

**PERSONAL ASSISTANT FOR SENIORS WHO ARE SELF  
RELIANT**

**IBM PROJECT REPORT**

**Team ID - PNT2022TMID36821**

**SUBMITTED BY**

<b>R ANITHA</b>	<b>411819205706</b>
<b>S SHRUTHI LAYA</b>	<b>411819205705</b>
<b>S VISHAL</b>	<b>411819205703</b>
<b>S VISHWA</b>	<b>411819205702</b>

**FROM**

**RRASE COLLEGE OF ENGINEERING ,PADAPPAI-603 301  
ANNA UNIVERSITY : CHENNAI 600 025**

**In fulfillment of project in IBM -NALAIYATHIRAN 2022**

**Team Id: PNT2022TMID36821**

**PROJECT GUIDES**

**Industry Mentor : KUMAR JULURI**

**Faculty mentao : A VIJAYAN**

# **INTEX**

## **1.INTRODUCTION**

### **1.1 Project Overview**

### **1.2 Purpose**

## **2.LITERATURE SURVEY**

### **2.1 Existing problem**

### **2.2 References**

### **2.3 Problem Statement Definition**

## **3.PROJECT DESIGN AND PLANNING**

### **3.1 Ideation Phase**

#### **3.1.1 Brainstorming Idea Clear Image**

#### **3.1.2 Define Problem Statement**

#### **3.1.3 Empathy Map**

#### **3.1.4 Literature Survey**

### **3.2 Project Phase I**

#### **3.2.1 Solution Architecture**

#### **3.2.2 Problem Solution Fit**

### **3.3 Project Phase II**

#### **3.3.1 Customer Journey Map**

#### **3.3.2 Data Flow Diagrams&User Stories**

#### **3.3.3 Solution Requirementa**

#### **3.3.4 Technology Stack**

### **3.4 Project Planning**

#### **3.4.1 Milestone Template Ibm**

#### **3.4.2 Project Planning**

## **4.DEVELOP A WEB APPLICATION USING NODE RED APP**

### **4.1 Creat A From**

## **5.PROJECT DEVELOP PHASE**

### **5.1 Sprint 1**

### **5.2 Sprint 2**

### **5.3 Sprint 3**

### **5.4 Sprint 4**

## **6.PREREQUISITES**

### **6.1 Software**

### **6.2 IBM cloud service**

## **7.FINAL DELIVERABLES**

### **7.1 Final deliverable**

### **7.2 Device**

## **8.RESULTS**

### **8.1 Performanc Metrics**

## **9.ADVANTAGES& DISADVANTAGES**

## **10.CONCLUSION**

## **11.FUTURE SCOPE**

## **12.APPENDIX**

### **12.1 Source Code**

### **12.2 GitHub & Project Demo Link**

# **Personal Assistance for Seniors Who Are Self-Reliant**

## **Project Overview**

### **Introduction**

- 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.

### **Purpose**

- 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.

### **1. Literature survey**

### **Existing problem**

Elderly people let slip the medications at the correct time and the existing solutions for this problem is setting reminders or using pill boxes, calendars, Personal Assistance. Though the solutions give reminders, the voice commands or assistance given by this system is more efficient.

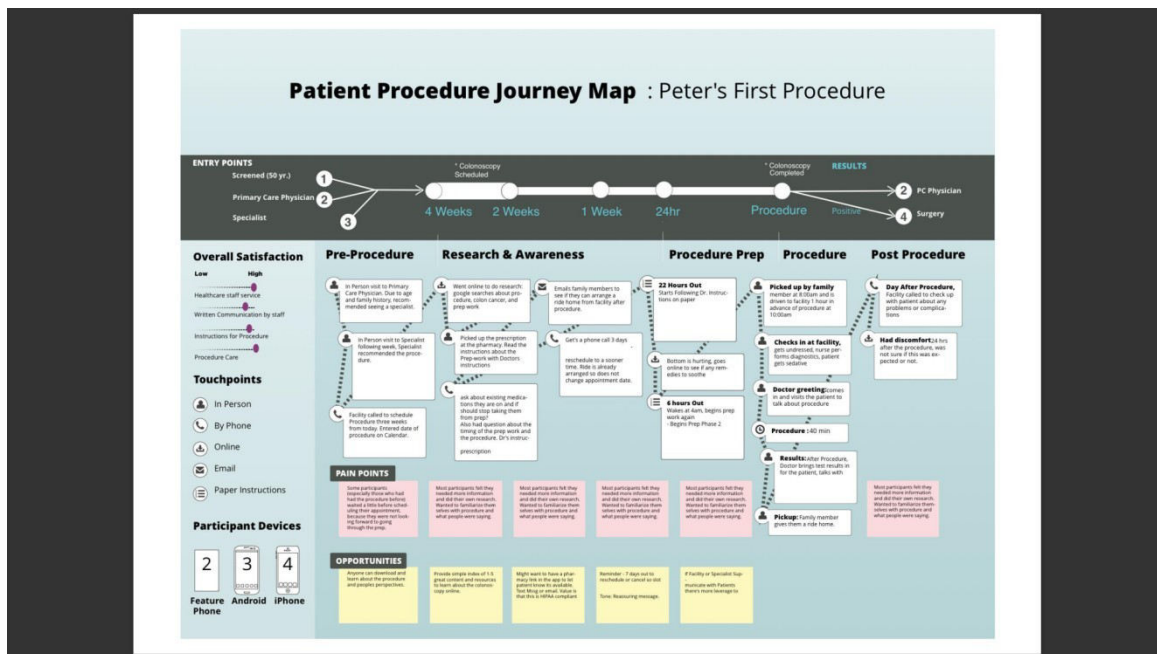
## Problem statement definition

Skipping medicines can be serious for some medical health conditions; Sometimes elderly people forget to take their medicine at the correct time. They also forget which medicine one should take at that particular time. And it is difficult for doctors/caretakers to monitor the patients around the clock.

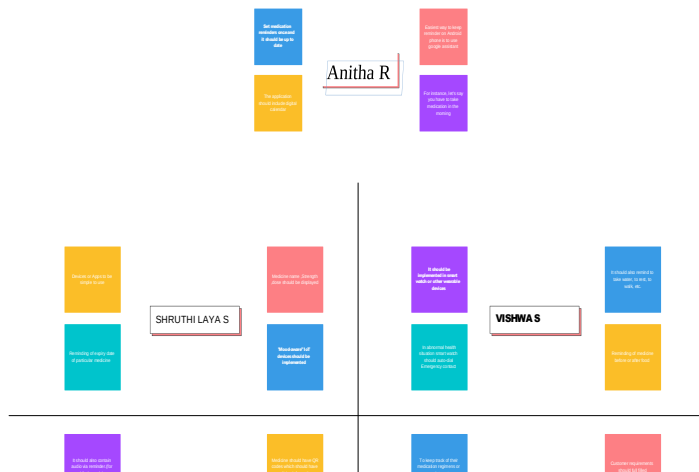
## 3. PROJECT DESIGN AND PLANNING

### 3.1 Ideation phase

### Empathy Map Canvas



## Brainstorming Idea Clear Image



## Define Problem Statement

### **PROBLEM STATEMENT**

A medicine reminder app designed for people who frequently forget to take their medications. You may also keep track of your appointments. Its parental feature distinguishes it from other apps on the market, allowing you to keep track of and remotely assist your loved ones who find it difficult to utilize such an app with their reminders.

### **CUSTOMER PROBLEM STATEMENT:**

>> I am shankar,  
Age-48

I have low sugar and high blood pressure.

>> I am trying to:

remind to take injectable glucagon for low sugar And enalapril, lisinopril, perindopril and ramipril for high bp

>>But :

lifestyle challenges, patient incompatibility, forgetting of medicine use, and nonexpert advice.

>>Because:

Due to patient Forgot, it will risk health of patient .

>>Which makes me feel:

Worry about patients health for careless of taking medicine due to forgotten .



Based on wireless sensor networks and telemedicine technology(2014), Vital sign monitor can be implemented with Bluetooth technology which is embedded with sensor, the transmitter will include the application oriented smart phone enable with 3G or IEEE 802.11 i.e. wi fi based transmission. The data from transmitter will be sending to cloud for centralized monitoring takes place; the expert in remote place can view all patient data and in case of emergency can take appropriate action. Ajmal Sawand et al<sup>1</sup> proposed Multidisciplinary approaches to achieving efficient and trustworthy eHealth monitoring systems(2014),The technological merging between IOT, wireless body area network and cloud computing have vital contribution in e health care which improve the quality of medical care, basically patient centric monitoring play a role in e health care services which involve medical data collection, aggregation, data Transmission and data analysis here entire monitoring lifecycle and essential services component have discus as well as design challenges in designing the quality and patient centric monitoring scheme along with potential solution. Huang et al<sup>8</sup> proposed the intelligent pill box—Design and implementation (2014), the implementation of pill box has proposed by keeping the problems of old age people in mind to provide full medication safety. The pill box will remind the patient about timing by doing this drug abusing can be controlled. Al-Majeed et al<sup>10</sup> proposed Home telehealth by Internet of Things (IoT) (2015), The real time monitoring can be possible through IOT which helps in development of low cost medical sensing, communication and analytic devices

Which make quality of life, in case of density of messages there is fear of information degradation but by using proper algorithm we can resolve the problem and can make the low cost imaging, sensing and human computer interaction technology. Lin et al<sup>9</sup> proposed A Selfpowering Wireless Environment Monitoring System Using Soil Energy (2015), The monitoring system can uses the self-powering wireless environment with the help of renewable energy which can be beneficial in remote places where the power problem in wide manner, in this the system have demonstrated which will uses soil energy with carbon, zink electrodes. Moga et al<sup>11</sup> proposed Embedded platform for Web-based monitoring and control of a smart home (2015), Present the low cost embedded platform for web based monitoring and controlling and the platform consist of distributed sensing and control network and touch screen to easy use interface to the user and remote web based access.



## Project Design Phase-I

### Solution Architecture

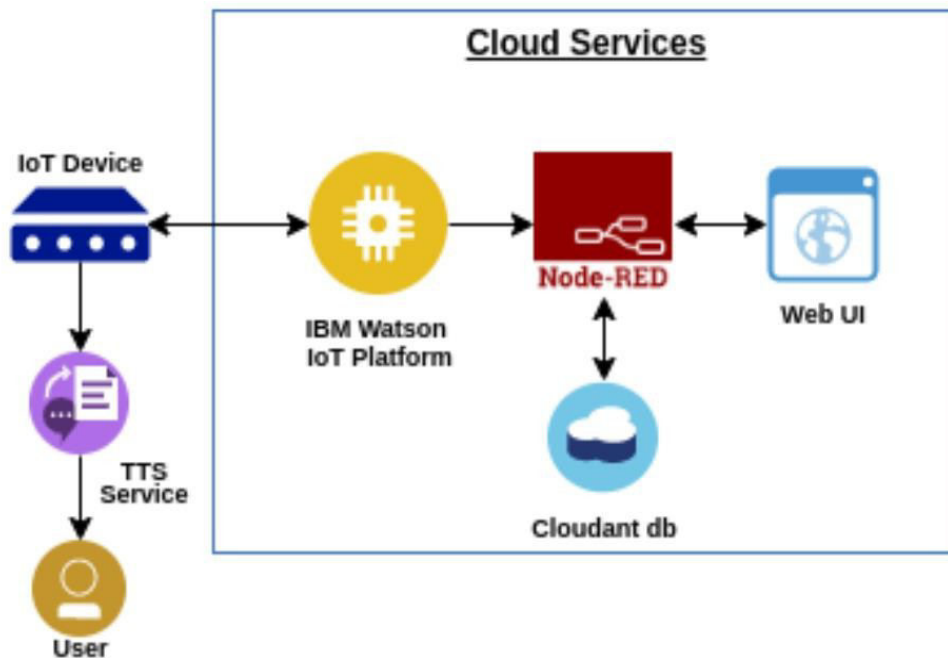
Date	18-11-2022
Team ID	PNT2022TMID36821
Project Name	Project – IOT (Medicine Remainder)
Maximum Marks	4 Marks

#### Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- ❖ Medicine Reminders serve as good way to stay on track and uphold appropriate schedule.
- ❖ It helps in decreasing medication dispensing errors and wrong dosages.
- ❖ It is used to organize your medication doses for a certain length of time.

#### Solution Architecture Diagram:



# Problem Solution fit

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> Our customers are people who require medical support; Also, our alert system can be used in hospitals and old age homes where people will require medical assistance.	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> Healthcare costs, lack of financial support, Difficulty with everyday tasks and mobility, Finding the right care provision and seclusion.	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> The existing solutions for this project is setting reminders or using pill boxes, calendar, Personal Assistance. Though, the solutions give reminders, the voice commands or assistance given by this system is more efficient.	Explore AS, differentiate
	<b>2. JOBS-TO-BE-DONE / PROBLEM</b> <span>J&amp;P</span> Skipping of medicines can be serious for some medical health conditions; in such cases this system would help the individual to take their medication on time.	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> Sometimes elderly people forget to take their medicine at the correct time. They also forget which medicine one 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.	<b>7. BEHAVIOUR</b> <span>BE</span> Directly related: To download the web application so that the user can receive voice notifications on the connected IoT device. Through this application, the user can set the details of the medicine name and other details. Indirectly associated: Customers can be carefree and don't need a person round the clock to check on them.	
Focus on J&P, fit into BE, understand RC	<b>3. TRIGGERS</b> <span>TR</span> There are applications which already exist that give regular reminders to take medicines.	<b>10. YOUR SOLUTION</b> <span>SL</span> 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.	<b>8. CHANNELS of BEHAVIOUR</b> <span>CH</span> <b>8.1 ONLINE</b> The customers should have the mobile application on their devices so that they can get regular voice commands. <b>8.2 OFFLINE</b> The customer should have the device or mobile near them. Also, the customer should update the schedule.	Identify strong TR and EM
	<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> With this application built, which gives voice commands and alerting system which is more efficient in helping the elderly to take their medicines on time and can be carefree.			

