

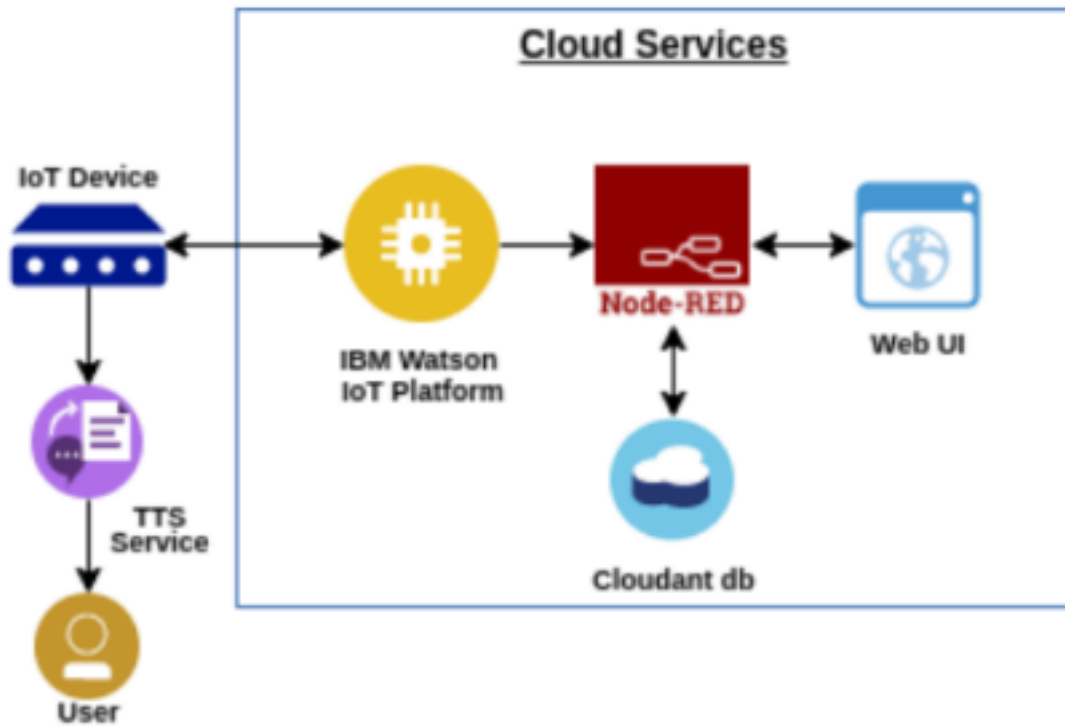
Final Deliverables

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 note 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	6x15xp
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

 dharanigk02@gmail.com
ID: 6x15xp

[← 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

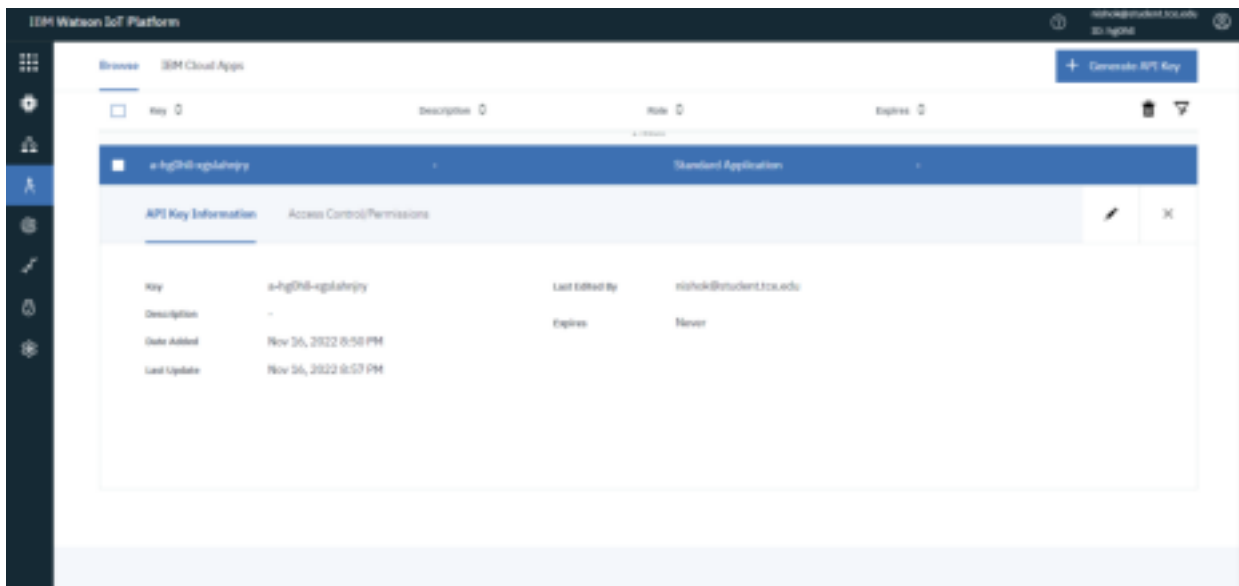
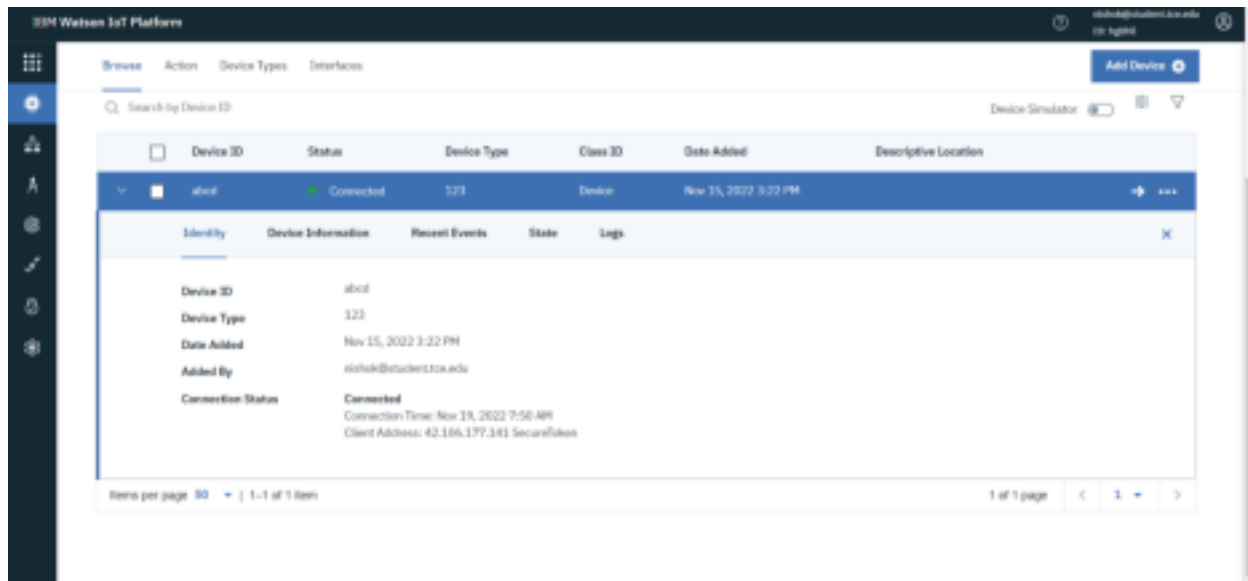
 Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.

