Date	12 November 2022
Team ID	PNT2022TMID21926
Project Name	Project - Personal Assistance for Seniors Who Are Self-Reliant

1. INTRODUCTION

1.1 Project Overview

A pill reminder is **any device that reminds users to take medications**. Traditional pill reminders are pill containers with electric timers attached, which can be preset for certain times of the day to set off an alarm.

1.2 Purpose

Some peoples are forgetting to take their medicines at the time mentioned due to some works or any other stuffs.In this system the main purpose of the project is to to develop a prototype of a smart medicine reminder for elderly people that helps them consume the medicines right on time.The main intention in developing this project is to usually remind the medicines which have to be taken in the daily schedule and it is especially useful for old age people, patients, and busy people. Medication reminders serve as a good way to stay on track and uphold an appropriate schedule. Ensuring that you or your loved one is properly taking their medications can help avoid unnecessary risk and serious illness.

2.LITERATURE SURVEY

1. Personal Assistance Device for Independent Senior Citizens/Patients(A. Yuvaraj K et al., Jan 2020)

- The personal assistance gadget for elderly people's health monitoring proposed in this research uses various sensors that can measure the elderly's posture and pulse rate. Therefore, if the device is utilised in a hospital, the doctor may quickly recognise the abnormal results and treat the patient. The display on the OLED screen and buzzer both serveas indicators of proper medication intake at the appropriate time.

- The mobile application for this device displays the pulse ratemeasurements as well as the accelerometer and pulse sensor readings for the person being tracked.IOT pulse sensor and accelerometer connections allow for information sharing and communication between patientsand doctors.

2. A Smart-Home IoT Infrastructure for the Supportof Independent Living of Older Adults(StefanosStavrotheodorosset al., June 2018)

- The smart home IOT architecture in this researchis designed to assistindependent living for older adults over 65 who have ongoing health issues or are fragile. Here, a gateway connects the installed supporting devices to the internet, establishing a machine-to-machine network. There are three layers in the architecture.
- First perception layer, which consistsof two components and gathersuser
- data for integration into the following layer.
- All sensor data is collected by the second gateway layer and sent to the third tier. Here, message queuing telemetry transport protocols are utilised to enable communication across devices with the usage of a specific device calledan aggregation point(MQTT).
- The third cloud layer is based on software that gathers data from installed sensors, offers data storage, and supports decision-making by analysing the data throughsophisticated Data Analyticsand visual analyticsto end users.

3. Development of an IoT-Based HealthPromotion System forSeniors(Chia-Hui Liu.,Oct 2020)

- Using wireless technology in combination with physiological measuring methods, home care devices, and can help seniors maintain their health and get home health care services. This paper includes a wireless sensor network to create context-sensitive health promotion for an aged care system. The system is separated into three subsystems: the context awareness-based service subsystem, the elderly nutrition diet and health promotion subsystem, and the IoT-basedphysiological information subsystem.
- Long-term elderlydiet and activity records can be integrated by the

- system, which can then help the elderly complete their own nutrition assessment and healthmanagement.

4. Virtual Agents as Daily Assistants for Elderly or CognitivelyImpairedPeople(Ramin Yaghoubzadeh.,2013)

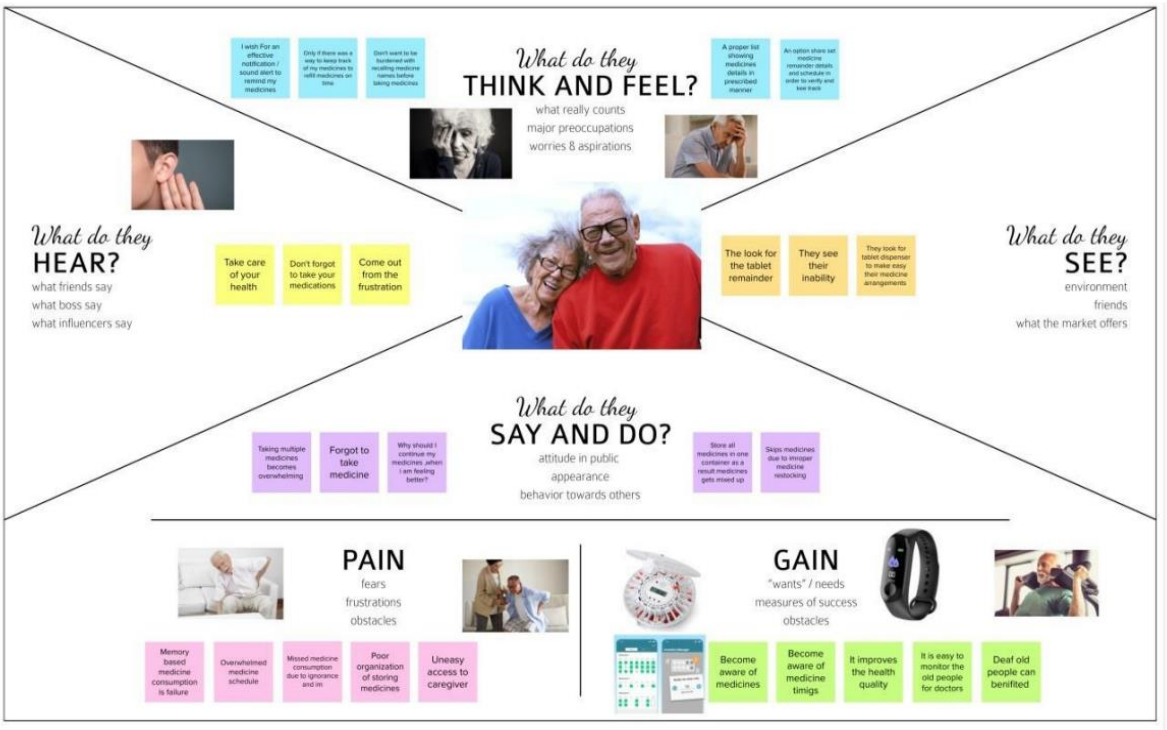
- In this paper, studies that address these issues for older users and users with cognitive impairment are presented. Results from focus groups and interviews indicate that using a participatory design approach can boost acceptance. Actual experiments of interaction with a prototype show thatspoken-language interaction is possible and provide techniques to reduceunderstandingissues.
- Researchers have examined whetherand how virtual agents can help.
- people with cognitive (and possibly other) disabilities manage their dailyschedule and calendar, and we have shared our preliminary findings in this study.