## Proposed solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Sometimes elderly people forget to take their medicine at the correct time. They also forget which medicine should be taken at that particular time. And it is difficult for doctors/caretakers to monitor the patients around the clock.
2.	Idea / Solution description	<ul style="list-style-type: none"><li>➤ A medicine reminder system is developed. An app is built for the user (caretaker) which enables him to set the desired time and medicine.</li><li>➤ 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.</li></ul>

		<ul style="list-style-type: none"> <li>➤ The device will receive the medicine name and notify the user with voice commands.</li> </ul>
3.	Novelty / Uniqueness	<ul style="list-style-type: none"> <li>➤ Keeping track of the medicines taken by the user at each time interval.</li> <li>➤ Information is stored in the secured IBM cloud.</li> </ul>
4.	Social Impact / Customer Satisfaction	The reminder system enables the user to take tablets at regular intervals prescribed by the physicians.
5.	Business Model (Revenue Model)	<p><b>Direct Mode:</b> We gain revenue from selling the medical reminder system to hospitals, medical health centres and even in old age homes.</p> <p><b>Indirect Mode:</b> We gain profit by having partnership with pharmaceutical companies.</p>
6.	Scalability of the Solution	The medical alert system can be used in hospitals, medical health centres and even in old age homes for dispensing medicines.

### 3. Requirement

#### analysis Functional

#### Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Access Cloud services	Accessing the cloud service with correct credentials. Storing the details in the cloud database.
FR-4	IOT configuration	Fine Tuning the IOT device based Cloud DB access via device. Manage the data request and response effectively

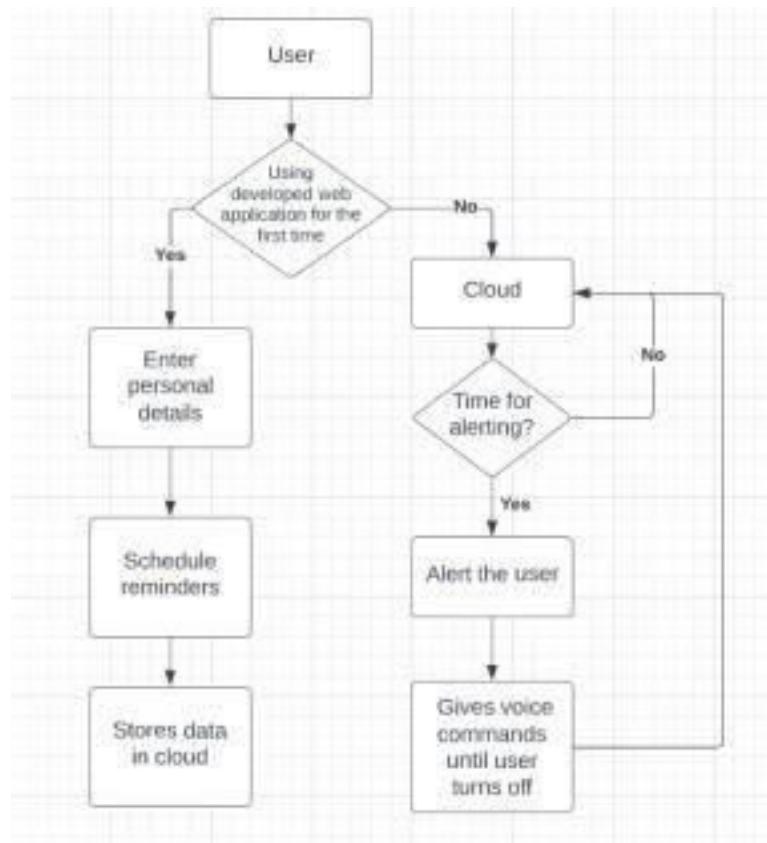
#### Non-functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	App can be used by anyone who has knowledge about applications and computers.
NFR-2	Security	For security, TFA is enabled and biometrics are also added for user safety.
NFR-3	Reliability	Highly reliable since, It uses trusted and authentic cloud services like IBM

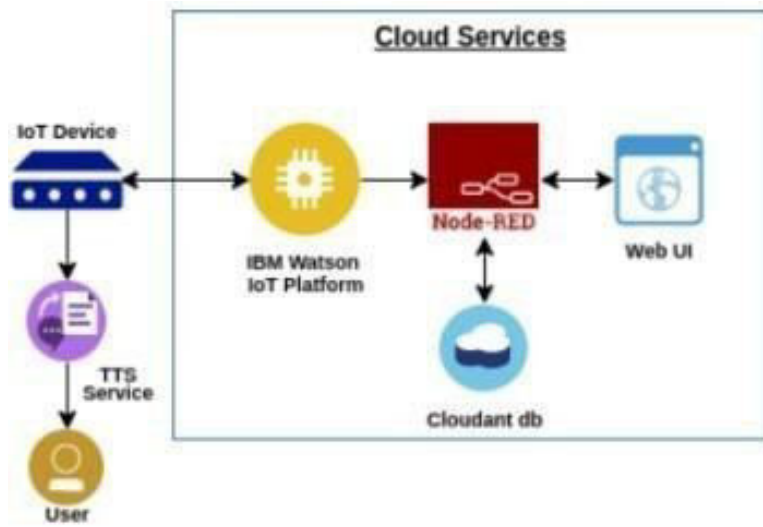
NFR-4	<b>Performance</b>	Performance is better compared to other marketproducts.
NFR-5	<b>Availability</b>	Available on mobile app.
NFR-6	<b>Scalability</b>	Using Cloud services, makes the scalability higher the using traditional locally stored database.

#### 4. Project Design

##### Data Flow Diagrams



## Technical architecture



## User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email or mobile number, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
Confirmation mail	Gmail	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
Accessing	Link	USN-3	As a user, I can register for the application through Gmail		Medium	Sprint-1
Medicine Name	Login	USN-4	As a user, I can log into the application by entering email or mobile	I can access my account /	High	Sprint-1

Entering Credentials	Medicine Name	USN -5	number & password	dashboard	High	Sprint-1
-------------------------	------------------	-----------	-------------------	-----------	------	----------



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
	Dashboard	USN-5	As a user, I can update my reminders and medicines wherever required		High	Sprint-2
		USN-6	As a user, I can check the application whether the medicine dosage is completed.		Medium	Sprint-2
Customer Care Executive		USN-7	For any troubleshooting, the user can send a mail to the technical team.		Low	
Administrator		USN-8	Ensures smooth functioning and data warehousing strategies		Medium	Sprint-3

## 5. Project Planning and

### Scheduling Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email or mobile number, password, and confirming my password.	2	High	Anitha
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the	1	High	Shruthilay

1			for the application through Gmail		m	
Sprint-1		USN-4	As a user, I can log into the application by entering email or mobile number & password	2	High	Anitha
Sprint-2	Login	USN-5	As a user, I can update my reminders and medicines wherever required	1	High	Shruthilaya
Sprint-2	Dashboard	USN-6	As a user, I can check the application whether the medicine dosage is completed	1	Medium	Vishal
		USN-7	For any troubleshooting, the user can send a mail to the technical team	1	Low	Vishwa

### Sprint Delivery Schedule

Sprint	Total Story Points	Duration Sprint Start Date End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	10 Days 25 Oct 2022 04 Nov 2022	20	04 Nov 2022
Sprint-2	20	5 Days 5 Nov 2022 10 Nov 2022	20	10 Nov 2022
Sprint-3	20	5 Days 11 Nov 2022 15 Nov 2022	20	16 Nov 2022
Sprint-4	20	2 Days 17 Nov 2022 18 Nov 2022	20	18 Nov 2022

## PROJECT DESIGN PHASE - II

### CUSTOMER JOURNEY MAP :

Date	18- 11 -2022
Team ID	PNT2022TMID36821
Project Name	Personal Assistance for Seniors Who Are Self-Reliant
Maximum Marks	4 Marks

# Customer Journey

Customer Journey Maps give an overview of the customer experience. How do you want your business to reach users?

MEDICINE REMINDER	ENTICE	ENTER	ENGAGE		EXIT
STEPS	Their insight into how their emotional makeup influences patient care.	Searching best Product on Market	Browsing the Best Product	Suitable for the customer Point of views	At the end of our customer Follow Proper Medication
INTERACTION	At the hospital By Caretakers	A Smart Medicine Box	Managing Patients Prescription	Reminding About the Insulin	Caretaker Free from 24/7 monitoring
GOALS	Solution For Proper medication Reminder	It Begins with the self care or patient care to take medicines regularly on time	They take the medication on time	The caretaker Takes care of Patient	At the End They find Smart Medicine Box
POSITIVE MOMENTS	Public Suggestions	User Friendly App Environment	Proper Notification Via Voice Command	App Notification to Caretaker	It Regularly Reminds the Medication Times
NEGATIVE MOMENTS	Hard To Find The Best Smart Medicine Box in the Market	Difficult to operate the Medic app	The user Should Keep the Product near to them	Always Wifi should be in on condition online	A Smart Medicine box with Complex Architecture Only fit Elderly people's

**PROJECT DESIGN PHASE -II**  
**SOLUTION REQUIREMENTS(Functional & Non-Functional)**

Date	18-11-2022
Team ID	PNT2022TMID36821
Project Name	Personal Assistance for Senior Who are Self - Reliant
Maximum Marks	4 Marks

**Solution Requirements (Functional & Non functional)**

**Functional Requirements:**

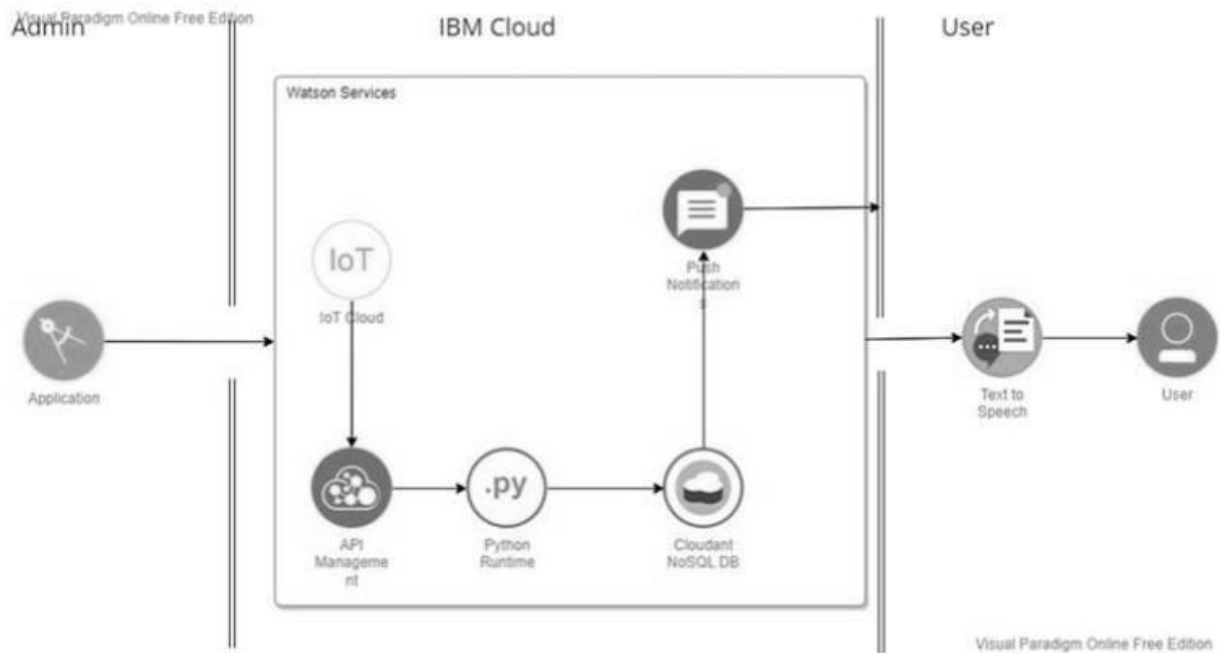
Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Access Cloud services	Access the cloud service with correct credentials Store the details in the database Retrieve needed information for the user's operation
FR-4	IOT configuration	Fine Tuning the IOT device based on preference Access the Cloud DB via device Manage the request and response effectively

## PROJECT DESIGN PHASE - II TECHNOLOGY STACK( ARCHITECTURE & STACK)

Date	18-11-2022
Team ID	PNT2022TMID36821
Project Name	Personal Assistance for Seniors Who Are Self-Reliant
Maximum Marks	4 Marks

### Technical Architecture :



## **Project Planning Phase Milestone**

Date	18-11-2022
Team ID	PNT2022TMID36821
Project Name	Personal Assistant For Seniors Who Are Self Reliant