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](#)

Manage

Service credentials

Plan

Connections

Overview

Capacity

Docs

Launch Dashboard

Deployment details

CRN

cm:v1:bluemix:public:cloudantnosqldb:au-syd:a/9154bf298d244d8c918065a7b7ae513f1df6581-7567-4252-8858-61365168e092::

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](#)

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

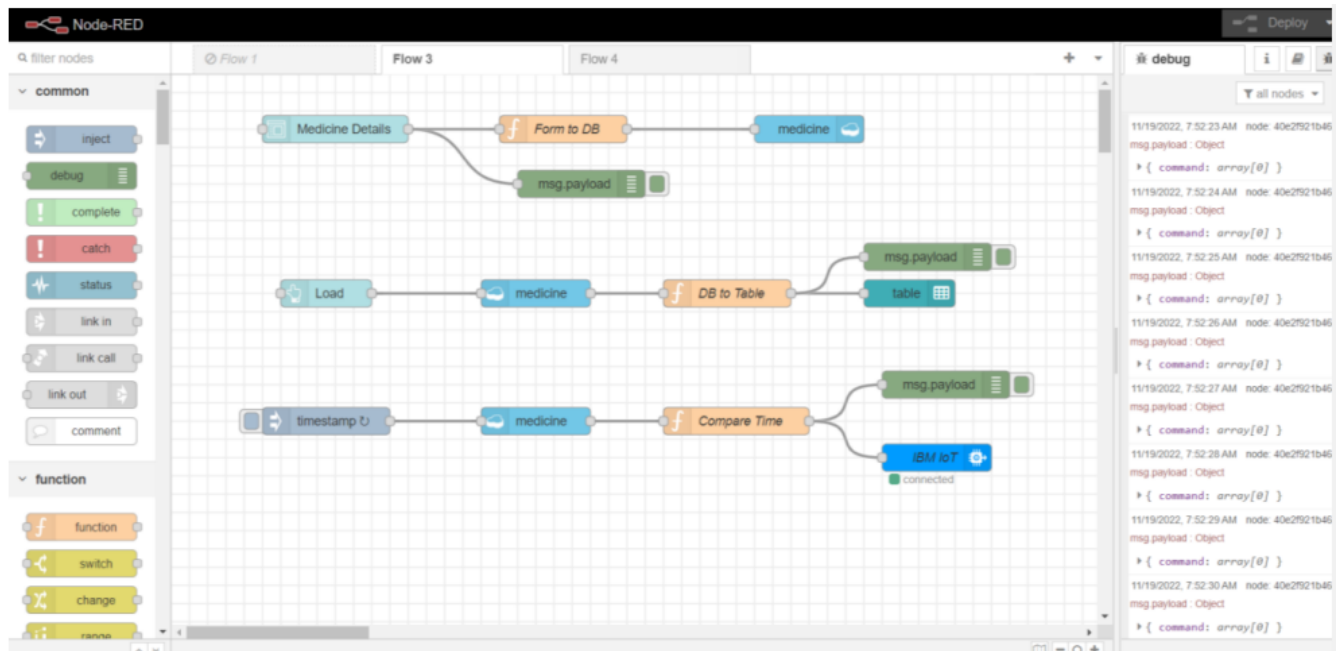
Create service

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 use 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.

Delete
Cancel
Done

Properties

Group
[Remainder] Medicien

Size
auto

Label
Medicine Details

Form elements

	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

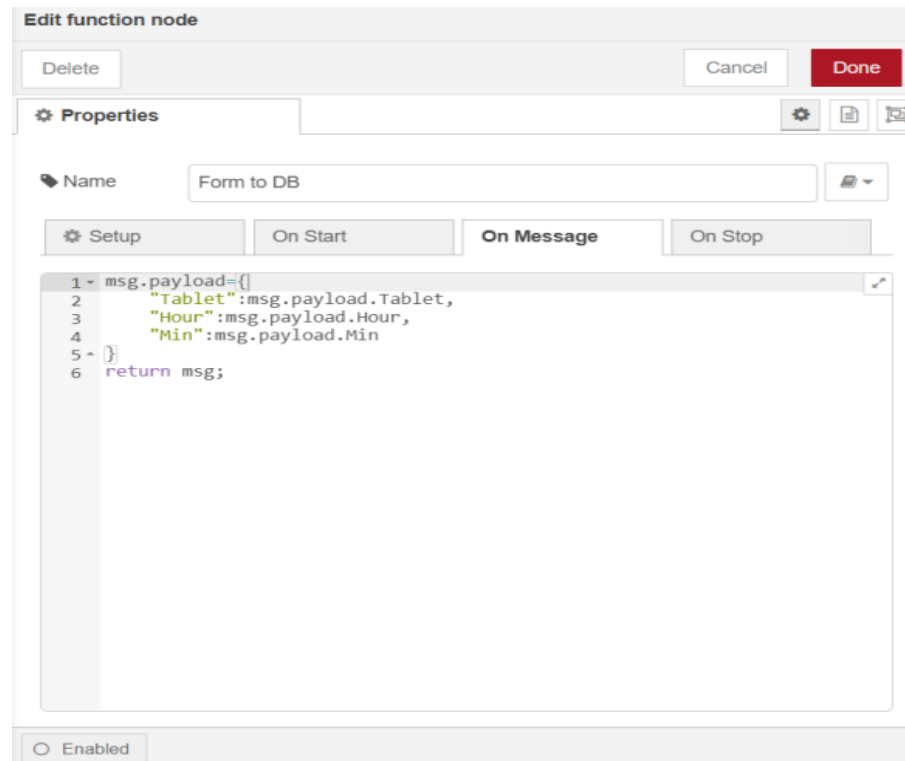
2. Function Node:

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:

Edit function node

Delete

Cancel

Done

Properties

Name

DB to Table

Setup

On Start

On Message

On Stop

1 var m = [];

2 var c=msg.payload;

3 if (msg.payload.length>0){

4 for (let i = 0 ; i<msg.payload.length; i++){

5 m.push({"Tablet":c[i].payload.Tablet,"Hour":c[i].payload.Hour,"M

6 }

7 msg.payload=m

8 return msg;

9 }

Enabled

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.

Edit cloudant out node

Delete Cancel Done

Properties

Service node-red-zhhhd-2022--cloudant-1666359856

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.

Edit cloudant in node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖨

Service

node-red-zhhhd-2022--cloudant-166635985€ ▾

🗄 Database

medicine

🔍 Search by

all documents ▾

🔑 Name

Name

☐ Enabled

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.

Edit table node

Delete Cancel Done

Properties

Group [Remainder] Tablet List

Size auto

Name Name

Columns Send data on click ☐

+ add

Enabled

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

Edit ibmiot out node

Delete
Cancel
Done

Properties

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.

Database name

Create Database

Your Databases

Name	Size	# of Docs	Partitioned	Actions
medicine	0.9 KB	7	No	
noderedzhhd20221021	33.2 KB	4	No	
test	0 bytes	0	No	
test1	359 bytes	3	No	

Showing 1–4 of 4 databases. Databases p

medicine

Document ID

Options

1/200

bell

All Documents

Query

Permissions

Changes

Design Documents

Table

Metadata

JSON

Create Document

	_id	payload	socketId	topic
	2f220b7493ebefed3a389w9bbd...	[{"table": "Paracetamol", "Hour": "...	F4QzOnWgYwAh6CAAM	[{"table": "Paracetamol", "Hour": "...
	602ee3390bdcac7a18ec995c9...	[{"table": "Dolo 650", "Hour": 8, "...	F4QzOnWgYwAh6CAAM	[{"table": "Dolo 650", "Hour": 8, "...
	62983547320b4ecda430a2929...	[{"table": "Deplata", "Hour": 17, "...	D0Fb8_3kByUp0AH9AAAs	[{"table": "Deplata", "Hour": 17, "Ph...
	9a5d91a280e03ed2b53454430f...	[{"table": "Mikar", "Hour": 17, "Ph...	D0Fb8_3kByUp0AH9AAAs	[{"table": "Mikar", "Hour": 17, "Ph...
	a96b38e7e2f5eb2f563444c88...	[{"table": "Concor", "Hour": 17, "...	D0Fb8_3kByUp0AH9AAAs	[{"table": "Concor", "Hour": 17, "Ph...
	f04caf9f9eacbf4af776e2672ff...	[{"table": "Cardace", "Hour": 17, "...	D0Fb8_3kByUp0AH9AAAs	[{"table": "Cardace", "Hour": 17, "Ph...
	f7bc3a6f18a517b3a395c54ef9...	[{"table": "Rosover", "Hour": 17, "...	D0Fb8_3kByUp0AH9AAAs	[{"table": "Rosover", "Hour": 17, "Ph...

Showing 4 of 5 columns. Show all columns.

Showing document 1 of 7. Documents per page: 20


```

import ibmiotf.device
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
}
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(url)

# Text To Speech Conversion
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

        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")

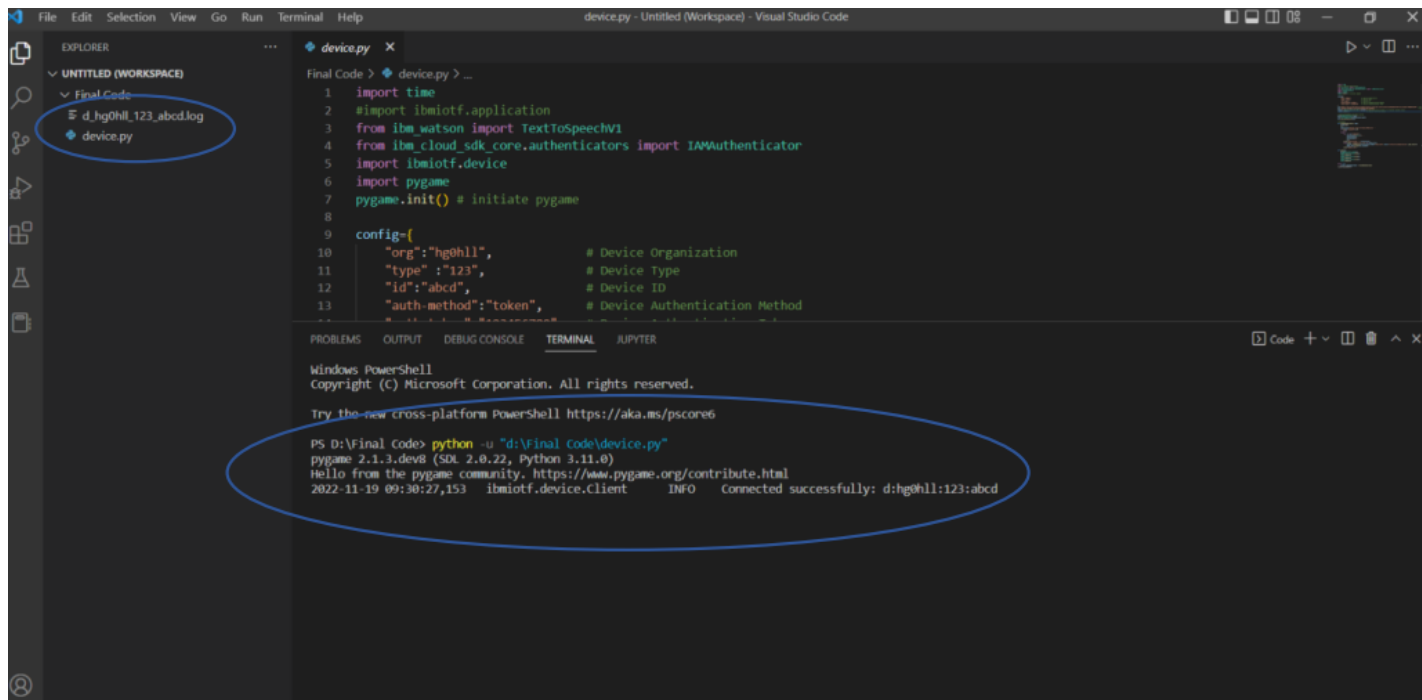
    def play(a):
        p=pygame.mixer.Sound(a)
        pygame.mixer.Sound.play(p)
        time.sleep(20)
        pygame.mixer.Sound.play(p)
        time.sleep(20)
        pygame.mixer.Sound.play(p)
        time.sleep(20)

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

