PERSONAL ASSISTANCE FOR SENIORS WHO ARE SELF-RELIANT

Team ID : PNT2022TMID16081

TABLE OF CONTENT

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. **IDEATION & PROPOSED SOLUTION**
    3.1. Empathy Map Canvas
    3.2. Ideation & Brainstorming
    3.3. Proposed Solution
    3.4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    4.1. Functional requirement
    4.2. Non-Functional requirements
5. **PROJECT DESIGN**
    5.1. Data Flow Diagrams
    5.2. Solution & Technical Architecture
    5.3. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    6.1. Sprint Planning & Estimation
    6.2. Sprint Delivery Schedule
7. **CODING & SOLUTIONING**
    7.1. Feature 1
    7.2. Feature 2
8. **TESTING**
    8.1. Test Cases
9. **RESULTS**
    9.1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code
GitHub & Project Demo Link

INTRODUCTION

PROJECT OVERVIEW:

In modern society, busy life has made people forget many things in day to day life. The elderly people and the people victims of chronicle diseases who need to take the medicines timely without missing are suffering from dementia, which is forgetting things in their daily routine. Considering this situation study has been done in this. Paper reviewing the technologies of home health care which are currently used for improving this situation by reminding the scheduled of medicine, remote monitoring and update new medicine data of patients, which can be done by prescriber through web. Elderly people forget to take their medicine at the correct time. They also forget which medicine they 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.

PURPOSE:

- Today medicine is used to control blood pressure, insulin, cholesterol and even the rate at which our hearts beat. Yet medicines are both a godsend and a curse.
- 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.
- 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.

LITERATURE SURVEY

EXISTING PROBLEM

Kartheek.K et al.,(2019) presented IOT devices to set reminders to the people though a medication box. which will alert them time to time to take medication. If a person is not carrying IOT Devices, it will send alerts to mobile via SMS and if a person is harrying a laptop, it will send an email alert. Suganya G et al.,(2019) proposed a dispenser that can dispense medicines to the patients, based on the prescriptions suggested by the doctor secured by individual barcodes. These dispensers are to be owned by the pharmacies and to be filled with medicines on the time of requirement (like ATM machines). Users of Online Health Communities are the target users of these machines who can get appropriate suggestions from the doctors associated with the same group. Mathur P. et al., (2019) proposed the system that an android application that can be used by doctors to create prescriptions for patients registered in OHC, taking medications on time is more crucial than ever because the human body requires certain medicines to prevent infections and other disorders while also giving the barriers and challenges. The elderly people requires a timely remainder to take their medicines. The urgent need is to create a system that allows physicians, care givers and workers to oversee their elderly people's health. It is laborious to nurture daily multiple medications. The Senior Citizen's is at prominent risk for medication-related problems.

The existing system uses IOT devices to set remainders, which will periodically remind them to take their prescription. It will send SMS notifications to mobile phones if a person is not carrying an IOT device and an email alert if a person is carrying a laptop, defined in the paper[1]. An IOT device that can get the schedule from the cloud that has been provided by a doctor's medication will remind the medicines in time and help to treat the illness or contamination the person is experiencing.

REFERENCE:

[1].Kartheek K, Saddam Hussain.SK, Medical Dispense System Using IOT, 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking.

[2]. Suganya G, Premalatha M, Anushka Sharma, Muktak Pandya, Abhishek Joshi February 2019 International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7, Issue-5S4.

[3]. A.P. Sankar, D.C. Nevedal, S. Neufeld, M.R. Luborsky.
[4].Huh J, Mcdonald DW, Hartzler A, Pratt W. Patient Moderator Interaction in Online Health Communities. AMIA. 2013 In Press. [PMC free article] [PubMed].

[5]. Shawn Benedict Kumar, Wei Wei Goh, Sumathi Balakrishnan, Smart Medicine Reminder Device For The Elderly, 26-28 October 2018.

[6]. Kuperman, G. J., Bobb, A., Payne, T. H., Avery, A. J., Gandhi, T. K., Burns, G., Classen, D. C., … Bates, D. W. (2007). Medication-related clinical decision support in computerized provider order entry systems: a review. Journal of the American Medical Informatics Association : JAMIA, 14(1), 29-40.

[7]. Sawand, S. Djahel, Z. Zhang, F. NaMultidisciplinary approaches to achieving efficient and trustworthy eHealth monitoring systems. Commun. China (ICCC), 2014 IEEE/CIC Int. Conf. (2014), pp. 187-192

[8]. Clifton, D. Wong, L. Clifton, S. Wilson, R. Way, R. Pullinger, *et al.*
A large-scale clinical validation of an integrated monitoring system in the Emergency Department.IEEE J Biomed Heal Informatics, 17 (4) (2013), pp. 835-842

[9]. Mathur P. (2019) Key Technological advancements in Healthcare. In: Machine Learning Applications Using Python. Apress, Berkeley, CA

PROBLEM STATEMENT:

The existing smart medicine reminder device's inability to assist patients who are prone to forgetfulness and also lack of a voice reminder is the problem. The main objective of this model is to address the mentioned issues by inventing and developing a tool that will allow the owner to track each pill they take in an easy and straightforward manner without the need for complicated training on their part. Sometimes elderly people forget to take their medicine at the correct time. The elderly population is at great risk for medication-related problems as a result of age-related physiological changes, the presence of multiple chronic diseases and conditions, and the types and numbers of prescription and non prescription medications they consume. They also forget which medicine they should take at that particular time. And it is difficult for doctors(caretakers) to monitor the patients around the clock. Common side effects of medicines in older adults can be dizziness and falls, weight loss or weight gain, and changes in memory or our ability to think and process information. These, in turn, can cause older adults to get hurt and may ultimately lessen their ability to function in day-to-day life. 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.

PROBLEM STATEMENT DEFINITION:

1. Who does the problem affect?

   People who are case-patient and forget to take medicine on time.
2. What are the boundaries of the problem?

   People who are the age (35-60) and majorly over 50
3. What is the issue?

   An aged person often has to take a variety of different medicines at various times. The elderly may find it difficult to remember to take the proper medication at the right time each day since it is not as simple as it could be for a younger person. Due to their impaired vision and the similarity in the shapes and colors of the pills, it may be challenging for them to remember which pill to take at the right time, to remember to take them, or to identify one pill from another.
4. When does the issue occur?

   When patients forget to take the medicine on time or when they face difficulty in distinguishing the pills. This leads to greater health problems
5. Why is it important that we fix the problem?

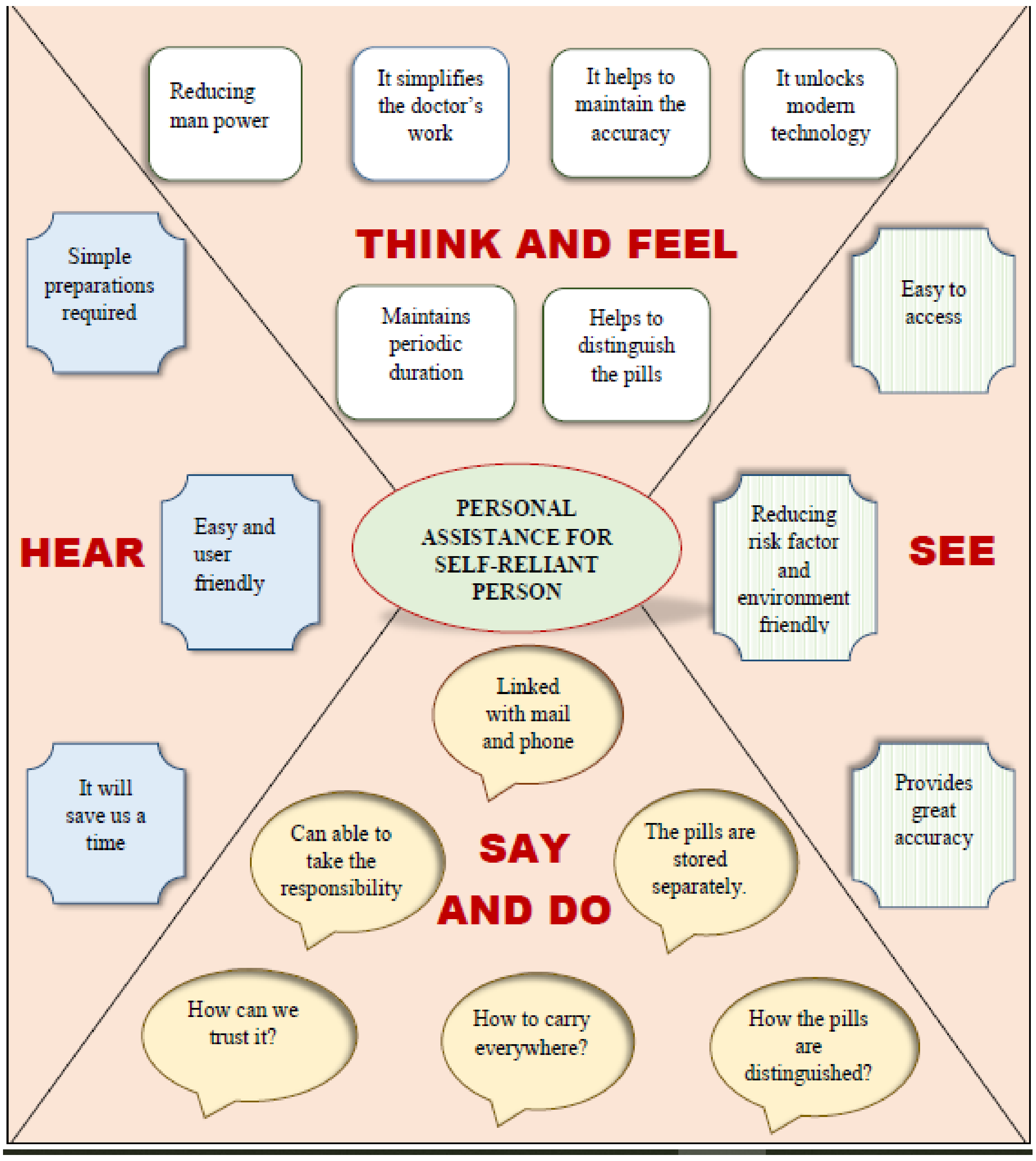
   It is very crucial to develop a application that gives instruction to the patients on time even when they have no alarming device which leads to various health problems.
6. Which solution can be used to address this issue?