3.PROBLEM STATEMENT

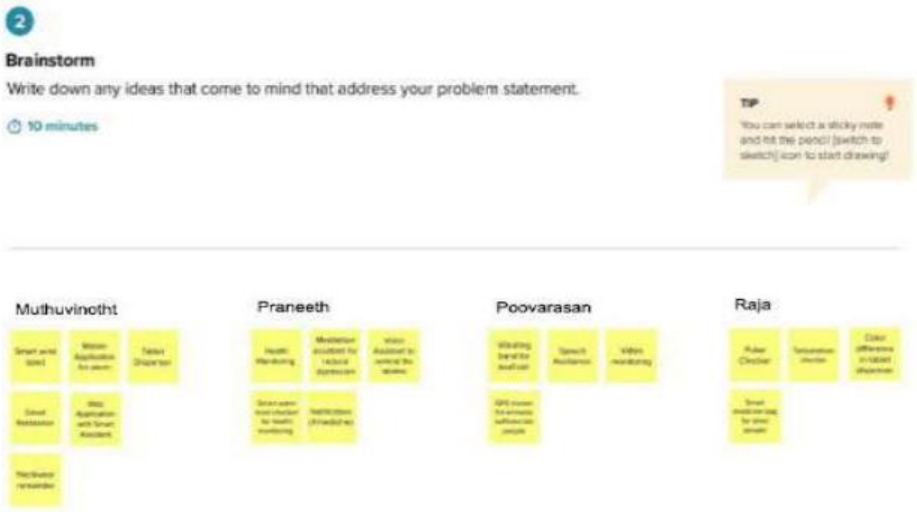
Some people find it difficult to learn new apps in this ever-expanding digital environment, and people nowadays tend to forget things more easily, such as taking their prescriptions. People need a way toremember to take their prescriptions without having to learn how to use sophisticated programs. To develop a prototype of a smart medicine reminder for elderly people that helps them consume the medicines right on time.

4.IDEATION AND PROPOSED SOLUTION

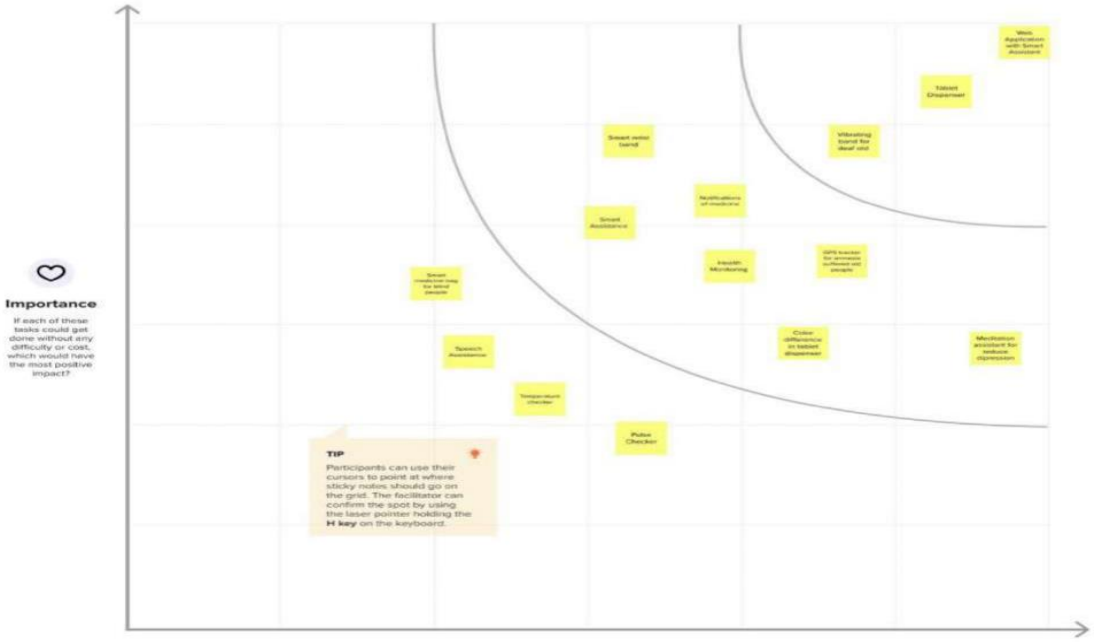
4.1 Empathy Map



4.2 Brainstorming



4.3 Prioritazation



4.4 Proposed Solution

S.No	Parameter	Description
1	Problem statement (problem to be solved)	During old age Seniors may face Memory loss and Health issues, so they may not be able to take care of themselves. It leads to chain of issues.
2	Idea / Solution description	The smart Alert present in the Mobile can rescue the Seniors from situation getting worsen.
3	Novelty / Uniqueness	In this smart dual alert application, there is a Medicine reminder as well as Emergency Notification alert for the Doctor and Care taker.
4	Social Impact / Customer Satisfaction	This provides them with the confidence that Seniors may lead a smoother life with this support.
5	Business Model (financial Benefit)	This feature in the mobile urges the society to buy the Mobile Phone.
6	Scalability of Solution	It gives secured feeling for their children's that their parents are safe.