```

RESULT:

1. Connect with the device.



The screenshot shows the Visual Studio Code interface. In the Explorer on the left, the file `device.py` is selected. The main editor displays the code for `device.py`, which imports `time`, `ibmiotf.application`, `ibm_watson.TextToSpeechV1`, `ibm_cloud_sdk_core.authenticators.IAMAuthenticator`, `ibmiotf.device`, and `pygame`. It also initializes `pygame` and defines a configuration dictionary with organization, type, ID, and authentication method. The terminal at the bottom shows the command `python -u "d:\Final Code\device.py"` being executed, resulting in a successful connection message: `INFO connected successfully: d:hg0hl1:123:abcd`.

```
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 }
```

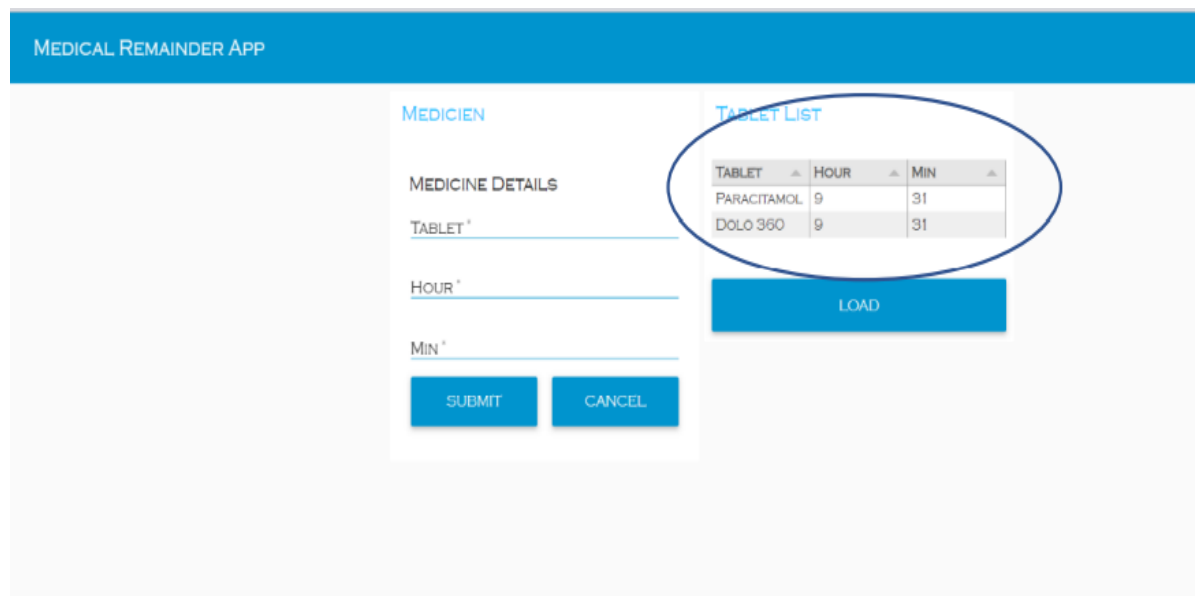
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS D:\Final Code> python -u "d:\Final Code\device.py"
pygame 2.1.1.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:hg0hl1:123:abcd