   We build 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 We build an effective device with IOT platform which stores the timing of pills to be taken according to prescription in the cloud.
7. What methodology used to solve the issue?

   IBM Cloud, IBM Watson Studio,IBM Cloudant DB,IOT platform.

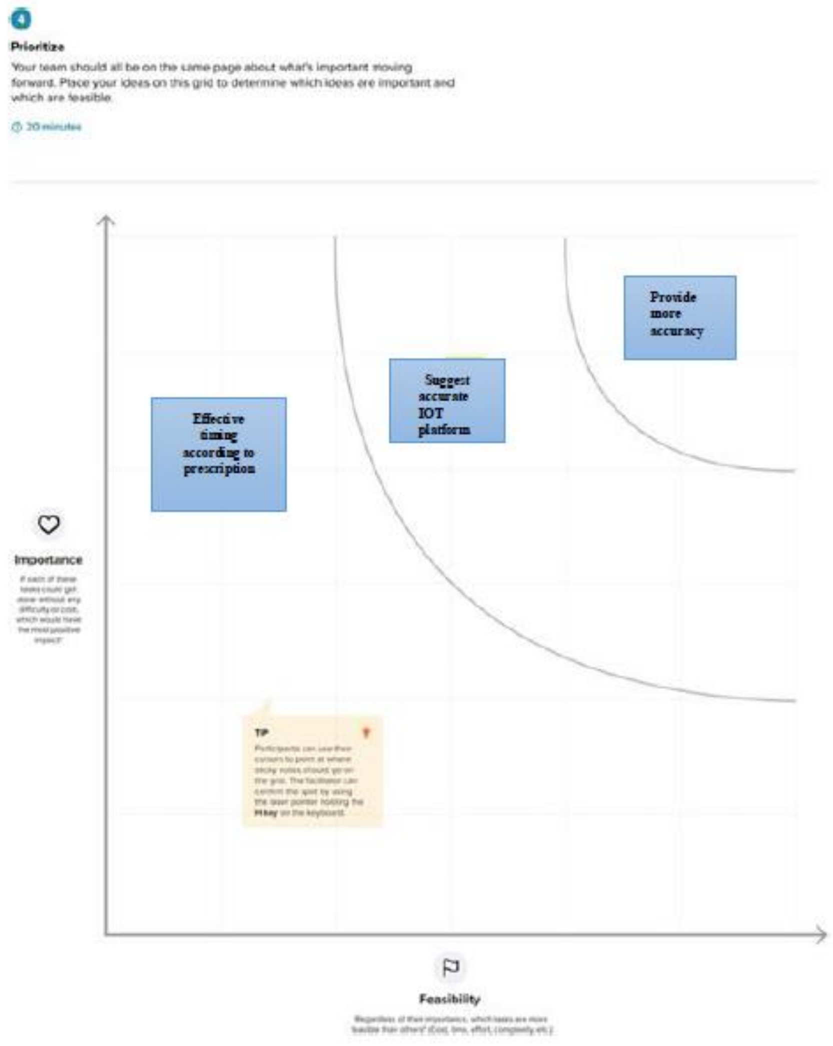
IDEATION AND PROPOSED SOLUTION

EMPATHY MAP CANVAS


Reducing man power
It simplifies the doctor's work
It helps to maintain the accuracy
It unlocks modern technology
THINK AND FEEL
Maintains periodic duration
Helps to distinguish the pills
Simple preparations required
Easy to access
HEAR
Easy and user friendly
PERSONAL ASSISTANCE FOR SELF-RELIANT PERSON
Reducing risk factor and environment friendly
SEE
Linked with mail and phone
It will save us a time
Can able to take the responsibility
SAY AND DO
The pills are stored separately.
Provides great accuracy
How can we trust it?
How to carry everywhere?
How the pills are distinguished?

<u>**IDEATION AND BRAINSTORMING**</u>

IDEATION:



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

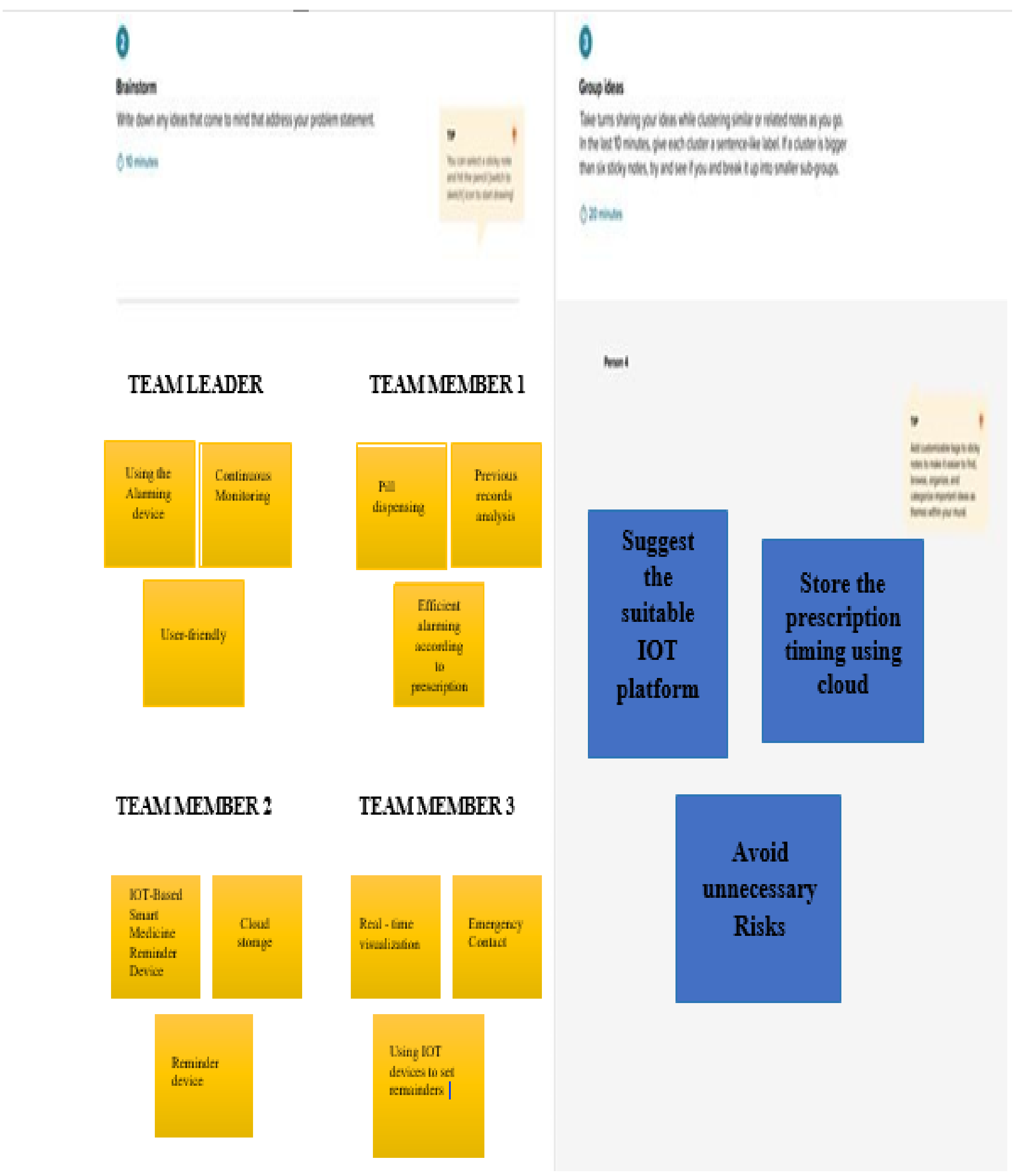
20 minutes

Effective timing according to prescription
Suggest accurate IOT platform
Provide more accuracy
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