4.5 Problem Solution Fit

5.PROJECT DESIGN

5.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

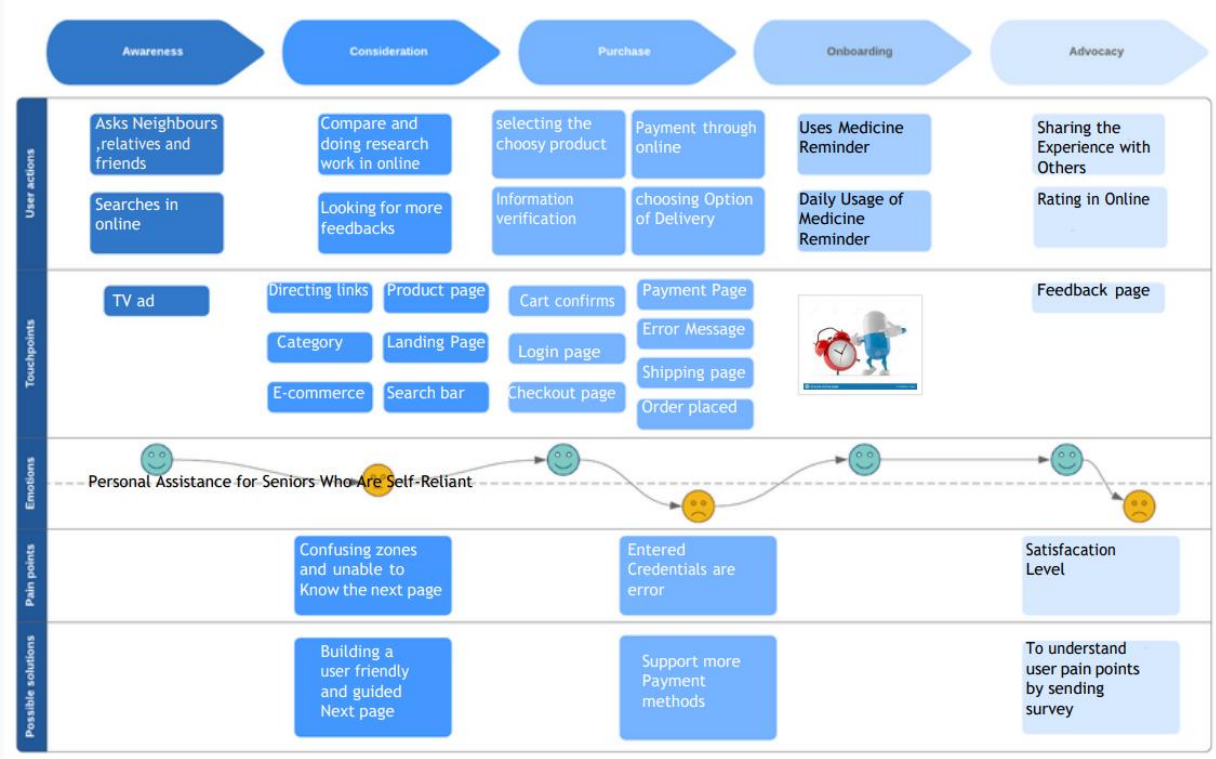
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login Id	Login Credentials
FR-4	User Dashboard	History of medicines
FR-5	Medicine Reminder (Time, medicine Name)	Speaker must be connected

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system should be easy to install and simple to use
NFR-2	**Security**	The system should authenticate users
NFR-3	**Reliability**	The system should perform the intended tasks for a specific time
NFR-4	**Performance**	The system should perform well under different conditions
NFR-5	**Availability**	The system should be available all the time when required
NFR-6	**Scalability**	The system should be able to add new features

