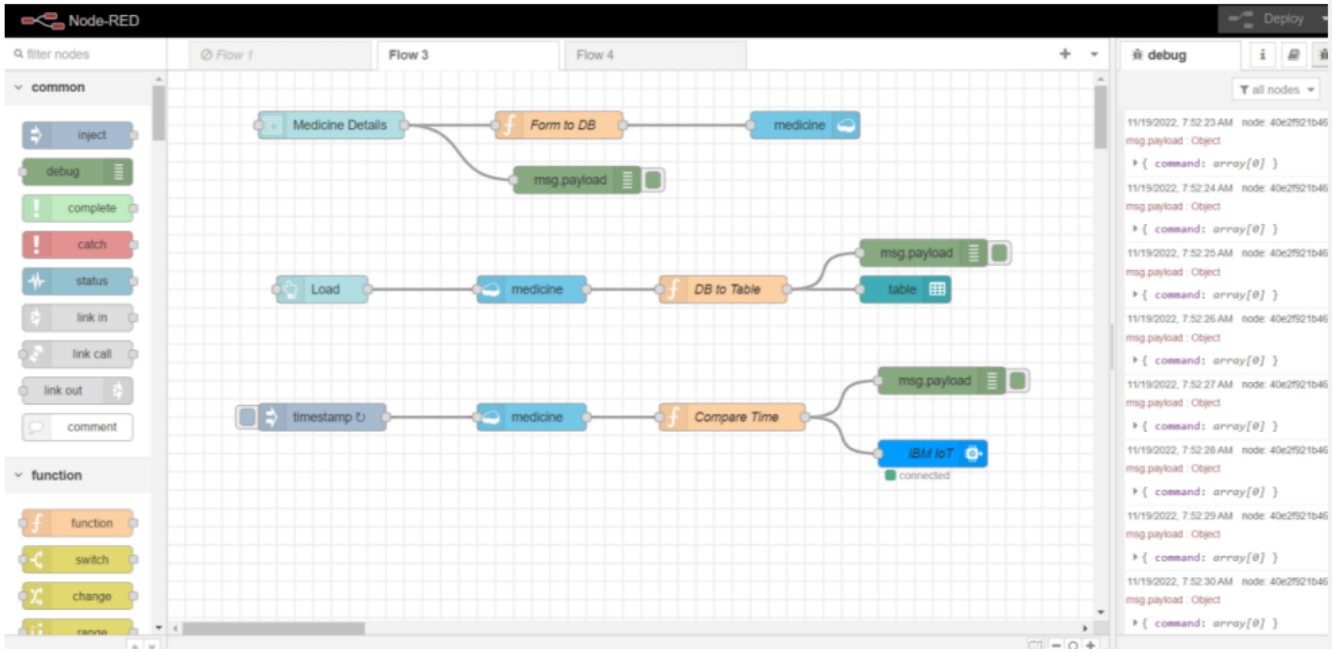
Connect existing services + Create service +

Deployment Automation

Configure Continuous Delivery

Continuous Delivery is not enabled for this app. Enable Continuous Delivery to automate builds, tests, and deployments through Delivery Pipeline, GitLab, and more.

Deploy your app

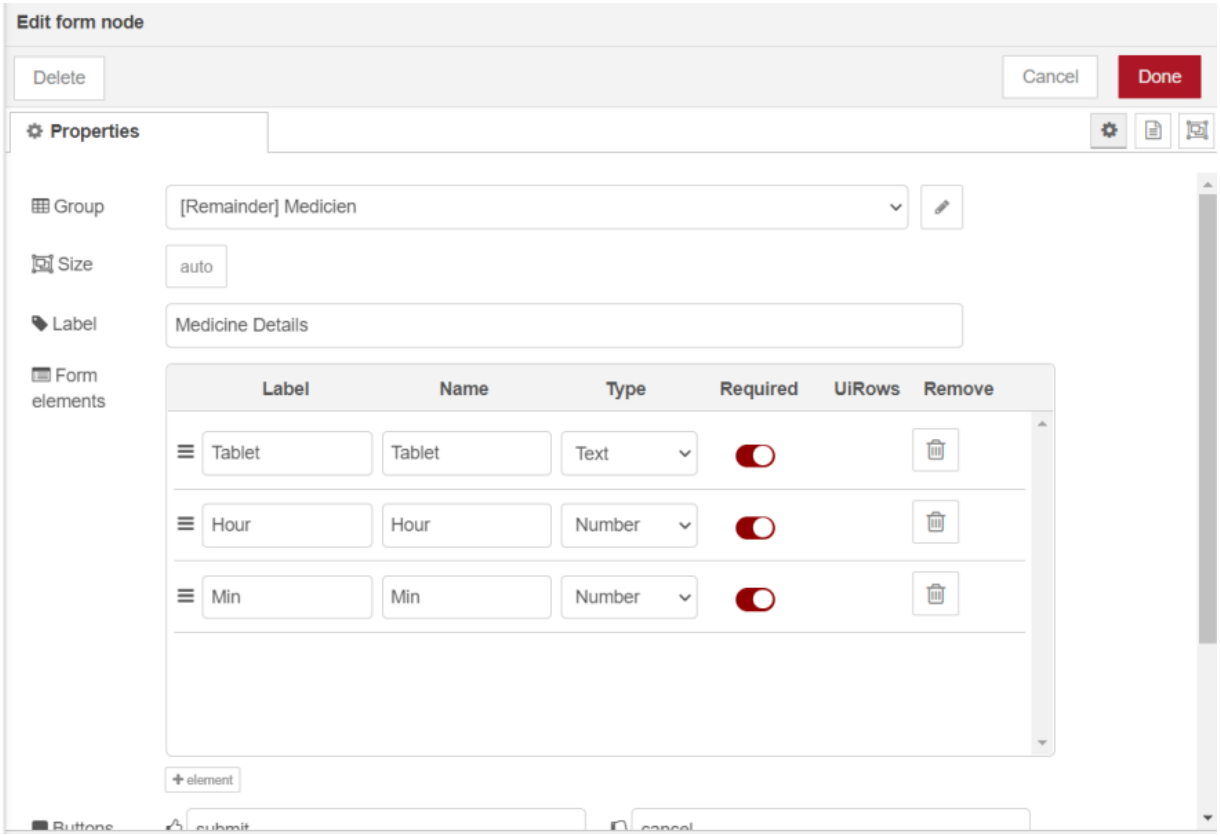


Nodes we user for this project:

1. Form
2. Function
3. Cloudant out
4. Botton
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.



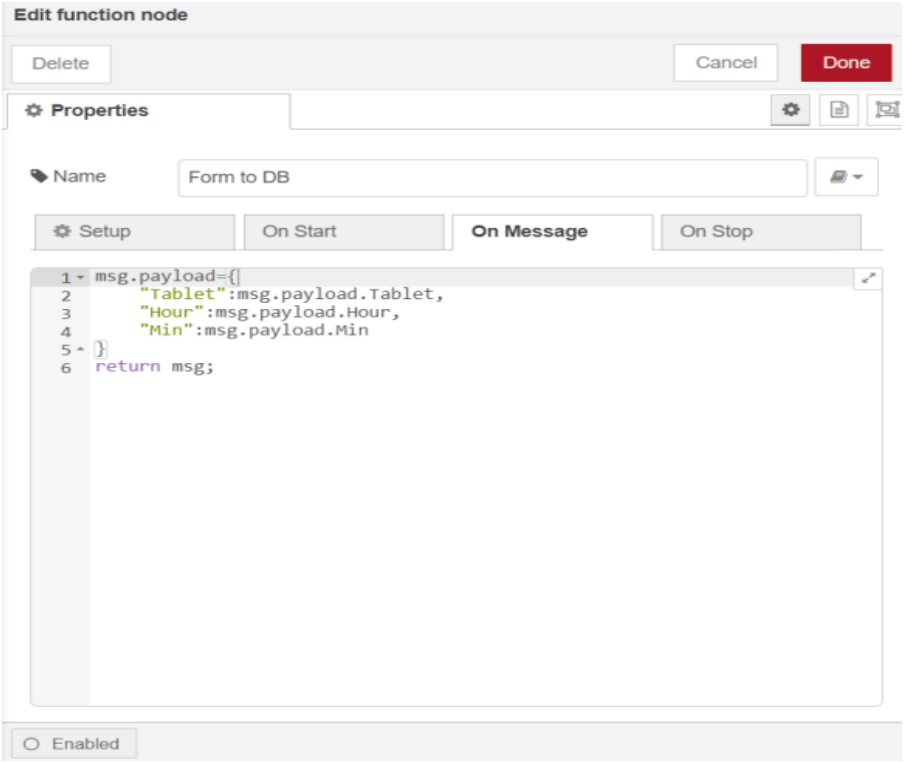
2. Function Node:

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

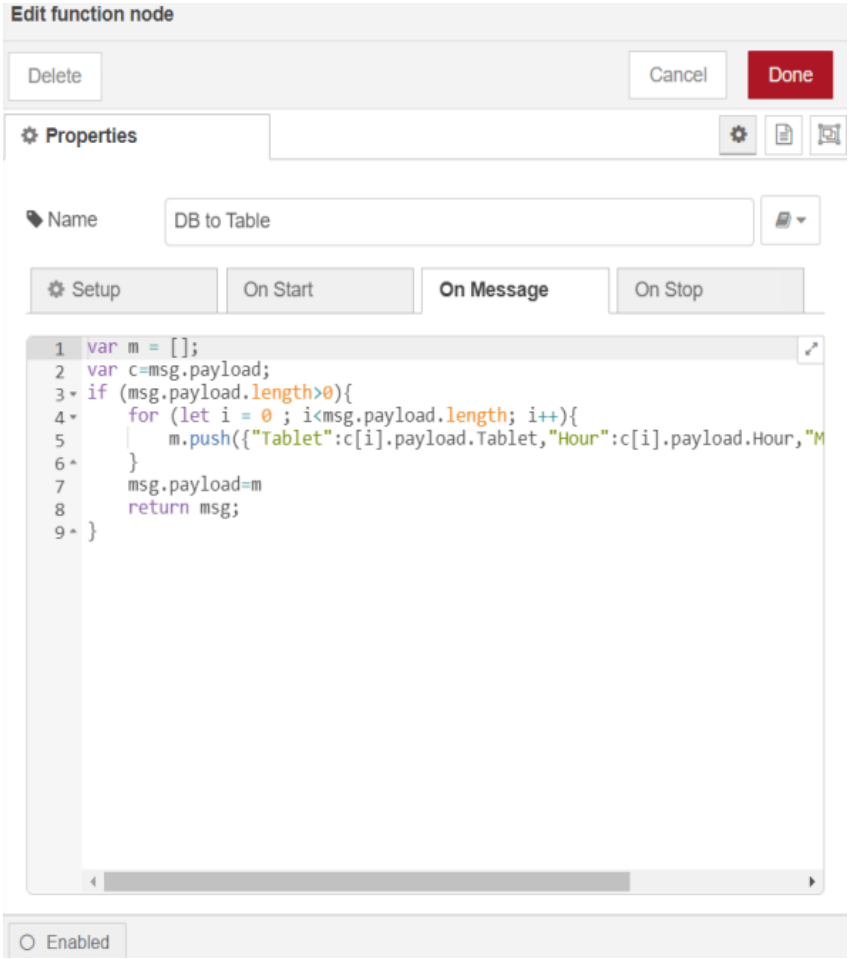
Function;

- Form to DB
- DB to Table
- Compare Time

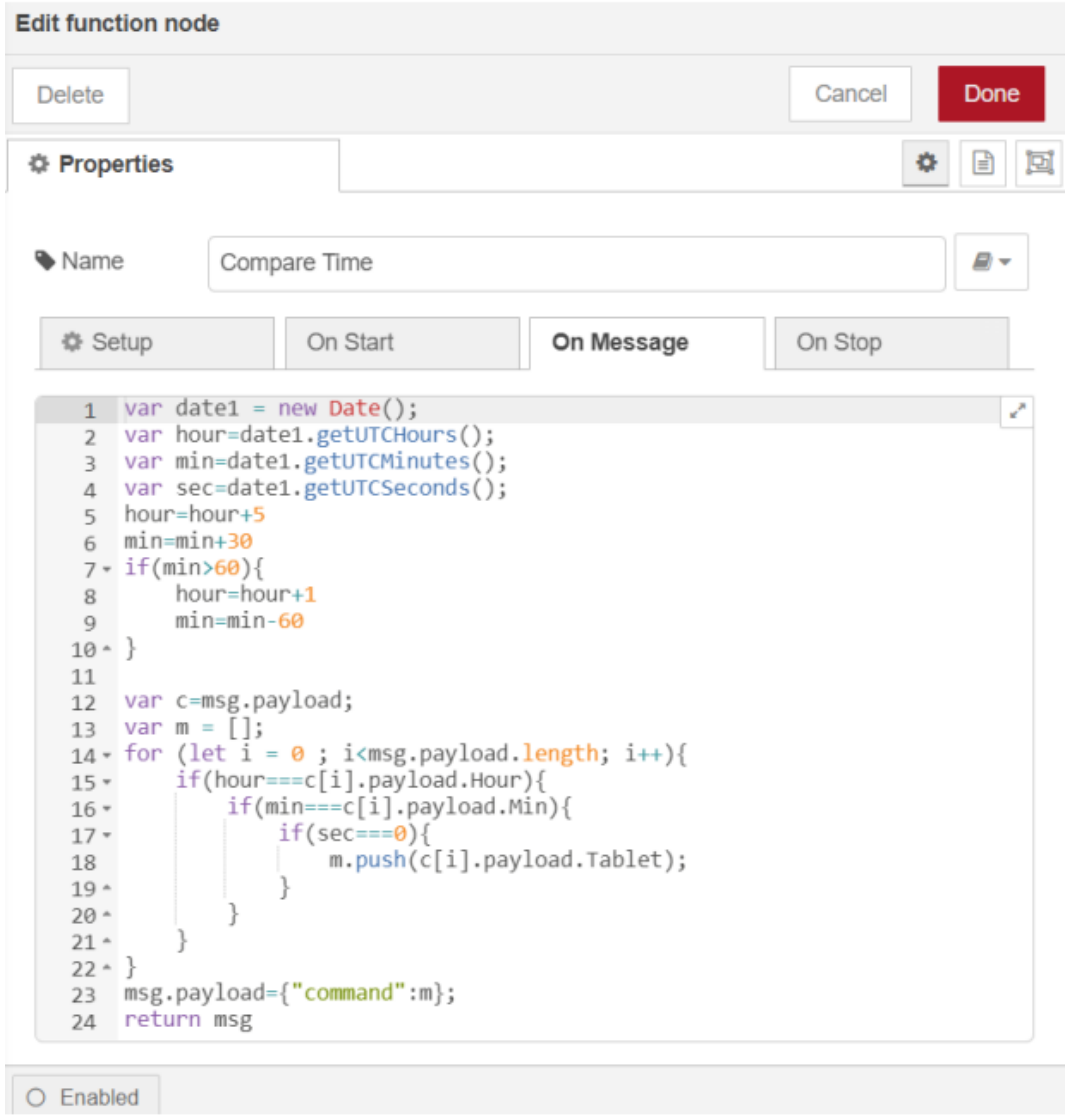
2.1.Form to DB:



2.2.DB to Table:

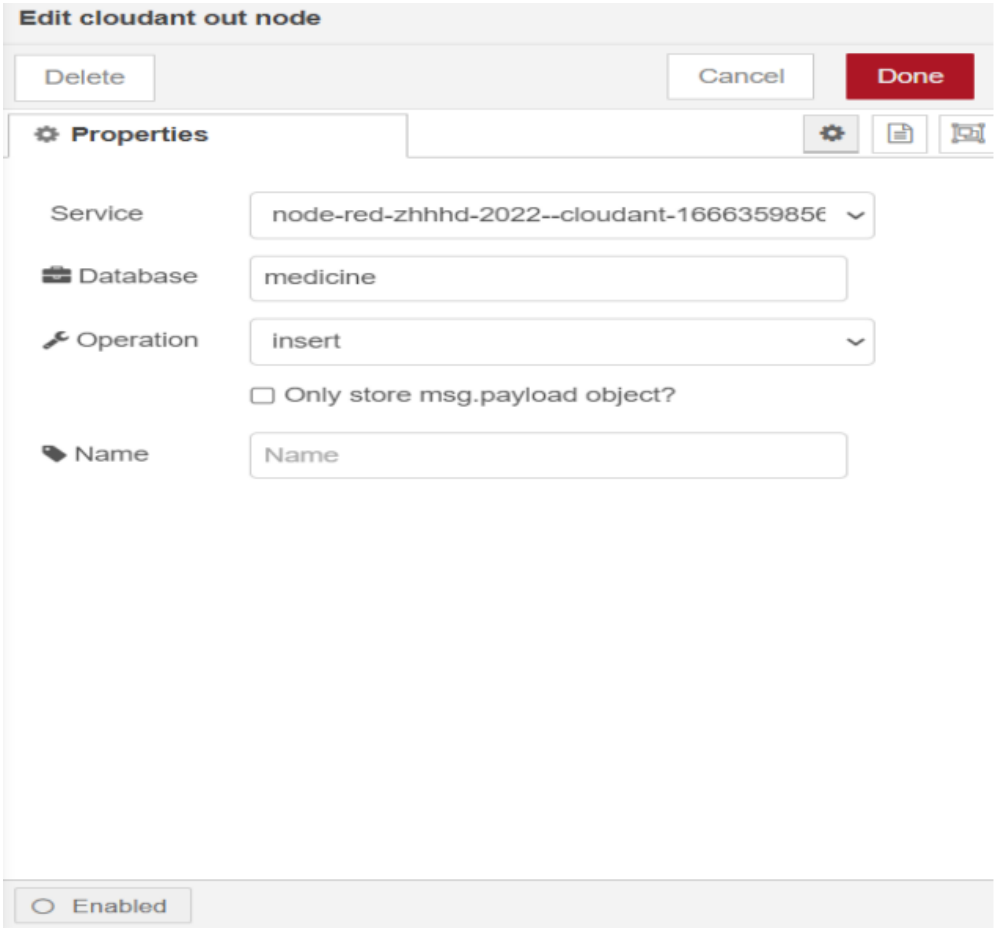


2.3.Compare Time:



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.