BRAINSTROMING:



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Tip

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

TEAM LEADER

Using the Alarming device

Continuous Monitoring

User-friendly

TEAM MEMBER 1

Pill dispensing

Previous records analysis

Efficient alarming according to prescription

TEAM MEMBER 2

IOT-Based Smart Medicine Reminder Device

Cloud storage

Reminder device

TEAM MEMBER 3

Real - time visualization

Emergency Contact

Using IOT devices to set remainders

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

Person 4

Tip

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Suggest the suitable IOT platform

Store the prescription timing using cloud

Avoid unnecessary Risks

PROPOSED SOLUTION

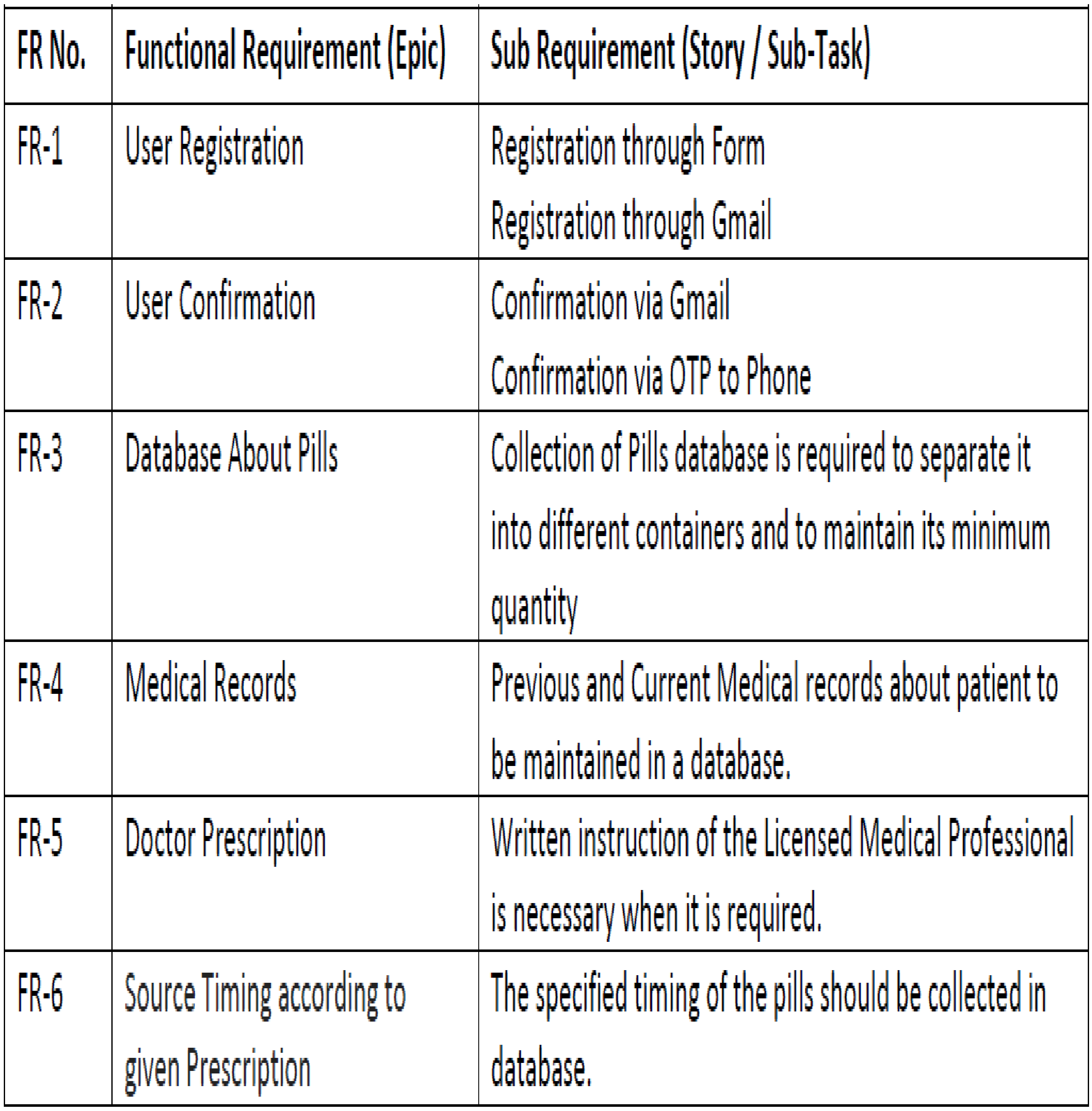
Elders who are in need to take medicine on time may forget the timing and they cannot able to distinguish the pills. This leads to various health problems. The IoT platform stores the database of medicine timing in the cloud and sends the notification to the device and the device notifies the person to take medicine on time. Specifically, we have used the IBM Cloudant to store the database and IoT platform to convert text message into voice message to notify the patient. As the elderly people cannot able to remember the regular timing to take medicine this system notifies the patients on time and it can be connected to email. Also they may not be able to read the text message so we use IoT platform of converting text message to voice message. This application is recommended to patients in low cost with subscription basis. Through IBM cloudant we can also store the patient database for regular medicine time. We can measure the accuracy through the database stored by the patient

PROPOSED SOLUTION FIT

- **CUSTOMER SEGMENT :** Elderly person who needs to take medicine on time for their diagnosed illness.
- **CUSTOMER CONSTRAINTS :** Storing database of medicine timing and connecting the device.
- **AVAILABLE SOLUTION:** Database is uploaded through the app and IoT is connected while distributing.
- **JOB-TO-BE DONE/PROBLEMS:** The collection of data must be stored in the Cloudant and the device can be connected to phone through mail.
- **PROBLEM ROOT CAUSE**:Taking medicine as prescribed is important for controlling chronic conditions, treating temporary conditions, and overall long-term health and well-being BE
- **BEHAVIOUR :** They must have to check the nearby hospitals. TR
- **TRIGGERS**: They may have trigger while not taking their medicine on time. **EMOTIONS:BEFORE/AFTER:**
  Before: Patients may get triggered emotionally after they cannot be able to distinguish the pills. After: But it helps to distinguish the pills by storing it in the IOT device.
- **YOUR SOLUTION :**Our project is about providing the IoT device that notifies the patient to take the medicine on time and also to distinguish the similar pills.
- **CHANNEL OF BEHAVIOUR** Online: The user can able to analyze the report of medicine timing. Offline: The used have to take pill on timing and can visit the pharmacy to distinguish the pills.