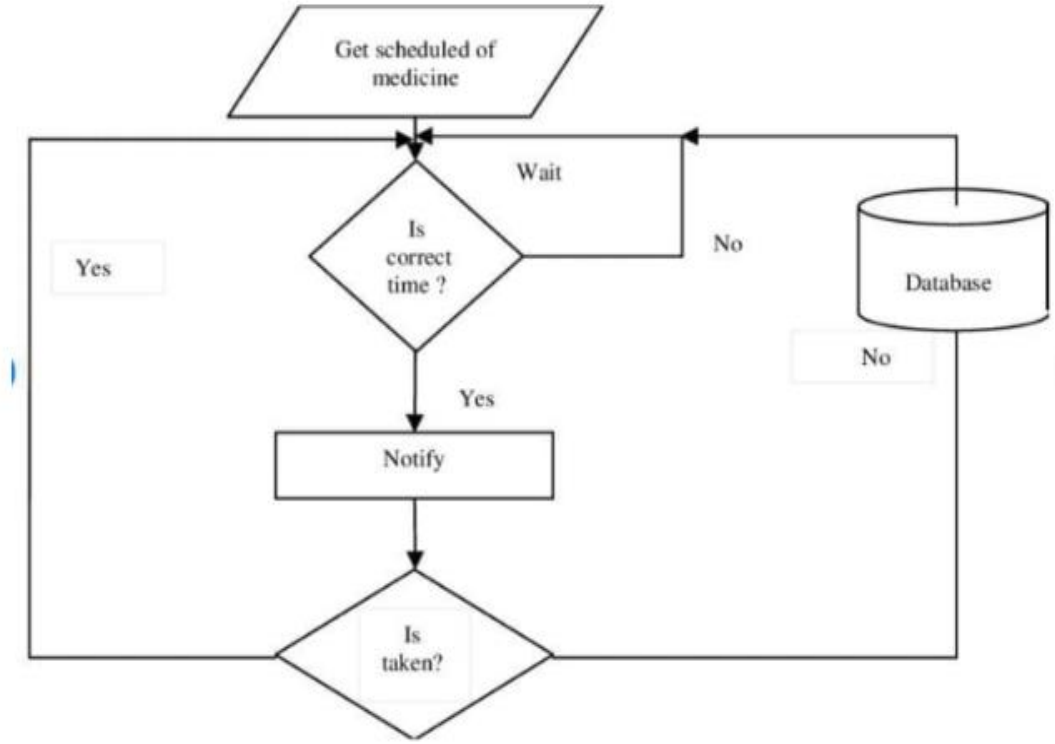
5.2 Customer Journey Map



5.3 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, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		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
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user, I can login into dashboard and search the access account and receive mail			
Customer (Web user)	Login	UI	As a user, I need to create an account by providing all the necessary information (e.g.name, password, mobile, email, address)		High	Sprint-1
Customer Care Executive	Registration	UX	As a customer, I need register for the care executive for the application	I can register and access the account	High	Sprint-1
Administrator	Confirmation		As a customer, confirmation mail once registered for the web user.		High	Sprint-1

5.4 DataFlow Diagrams



5.5 Technology Architecture

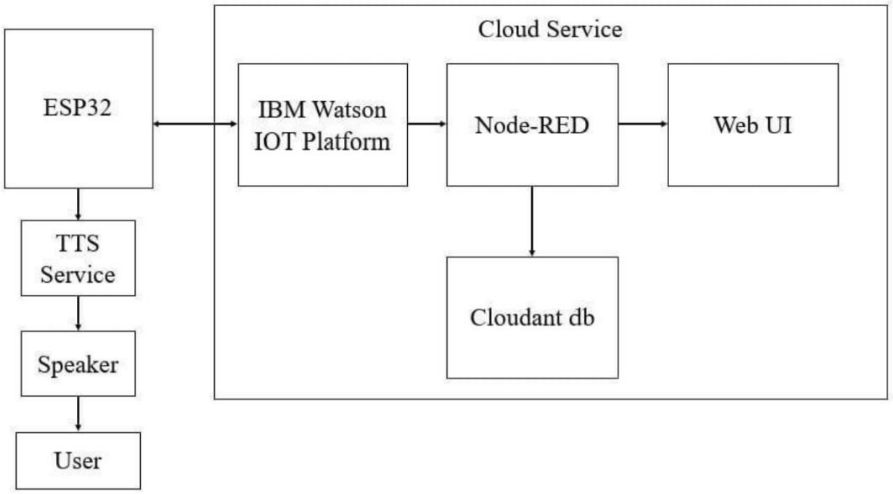


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	JavaScript
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Cloud Database	Database Service on Cloud	IBM Cloudant
6.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem

7.	External API-1	Purpose of External API used in the application	IBM Geolocation API, etc.
8.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Cloud Foundry

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Technology used
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

6.PROJECT PLANNING AND SCHEDULING

6.1 Sprint Delivery Plan

Sprint	Functional Requiremen t (Epic)	User Story Numb er	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	Muthuvinoth, Praneeth, Poovarasan, Raja
Sprint 1		USN-2	As a user, I can Activate and Deactivate the alarm	10	High	Muthuvinoth, Praneeth, Poovarasan, Raja
Sprint 2	Notification	USN-3	As a user once I can the set the alarm then I gets the notification	10	High	Muthuvinoth, Praneeth, Poovarasan, Raja
Sprint 2		USN-4	As a user, If I requires this system then a notification will be sent into his device.	10	High	Muthuvinoth, Praneeth, Poovarasan, Raja
Sprint 3	Medication Detail	USN-5	As a user, I have multiple medications each day, I can put each pill in the box for the corresponding day.	10	High	Muthuvinoth, Praneeth, Poovarasan, Raja
Sprirnt 3		USN-6	As a user , between setting alarm and using a pillbox, I'll be able to stay on the top of your medications and not miss a dose.	5	Low	Muthuvinoth, Praneeth, Poovarasan, Raja
Sprint 3		USN-7	As a user ,I can store the name of the medicine with its description.	10	High	Muthuvinoth, Praneeth, Poovarasan, Raja
Sprint 4	GPS Tracking	USN-8	As a user, they can also help large hospitals and their inventory more effectively. reminding the schedule of medicine.We have used the IOT enabled arduino device for monitoring the System.	5	Low	Muthuvinoth, Praneeth, Poovarasan, Raja Poovarasan, Raja