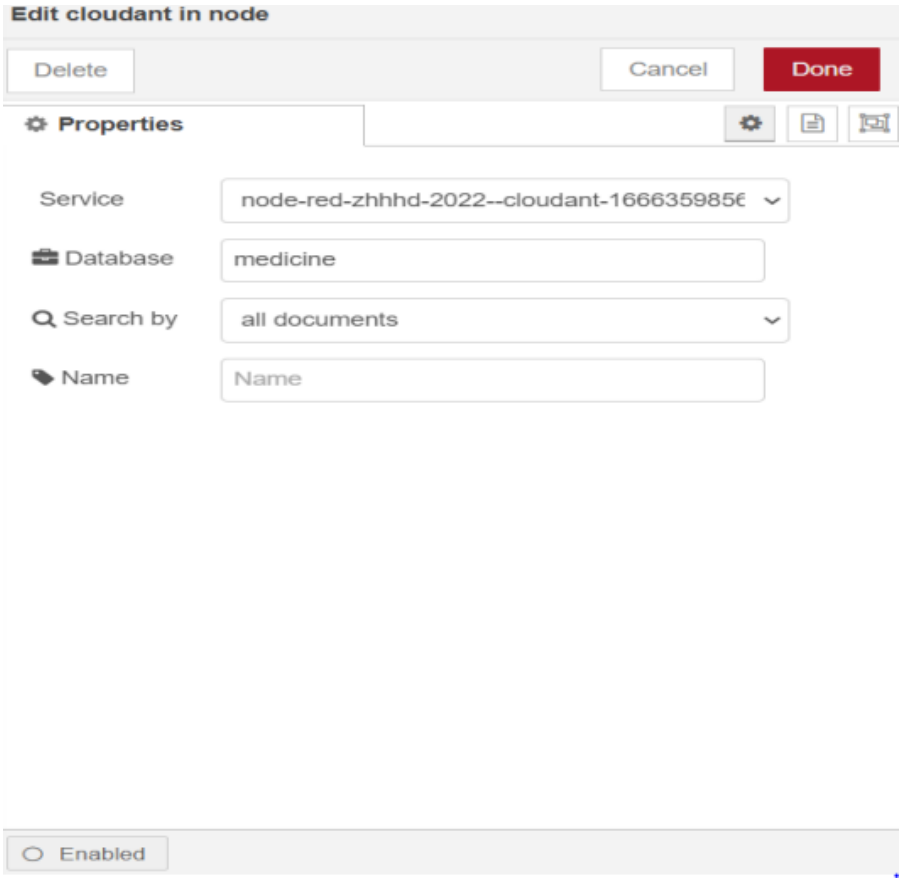


4. Button Node:

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.

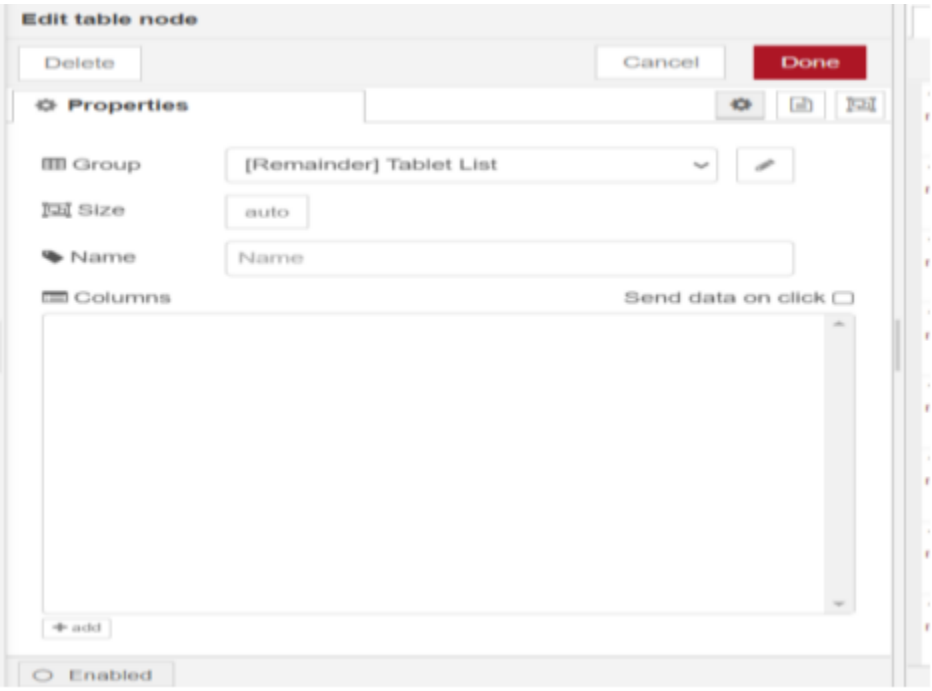
5. Cloudant In Node:

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.



6. Table UI Node:

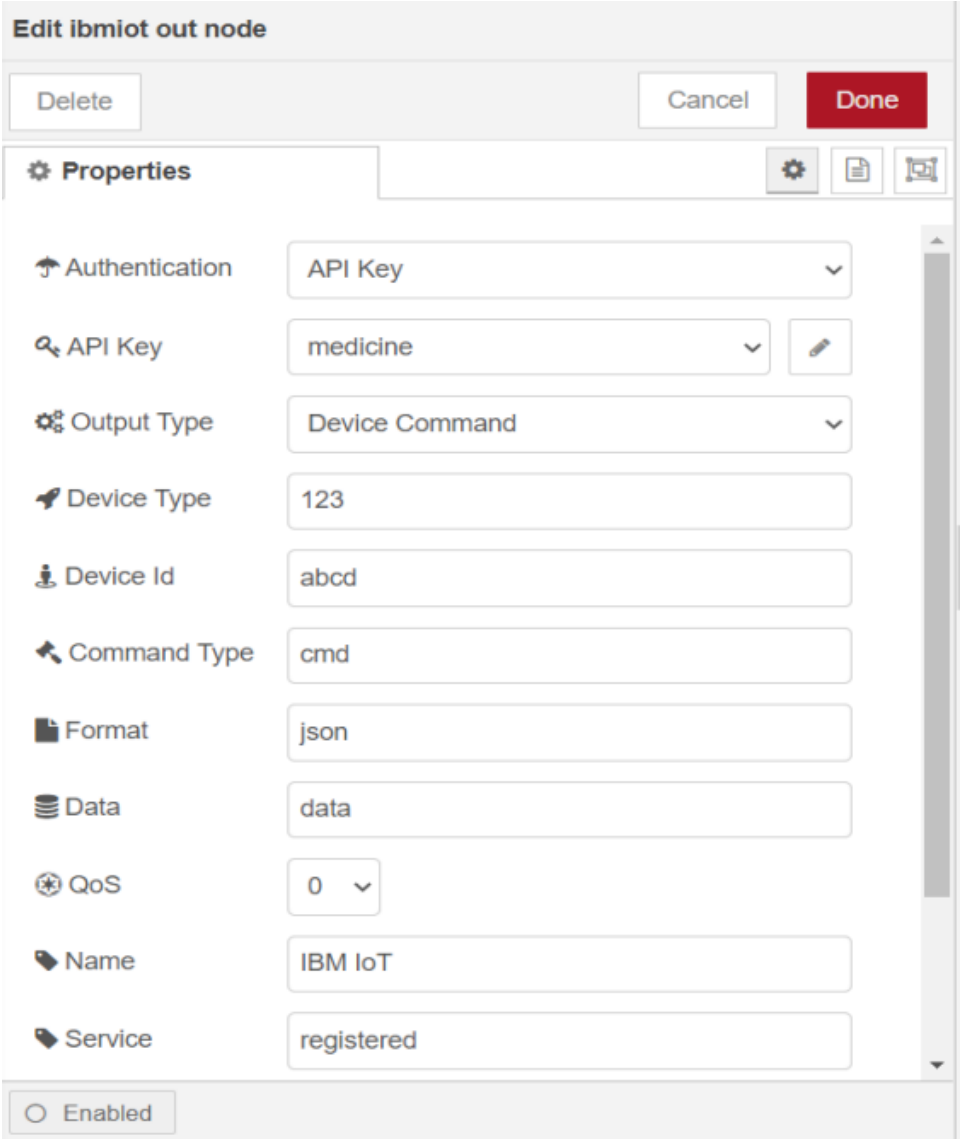
Drag “Table UI Node” from Dashboard Nodes and name the table. The table is use to see the loaded medicine in the database by the user it seen in the user IU dashboard.