2. Load Tablet name and time in User UI

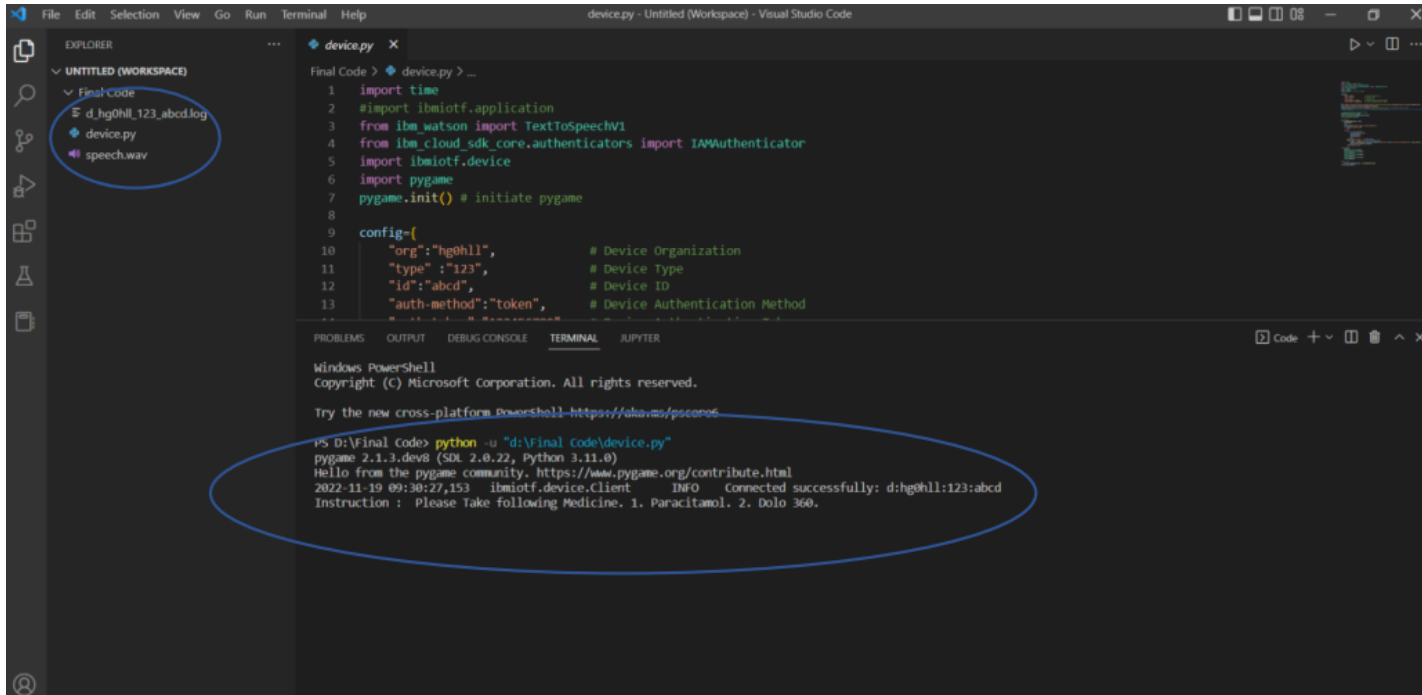
We load Two tablet Paracetamol and Dolo 650 set remainder and time 9.31 Am



The screenshot shows the 'MEDICAL REMAINDER APP' interface. It features a 'MEDICINE DETAILS' section with input fields for 'TABLET', 'HOUR', and 'MIN', and 'SUBMIT' and 'CANCEL' buttons. To the right, a 'TABLET LIST' section displays a table of loaded tablets. The table has columns for 'TABLET', 'HOUR', and 'MIN'. The loaded tablets are 'PARACETAMOL' (9 hours, 31 minutes) and 'DOLO 360' (9 hours, 31 minutes). A 'LOAD' button is located below the table.

TABLET	HOUR	MIN
PARACETAMOL	9	31
DOLO 360	9	31

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.



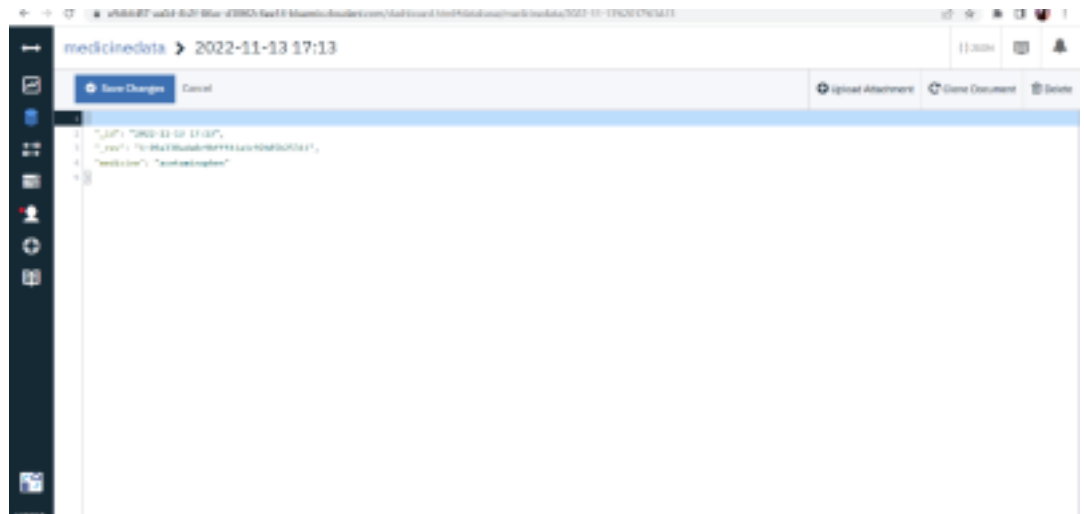
Web Application

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:

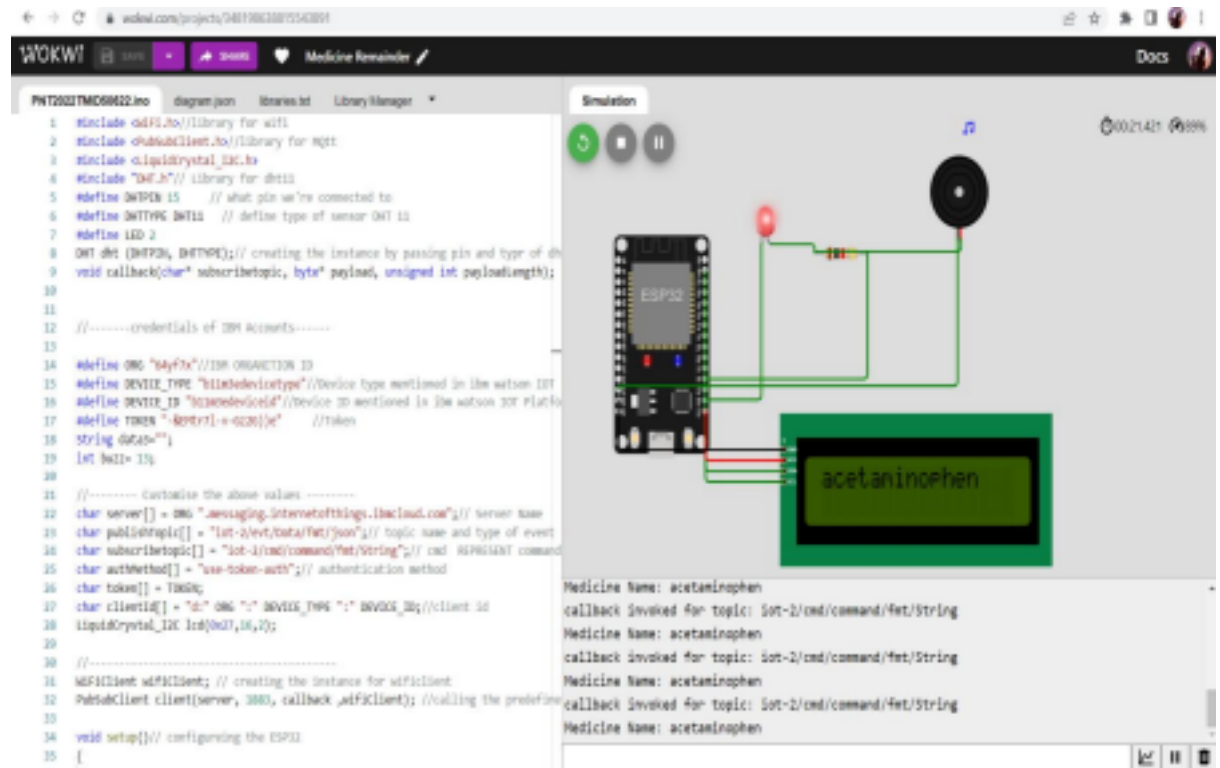
A screenshot of a web application interface. At the top, there is a blue header bar with the text 'Main'. Below the header, the main content area is light gray. In the center, there is a white form box titled 'Form'. The form contains four input fields: 'username' with the value 'anthonynophan', 'password' with the value '1234', 'date' with the value '2022-11-13', and 'time' with the value '17:13'. At the bottom of the form, there are two blue buttons: 'SUBMIT' and 'CANCEL'.

2. Stored in Cloudant



3. Display in Node-red

5. Simulation

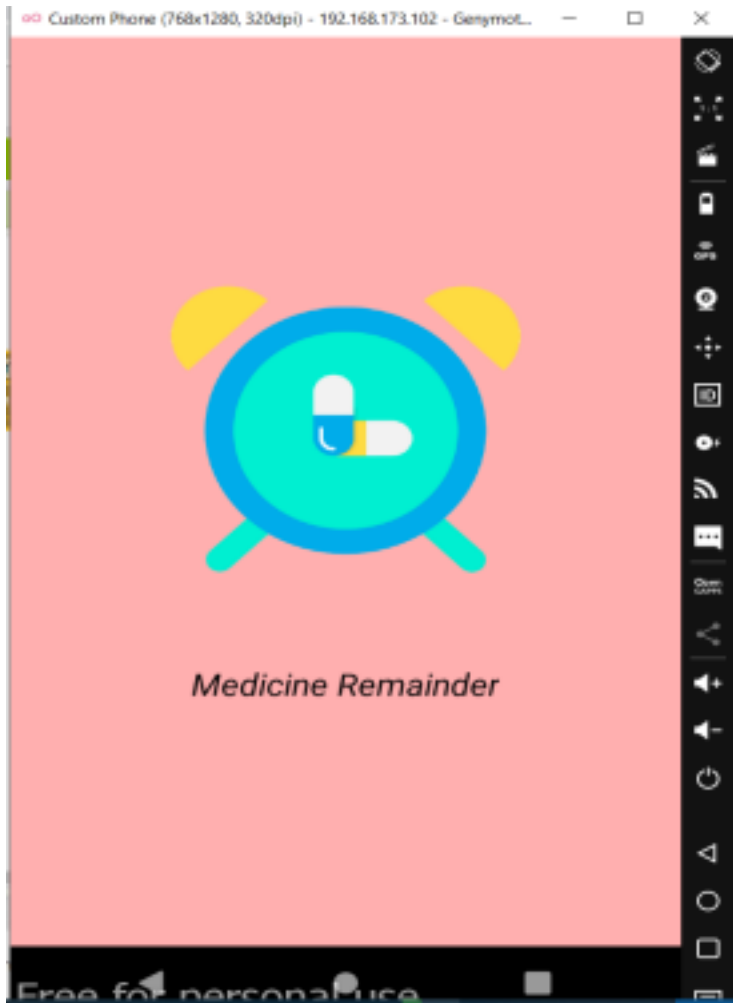


Link:

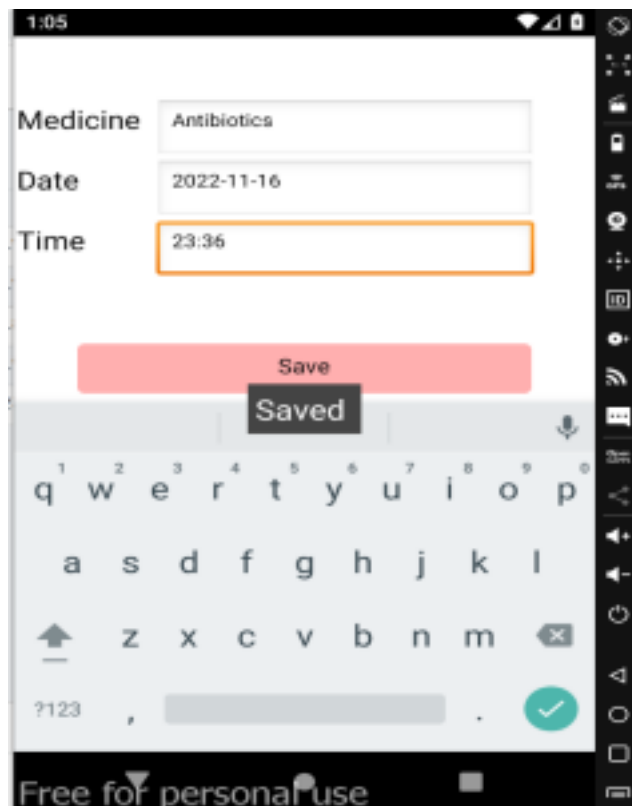
<https://wokwi.com/projects/348198638815543891>