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint 1	20	8 days	29-10-2022	5-11-2022	20	4-11-2022
Sprint 2	10	8 days	7-11-2022	14-11-2022	10	13-11-2022
Sprint 3	20	8 days	16-11-2022	23-11-2022	20	23-11-2022
Sprint 4	10	8 days	23-11-2022	30-11-2022	10	30-11-2022

7.Coding and Solutioning

```
<LiquidCrystal.h>

<RTClib.h>   (https://github.com/adafruit/RTClib)

<EEPROM.h>

<Wire.h>

 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__)));

//rtc.adjust(DateTime(2019, 1, 10, 7, 59, 52));
```

```
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 = 01;
      break;
     case 2:
      Serial.println("Set for 2/day");
      push1state = 0;
      push2state = 1;
      push3state = 0;
      pushVal = 10;
      break;
     case 3:
      Serial.println("Set for 3/day");
      push1state = 0;
```

```
    push2state = 0;
    push3state = 1;
    pushVal = 11;
    break;
  }
```

This statement is used to get the millis to use for timing and control of the defined interval screen cycling.

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

Start reading the digital pins connected to push buttons.

```
push1state = digitalRead(push1pin);
push2state = digitalRead(push2pin);
push3state = digitalRead(push3pin);
stopinState = digitalRead(stopPin);
```

Below function is used to read the push button state and write it to EEPROM. Whenever the push button gets pressed the state is written to EEPROM. Also it prints the message on LCD display of the selected user input choice. Similarly the functions push2() and push3() is used.

```
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();
  }
}
```

Below function is used to stop the buzzer and led. It is always good to give suggestions. Print a suggestion message on display “Take medicine with warm water”.

```
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();
  }
}
```

The below function is independent of the time keeping but always cycles in three screens which helps user. As we are keeping a care to patients lets print a greeting message as we know that emotional support is very helpful in healing patients in more quick time. You can choose your own creative message. Let's print a message as "Stay healthy, Get well soon".

The second screen is for giving instruction to patients as "Press buttons for reminder..". The Third screen is used to simply show current date and time.

```
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) {
      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;
    }
  }
}
```

This function is used to start buzzing and blinking the LED simultaneously if the selected time has reached.

```
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);
  }
}
```

This function is used to compare the user selected time slot at 8am and starts buzzing the buzzer and blinking the LED until user presses the stop push button. Similarly the functions void at2pm() and void at8pm is used to start buzzer and led at 2pm and 8pm.

```
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();
        ////////////////////////////////////////////////
      }
    }
```

```
  }

}
```

This is how you can simply make your own Automatic Medicine Reminder using Arduino. You can also use ESP8266 with Arduino to make it an IoT project which will be able to send email alert to the user.

```
//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

//bool pushPressed;                          //flag to keep track of push button state


int pushpressed = 0;

const int ledPin =  LED_BUILTIN;                    // buzzer and led pin

int ledState = LOW;

int Signal = 0;

int buzz = 13;
```

```
int push1state, push2state, push3state, stopinState = 0;     //

int push1Flag, push2Flag, Push3Flag = false;              // push button flags

int push1pin = 9;

int push2pin = 8;

int push3pin = 7;

int stopPin = A0;

int screens = 0;              // screen to show

int maxScreen = 2;            // screen count

bool isScreenChanged = true;

long previousMillis = 0;

long interval = 500;                  // buzzing interval

unsigned long currentMillis;

long previousMillisLCD = 0;    // for LCD screen update

long intervalLCD = 2000;          // Screen cycling interval

unsigned long currentMillisLCD;

//   Set Reminder Change Time


int buzz8amHH = 8;          //    HH - hours         ##Set these for reminder time in 24hr Format

int buzz8amMM = 00;          //    MM - Minute

int buzz8amSS = 00;          //    SS - Seconds
```

```
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 messeges

void gwsMessege(){               // print get well soon messege

   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 debuggin
    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) {
```