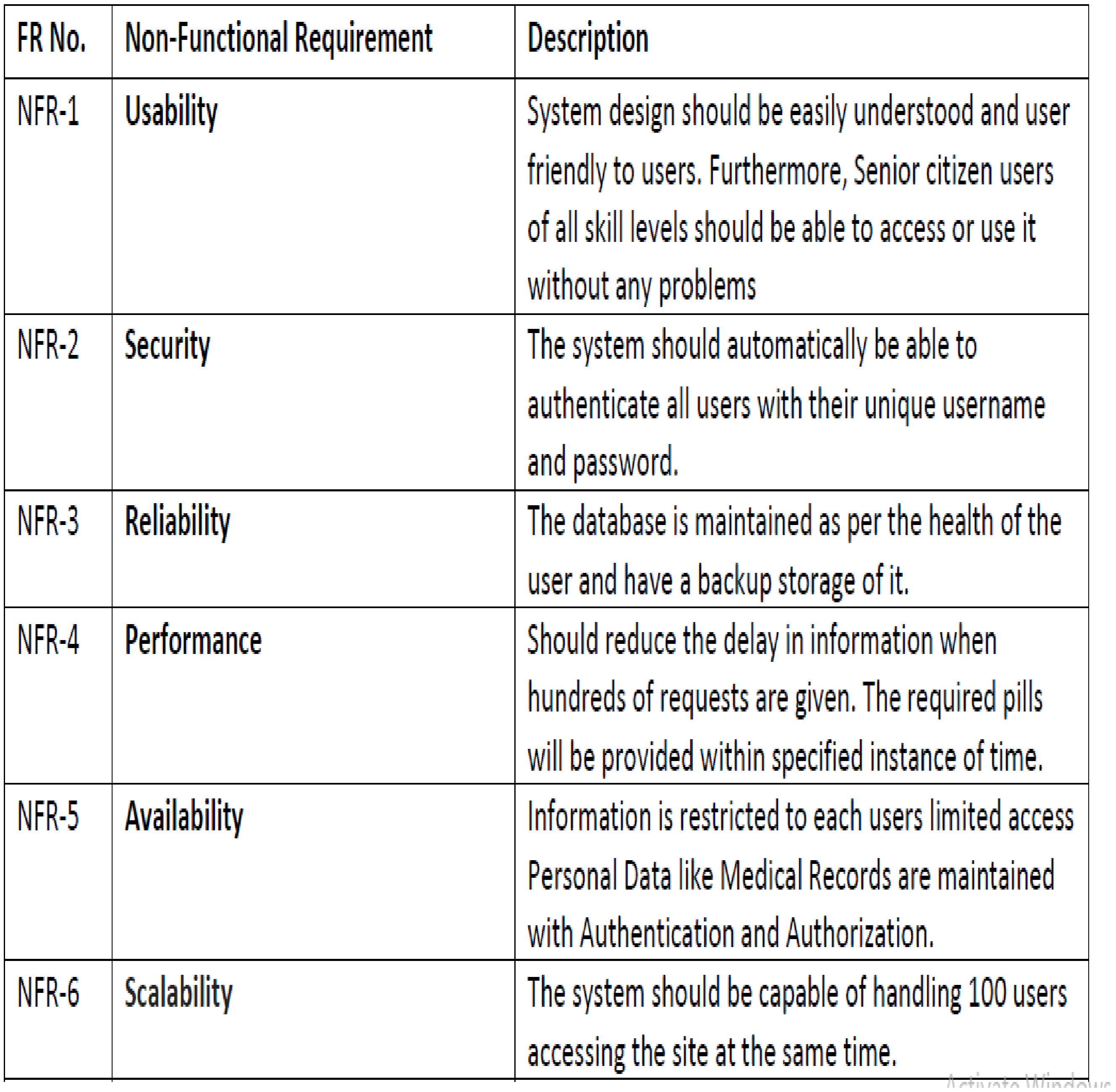
REQUIREMENT ANALYSIS

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 Gmail Confirmation via OTP to Phone
FR-3	Database About Pills	Collection of Pills database is required to separate it into different containers and to maintain its minimum quantity
FR-4	Medical Records	Previous and Current Medical records about patient to be maintained in a database.
FR-5	Doctor Prescription	Written instruction of the Licensed Medical Professional is necessary when it is required.
FR-6	Source Timing according to given Prescription	The specified timing of the pills should be collected in database.

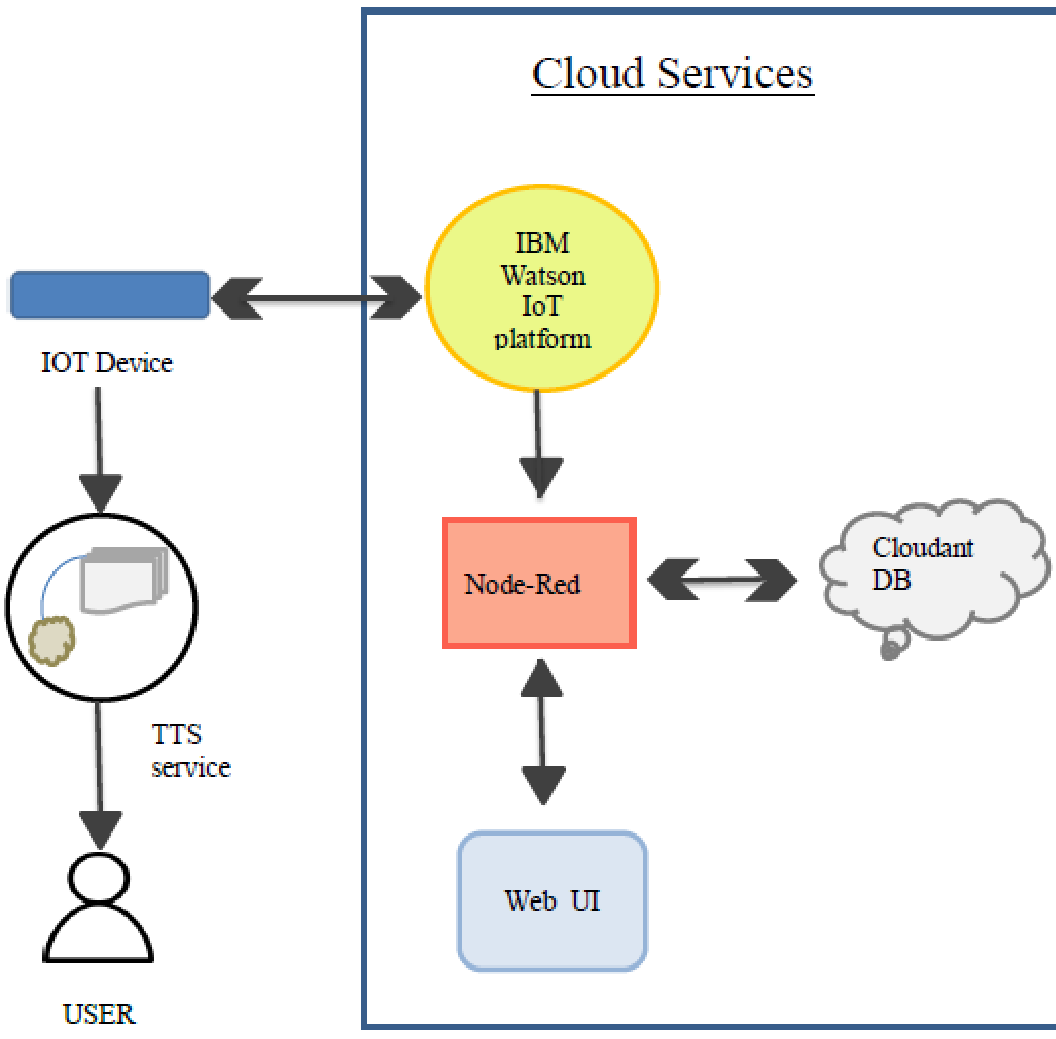
NON FUNCTIONAL REQUIREMENTS



FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	System design should be easily understood and user friendly to users. Furthermore, Senior citizen users of all skill levels should be able to access or use it without any problems
NFR-2	**Security**	The system should automatically be able to authenticate all users with their unique username and password.
NFR-3	**Reliability**	The database is maintained as per the health of the user and have a backup storage of it.
NFR-4	**Performance**	Should reduce the delay in information when hundreds of requests are given. The required pills will be provided within specified instance of time.
NFR-5	**Availability**	Information is restricted to each users limited access Personal Data like Medical Records are maintained with Authentication and Authorization.
NFR-6	**Scalability**	The system should be capable of handling 100 users accessing the site at the same time.

PROJECT DESIGN

DATA FLOW DIAGRAMS



SOLUTION AND TECHNICAL ARCHITECTURE

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:

- The best technological solution to remind the patient to take the medicine on time is to store the medicine time in the IBM cloudant. We employ IOT platform to transfer the medicine name to the device.
- Installing python SDK to connect the code with IBM Watson IOT platform.
- Development phases:

  Step 1: Create IBM Watson IOT platform.

  Step 2: Create the device and configure the IOT platform.