Mobile Application

1. Splash Screen

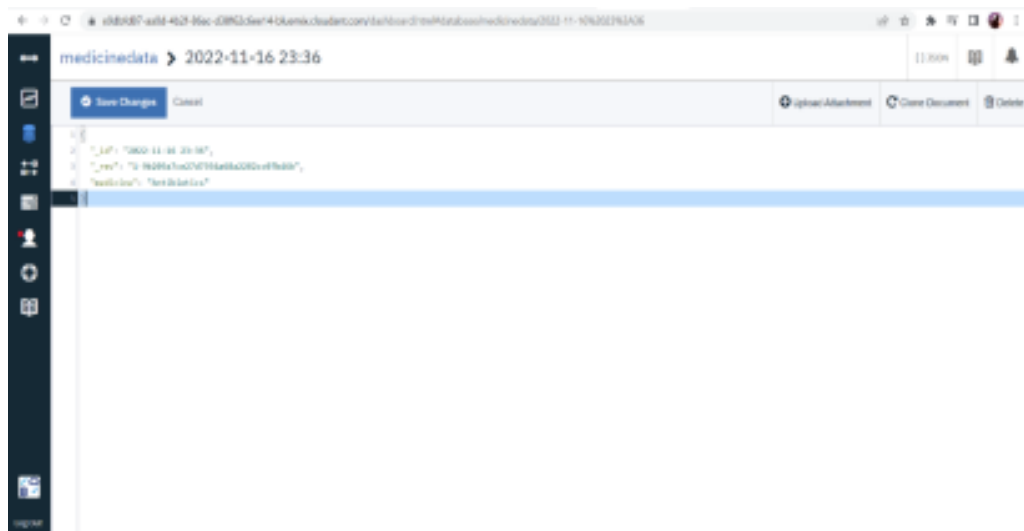


2. Get Data From User

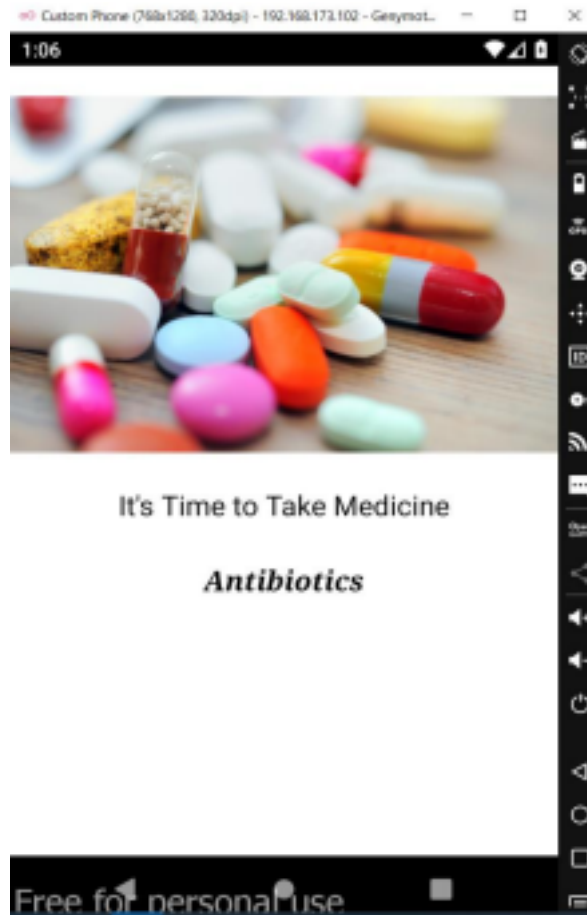


A screenshot of a mobile application interface for entering medicine data. The form has three input fields: "Medicine" with the value "Antibiotics", "Date" with the value "2022-11-16", and "Time" with the value "23:36". The "Time" field is highlighted with an orange border. Below the fields is a red "Save" button. A black "Saved" toast message is visible above the keyboard. The keyboard is open, showing a QWERTY layout. The status bar at the top shows the time "1:05" and various icons. The bottom of the screen has a black bar with the text "Free for personal use".