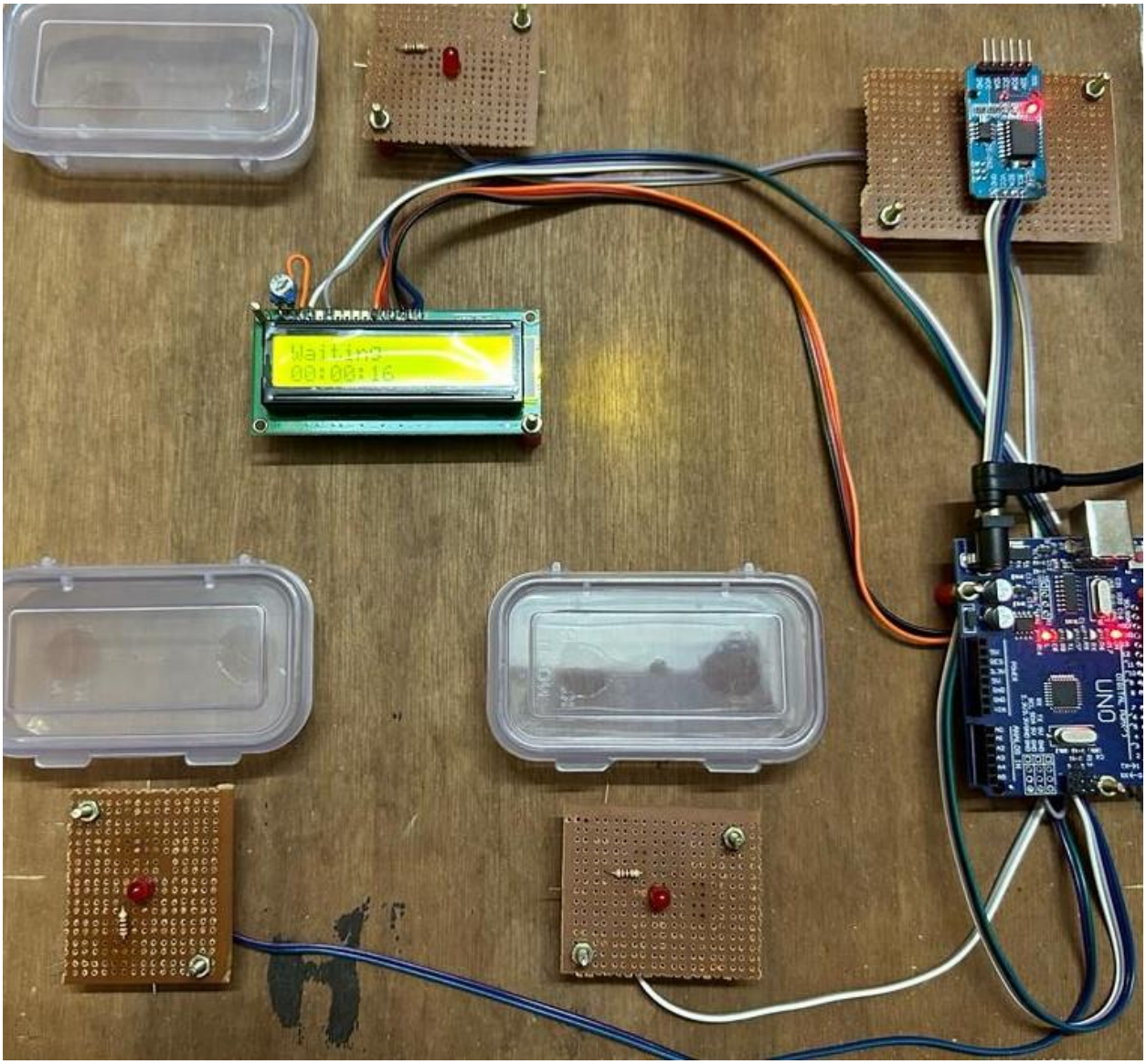
```
    if (int(now.second()) > buzz8amSS) {
      ////////////////////////////////////////////////
      startBuzz();
      ////////////////////////////////////////////////
    }
   }
  }
}
void at2pm() {                    // function to start buzzing at 2pm
  DateTime now = rtc.now();
  if (int(now.hour()) >= buzz2pmHH) {
   if (int(now.minute()) >= buzz2pmMM) {
    if (int(now.second()) > buzz2pmSS) {
      ////////////////////////////////////////////////
      startBuzz();
      ////////////////////////////////////////////////
    }
   }


  }
```

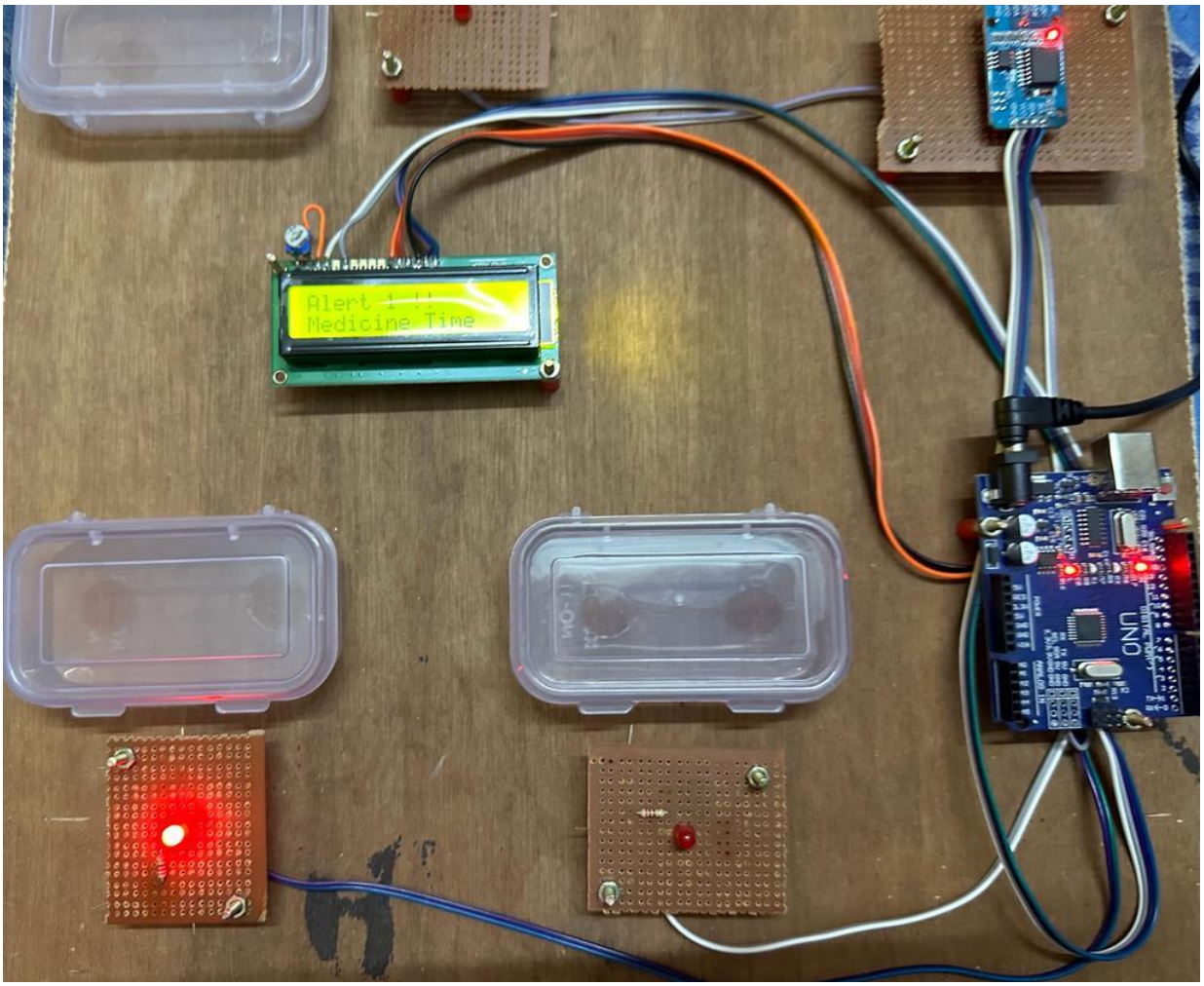
```
}

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) {
    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
```