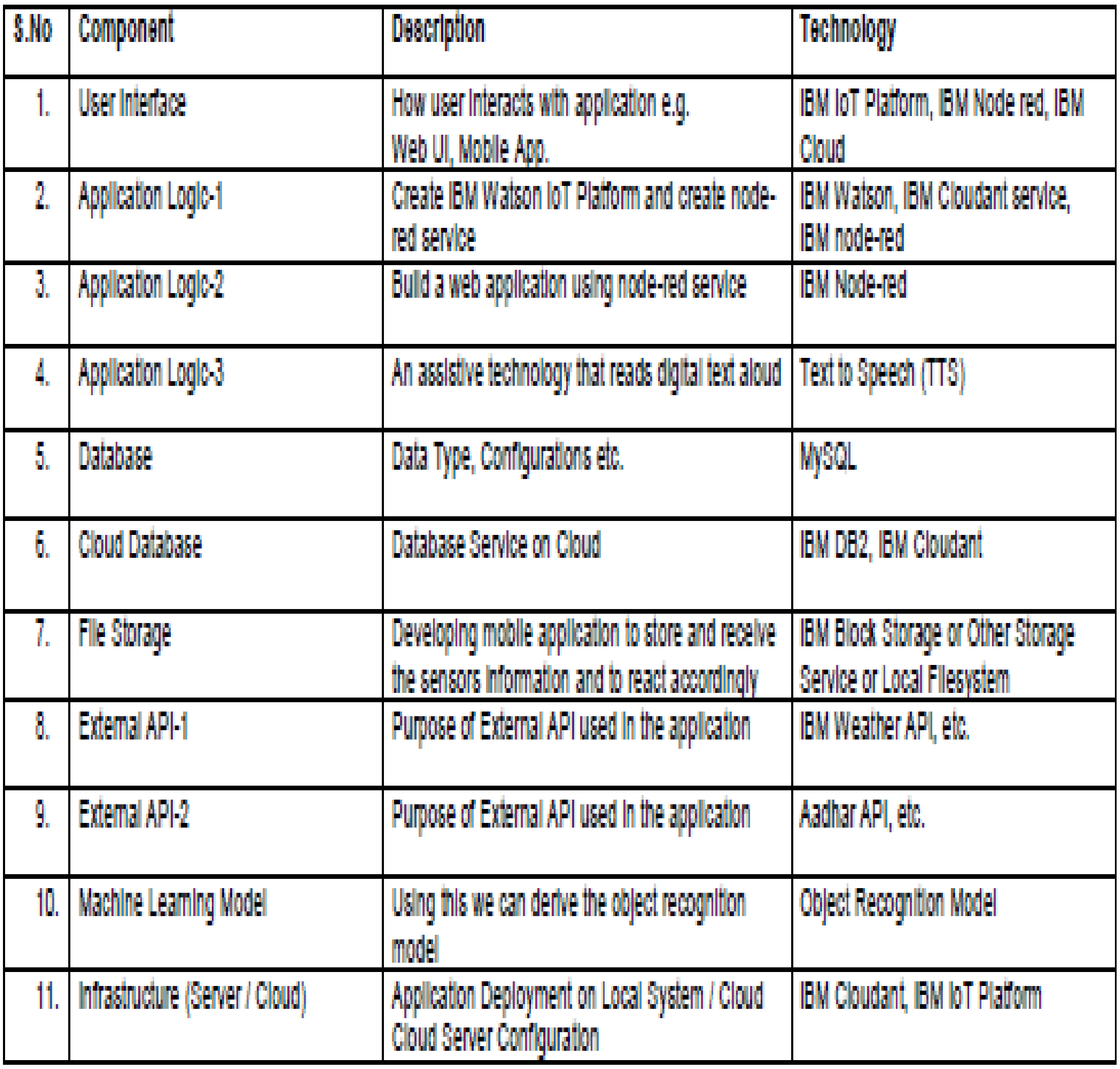
  Step 3: Create Node-RED device.

  Step 4: Create text to speech service.

  Step 5: Create Database in Cloudant DB to store the medicine details.

- Requirements: IBM Watson IOT Platform, Node-RED Service, Cloudant DB, TTS Service are required.

TECHNICAL SOLUTIONS:



S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App.	IBM IoT Platform, IBM Node red, IBM Cloud
2.	Application Logic-1	Create IBM Watson IoT Platform and create node-red service	IBM Watson, IBM Cloudant service, IBM node-red
3.	Application Logic-2	Build a web application using node-red service	IBM Node-red
4.	Application Logic-3	An assistive technology that reads digital text aloud	Text to Speech (TTS)
5.	Database	Data Type, Configurations etc.	MySQL
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant
7.	File Storage	Developing mobile application to store and receive the sensors information and to react accordingly	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
10.	Machine Learning Model	Using this we can derive the object recognition model	Object Recognition Model
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Cloud Server Configuration	IBM Cloudant, IBM IoT Platform

USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile User,Web User,Care Executive, Administrator)	Registration	USN-1	As a user, I can register for the application byentering my email, password, and confirmingmy password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation emailonce I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
	Dashboard	USN-3	As a user, I can register for the applicationthrough 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	I can confirm the registration in Gmail	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application byentering email & password	I can login with my id and password	High	Sprint-1

PROJECT PLANNING & SCHEDULING

SPRINT PLANNING AND ESTIMATION

- Identify the Problem
- Prepare an Abstract, Problem Statements
- List a Requirements Needed
- Create a Code and Run the code
- Make a proper prototype
- Test with the created code and check the deigned prototype
- Solution for the problem is Found

<u>SPRINT DELIVERY SCHEDULING</u>

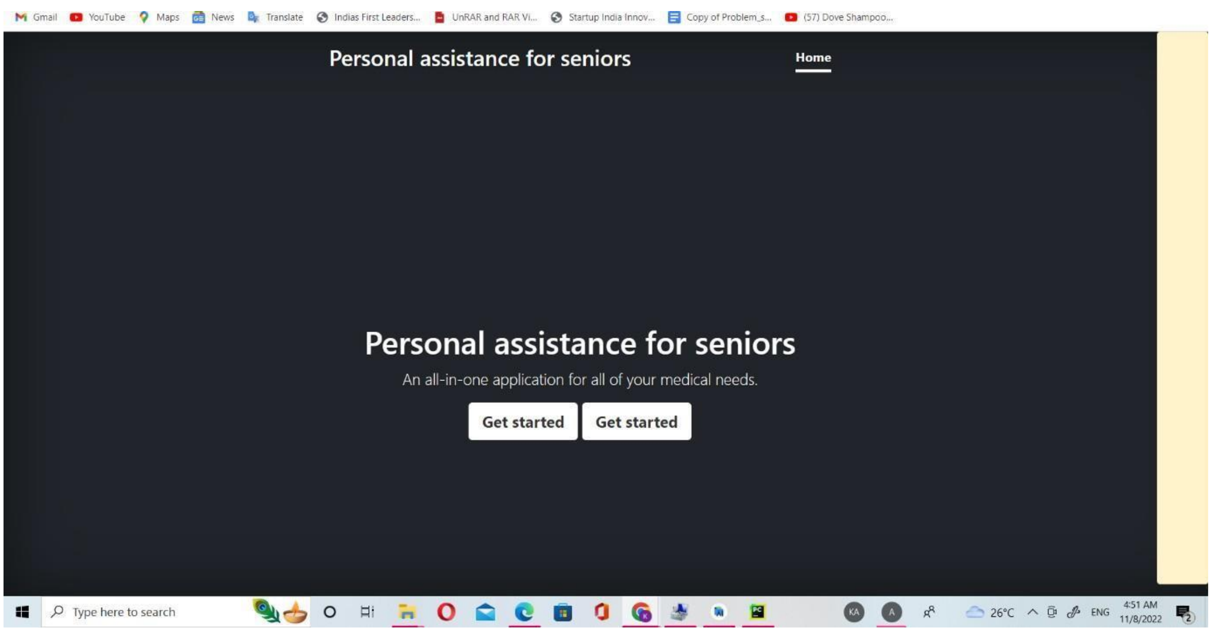
CONTENTS:

1. INDEX PAGE

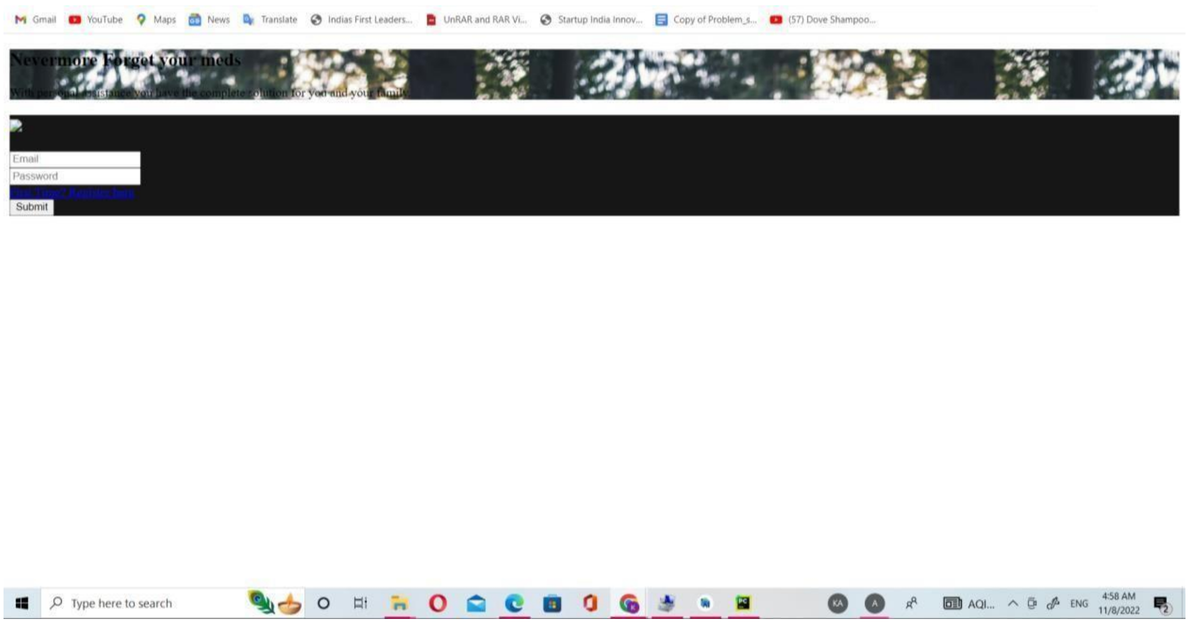
2. LOG IN

3. SIGN UP

1.INDEX PAGE



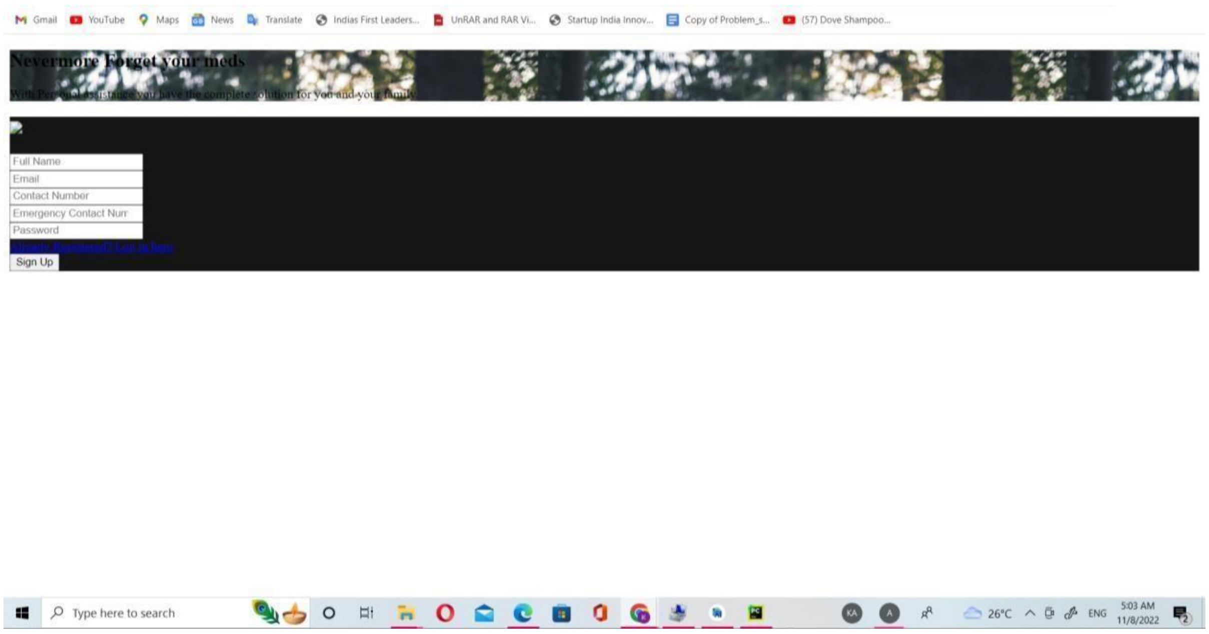
2.LOGIN PAGE



Nevermore Forget your meds

With personal assistance you have the complete solution for you and your family

3.SIGNUP PAGE



Nevermore Forget your meds

With Personal assistance you have the complete solution for you and your family

CODING & SOLUTIONING

FEATURE 1

In this code we have designed that the message or information can be passed through email even the user don't have any own source for example laptop ,mobile phone etc. so they can receive messages where ever they are.

- ✓ IoT Device
- ✓ IBM Watson platform
- ✓ Node Red
- ✓ Cloudant DB
- ✓ Web UI
- ✓ MIT App
- ✓ Python Code

Programing:

```
import smtplib
from email.mime.multipart import MIMEMultipart
 EMAIL_ADDRESS = 'varshignans448@gmail.com'
PASSWORD = ''
 def sendmail(receiver, html):
   server = smtplib.SMTP('smtp.gmail.com:587')
   server.ehlo()
   server.starttls()
   msg = MIMEMultipart('alternative')
   msg['Subject'] = "Reminder – Personalassistance"
   msg['From'] = 'varshignans448@gmail.com'
   msg['To'] = receiver
   msg.attach(html)
   server.login(EMAIL_ADDRESS, PASSWORD)
   server.sendmail(EMAIL_ADDRESS, receiver, msg.as_string())
   server.quit()
```

FEATURE 2

- ✓ Registration
- ✓ Login
- ✓ Verification
- ✓ Medicine name
- ✓ Schedule
- ✓ Reschedule
- ✓ Notification

Programming:

```
from pymongo import MongoClient
client =
MongoClient('mongodb+srv://pancham:pancham@niggaballs.tjmtx.mongodb.net/
myFirstDatabase?retryWrites=true&w=majority')
db = client['medicine_schedule']
users = db['users']
scheduledb = db['schedule']
def get_all_medicines(user):
    document = scheduledb.find_one({"_id": user.lower()})
    medicines = document['medicines']
    list = []
    for medicine in medicines:
        list.append(medicine.title())
    return list
def medicine_card(medicine, price, href):
    card = f"""<div class="flex flex-col card rounded-lg my-5 p-3 shadow-md">
    <p class="text-gray-800 my-3">{medicine.title()}</p>
    <div class="flex">
    <a href='{href}' target='_blank'>
      <button class="bg-primary-blue-light text-white p-1 rounded-lg flex">
        <i class="fas fa-external-link-alt mt-1.5 mx-1"></i>
        <p class="mt-1 font-medium">₹{price} | Buy now</p>
      </button></a>
    </div>
  </div>"""
Return
```

TESTING

Test Cases:

a. Login testing

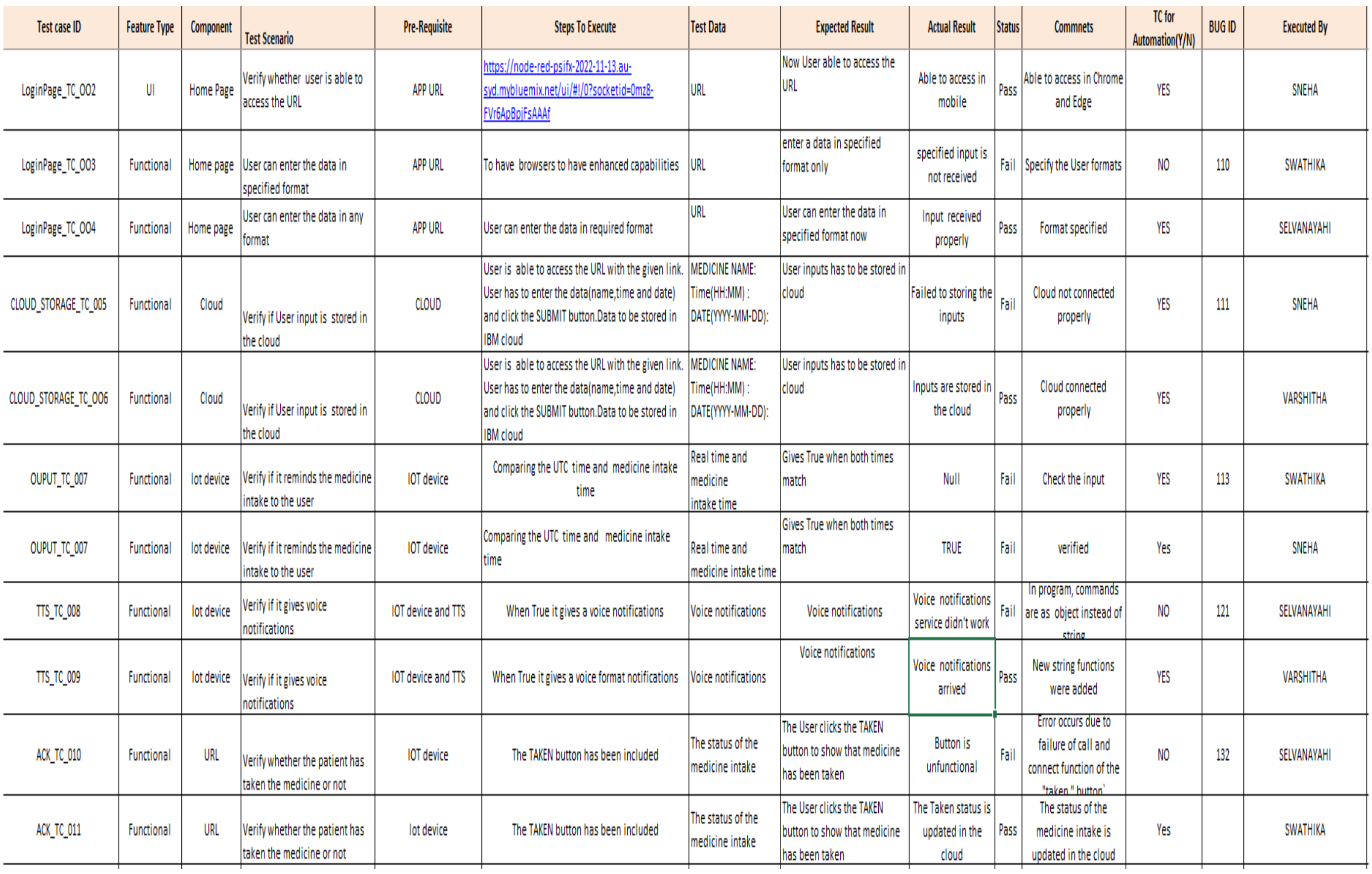
We are creating login page for the users and deployed into the testing environment. This login page is tested using credentials.

b. Storage testing

We are testing whether the information from the user is stored as database using cloud login credentials and verifying it for the user.

c. Device testing

Here the IOT device is created using the cloud login credentials, the sample test case is deployed in the device and code is uploaded and tested in the testing environment.