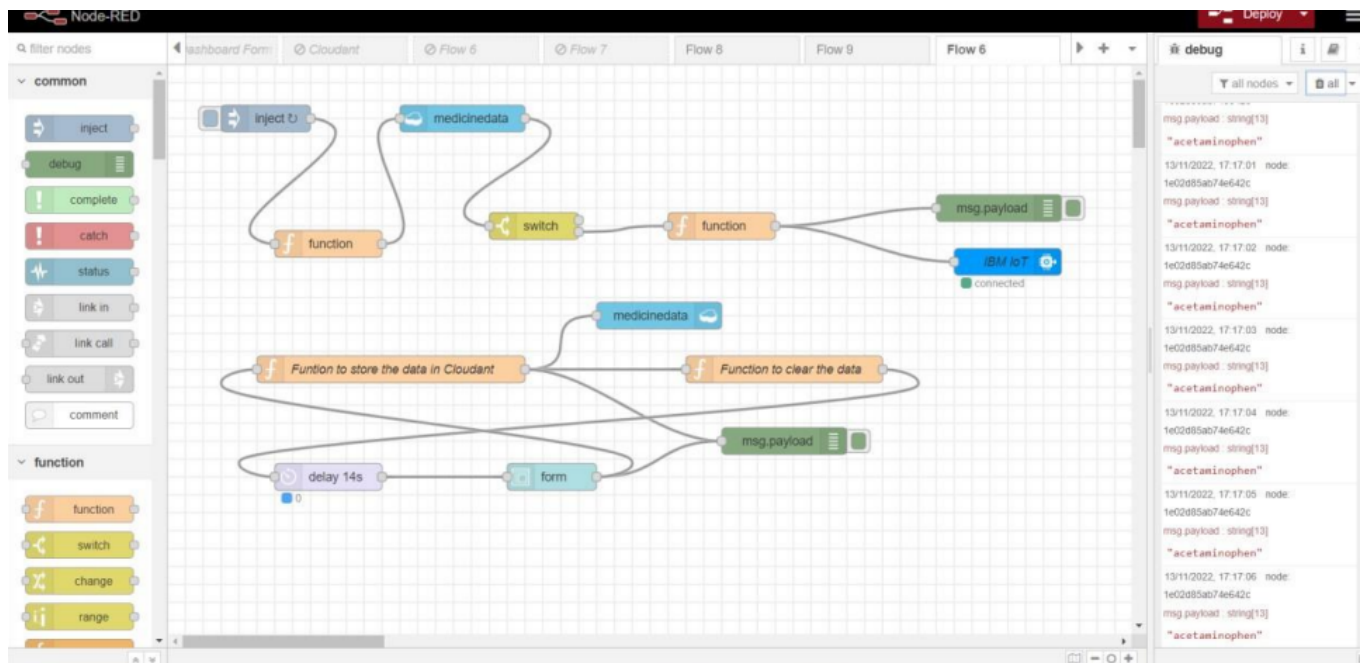
3. Store in Cloudant DB



4. Display Remainder with audio



5. Display in node-red



6. Remainder in Simulation

WOKWI

SAVE

SHARE

Docs

PNT2022TMD50622.ino

diagram.json

libraries.txt

Library Manager

```

97   Serial.println("subscribe to cmd FAILED");
98   }
99   }
100
101 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
102 {
103
104   Serial.print("callback invoked for topic: ");
105   Serial.println(subscribetopic);
106   for (int i = 13; i < payloadLength-2; i++) {
107     //Serial.print((char)payload[i]);
108     data3 += (char)payload[i];
109   }
110
111   Serial.println("Medicine Name: " + data3);
112   if(data3 != "")
113   {
114     lcd.init();
115
116     lcd.print(data3);
117     digitalWrite(LED,HIGH);
118     tone(buzz, 100, 1000);
119     delay(2000);
120     digitalWrite(LED,LOW);
121     noTone(buzz);
122     delay(1000);
123
124   }
125
126   else
127   {
128     digitalWrite(LED,LOW);
129   }
130
131   data3="";
132

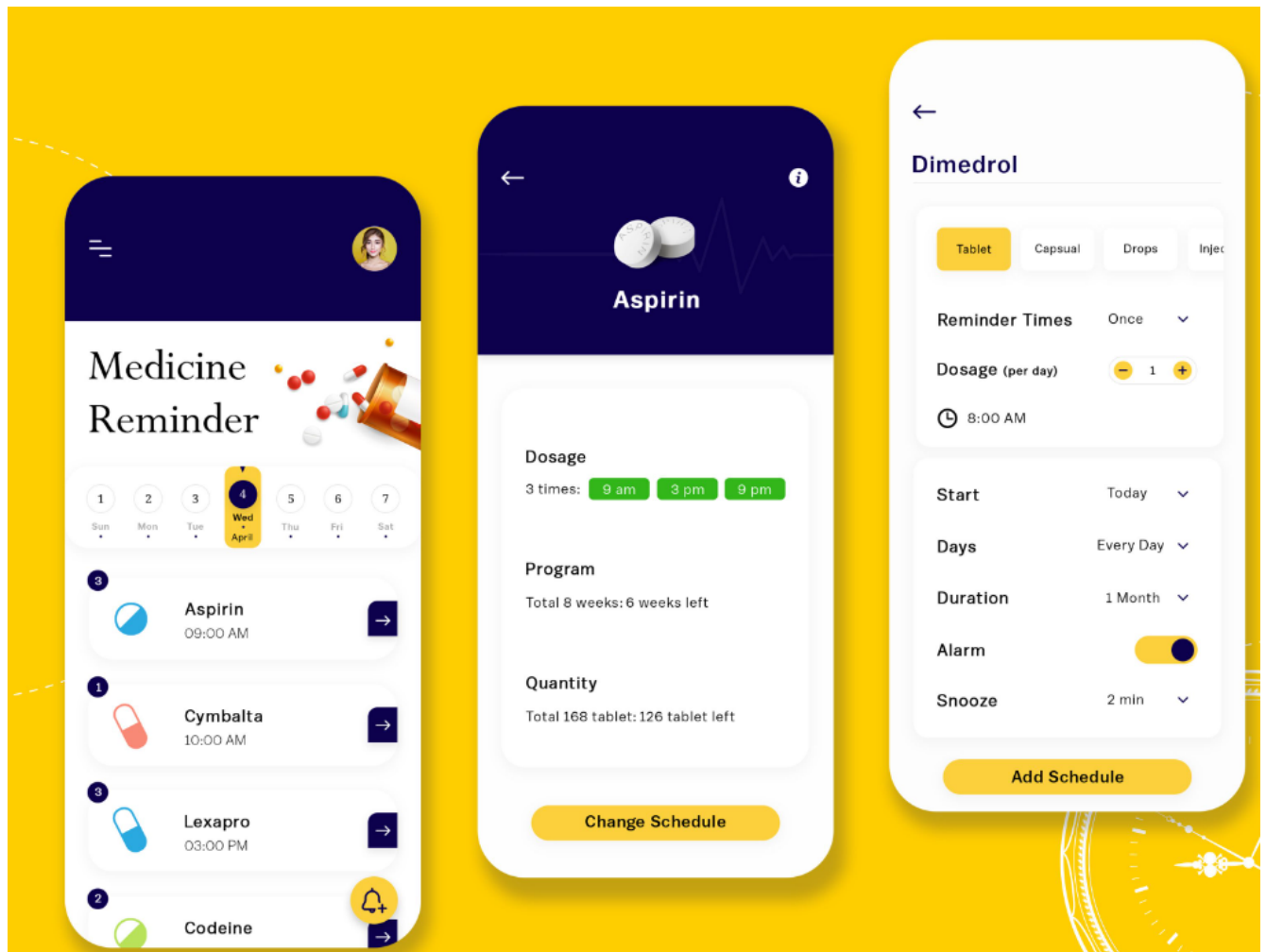
```

Simulation

01:22.454 4%

10.10.0.2
Reconnecting client to 64yf7x.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK

Reconnecting client to 64yf7x.messaging.internetofthings.ibmcloud.com
.....



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.