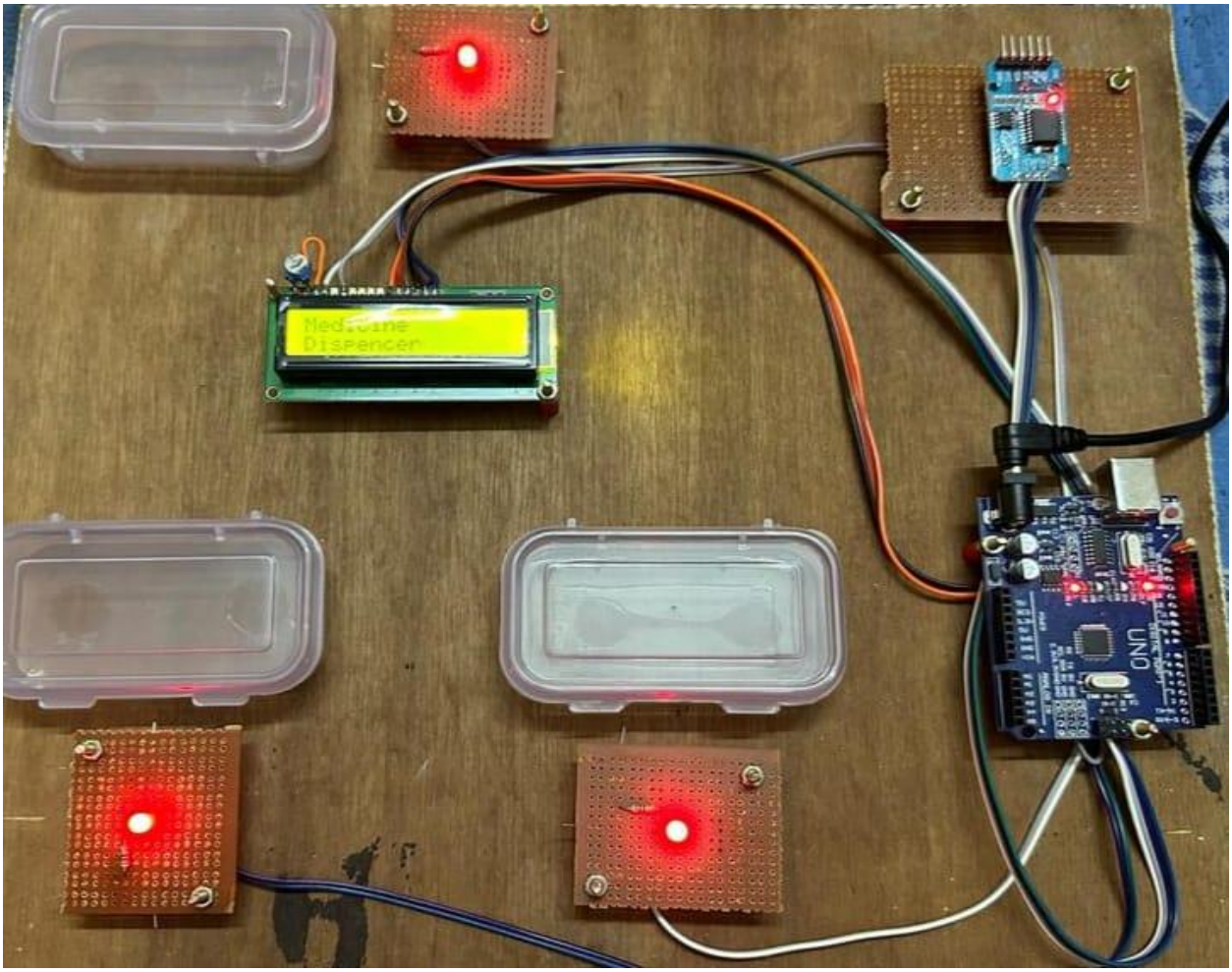
```
        break;
      default:


        //NOT SET.
         break;
    }
  }
}
```

OUTCOME


Waiting
00:00:16
UNO


Alert !!!
Medicine Time
UNO


Waiting
08:08:24


Medicine
Dispencer
UNO


Alert 1 !!
Medicine Time

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Medication tracking history.
- Flexible scheduling.
- Compliance reminders and alerts.
- Shape and colour identification feature.
- Repeat or refill reminder.
- Notification for other people.
- Reports for sharing.