### **Milestone Template :**

<b>Sprint</b>	<b>Sprint Topic</b>	<b>Start Date</b>	<b>Expected Delivery</b>
Sprint 1	Set alarm	29-10-2022	5-11-2022
Sprint 2	Notification	7-11-2022	14-11-2022
Sprint 3	Medication details	16-11-2022	23-11-2022
Sprint 4	GPS Tracking	23-11-2022	30-11-2022

**Project Planning Phase**  
**Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)**

Date	18-11-2022
Team ID	PNT2022TMID36821
Project name	Personal Assistant For Seniors Who Are Self Reliant

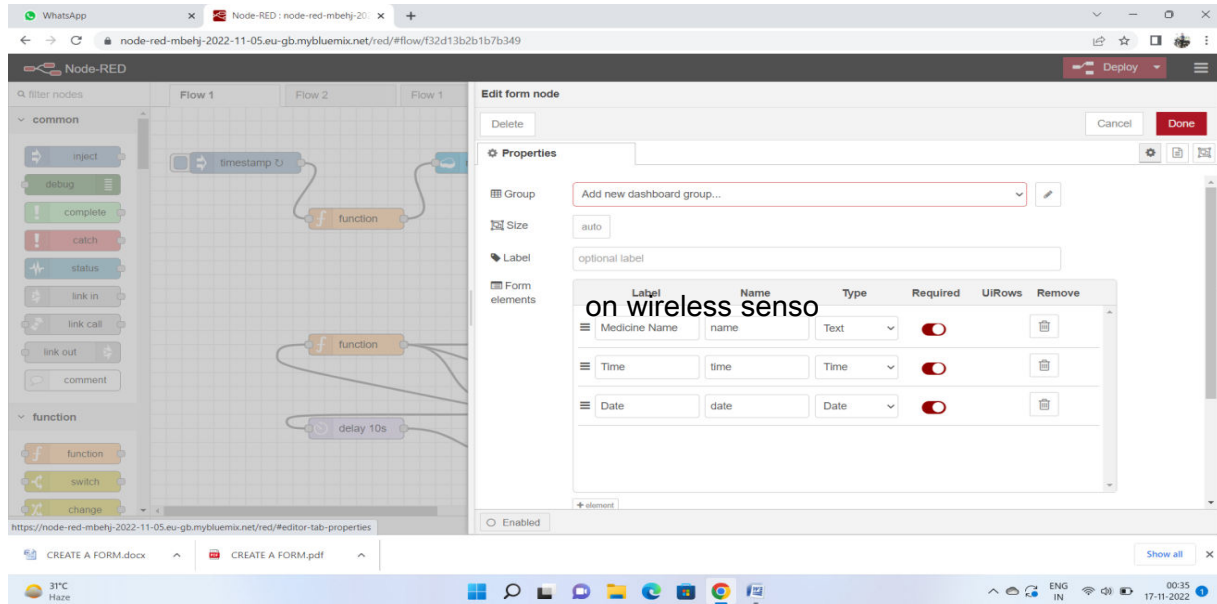
**Product Backlog, Sprint Schedule, and Estimation**

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story points	Priority	Team Members
Sprint 1	Set Alarm	USN-1	As a user, I can set an alarm to alerting a medicine through medicine remainder system	10	High	R.ANITHA
Sprint 1		USN-2	As a user, I can Activate and Deactivate the alarm	10	High	S.SHRUTHI LAYA
Sprint 2	Notification	USN-3	As a user once I can the set the alarm then I gets the notification	10	High	S.VISHAL
Sprint 2		USN-4	As a user, If I requires this system then a notification will be sent into his device.	10	High	S.VISHWA

## DEVELOP A WEB APPLICATION USING NODE RED APP

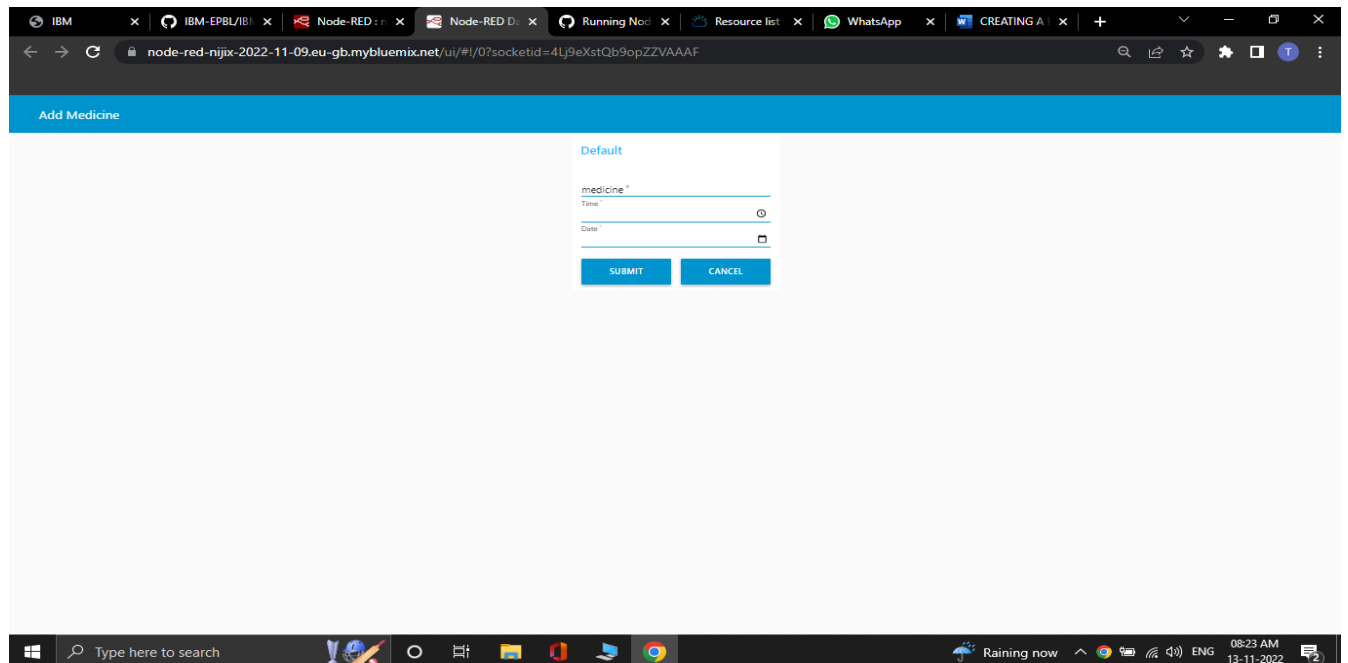
## CREATING A FORM IN NODE RED

## NODE RED – FORM CREATION



Click on Done after creating the form

## NODE RED – FORM USAGE





<b>TEAM ID</b>	<b>PNT2022TMID36821</b>
<b>PROJECT NAME</b>	<b>Personal Assistance for Seniors Who Are Self-Reliant</b>

## **Sprint 1**

**The AIM of Sprint 1 is SIMULATION CREATION.**

### **Duration : 6 days**

In this Sprint 1 we have found out what the elderly people used to and suffer in Alzheimer disease .So the we had decided that we should find the solution of four different Ideas from our teammates

- 1) Our teammate Ramesh P, Medicine Remainder for Alzheimer disease person to keep an instant remainder for their medicine that is taken by them and keep them on track of their medicine taken.
- 2) Our teammate Prasath V,A Medicine Remainder for elderly person that are elderly person in home and Hospital and keep them check in Medicine that are forgotten by Caretaker and Person around them.
- 3) Our teammate Praveen Kumar A, A Stock Remainder of Medicine for Elderly Person and keep the medicine in check every time if the medicine prolonged for another week and have the refilled alarm for the medicine that need to taken.
- 4) Our teammate Maria Antony B ,A Scan of medicine to zoom and see the medicine those are taken by the elderly person and keep in check of medication of elderly person. If needed the medication can be upload the App.

By considering all the ideas that are given teammate .We conclude that are of decision made by teammate is make a combine idea of Medicine Remainder App that feature of scanning of medicine , medicine remainder at correct timing, Stock remainder.

What we should complete in sprint 1?The Issue collected from user

USN 1:As a user, I want to take Medicines on time and monitor my health

USN 2:As a user, I want to take my tablets on time by voice command

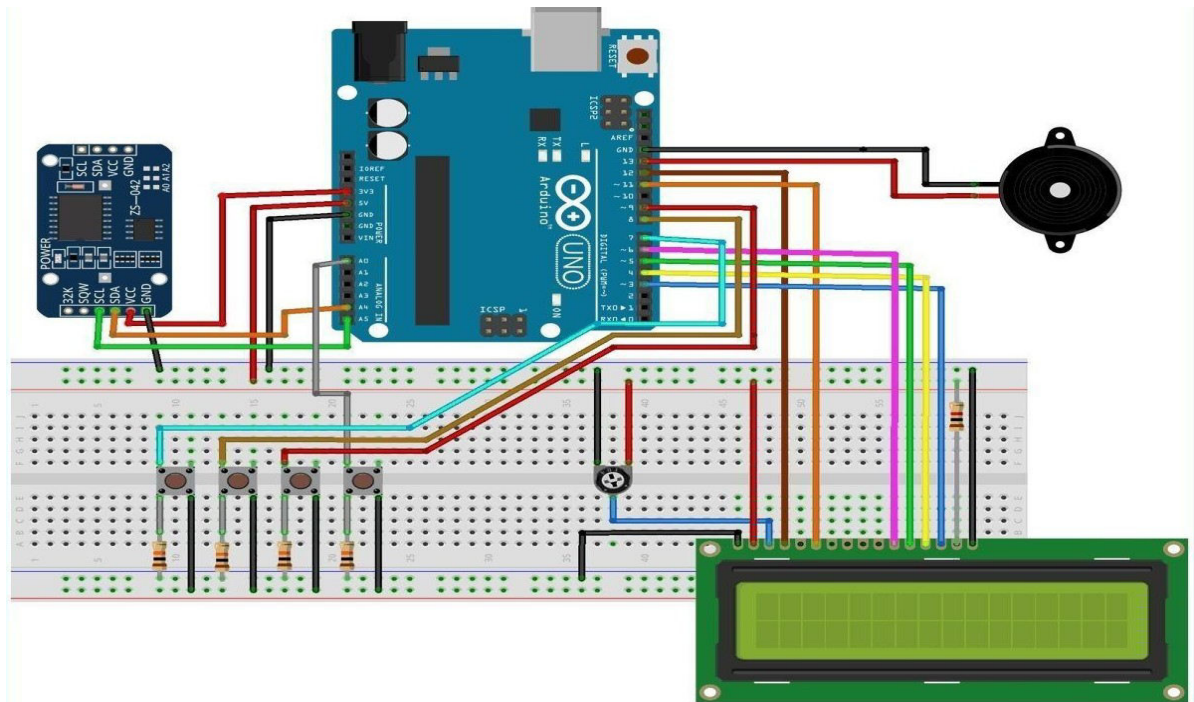
USN 3:As a user, I need to take my medicine and I am not able see the dosage of medicine properly

USN 4:As a user, Sometimes my medicine aren't in stock and I usually forget the Stock of my medication

## REQUIRED MATERIALS:

1. RTC DS3231 module
2. 16x2 LCD Display
3. Buzzer
4. Led(any color)
5. Breadboard
6. Push Buttons
7. 10K Potentiometer
8. 10K,1K Resistors
9. Jumper Wires
10. Arduino Uno

Simulation:



Code:

```
//Medicine Reminder using Arduino Uno
```

```
// Reminds to take medicine at 8am, 2pm, 8pm
```

```

/* The circuit:

LCD RS pin to digital pin 12

LCD Enable pin to digital pin 11

LCD D4 pin to digital pin 5

LCD D5 pin to digital pin 4

LCD D6 pin to digital pin 3

LCD D7 pin to digital pin 2

LCD R/W pin to ground

LCD VSS pin to ground

LCD VCC pin to 5V

10K resistor:

ends to +5V and ground

wiper to LCD VO pin (pin 3)*/

#include <LiquidCrystal.h>

#include <Wire.h>

#include <RTClib.h>

#include <EEPROM.h>

int pushVal = 0;

int val;

int val2;

int addr = 0;

RTC_DS3231 rtc;

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2; // lcd

Pins

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

#define getWellsoon 0

#define HELP_SCREEN 1

#define TIME_SCREEN 2

```

```

int buzz2pmHH = 14; // HH - hours

int buzz2pmMM = 00; // MM - Minute

int buzz2pmSS = 00; // SS - Seconds

int buzz8pmHH = 20; // HH - hours

int buzz8pmMM = 00; // MM - Minute

int buzz8pmSS = 00; // SS - Seconds

int nowHr, nowMin, nowSec; // to show current mm,hh,ss
// All messages

void gwsMessage(){ // print get well soon message

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Stay Healthy :)"); // Give some cheers

lcd.setCursor(0, 1);

lcd.print("Get Well Soon :)"); // wish
}

void helpScreen() { // function to display 1st screen in LCD

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Press Buttons");

lcd.setCursor(0, 1);

lcd.print("for Reminder...!");
}

void timeScreen() { // function to display Date and time in LCD screen

DateTime now = rtc.now(); // take rtc time and print in display

lcd.clear();