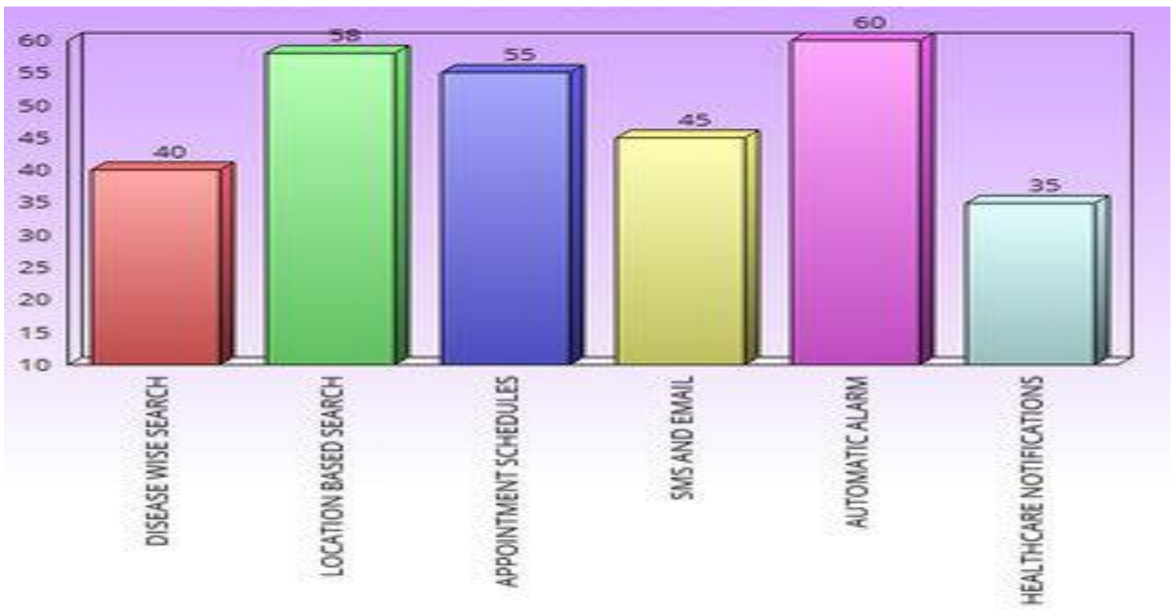


Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_002	UI	Home Page	Verify whether user is able to access the URL	APP URL	https://node-red-psifx-2022-11-13.au-syd.mybluemix.net/ui/#!/0?socketid=0mz8-FVr6ApBpjFsAAAf	URL	Now User able to access the URL	Able to access in mobile	Pass	Able to access in Chrome and Edge	YES		SNEHA
LoginPage_TC_003	Functional	Home page	User can enter the data in specified format	APP URL	To have browsers to have enhanced capabilities	URL	enter a data in specified format only	specified input is not received	Fail	Specify the User formats	NO	110	SWATHIKA
LoginPage_TC_004	Functional	Home page	User can enter the data in any format	APP URL	User can enter the data in required format	URL	User can enter the data in specified format now	Input received properly	Pass	Format specified	YES		SELVANAYAHI
CLOUD_STORAGE_TC_005	Functional	Cloud	Verify if User input is stored in the cloud	CLOUD	User is able to access the URL with the given link. User has to enter the data(name,time and date) and click the SUBMIT button.Data to be stored in IBM cloud	MEDICINE NAME: Time(HH:MM) : DATE(YYYY-MM-DD):	User inputs has to be stored in cloud	Failed to storing the inputs	Fail	Cloud not connected properly	YES	111	SNEHA
CLOUD_STORAGE_TC_006	Functional	Cloud	Verify if User input is stored in the cloud	CLOUD	User is able to access the URL with the given link. User has to enter the data(name,time and date) and click the SUBMIT button.Data to be stored in IBM cloud	MEDICINE NAME: Time(HH:MM) : DATE(YYYY-MM-DD):	User inputs has to be stored in cloud	Inputs are stored in the cloud	Pass	Cloud connected properly	YES		VARSHITHA
OUPUT_TC_007	Functional	Iot device	Verify if it reminds the medicine intake to the user	IOT device	Comparing the UTC time and medicine intake time	Real time and medicine intake time	Gives True when both times match	Null	Fail	Check the input	YES	113	SWATHIKA
OUPUT_TC_007	Functional	Iot device	Verify if it reminds the medicine intake to the user	IOT device	Comparing the UTC time and medicine intake time	Real time and medicine intake time	Gives True when both times match	TRUE	Fail	verified	Yes		SNEHA
TTS_TC_008	Functional	Iot device	Verify if it gives voice notifications	IOT device and TTS	When True it gives a voice notifications	Voice notifications	Voice notifications	Voice notifications service didn't work	Fail	In program, commands are as object instead of string	NO	121	SELVANAYAHI
TTS_TC_009	Functional	Iot device	Verify if it gives voice notifications	IOT device and TTS	When True it gives a voice format notifications	Voice notifications	Voice notifications	Voice notifications arrived	Pass	New string functions were added	YES		VARSHITHA
ACK_TC_010	Functional	URL	Verify whether the patient has taken the medicine or not	IOT device	The TAKEN button has been included	The status of the medicine intake	The User clicks the TAKEN button to show that medicine has been taken	Button is unfunctional	Fail	Error occurs due to failure of call and connect function of the "taken " button`	NO	132	SELVANAYAHI
ACK_TC_011	Functional	URL	Verify whether the patient has taken the medicine or not	Iot device	The TAKEN button has been included	The status of the medicine intake	The User clicks the TAKEN button to show that medicine has been taken	The Taken status is updated in the cloud	Pass	The status of the medicine intake is updated in the cloud	Yes		SWATHIKA

RESULT

PERFORMANCE METRICES

In this testing, the application is tested with user for compatibility in the real world.



ADVANTAGES AND DISADVANTAGES

Advantages:

- It is easy for seniors to take the wrong meds or even skip doses. Medication reminders prevent this from happening. There is nothing your senior has to read or figure out. They simply need to take the pills in the compartment after the reminder beeps.
- Increase patient satisfaction, keep treatment plans on track and aid your office staff in doing the work that they're best suited to do.
- These can range from a watch that the patient or caregiver wears with alarms that sound at various times throughout the day to computer software that can be programmed to let people know when it is time to take the medication.

Disadvantages:

- Reminders cannot be set automatically. There is a need for manual work in setting the reminder.
- A lot of time is consumed in manually setting the reminders.
- The possibility exists for the existing systems to hang down due to the manual work involved.

CONCLUSION

For home health care various technology have evolved as review considered, in this paper medicine, its scheduling have well focused which is beneficial to improve efficiency of prescribed drug and reduce economic factor. To improve the existing home health care technique number of monitoring technology has observed which leads to home health monitoring system. The monitoring system can be implemented with sensing element and wireless module which should need to secure so that message containing the health related information should not be corrupt. IOT (Internet of Things) play a vital role in communicating the two devices, the use of messaging standard and communication protocol we can securely transfer the important messages regarding to health. open source IOT cloud will be effective for storing sensors data,the benefit of digitally storing is the retrieving of data is easy and faster manner in case of emergency for secure health.

FUTURE SCOPES

In future, efforts can be made to improve the accuracy of the character recognition. This reminder reminds user about their medicine in-take schedule. The system which we are implementing will also give the reminder about doctor's next appointment. It will also tell the user of the end of the medicines. The scheduled reminder will not suggest any kind of medicine, dose of medicine, etc. Also the facility of adding names & dose of the medicine will be included in the reminder.