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



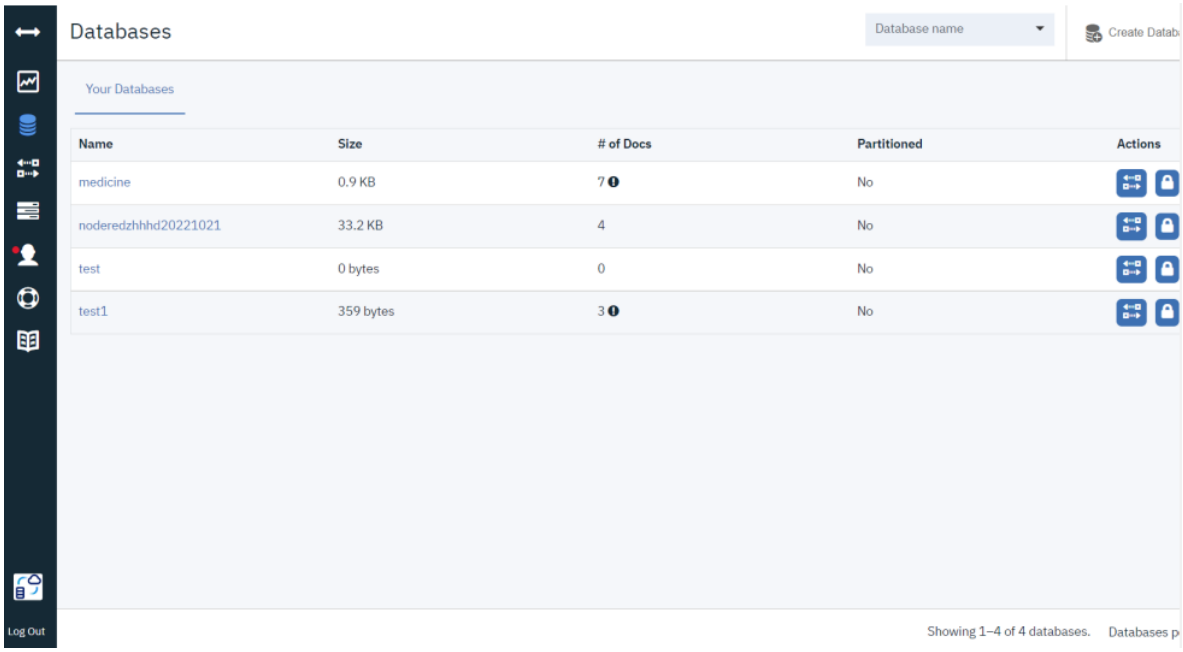
8. Inject node:

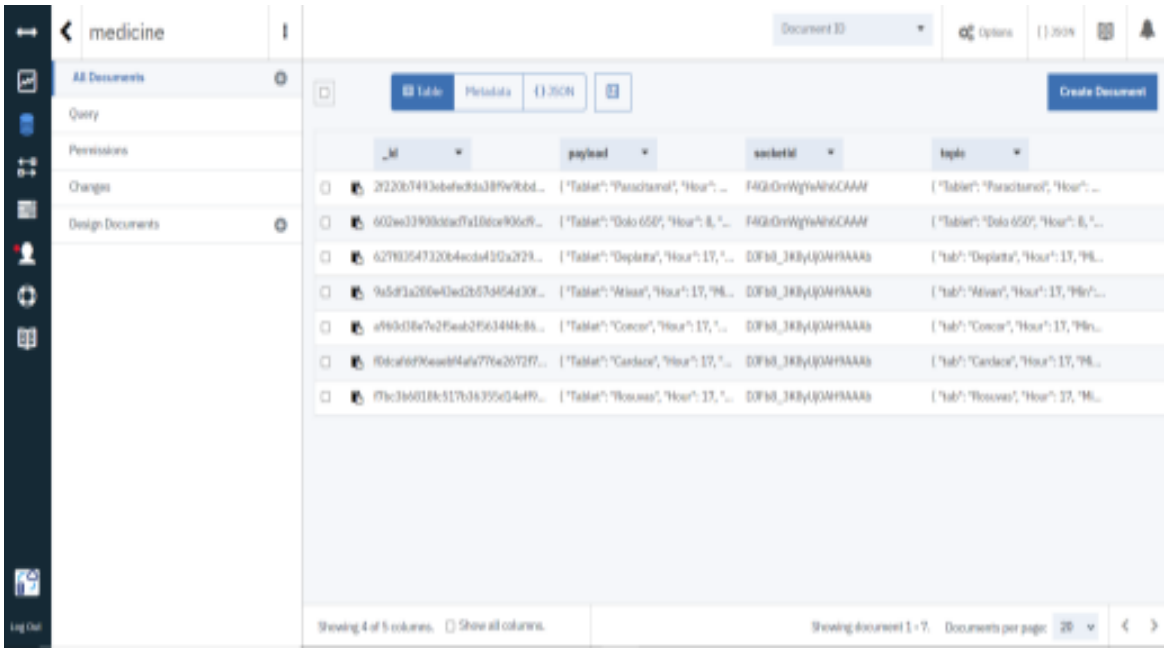
Drag "Inject Node" from Common nodes. Which is used for inject the time stamp every second for retrieve the data from database and compare the data and the current time. 9. Debug node:

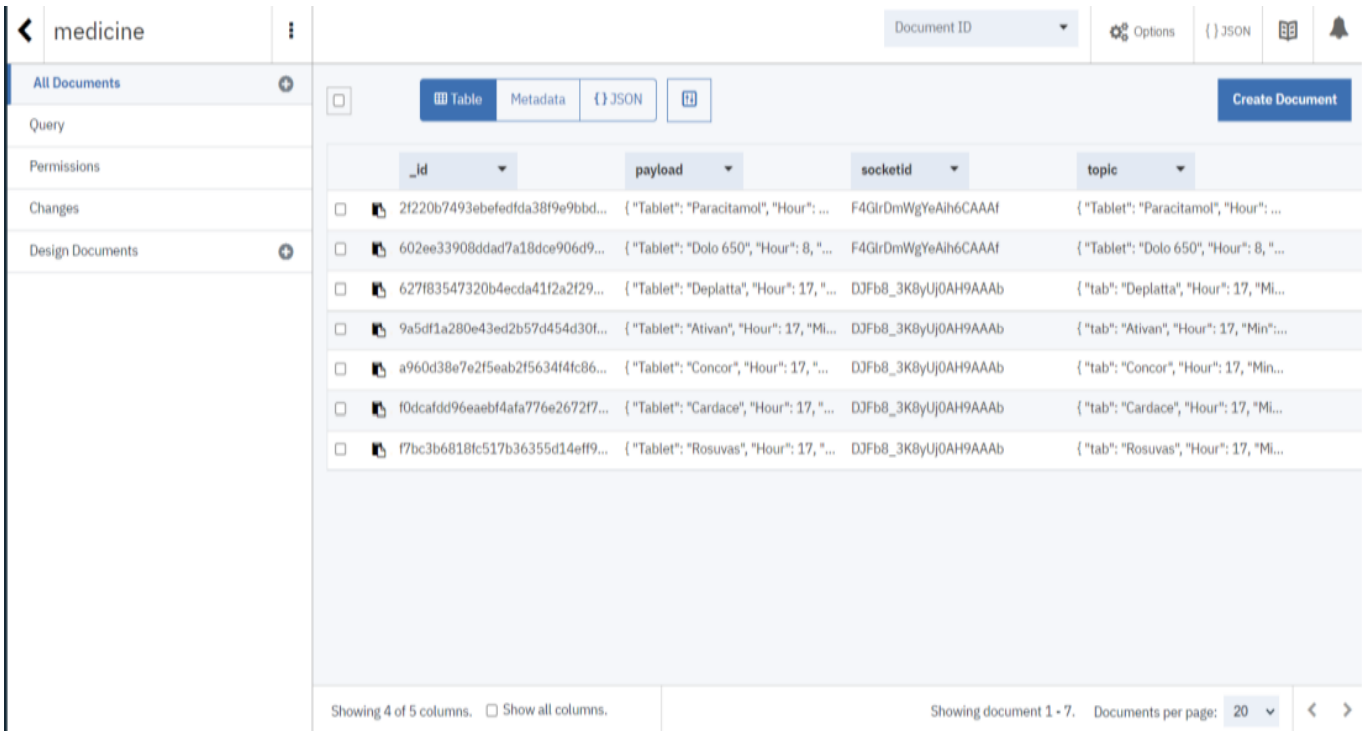
Drag "Debug Node" rom Common Nodes. Which is used to view the payloads.

Step 3:

Create Cloundant Database to save the incoming data.