```

```

lcd.setCursor(0, 0);
lcd.print("Time:");
lcd.setCursor(6, 0);
lcd.print(nowHr = now.hour(), DEC);
lcd.print(":");
lcd.print(nowMin = now.minute(), DEC);
lcd.print(":");
lcd.print(nowSec = now.second(), DEC);
lcd.setCursor(0, 1);
lcd.print("Date: ");
lcd.print(now.day(), DEC);
lcd.print("/");
lcd.print(now.month(), DEC);
lcd.print("/");
lcd.print(now.year(), DEC);
}

void setup() {
  Serial.begin(9600); // start serial debugging
  if (! rtc.begin()) { // check if rtc is connected
    Serial.println("Couldn't find RTC");
    while (1);
  }
  if (rtc.lostPower()) {
    Serial.println("RTC lost power, lets set the time!");
  }

  // rtc.adjust(DateTime(F( DATE ), F( TIME ))); // uncomment this to
    set the current time and then comment in next upload when u set the time

```

```

rtc.adjust(DateTime(2019, 1, 10, 7, 59, 30)); // manual time set
lcd.begin(16, 2);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Welcome To"); // print a messege at
startup
lcd.setCursor(0, 1);
lcd.print("Circuit Digest");
delay(1000);
pinMode(push1pin, INPUT); // define push button
pins type
pinMode(push2pin, INPUT);
pinMode(push3pin, INPUT);
pinMode(stopPin, INPUT);
pinMode(ledPin, OUTPUT);
delay(200);
Serial.println(EEPROM.read(addr));
val2 = EEPROM.read(addr); // read previosuly saved value of
push button to start from where it was left previously
switch (val2) {
case 1:
Serial.println("Set for 1/day");
push1state = 1;
push2state = 0;
push3state = 0;
pushVal = 1;
break;
case 2:

```

```
Serial.println("Set for 2/day");  
push1state = 0;  
push2state = 1;  
push3state = 0;  
pushVal = 2;  
break;  
  
case 3:  
Serial.println("Set for 3/day");  
push1state = 0;  
push2state = 0;  
push3state = 1;  
pushVal = 3;  
break;  
}  
}  
  
void loop() {  
push1(); //call to set once/day  
push2(); //call to set twice/day  
push3(); //call to set thrice/day  
if (pushVal == 1) { // if push button 1 pressed  
then remind at 8am  
at8am(); //function to start uzzing at  
8am  
}  
else if (pushVal == 2) { // if push button 2 pressed  
then remind at 8am and 8pm  
at8am();
```

```

at8pm(); //function to start uzzing at
8mm

}

else if (pushVal == 3) { // if push button 3 pressed
then remind at 8am and 8pm

at8am();

at2pm(); //function to start uzzing at
8mm

at8pm();

}

currentMillisLCD = millis(); // start millis for LCD screen
switching at defined interval of time

push1state = digitalRead(push1pin); // start reading all push
button pins

push2state = digitalRead(push2pin);
push3state = digitalRead(push3pin);
stopinState = digitalRead(stopPin);
stopPins(); // call to stop buzzing
changeScreen(); // screen cycle function
}

// push buttons

void push1() { // function to set reminder once/day
if (push1state == 1) {
push1state = 0;
push2state = 0;
push3state = 0;
// pushPressed = true;

```



```

EEPROM.write(addr, 1);
Serial.print("Push1 Written : "); Serial.println(EEPROM.read(addr)); // for
debugging
pushVal = 1; //save the state of push
button-1
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Reminder set ");
lcd.setCursor(0, 1);
lcd.print("for Once/day !");
delay(1200);
lcd.clear();
}
}

void push2() { //function to set reminder twice/day
if (push2state == 1) {
push2state = 0;
push1state = 0;
push3state = 0;
// pushPressed = true;
EEPROM.write(addr, 2);
Serial.print("Push2 Written : "); Serial.println(EEPROM.read(addr));
pushVal = 2;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Reminder set ");
lcd.setCursor(0, 1);
lcd.print("for Twice/day !");

```

```

delay(1200);
lcd.clear();
}
}
void push3() { //function to set reminder thrice/day
if (push3state == 1) {
push3state = 0;
push1state = 0;
push2state = 0;
// pushPressed = true;
EEPROM.write(addr, 3);
Serial.print("Push3 Written : "); Serial.println(EEPROM.read(addr));
pushVal = 3;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Reminder set ");
lcd.setCursor(0, 1);
lcd.print("for Thrice/day !");
delay(1200);
lcd.clear();
}
}
void stopPins() { //function to stop buzzing when user pushes stop
push button
if (stopinState == 1) {
// stopinState = 0;
// pushPressed = true;

```

```

pushpressed = 1;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Take Medicine ");
lcd.setCursor(0, 1);
lcd.print("with Warm Water");
delay(1200);
lcd.clear();
}
}

void startBuzz() { // function to start buzzing when time reaches
to defined interval
// if (pushPressed == false) {
if (pushpressed == 0) {
Serial.println("pushpressed is false in blink");
unsigned long currentMillis = millis();
if (currentMillis - previousMillis >= interval) {
previousMillis = currentMillis; // save the last time you blinked the
LED
Serial.println("Start Buzzing");
if (ledState == LOW) { // if the LED is off turn it on and
vice-versa:
ledState = HIGH;
}
else {
ledState = LOW;
}
digitalWrite(ledPin, ledState);
}
}
}

```

```

}
else if (pushpressed == 1) {
Serial.println("pushpressed is true");
ledState = LOW;
digitalWrite(ledPin, ledState);
}
}

void at8am() { // function to start buzzing at 8am
DateTime now = rtc.now();
if (int(now.hour()) >= buzz8amHH) {
if (int(now.minute()) >= buzz8amMM) {
if (int(now.second()) > buzz8amSS) {
////////////////////////////////////
startBuzz();
////////////////////////////////////
}
}
}
}

void at2pm() { // function to start buzzing at 2pm
DateTime now = rtc.now();
if (int(now.hour()) >= buzz2pmHH) {
if (int(now.minute()) >= buzz2pmMM) {
if (int(now.second()) > buzz2pmSS) {
////////////////////////////////////
startBuzz();

```

```

////////////////////////////////////
}
}
}
}

void at8pm() { // function to start buzzing at 8pm
DateTime now = rtc.now();
if (int(now.hour()) >= buzz8pmHH) {
if (int(now.minute()) >= buzz8pmMM) {
if (int(now.second()) > buzz8pmSS) {
////////////////////////////////////
startBuzz();
////////////////////////////////////
}
}
}
}

//Screen Cycling
void changeScreen() { //function for Screen Cycling
// Start switching screen every defined intervalLCD
if (currentMillisLCD - previousMillisLCD > intervalLCD) // save the
last time you changed the display
{
previousMillisLCD = currentMillisLCD;
screens++;
if (screens > maxScreen)
{

```

<b>TEAM ID</b>	<b>PNT2022TMID36821</b>
<b>PROJECT NAME</b>	<b>Personal Assistance for Seniors Who Are Self-Reliant</b>

**Time Taken : 6 days**

**Sprint 2: software (Create device in the Iot watson platform, workflow for Iot scenarios using local NODE-RED)**

**Time Taken : 6 days**

As we have completed our simulation test and briefing every objectives of the project . So we have

## **REQUIRED SOFTWARE:**

MIT App, IBM Watson IoT Platform, IBM Cloud, IBM Node-RED, Jira Software

The question that are raised during our session How, When, why are doing the project .And had the session by Service was the main motive to be produced.

And what we did in this SPRINT 1 is that we found the component that are needed for the project and conducted a dry test and had a live discussion session of how the project should be designed and how it is used efficiently by using required software.

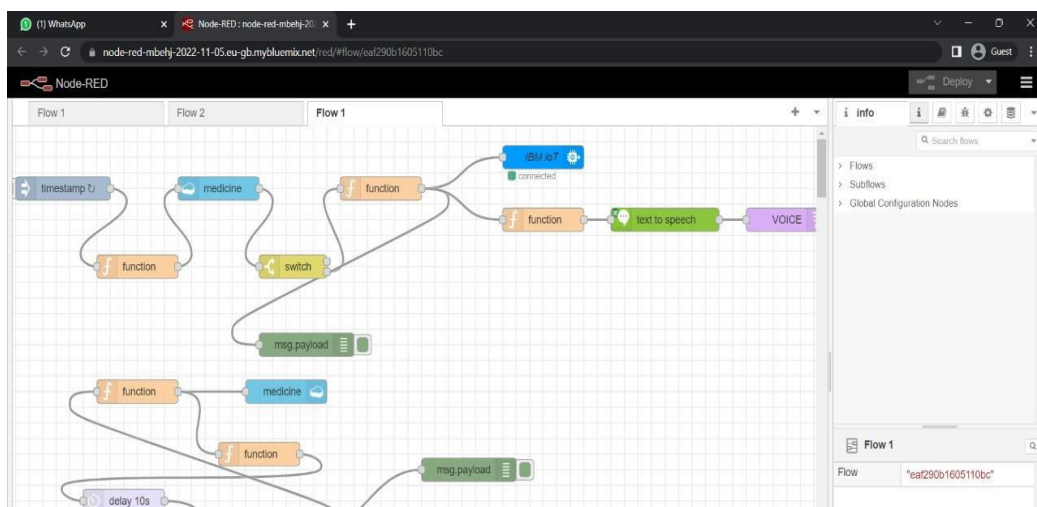
How was required Software useful and how our team did it?

We used the **MIT App** to built to use the App that can be reached to End User.

**IBM Node – RED** used to connect all the process that are made by the MIT App and process the data .

**IBM Watson IoT Platform** is used to output to user but it is useful that it run on IBM Cloud.

**Jira Software** it is used for remainder for work is seen in the software and essentially gives the burn down chart which can seen that our work done of every week is reflected in the Jira Software.

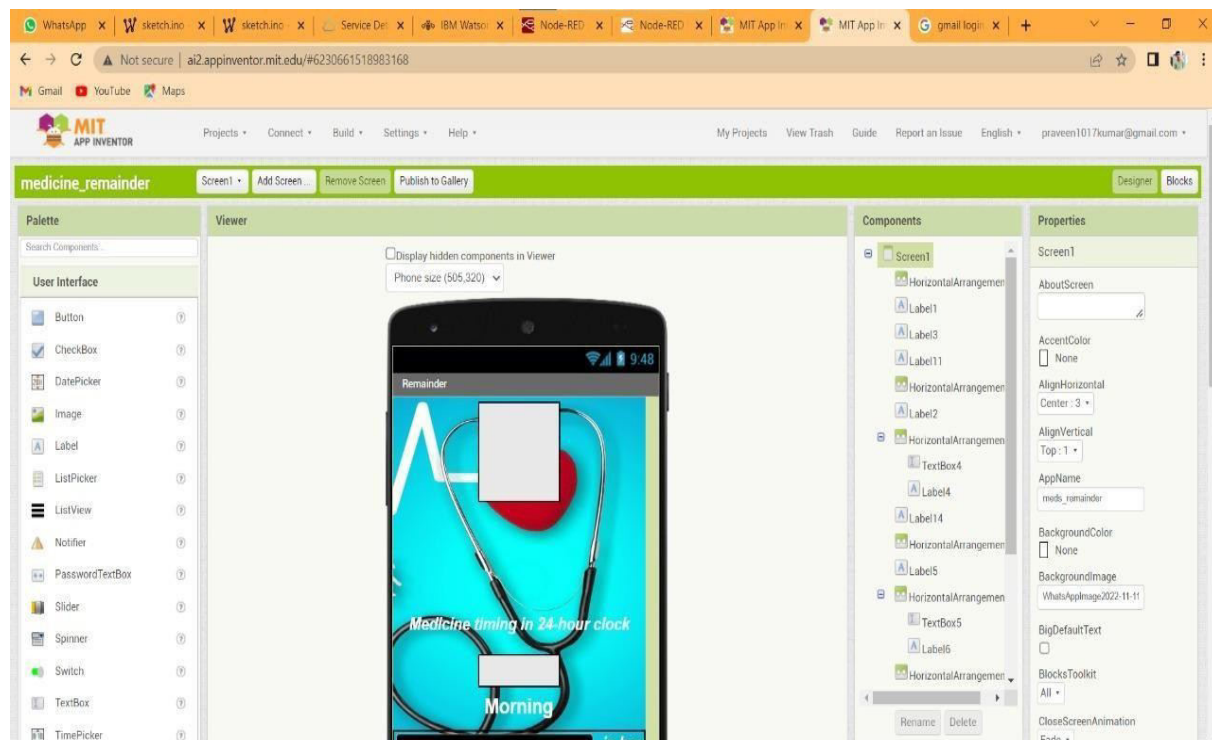
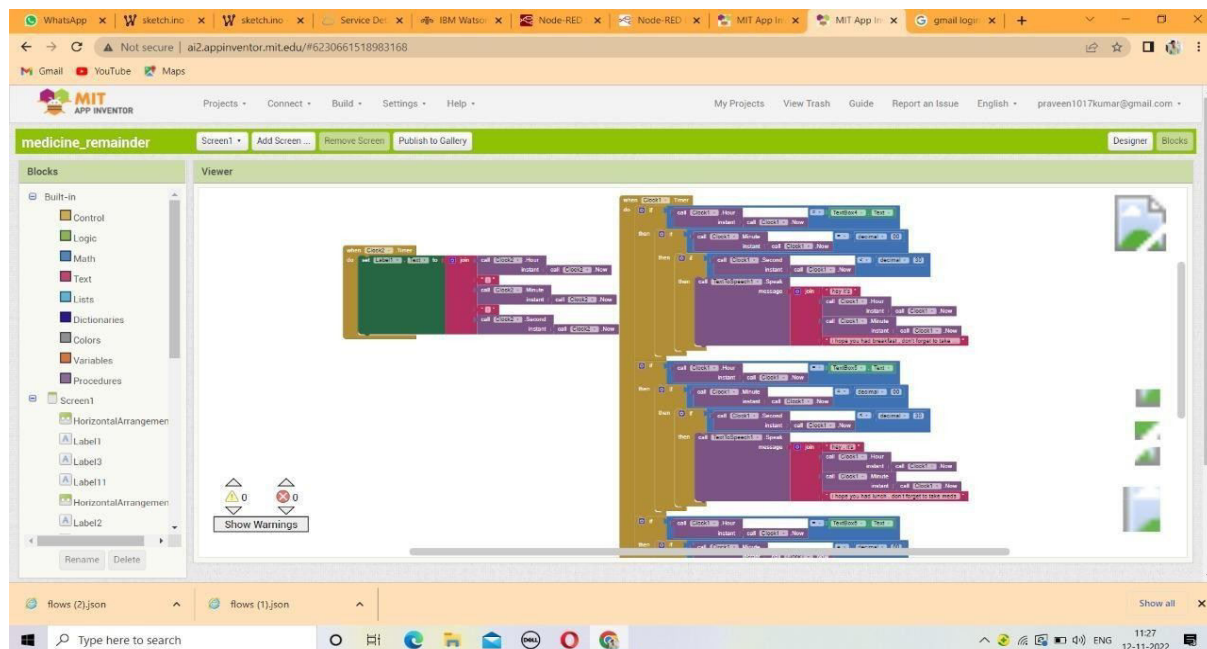