DISADVANTAGES

- costs associated with running an IT system.
- possibility of technical problems
- Does not encourage cancellation or rescheduling in patients who cannot attend or who no longer wish to attend.
- Inner city populations may have less stable contact details (either address or phones) and this may put these patients at a specific disadvantage.

8.CONCLUSION

These three associated reviews have found strong, consistent evidence to support the use of all reminder systems for all patients in any outpatient setting for increasing attendance, cancellation or rescheduling. There is additional evidence that ‘reminder plus’, which provides additional information over and above date, time and location of the appointment, may be more effective than simple reminders at reducing non-attendance, particularly at first appointments. However, there is a need for high-quality studies investigating the differential influence of providing additional information as part of the reminder system in different contexts (first compared with follow-up appointment, the use of loss- compared with gain-framed messages and orientation information for facilitating attendance behaviours)There was strong evidence that:the timing of reminders, between 1 and 7 days prior to the scheduled appointment, has no effect on attendancea substantial number of reminders may not be received by patientsreminders promote cancellation of appointmentspatients find difficulty with cancelling appointments because of structural factors affecting reminder systems (e.g. busy telephone line, nobody answers the telephone).This leads to the conclusion that, unless patients indicate otherwise, all patients should receive a reminder or ‘reminder plus’ that actively encourages patients who are unable to attend to cancel their appointment and to reschedule if further appointments are required. The reminder should be sent around 3 days in advance of the appointment as timing of a reminder, between 1 and 7 days prior to the scheduled appointment, has no effect on patient attendance behaviour. This will allow sufficient time for patient cancellation and health service reallocation of the appointment to another patient or allow the clinician to undertake care-related administrative tasks. To optimise attendance, cancellation and rescheduling there needs to be robust procedures to ensure that patient contact details are up to date and that there are easy to use, probably multiple, systems for cancelling appointments that suit the needs of the patients, e.g. automated SMS cancellation, answerphone, e-mail, etc. Robust 24-hours-per-day rescheduling procedures should allow easy rescheduling of appointments for patients. Finally, an effective reminder system will increase the workload on clinical staff and alternative time will need to be scheduled for staff to undertake health-care-related administration. Further research is required to investigate the differential effectiveness and cost-effectiveness of an ‘optimised’ reminder system over and above usual reminder systems.There were few studies investigating the differential effectiveness of alternative types of reminders for different segments of the population and this we believe is a key area for further research. Nevertheless, we have used the findings of our review to suggest possible reminder alternatives for key groups of patients who appear to be at higher risk of not attending appointments, namely deprived, ethnic,

substance abusers, and those with comorbidities and illness. Simple reminders and automated reminders to attend may be ignored or overlooked and may put these patient groups at a disadvantage compared with general outpatient populations. Reminders with direct personal contact might be appropriate in these groups. To facilitate attendance in the most at risk, vulnerable groups we have suggested that reminder systems of increasing intensity and interactivity could be introduced to ensure that disparities in health-care opportunities are not compounded. We have introduced the concept of a sequential reminder intervention in order to reach the most number of patients and maximise attendance, although their effectiveness in this context needs to be established. The re-engagement of these patient groups with treatment after they have missed their appointment may be important if they have particular health problems that need ongoing treatment. Intensive approaches, such as 'stepped reminders' and patient navigators have been effective at increasing attendance at screening and immunisation programmes in disadvantaged and vulnerable populations, although their effectiveness in this context needs to be investigatedReminder systems are a complex intervention, because of the potential number of interacting components within the interventions, the requirement for tailoring of the intervention to the health service and the number of difficulties and behavioural changes from those receiving and delivering the reminder. Therefore, in addition to following the general recommendations provided above, health service managers will need to tailor their reminder systems to meet the needs of the service and the patient population that it serves. This review provides a range of findings that will inform health service managers' decision-making processes. To this end, we are producing a practice guide to help health service managers consider specific issues that may be relevant to the design of reminder systems for their health service.

9.FUTURE WORKS

There are many Medication Reminder Systems developed on different platforms.These systems require either complex hardware or software applications to remindthe patients about the medicine intake timings. Many of them are costly and moretime consuming. Moreover these systems need adequate prior knowledge tooperate those complex systems. So in this work an attempt has been made toimplement a system which is economical, easily accessible and improvesmedication adherence .The system can be updated by adding some additionalfeatures like notification for next heath check-up, automatic consultation to thedoctor etcA button on the device, on pressed can send an alert call/text to the caregiver if the user is ill or needs help.A button, when not pressed after the notification, sends a message to the user and the caregiver an alert prompt Medicine not taken kindly take it.

10.APPENDIX

Source Code

https://github.com/IBM-EPBL/IBM-Project-7734-1658896808

GitHub Link

https://github.com/IBM-EPBL/IBM-Project-7734-1658896808

Project Demo Link

https://youtu.be/9ppelWkgLFw