Databases
Database name
Create Databa
Your Databases
Name
Size
# of Docs
Partitioned
Actions
medicine
0.9 KB
7
No
noderedzhhhd20221021
33.2 KB
4
No
test
0 bytes
0
No
test1
359 bytes
3
No
Log Out
Showing 1–4 of 4 databases.
Databases p


medicine
All Documents
Query
Permissions
Changes
Design Documents
Table
Metadata
{} JSON
Create Document
_id
payload
socketid
topic
Showing 4 of 5 columns.
Show all columns.
Showing document 1 - 7.
Documents per page:
20
Log Out



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":"hg0hll",                             # 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 Varible 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 divice
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(url)

# Text To Speech Convertion
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)
```

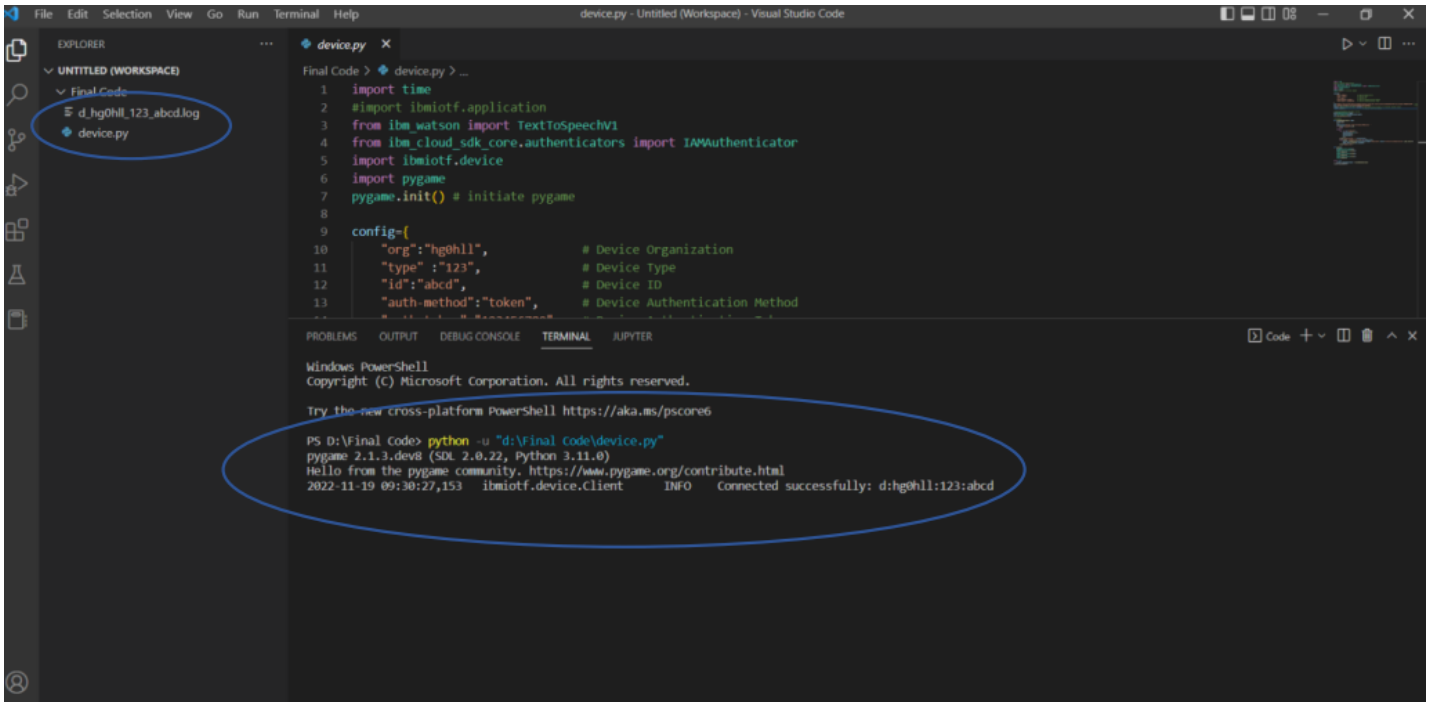
Step 5:

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":"hg0hll",             # 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 Varible 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


            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()
```

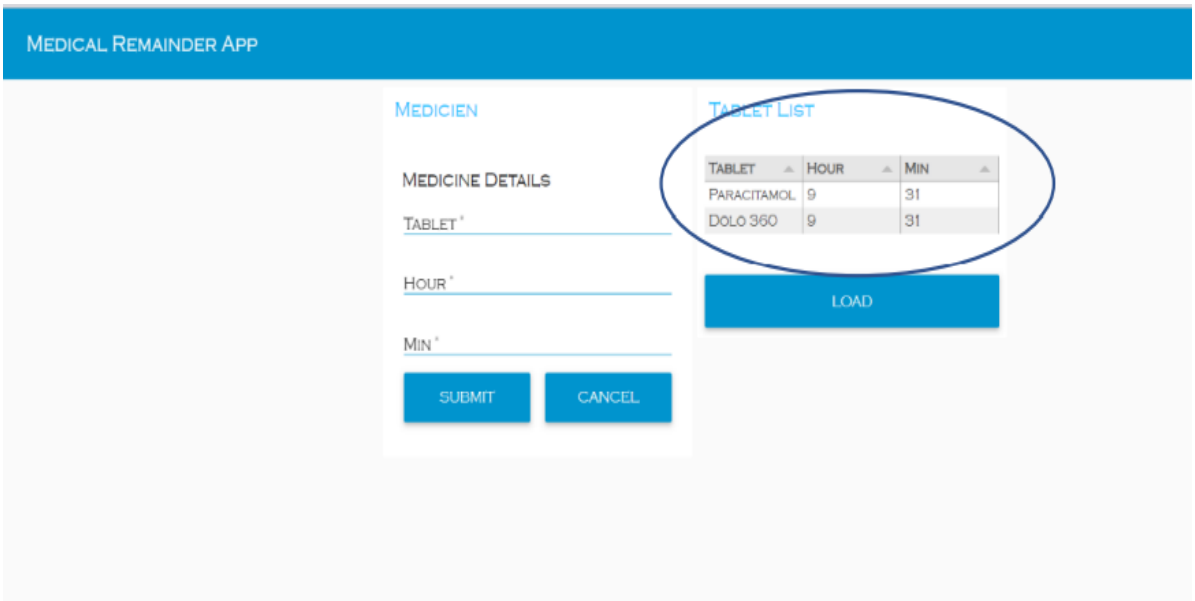
RESULT:

1. Connect with the device.

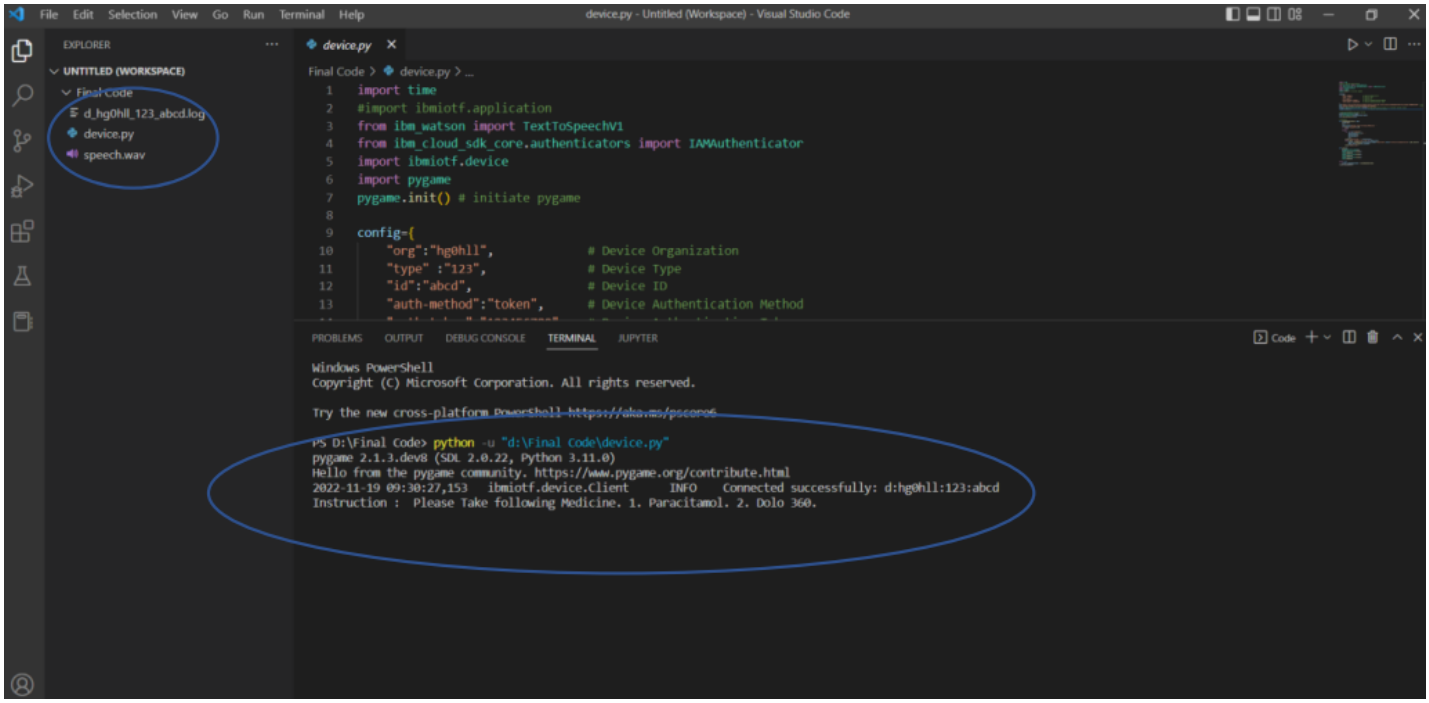


2. Load Tablet name and time in User UI

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



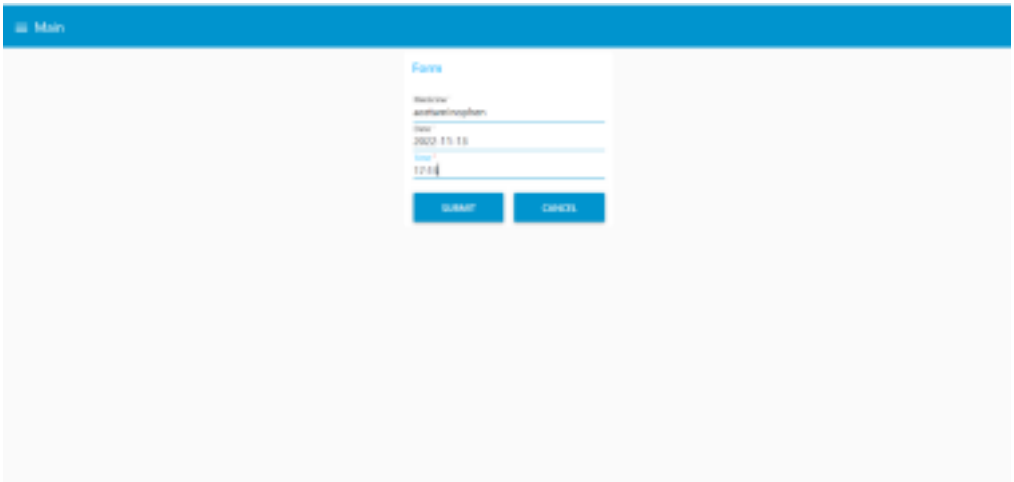
3. At 9.31 Am data saved in the DB is received and the audio file for instructions is generated and voice command was given to the user.