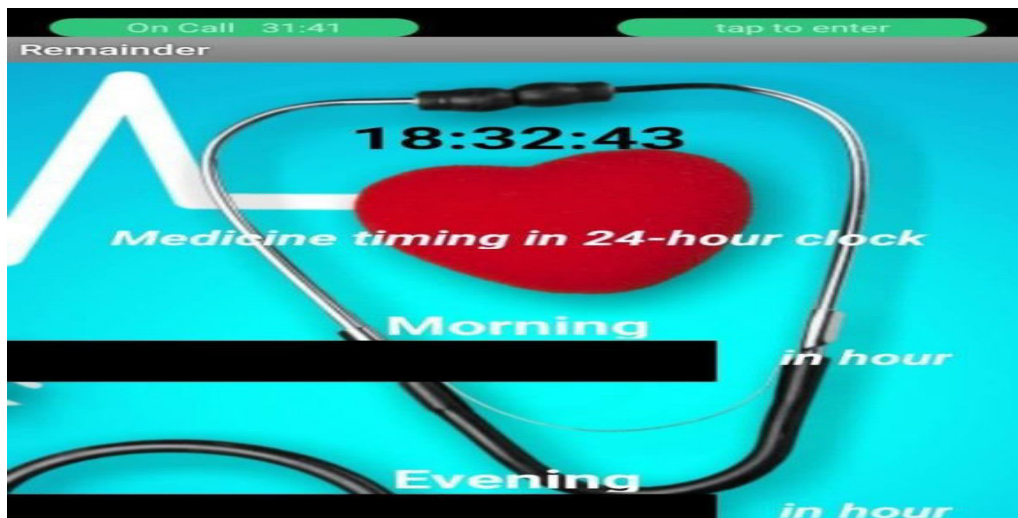
<b>TEAM ID</b>	<b>PNT2022TMID36821</b>
<b>PROJECT NAME</b>	<b>Personal Assistance for Seniors Who Are Self-Reliant</b>

### SPRINT 3:

MIT App inventor, dashboard (application for your project using MIT App, design the model and test the app)

### Simulation :





<b>TEAM ID</b>	<b>PNT2022TMID36821</b>
<b>PROJECT NAME</b>	<b>Personal Assistance for Seniors Who Are Self-Reliant</b>
<b>SPRINT 4 :</b>	

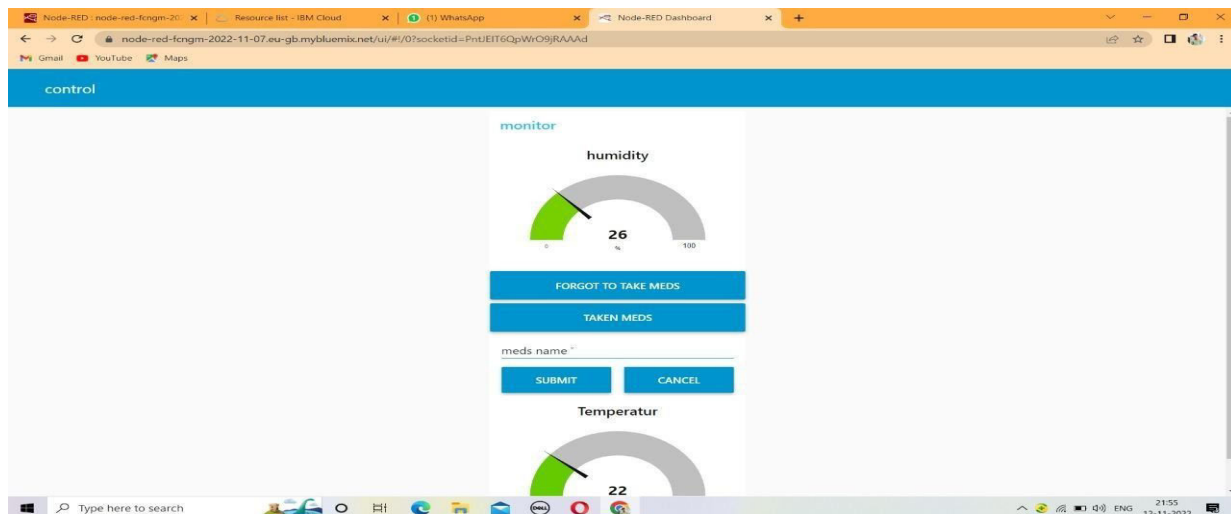
**web UI (to make the user interact with the software)**

**Time taken : 6 days**

The Medicine remainder App the below Web UI shows that the remainder is given morning, evening and night .If needed the App can Edit the time of medicine taken.

Feature

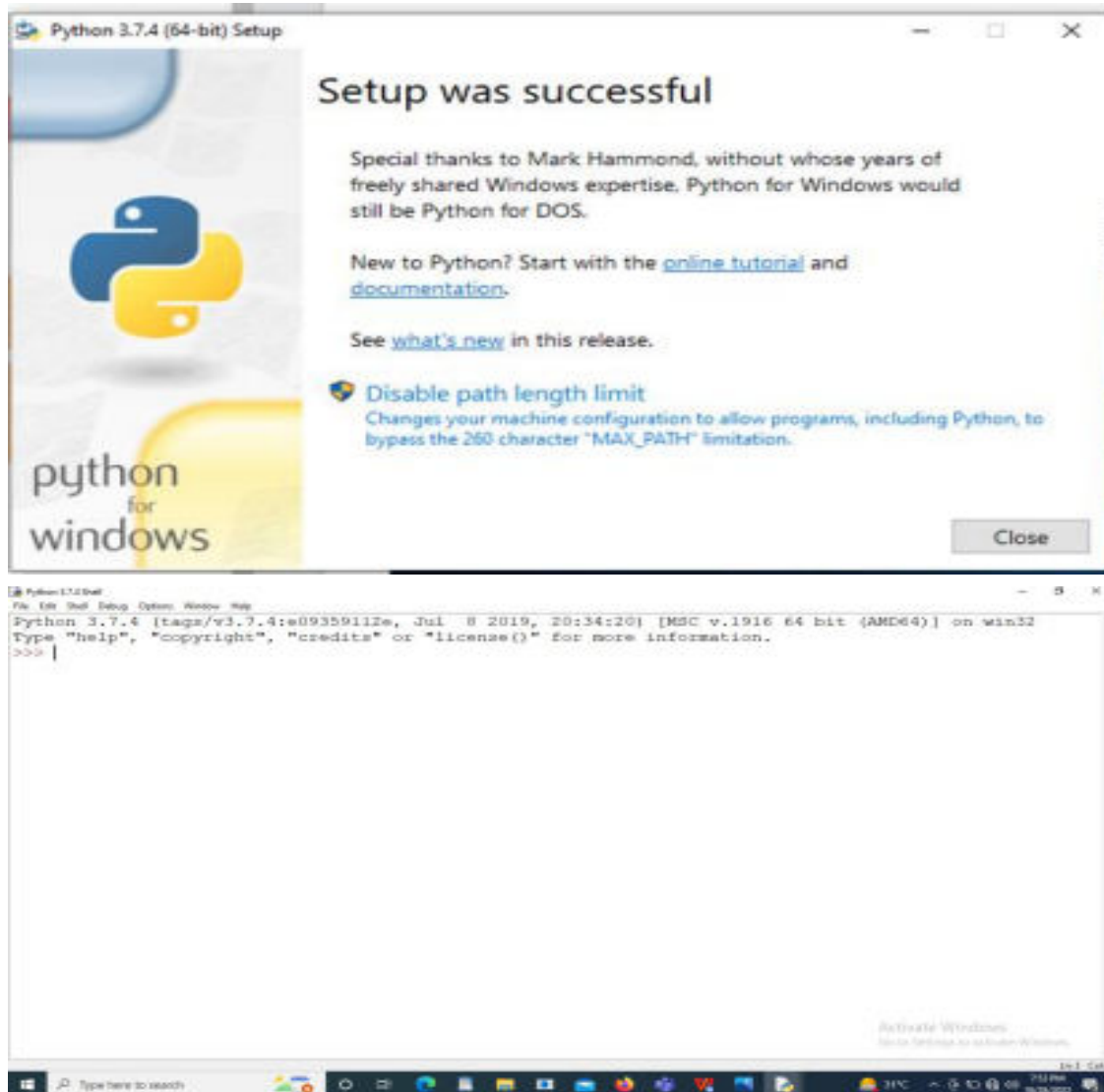
- Medicine remainder on time
- If any dose missed, alert will be provided for the emergency number in contact
- Provides information about the medicine intake such as the medicine name, dose and dosage route.
- It also reminds the reminds stock of medicine to be purchased a week priory
- The update of the patient Body Temperature and humidity measure before medication is taken.
- The remainder is as following steps The Message given as “TAKE THE MEDICINE” and following that the patient has not taken medication after 5 minutes of gap the another remainder is given “FORGOT TO TAKE MEDICIENE ” and if the medication is taken the message is given as “TAKEN MEDS”





## PREREQUISITES:

### Software:



# IBM Cloud Services

## IBM Watson IoT Platform :

This service is the hub for IBM Watson IoT and lets you communicate with and consume data from connected devices and gateways. Use the built-in web console dashboards to monitor your IoT data and analyze it in real time. Then, enhance and customize your IBM Watson IoT Platform experience by building and connecting your own apps by using messaging and REST APIs.

### Features

#### *Connect:*

Quickly and securely register and connect your devices and gateways. You can find simple step-by-step instructions for connecting popular devices, sensors, and gateways in our recipes site.

#### *Information Management*

Control what happens to the data that is received from your connected devices. Manage data storage, configure data transformation actions, and integrate with other data services and device platforms.

#### *Analyze in real time*

Monitor your real-time device data through rules, analytics, and dashboards. Define rules to monitor conditions and trigger automatic actions that include alerts, email, IFTTT, Node-RED flows, and external services to react quickly to critical changes.

#### *Risk and Security management*

Our secure-by-design control capabilities protect the integrity of your IoT solution through secure connectivity and access control for users and applications. Extend the base security with threat intelligence for IoT to visualize critical risks and automate operational responses with policy-driven mitigation actions.

## Node-RED Service:

### *Input Node:*

The input node receive device commands from the IBM Watson Internet of Things Platform.

The node can connect as either a Device or Gateway:

- **Device:** the node can be configured to either receive all commands for the Device, or just select a specific command type.
- **Gateway:** the node can be configured to receive commands for all devices connected through the gateway, or to select a subset of them.

### *Output Node*

Send device events to the IBM Watson Internet of Things Platform.

The node can connect as either a Device or Gateway, in registered mode or using the Quickstart service.

When connecting using the Quickstart service, the connection will use a device type of node-red-ibmwiots and a randomly generated device id, which can be configured in the node

## Cloudant DB:

IBM Cloudant is a fully managed JSON document database that offers independent serverless scaling of provisioned throughput capacity and storage. Cloudant is compatible with Apache CouchDB and accessible through a simple to use HTTPS API for web, mobile, and IoT applications.

### **Features:**

- 99.99% SLA

• **Secure:** All data encrypted over the wire and at rest. Optional BYOK and private service endpoints on Dedicated Hardware environments.

HA/DR: All Cloudant JSON documents are stored in triplicate for in-region HA/DR. In regions that support availability zones (Dallas,

• Washington DC, London, Frankfurt, Tokyo, Sydney), documents are stored across three separate availability zones.