APPENDIX

SOURCE PROGRAM

```
from pymongo import MongoClient
import bcrypt
from flask import session
import smtplib
from email.mime.multipart import MIMEMultipart
from pymongo import MongoClient
from pymongo import MongoClient
import json
import pytz
import owncloud
import os
from flask import flash



client = MongoClient(

'mongodb+srv://pancham:pancham@niggaballs.tjmtx.mongodb.net/myFirstDatabase?retr
yWrites=true&w=majority')
db = client['medicine_schedule']
user_collection = db['users']
schedules = db['schedule']
prescriptions = db['prescriptions']
```

```
def check_existing_user(email):
    if user_collection.find_one({'_id': email.lower()}):
        return True
    else:
        return False




def add_new_user(name, email, password, contact, emergency_contact):
    user_collection.insert_one({
        '_id': email.lower(),
        'name': name,
        'password': password,
        'contact':contact,
        'emergency_contact':emergency_contact
    })
    schedules.insert_one({
        '_id':email.lower(),
        'medicines':{}
    })
    prescriptions.insert_one({
        '_id':email.lower(),
        'prescription':{}
    })



def check_user_credentials(email, password):
    user = user_collection.find_one({'_id': email.lower()})
    if user:
        return bcrypt.checkpw(password.encode(), user['password'])



def login_check():
    if session['login'] is True:
        return True
    else:
        return False
EMAIL_ADDRESS = 'keerthanaanandh4@gmail.com'
PASSWORD = ''



def sendmail(receiver, html):
    server = smtplib.SMTP('smtp.gmail.com:587')
    server.ehlo()
    server.starttls()
    msg = MIMEMultipart('alternative')
    msg['Subject'] = "Reminder - Personalassistance"
    msg['From'] = 'keerthanaanandh4@gmail.com'
    msg['To'] = receiver
    msg.attach(html)
    server.login(EMAIL_ADDRESS, PASSWORD)
    server.sendmail(EMAIL_ADDRESS, receiver, msg.as_string())
    server.quit()

client = MongoClient(

'mongodb+srv://pancham:pancham@niggaballs.tjmtx.mongodb.net/myFirstDatabase?retr
yWrites=true&w=majority')
db = client['medicine_schedule']
users = db['users']
scheduledb = db['schedule']



def user_name(user):
    document = users.find_one({"_id":user})
    return document['name']
```

```
def today_card(medicine, time):
   card = f"""<div class="card">
             <div class="rounded-lg mx-8 my-3 bg-primary-blue-accent px-4 py-2"
onclick="show_slideover()">
              <p class="my-3 text-primary-blue-dark text-lg">{medicine.title()}</p>
              <!-- Use jinja to enter medicine name here -->
              <p class="my-3 text-primary-blue-dark text-lg">At {time}</p>
             </div>
             <!-- If it is the last card of the day then do not render the border -->
             <div class="mt-3 border-t mx-8 border-gray-200 text-right"></div>
            </div>"""
   return card


def tomorrow_card(medicine, time):
   card = f"""<div class="card">
             <div class="rounded-lg mx-8 my-3 bg-primary-green-light px-4 py-2">
              <p class="my-3 text-primary-blue-dark text-lg">{medicine.title()}</p>
              <!-- Use jinja to enter medicine name here -->
              <p class="my-3 text-primary-blue-dark text-lg">At {time}</p>
             </div>
             <div class="mt-3 border-t mx-8 border-gray-200  text-right"></div>
            </div>"""
   return card


def day_after_card(medicine, time):
   card = f"""<div class="card">
             <div class="rounded-lg mx-8 my-3 bg-primary-yellow-accent px-4 py-2">
              <p class="my-3 text-primary-blue-dark text-lg">{medicine.title()}</p>
              <!-- Use jinja to enter medicine name here -->
```

```
                <p class="my-3 text-primary-blue-dark text-lg">At {time}</p>
              </div>
              <div class="mt-3 border-t mx-8 border-gray-200  text-right"></div>
            </div>"""
    return card




IST = pytz.timezone('Asia/Chennai')


client = MongoClient(

'mongodb+srv://pancham:pancham@niggaballs.tjmtx.mongodb.net/myFirstDatabase?retr
yWrites=true&w=majority')
db = client['medicine_schedule']
scheduledb = db['schedule']


def add_medicine(email, object: dict):
    document = scheduledb.find_one({'_id': email.lower()})
    medicine = object['medicine_name']
    del object['medicine_name']
    medicines = document['medicines']
    medicines[medicine] = object
    update = {'$set': {'medicines': medicines}}
    scheduledb.update_one({'_id': email.lower()}, update)


def edit_medicine(email, old_medicine_name, new_medicine_name):
    document = scheduledb.find_one({'_id': email.lower()})
    new = str(document).replace(old_medicine_name, new_medicine_name)
    new = new.replace("'", "")
    new = json.loads(new)
    scheduledb.replace_one(document, new)


def fetch_user_schedule(day, user=None):
    document = scheduledb.find_one({'_id': user.lower()})
    medicines = document['medicines']
    med_dates = []
    for medicine in medicines:
        med_dates.append([medicine, medicines[medicine]['start_date'],
medicines[medicine]['end_date']])

    today_meds = {}
    for med_date in med_dates:
        if day < med_date[2]:
            today_meds[med_date[0]] = medicines[med_date[0]]['dose_time'][:-3]
        else:
            pass
    return today_meds


def no_data_check(user):
    document = scheduledb.find_one({"_id":user.lower()})
    medicines = document['medicines']
    if not medicines:
        return True
    else:
        return False


def card(medicine:str, time):
    card_html = f"""<div class="flex flex-col card rounded-lg my-5 p-3 shadow-md">
```

```
                <p class="text-gray-800 my-3">{medicine.title()}</p>
                <div class="flex">
                  <div class="bg-primary-blue-light text-white p-1 rounded-lg flex">
                    <i class="fas mt-1.5 mx-1 fa-clock"></i>
                    <p class="mt-1 font-medium">{time}</p>
                  </div>
                </div>
              </div>"""
    return card_html


client = MongoClient(

'mongodb+srv://pancham:pancham@niggaballs.tjmtx.mongodb.net/myFirstDatabase?retr
yWrites=true&w=majority')
db = client['medicine_schedule']
users = db['users']
scheduledb = db['schedule']


def get_all_medicines(user):
    document = scheduledb.find_one({"_id": user.lower()})
    medicines = document['medicines']
    list = []
    for medicine in medicines:
        list.append(medicine.title())
    return list


def medicine_card(medicine, price, href):
    card = f"""<div class="flex flex-col card rounded-lg my-5 p-3 shadow-md">
                <p class="text-gray-800 my-3">{medicine.title()}</p>
                <div class="flex">
                <a href='{href}' target='_blank'>
                  <button class="bg-primary-blue-light text-white p-1 rounded-lg flex">
                    <i class="fas fa-external-link-alt mt-1.5 mx-1"></i>
                    <p class="mt-1 font-medium">₹{price} | Buy now</p>
                  </button></a>
                </div>
              </div>"""
    return card




client = MongoClient(

'mongodb+srv://pancham:pancham@niggaballs.tjmtx.mongodb.net/myFirstDatabase?retr
yWrites=true&w=majority')
db = client['medicine_schedule']
prescriptiondb = db['prescriptions']

def add_prescription(path=None, user=None, filename=None):
    try:
        oc.mkdir(f'medSCHED/{user}')
    except:
        pass
    try:
        oc.put_file(f'medSCHED/{user}/{filename}', path)
    except FileExistsError:
        pass
    os.remove(path)
    link = oc.share_file_with_link(f'medSCHED/{user}/{filename}').get_link()
    return link
```

```
def add_prescription_record(user, file_name, link):
    document = prescriptiondb.find_one({'_id':user.lower()})
    prescriptions = document['prescription']
    prescriptions[file_name] = link

    update = {'$set':{'prescription':prescriptions}}
    prescriptiondb.update_one({'_id':user.lower()}, update)
    return flash('Prescription uploaded')


def fetch_prescriptions(user):
    document = prescriptiondb.find_one({'_id':user.lower()})
    prescriptions = document['prescription']
    data = []
    count = 0
    for prescription in prescriptions:
        count +=1
        data.append({
            'name':prescription,
            'link':prescriptions[prescription]
        })
    return data, count


def prescription_card(prescription, href):
    card = f"""<div class="flex flex-col card rounded-lg my-5 p-3 shadow-md">
                <p class="text-gray-800 my-3">{prescription.title()}</p>
                <div class="flex">
                <a href='{href}' target='_blank'>
                  <button class="bg-primary-blue-light text-white p-1 rounded-lg flex">
                    <i class="fas fa-external-link-alt mt-1.5 mx-1"></i>
                    <p class="mt-1 font-medium">View</p>
                  </button></a>
                </div>
              </div>"""
    return card



client = MongoClient(

'mongodb+srv://pancham:pancham@niggaballs.tjmtx.mongodb.net/myFirstDatabase?retr
yWrites=true&w=majority')
db = client['medicine_schedule']
users = db['users']
scheduledb = db['schedule']



def get_all_medicines(user):
    document = scheduledb.find_one({"_id": user.lower()})
    medicines = document['medicines']
    list = []
    for medicine in medicines:
        list.append(medicine.title())
    return list



def medicine_card(medicine, price, href):
    card = f"""<div class="flex flex-col card rounded-lg my-5 p-3 shadow-md">
                <p class="text-gray-800 my-3">{medicine.title()}</p>
                <div class="flex">
                <a href='{href}' target='_blank'>
                  <button class="bg-primary-blue-light text-white p-1 rounded-lg flex">
                    <i class="fas fa-external-link-alt mt-1.5 mx-1"></i>
                    <p class="mt-1 font-medium">₹{price} | Buy now</p>
                  </button></a>
                </div>
```

```
            </div>"""
    return
```

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-15217-1659595094