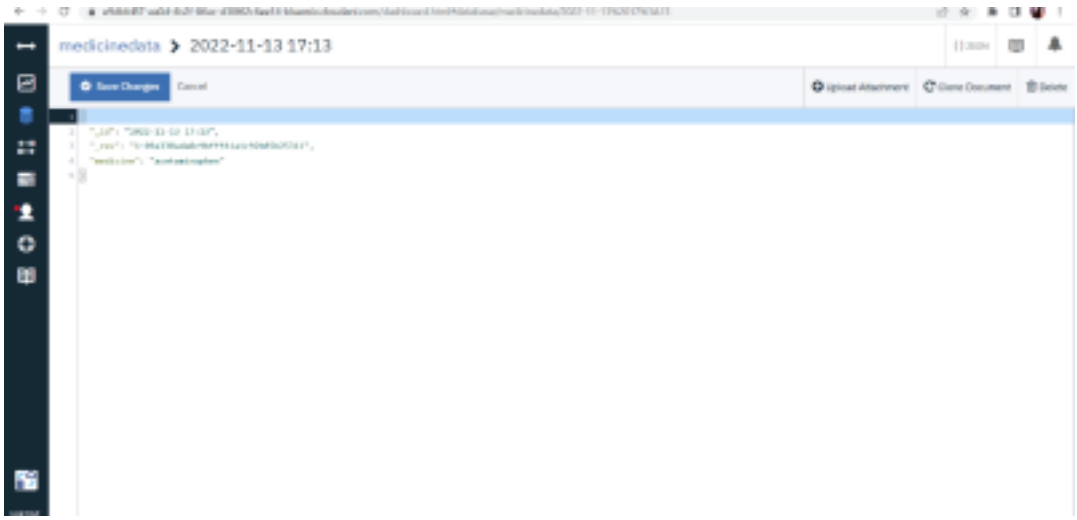
Web Application

Web Application using node-red To Create a form for taking the user inputs like medicine name, time, and date. Store all the details in the database.

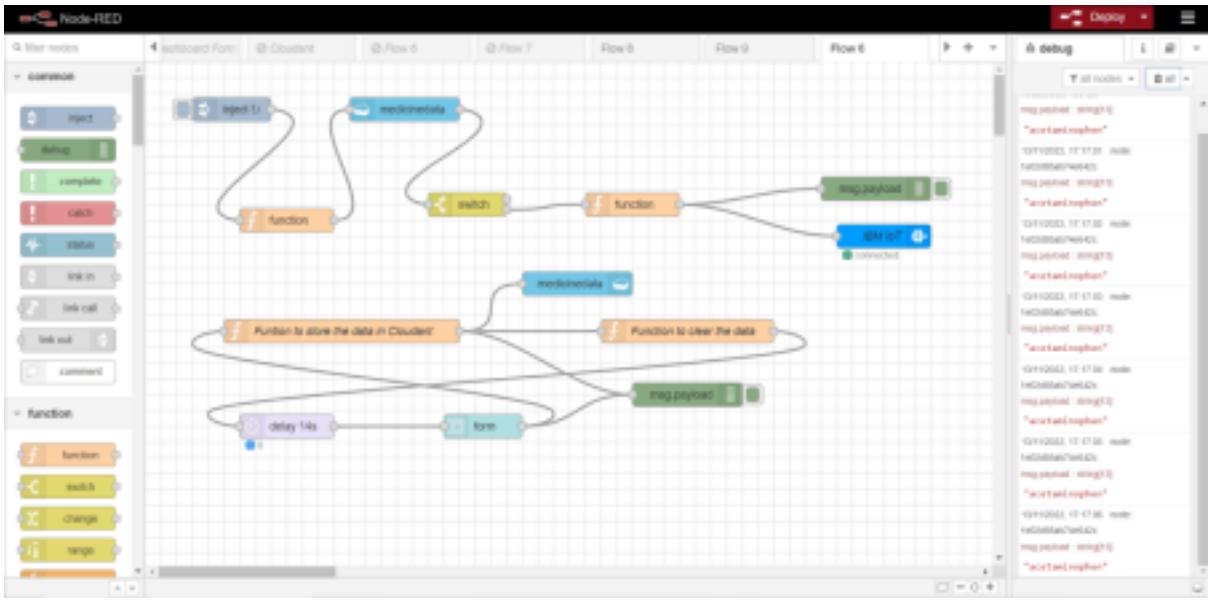
1. Get Data From User:



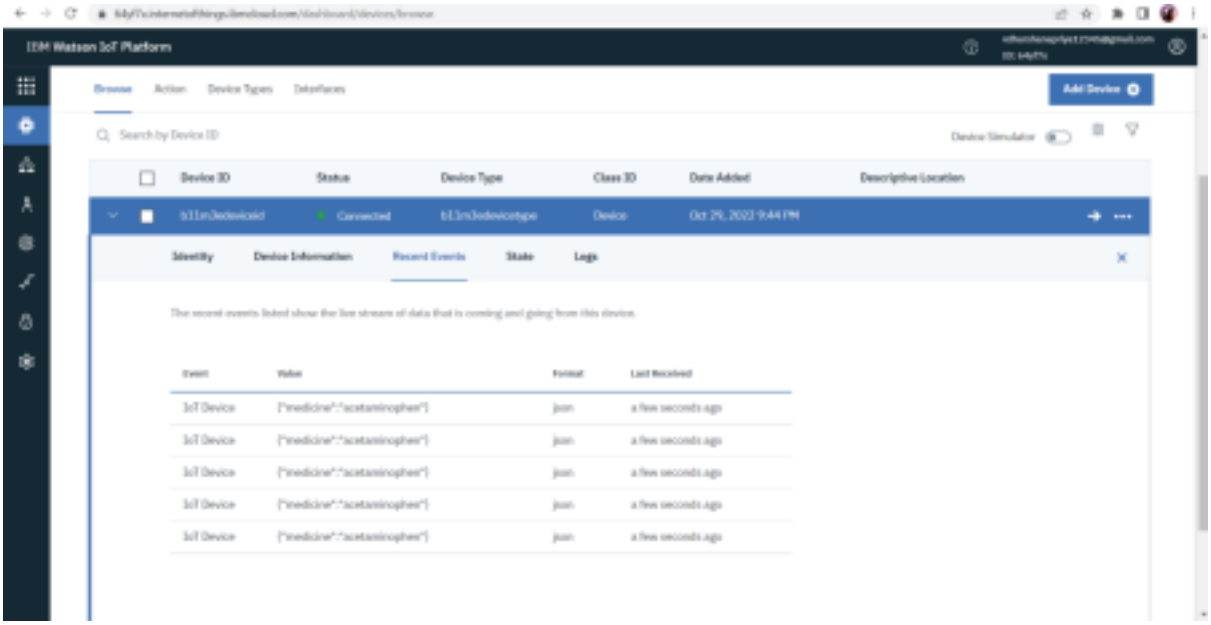
2. Stored in Cloudant



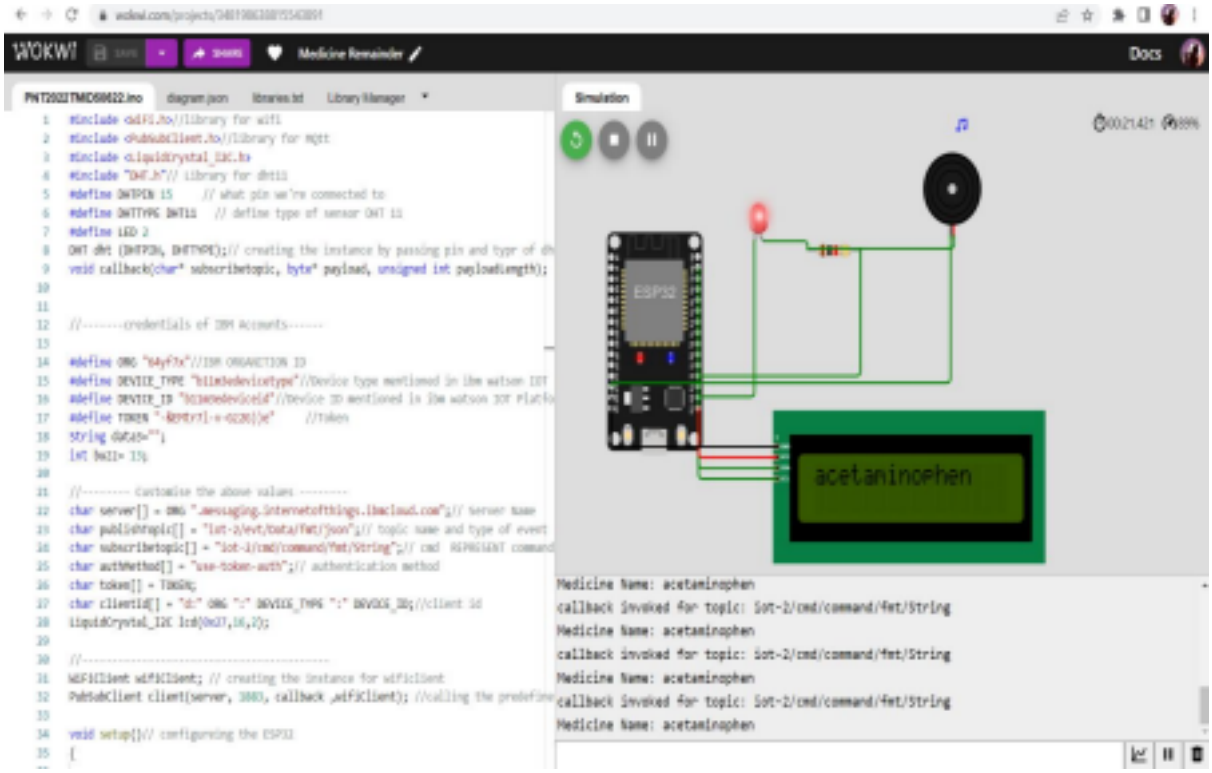
3. Display in Node-red



4. **Streaming in Watson IoT Platform**



5. Simulation

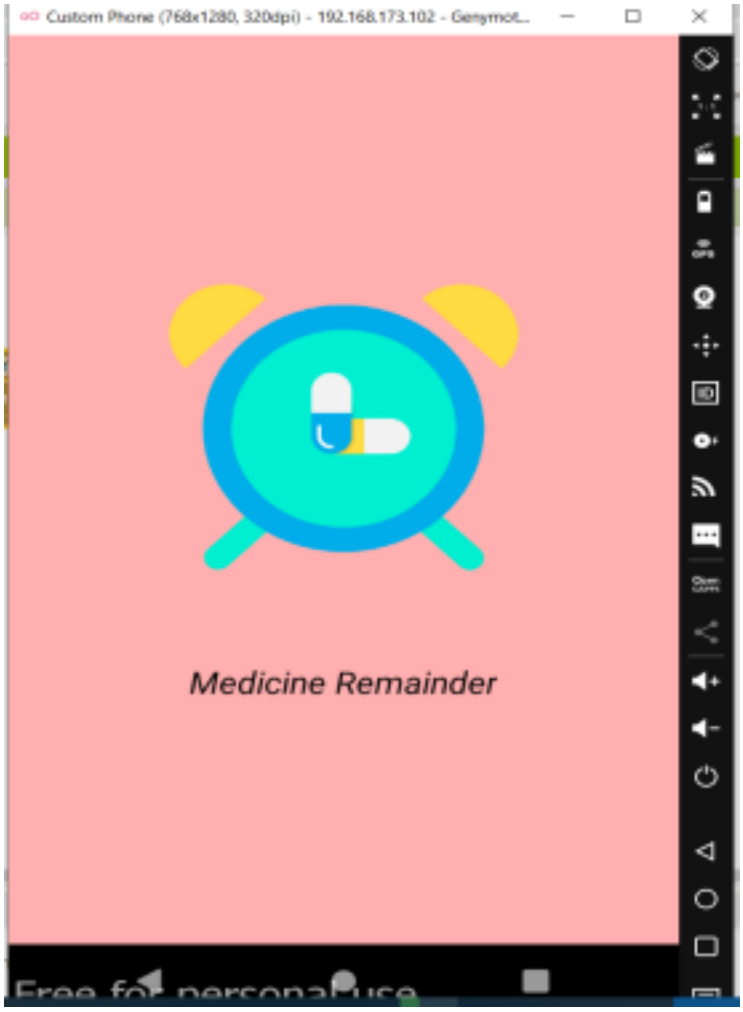


Link:

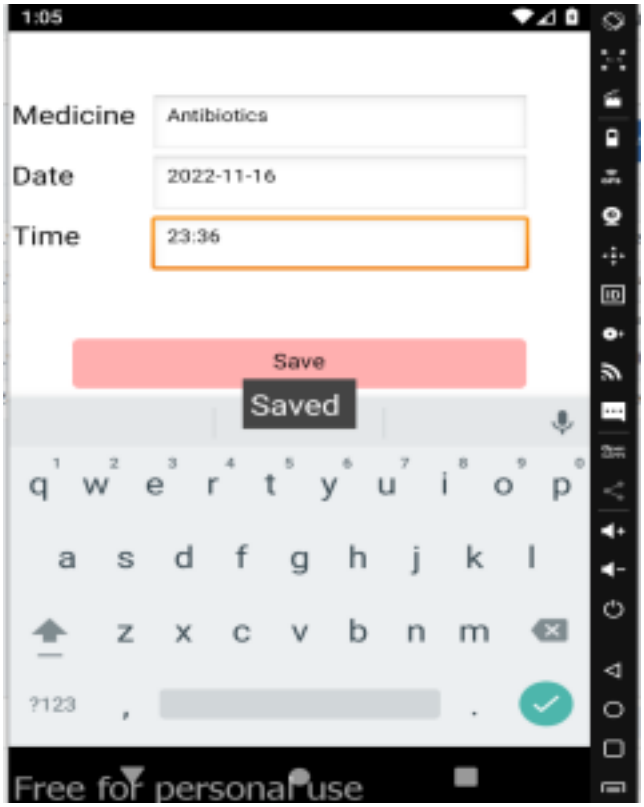
https://wokwi.com/projects/348198638815543891