• **Compliance:** GDPR, PCI, SOC2 & ISO 27001 Compliant. HIPAA available on Dedicated Hardware environments and requires opt into HIPAA Supported setting. Any Cloudant instance deployed from the Frankfurt region will be in an EU Supported environment.

- Max JSON document size of 1MB

## TTS Service:

The IBM Watson™ Text to Speech service provides APIs that use IBM's speech-synthesis capabilities to synthesize text into natural-sounding speech in a variety of languages, dialects, and voices. The service supports at least one male or female voice, sometimes both, for each language. The audio is streamed back to the client with minimal delay.

For speech synthesis, the service supports a synchronous HTTP Representational State Transfer (REST) interface and a WebSocket interface. Both interfaces support plain text and SSML input. SSML is an XML-based markup language that provides text annotation for speech-synthesis applications. The WebSocket interface also supports the SSML <mark> element and word timings

```
screens = 0; // all screens over -> start from 1st
}
isScreenChanged = true;
}
// Start displaying current screen
if (isScreenChanged) // only update the screen if the screen is changed.
{
isScreenChanged = false; // reset for next iteration
switch (screens)
{
case getWellsoon:
gwsMessege(); // get well soon message
break;
case HELP_SCREEN:
helpScreen(); // instruction screen
break;
case TIME_SCREEN:
timeScreen(); // to print date and time
break;
default:
//NOT SET.
break;
}
}
}
```

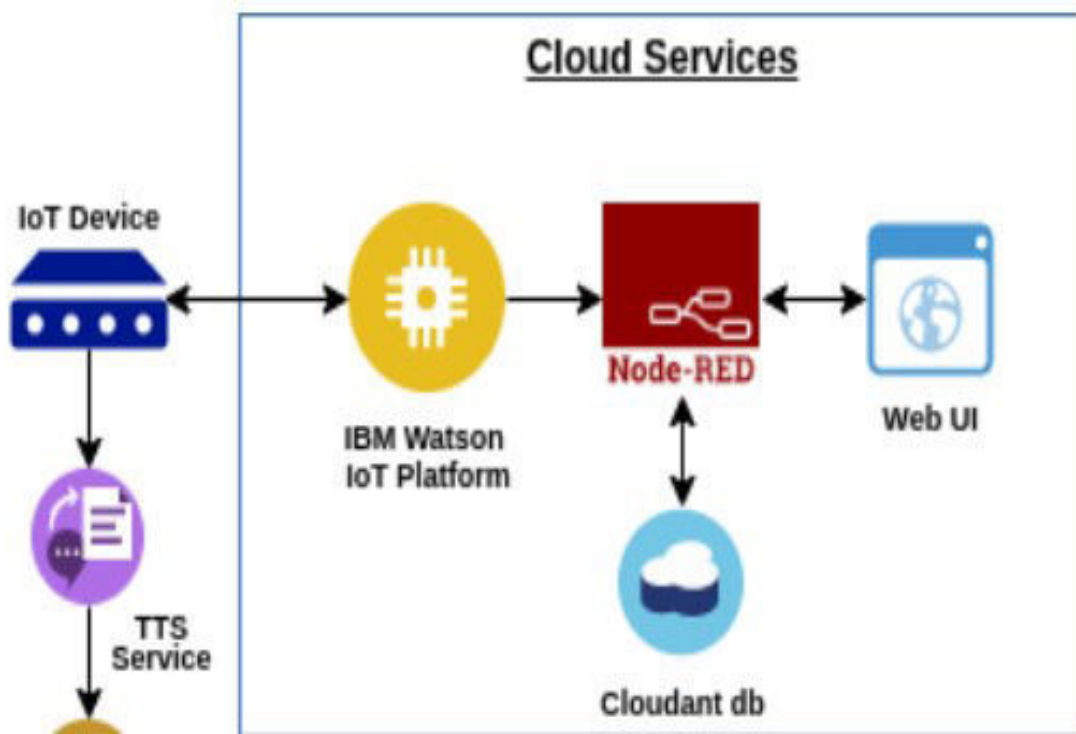
## Final Deliverables

Team ID	PNT2022TMID36821
Project Name	Personal Assistance for Seniors Who Are Self-Reliant.
Team Members	Anitha R Shruthilaya S Vishal S Vishwa S

### 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.

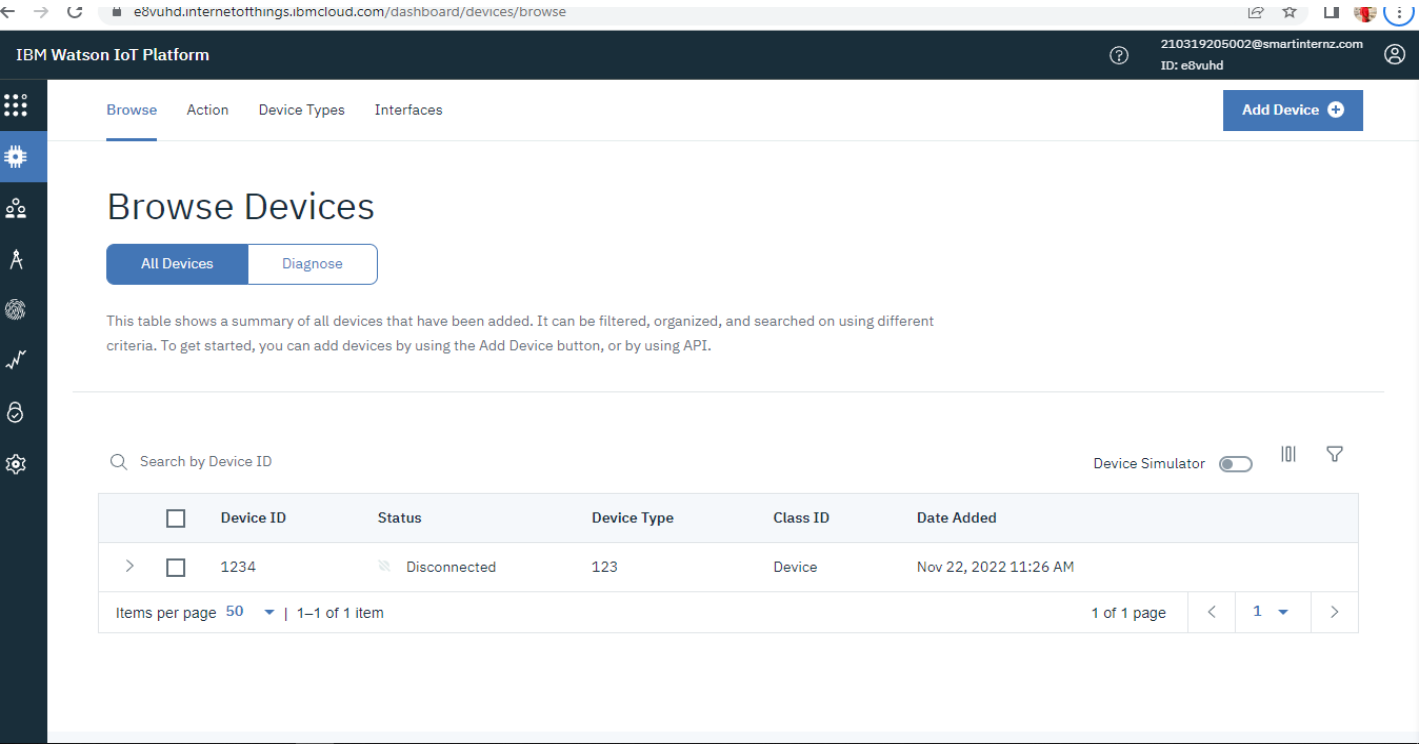
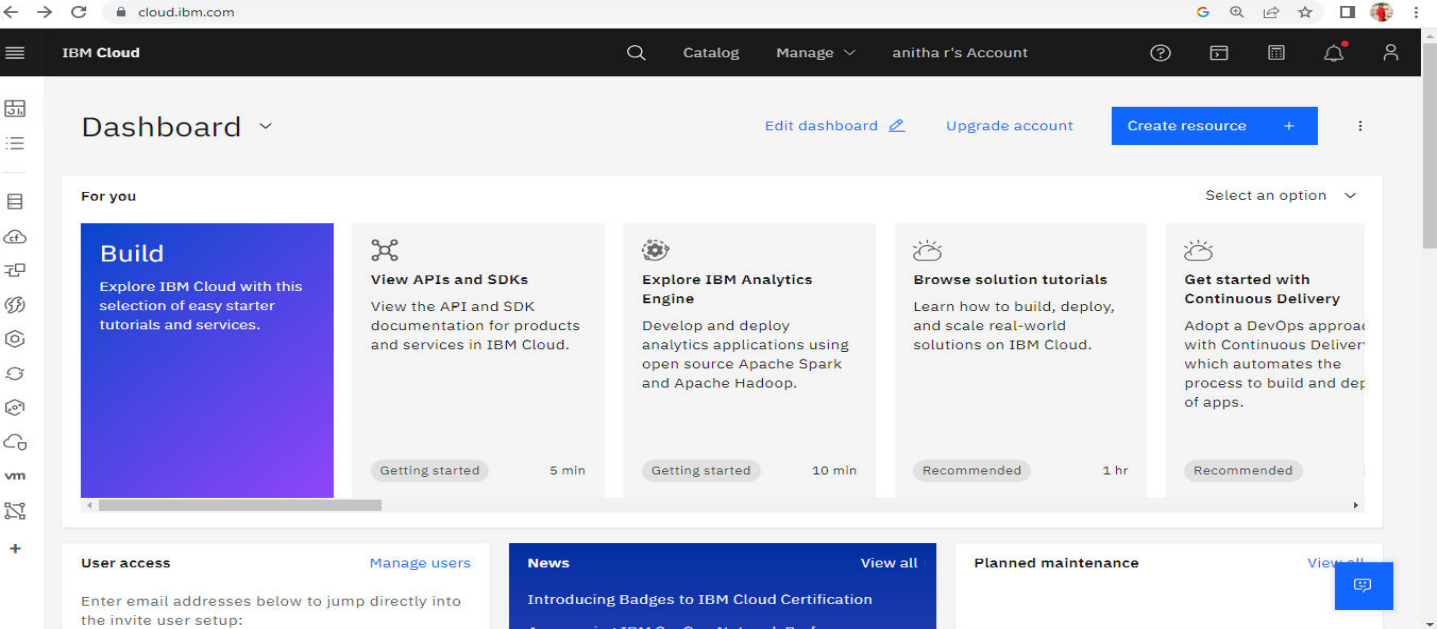
### FLOW OF THE PROJECT:



WAYS ACHIEVES THE PROJECT FLOW:

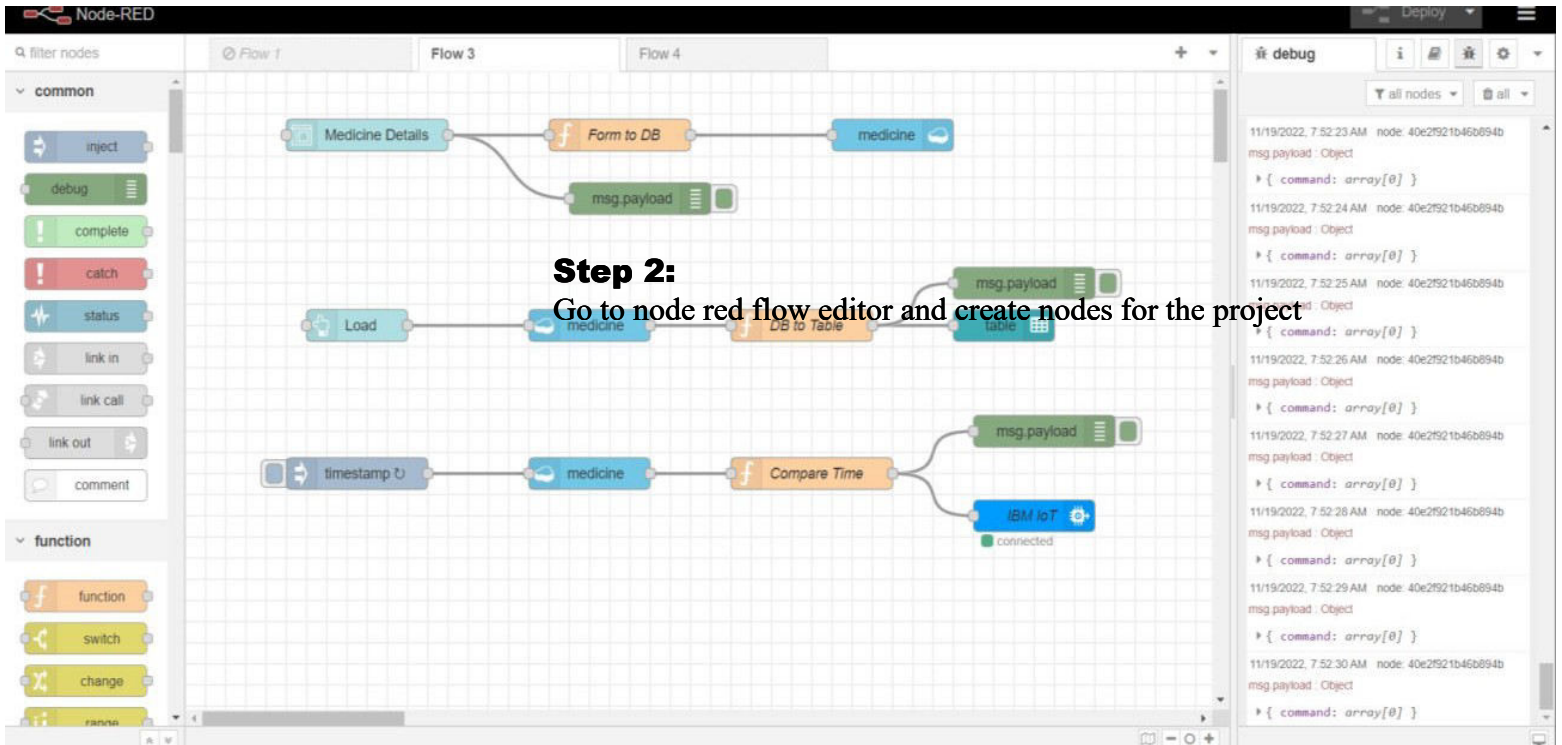
Step1:

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



## Step 2:

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



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.

**Edit form node**

Delete Cancel Done

**Properties**

Group: [Remainder] Medicien

Size: auto

Label: Medicine Details

Label	Name	Type	Required	UI Rows	Remove
Tablet	Tablet	Text	<input checked="" type="checkbox"/>		<input type="button" value="Remove"/>
Hour	Hour	Number	<input checked="" type="checkbox"/>		<input type="button" value="Remove"/>
Min	Min	Number	<input checked="" type="checkbox"/>		<input type="button" value="Remove"/>

+ element

## 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:

The screenshot shows the 'Edit function node' dialog with the 'Form to DB' function selected. The 'On Message' tab is active, displaying the following JavaScript code:

```
1- msg.payload={
2-   "Tablet":msg.payload.Tablet,
3-   "Hour":msg.payload.Hour,
4-   "Min":msg.payload.Min
5- };
6- return msg;
```

The dialog includes a 'Name' field, tabs for 'Setup', 'On Start', 'On Message', and 'On Stop', and an 'Enabled' checkbox at the bottom.

### 2.2.DB to Table

The screenshot shows the 'Edit function node' dialog with the 'DB to Table' function selected. The 'On Message' tab is active, displaying the following JavaScript code:

```
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- }
```

The dialog includes a 'Name' field, tabs for 'Setup', 'On Start', 'On Message', and 'On Stop', and an 'Enabled' checkbox at the bottom.

## IBM IoT Out Node:

The screenshot shows the 'Edit ibmiot out node' dialog. The 'Properties' tab is active, displaying the following configuration:

- 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

The dialog includes a 'Name' field, tabs for 'Properties', 'On Start', 'On Message', and 'On Stop', and an 'Enabled' checkbox at the bottom.



Create Cloundant Database to save the incoming data.

Log Out

medicine

All Documents

Query

Permissions

Changes

Design Documents

Document ID

Options

{ } JSON

Create Document

	_id	payload	socketid	topic
<input type="checkbox"/>	2f220b7493ebefedfa38f9e9bbd...	{ "Tablet": "Paracetamol", "Hour": ...	F4GlrDmWgYeAih6CAAAf	{ "Tablet": "Paracetamol", "Hour": ...
<input type="checkbox"/>	602ee33908ddad7a18dce906d9...	{ "Tablet": "Dolo 650", "Hour": 8, "...	F4GlrDmWgYeAih6CAAAf	{ "Tablet": "Dolo 650", "Hour": 8, "...
<input type="checkbox"/>	627f83547320b4ecda41f2a2f29...	{ "Tablet": "Deplatta", "Hour": 17, "...	D3Fb8_3K8yUj0AH9AAAb	{ "tab": "Deplatta", "Hour": 17, "Mi...
<input type="checkbox"/>	9a5df1a280e43ed2b57d454d30f...	{ "Tablet": "Ativan", "Hour": 17, "Mi...	D3Fb8_3K8yUj0AH9AAAb	{ "tab": "Ativan", "Hour": 17, "Min"...
<input type="checkbox"/>	a960d38e7e2f5eab2f5634f4fc86...	{ "Tablet": "Concor", "Hour": 17, "...	D3Fb8_3K8yUj0AH9AAAb	{ "tab": "Concor", "Hour": 17, "Min...
<input type="checkbox"/>	f0dcafdd96eaebf4afa776e2672f7...	{ "Tablet": "Cardace", "Hour": 17, "...	D3Fb8_3K8yUj0AH9AAAb	{ "tab": "Cardace", "Hour": 17, "Mi...
<input type="checkbox"/>	f7bc3b6818fc517b36355d14eff9...	{ "Tablet": "Rosuvas", "Hour": 17, "...	D3Fb8_3K8yUj0AH9AAAb	{ "tab": "Rosuvas", "Hour": 17, "Mi...

Showing 4 of 5 columns. ☐ Show all columns.

Showing document 1 - 7. Documents per page: 20

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.

- 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":"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

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":"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)

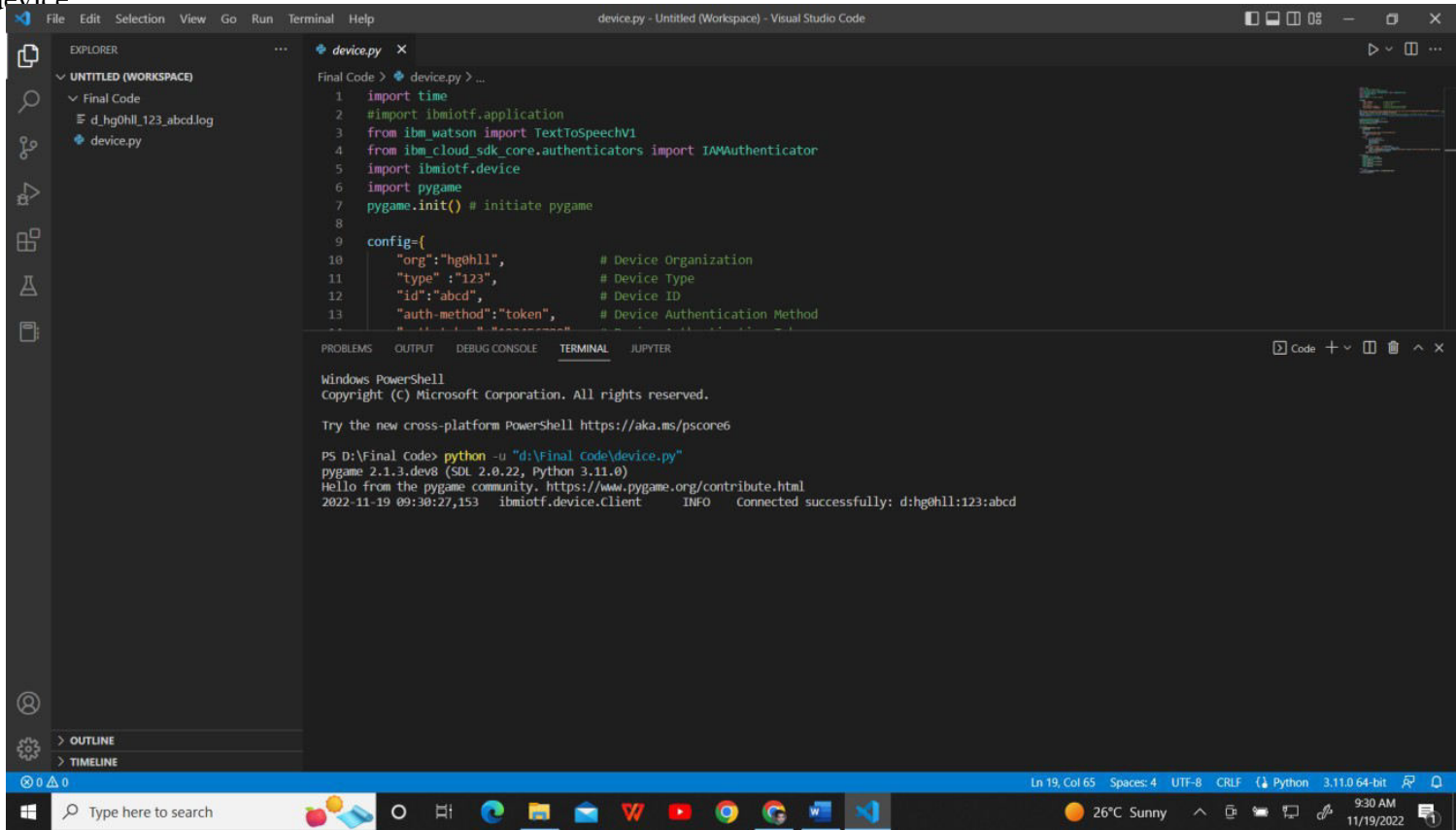
# callback and
def myCommandCallback (cmd):
    a=cmd.data
    c=1
    instruction="Please Take following Medicine. "
    if len(a["command"])==0:
        pass
    else:
        for i in a["command"]:
            instruction+=str(c)+". "
            instruction+=i
            instruction+=". "
     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

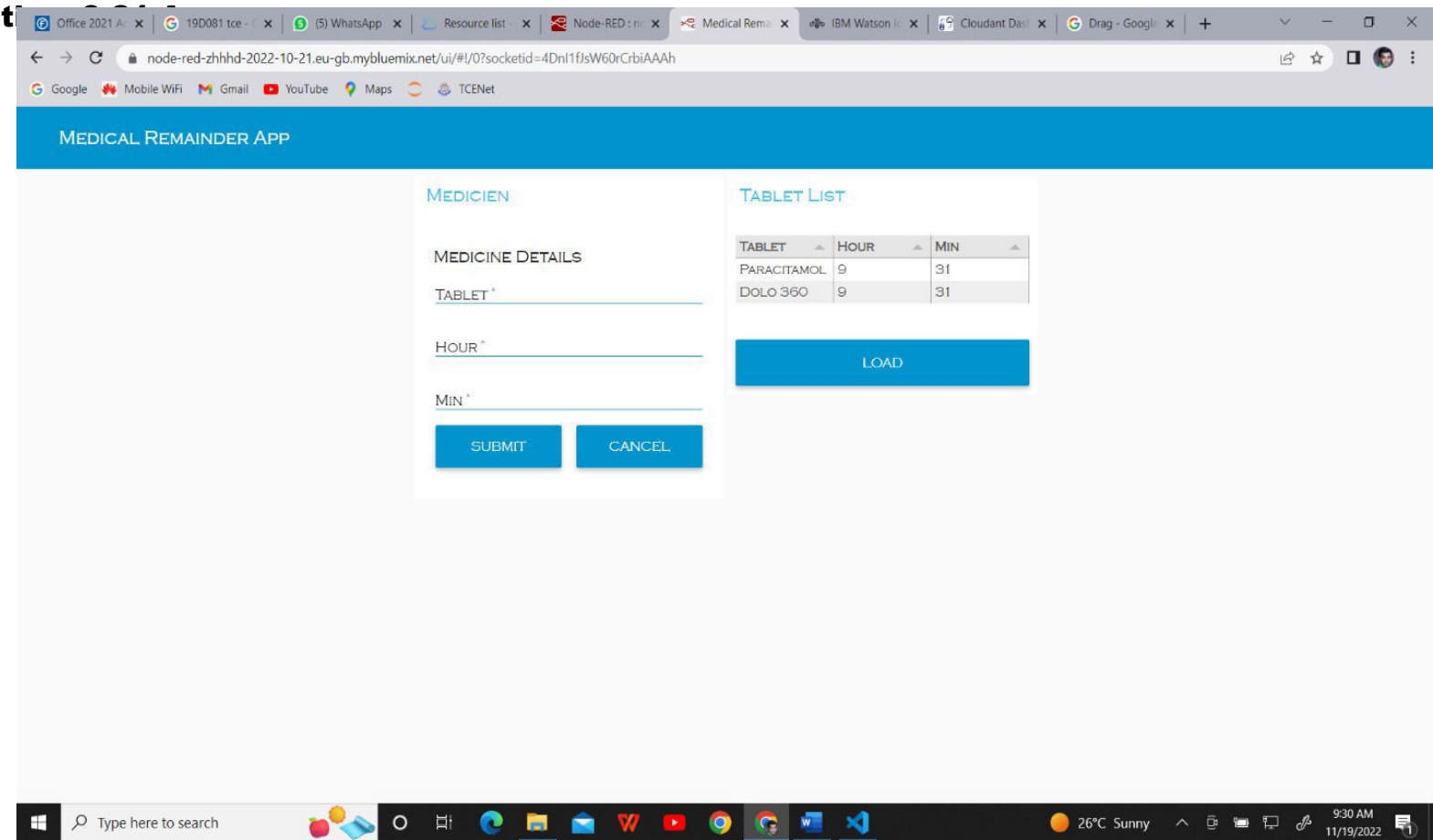
# RESULT:

## 1. Connect with the device

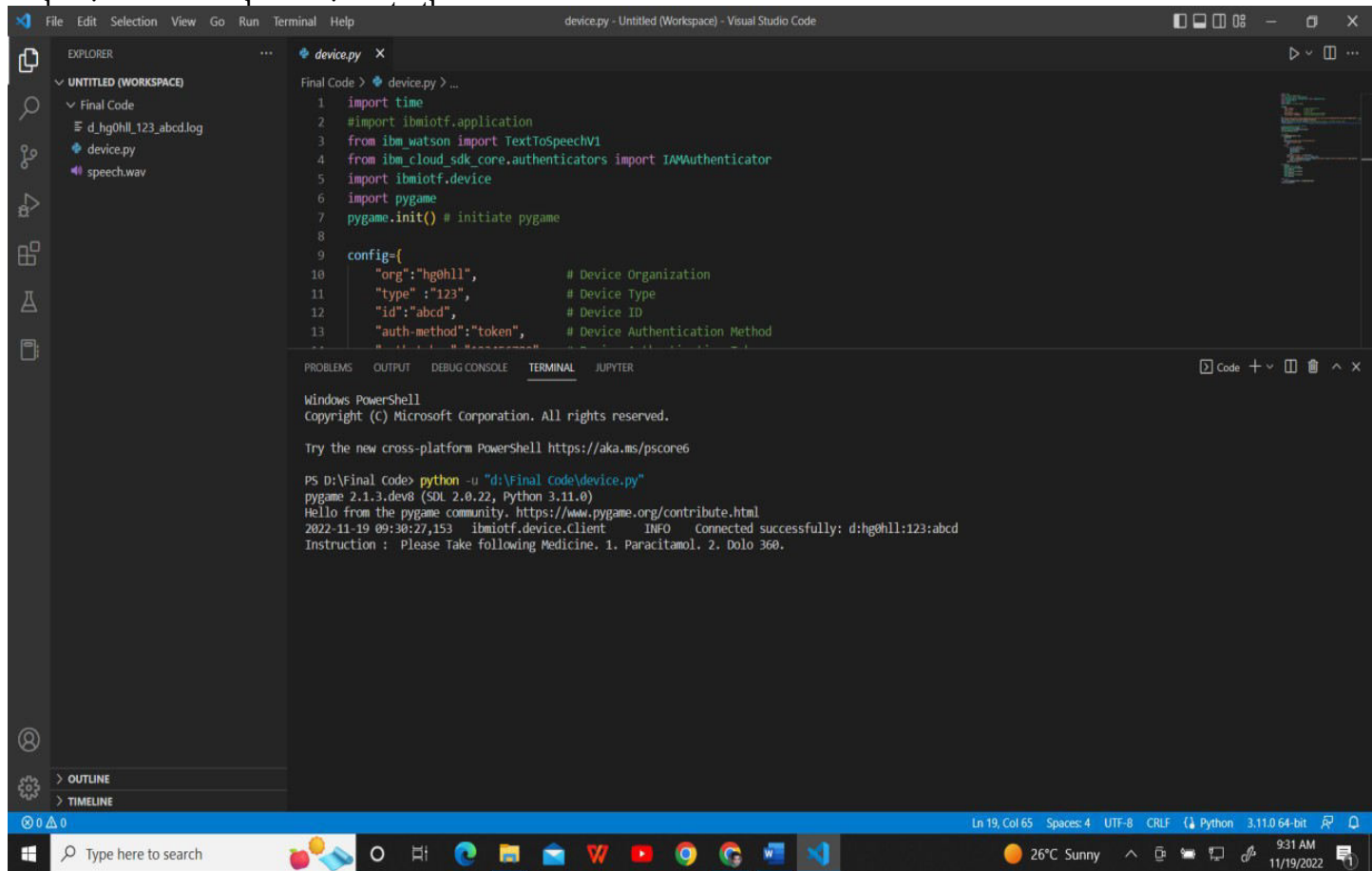


## 2. Load Tablet name and time in User UI

We load Two tablet Paracetamol and Dolo 650 set remainder and



3. At 9.31 Am data saved in the DB is received and the audio file for instructions is generated



The screenshot displays the Visual Studio Code interface. The Explorer pane on the left shows a workspace with files: 'Final Code', 'd\_hg0hll\_123\_abcd.log', 'device.py', and 'speech.wav'. The main editor shows the 'device.py' file with the following Python code:

```
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":"hg0hll",           # Device Organization
11     "type": "123",           # Device Type
12     "id": "abcd",           # Device ID
13     "auth-method": "token",  # Device Authentication Method
14 }
```

The TERMINAL pane at the bottom shows the execution of the script in a Windows PowerShell environment:

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

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

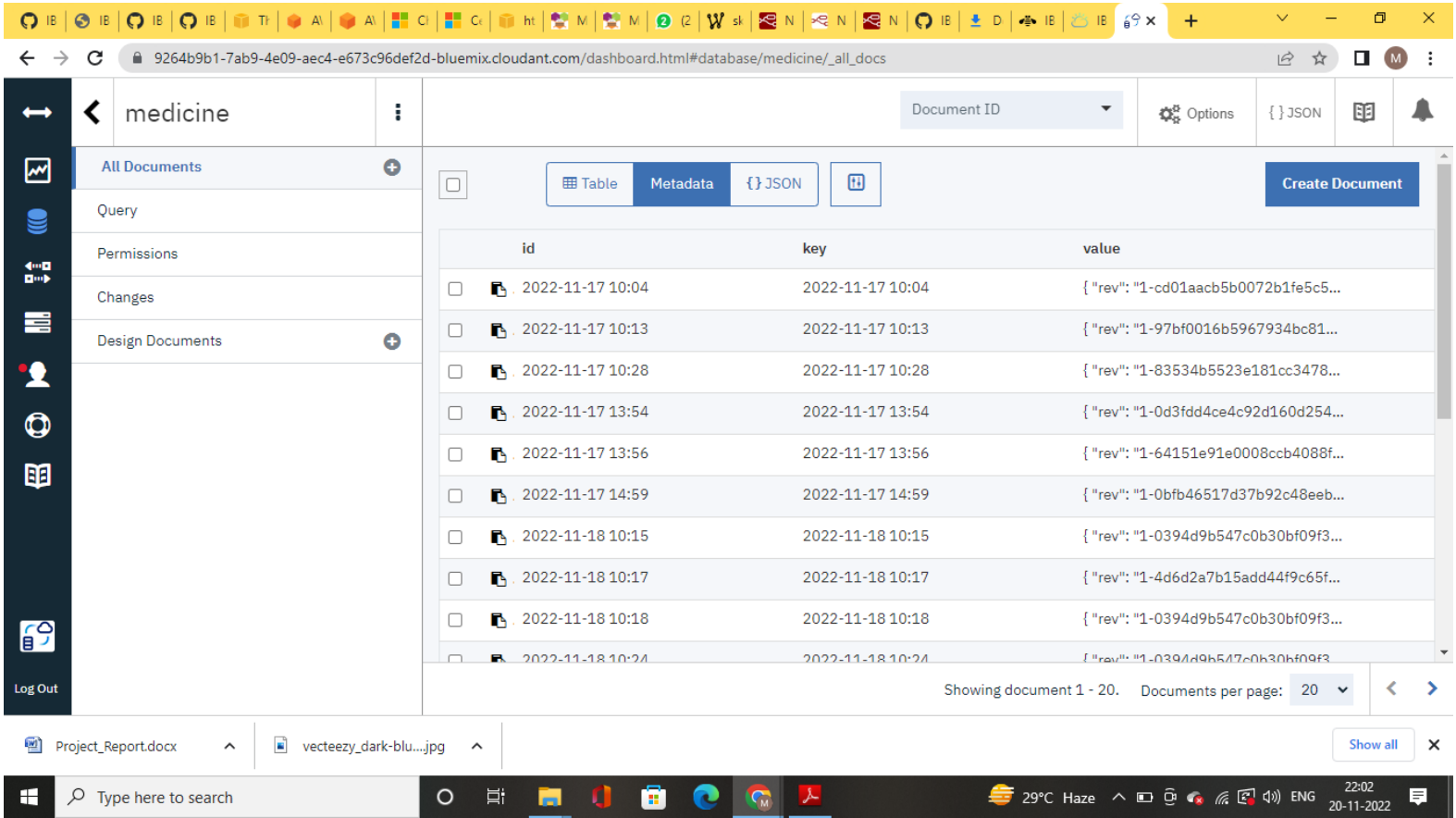
PS D:\Final Code> python -u "d:\Final Code\device.py"
pygame 2.1.3.dev8 (SDL 2.0.22, Python 3.11.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
2022-11-19 09:30:27,153 ibmiotf.device.Client INFO Connected successfully: d:hg0hll:123:abcd
Instruction : Please Take following Medicine. 1. Paracetamol. 2. Dolo 360.
```

**GIT HUB:** <https://github.com/IBM-EPBL/IBM-Project-55218-1667628039>

### CONCLUSION:

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

Feature 3



The project includes a cloud database system.

## 7. Testing

### Test cases

Test case	Precondition	Test steps	Expected result
Verify login with valid credentials	User should have a network Connection	1. Launch URL 2. Enter valid username. 3. Enter valid password. 4. Click on the “Login” button.	Users should be able to login successfully.
Verify login with invalid credentials	User should have a network Connection	1. Launch URL 2. Enter valid username. 3. Enter invalid password. 4. Click on the “Login” button.	Users should not be able to login.
Update the medicine name with the time.	User should have a network Connection	1. Enter valid medicine name. 2. Enter the time when the medicine has to be consumed. 3. Click on the “Submit” button.	Users should be able to update it successfully.

## 8. Results Performance Metrics

S. NO	Parameter	Performance
1.	Response Time	0.2s (Average of 10 trials)
2.	Workload	500 users ( Calculated based on Cloud Space)
3.	Revenue	Individual users and pharmaceutical industries.
4.	Efficiency	Simple and straightforward workflow, which makes the process efficient.
5.	Down Time	Almost no down time due to IBM Cloud enabled solution.

## 9. Advantages and Disadvantages

- Help the elderly people to take their medicine at the correct time.
- Avoid personal assistants or caretakers needed for medically sick people.
- Cost efficient.
- Can store multiple data and many notifications can be generated.
- Since it includes voice assistance, even blind people can use our device.

### Disadvantages

- Makes people lethargic and makes them dependent always on others.
- Requires a stable internet connection.

## 10. Conclusion

The project offers the elderly or medically sick people a personal assistant which reminds them of the medicines to be consumed at the particular time. Skipping tablets may lead to serious problems if the person has a severe illness and this can be avoided. Since the cloud is integrated with the mobile application, numerous data can be fed into the database and notifications can be generated. The mobile application developed is highly customisable by the user and easy to use.



## 11. Future Scope

The project can be further developed by bringing into the feature of informing the medicine name during the notification. The voice assistance which is given can be customized by adding the user's voice or the caretaker's voice. Further the mobile application can update medicines by taking voice commands as an input from the user.

## 12. Appen

### dix Source Code:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#include "SoundData.h"
#include "XT_DAC_Audio.h"
XT_Wav_Class
Sound("voice_command.wav");
XT_DAC_Audio_Class DacAudio(2,0);
uint32_t DemoCounter=0;

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "9a7os9"//IBM ORGANITION ID
#define DEVICE_TYPE " ESP" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID " ESP32"//Device ID mentioned in ibm IOT Platform #define
TOKEN " LC!x?+V9etumdVMaSR " //Token
String data3;
float h, t;

//----- Customise the above values -----
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and
format in which data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command type
AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
```

```
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
```

```
// _____
```

```
WiFiClient wifiClient; // creating the instance for wificlient
```

```
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by  
passing parameter like server id,portand wificredential
```

```
void setup()// configureing the ESP32
```

```
{  
  Serial.begin(115200);
```

```
  
  delay(10);  
  Serial.println();  
  wificonnect();  
  mqttconnect();  
}
```

```
void loop()// Recursive Function
```

```
{  
  
  
  
  
  
  
  
  
  delay(1000);  
  if (!client.loop())  
  { mqttconnect();  
  }  
}
```

```
/* .....retrieving to Cloud */
```

```
void mqttconnect() {  
  if (!client.connected())  
  { Serial.print("Reconnecting client  
to "); Serial.println(server);  
    while (!client.connect(clientId, authMethod, token)) {
```

```

    Serial.print(".");
    delay(500);
}

    initManagedDevice()
    ; Serial.println();
}
}
void wificonnect() //function defination for wificonnect
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the
connection
    while (WiFi.status() != WL_CONNECTED)
    { delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi
connected"); Serial.println("IP
address: ");
    Serial.println(WiFi.localIP());
}

void initManagedDevice() {
    if (client.subscribe(subscribetopic))
    { Serial.println((subscribetopic));
      Serial.println("subscribe to cmd OK");
    } else {
      Serial.println("subscribe to cmd FAILED");
    }
}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{

    Serial.print("callback invoked for topic: ");
    Serial.println(subscribetopic);

```

Is

```
for (int i = 0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]);
    data3 += (char)payload[i];
}

Serial.println("data: "+ data3);
if(data3=="announce")
{
Serial.println(data3);
for(int
i=0;i<5;i++){ DacAudio.F
illBuffer();
if(Sound.Playing==false)
    DacAudio.Play(&Sound);
    Serial.println(DemoCounter++);
}
}

else
{
    pass;

}
data3="";

}
```

**Git Hub Link :** <https://github.com/IBM-EPBL/IBM-Project-55218-1667628039>

---