Mobile Application

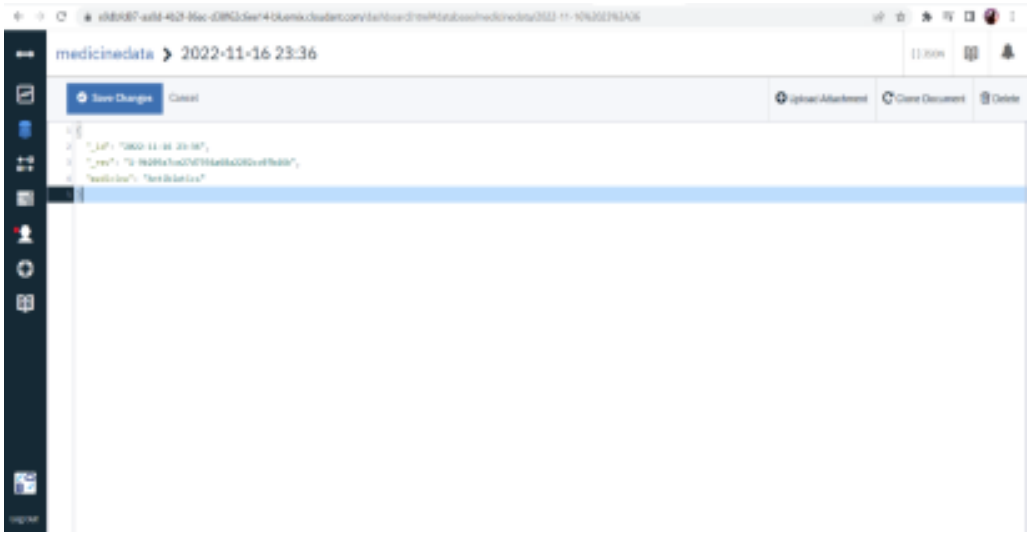
1. Splash Screen



2. Get Data From User

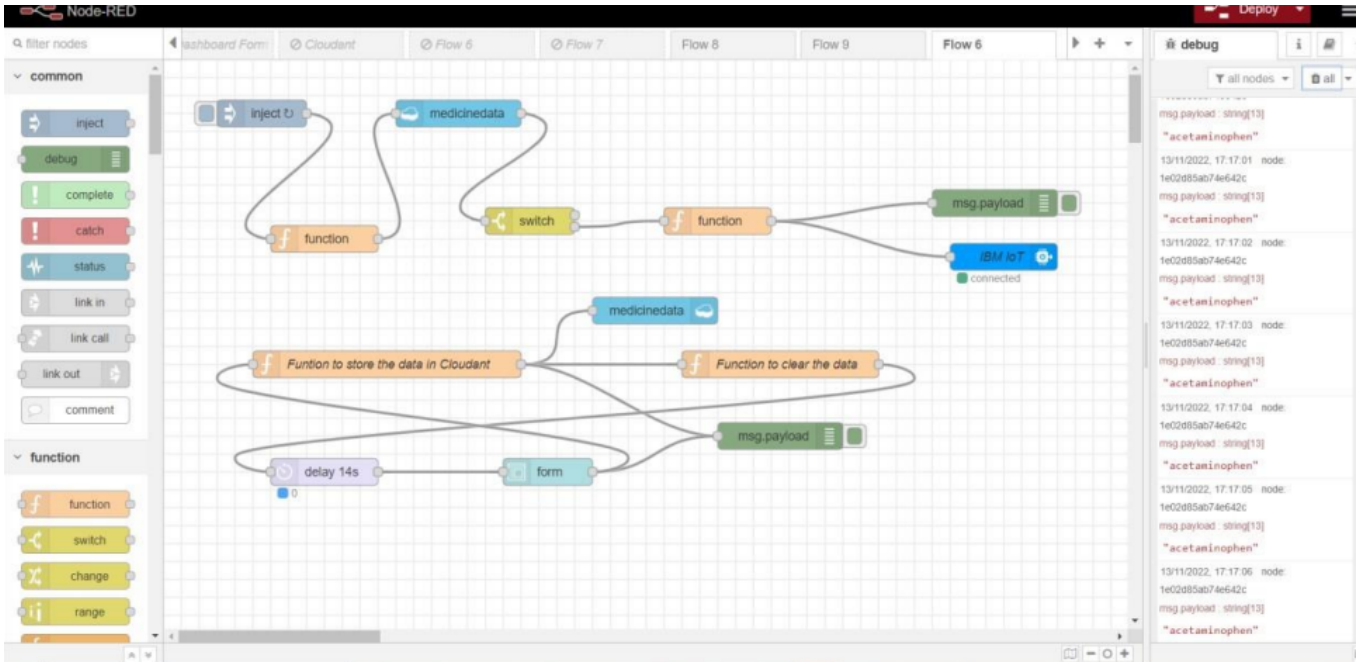


3. Store in Cloudant DB

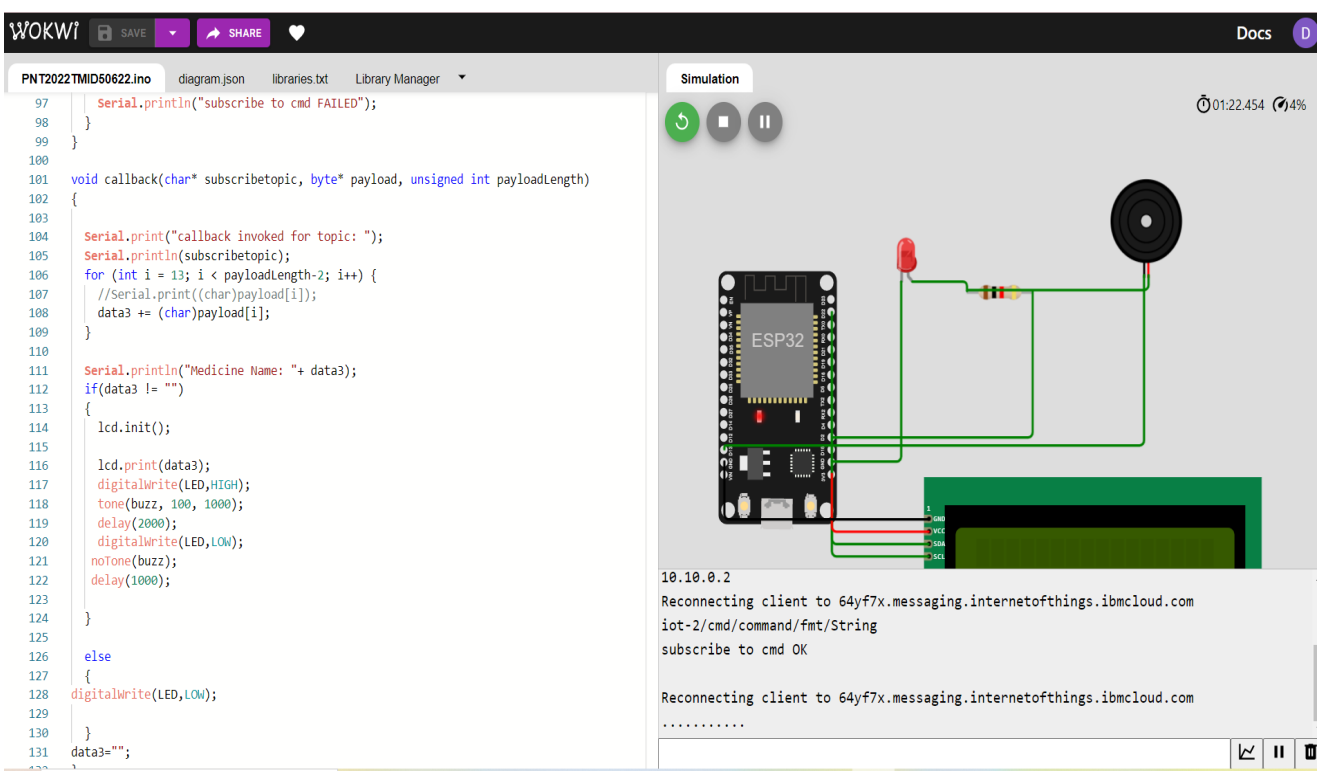


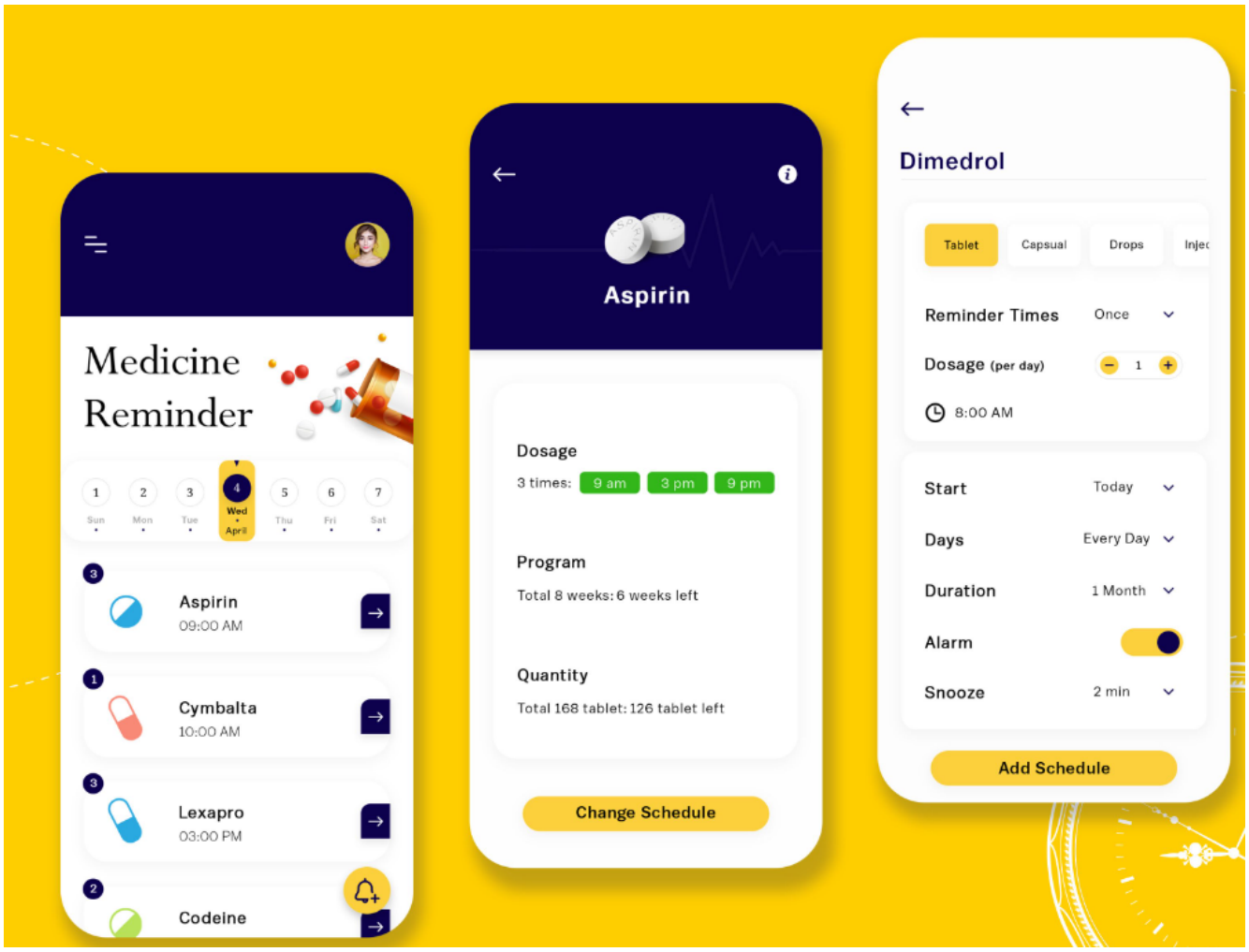
4. Display Remainder with audio

5. Display in node-red



6. Remainder in Simulation





GITHUB: https://github.com/IBM-EPBL/IBM-Project-20846-1659764798

CONCLUSION:

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