PERSONAL ASSISTANCE FOR SENIOR

WHO ARE SELF RELIANT

IBM NALAIYA THIRAN

PROJECT REPORT

Submitted By:

RANJITH KUMAR R

ABISHEK G

DHINAKARAN S

AABID KALEEM A

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

MUTHAYAMMAL ENGINEERING COLLEGE,

RASIPURAM-637408

NOVEMBER-2022

BONAFIDE CERTIFICATE

Certified that this project report titled "PERSONAL ASSISTANCE FOR SENIORS WHO ARE SELF RELIANT" is the bonafide work of "RANJITH KUMAR R, ABISHEK G, DHINAKARAN S, AABID KALEEM A" who carried out the project work under my supervision.

SIGNATURE **SIGNATURE**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

At the outset, we express our heartfelt gratitude to GOD, who has been our strength to bring this project to light. At this pleasing moment of having completed our project, we would like to express our gratitude and appreciation to all those who gave us the possibility to complete this report. we also sincerely thank for the time spent proof reading and correcting our mistakes. I would like to express my special thanks of gratitude to our faculty mentor, ibm mentor&Evaluator for their able guidance and support in completing our project. we came to know about many things and we learnt more . They have also helped us to complete our project on time. Secondly we thank our friends who helped us directly and indirectly in all aspects of the project work to get completed successfully.This project is not only developed for marks but also to gain knowledge.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
1	**INTRODUCTION**	**1**
	1.1 PROJECT OVERVIEW	1
	1.2 PURPOSE	1
2	**LITERATURE SURVEY**	**2**
	2.1 EXISTING PROBLEM	2
	2.2 REFERENCES	3
	2.3 PROBLEM STATEMENT DEFINITION	3
3	**IDEATION AND PROPOSED SOLUTION**	**4**
	3.1 EMPATHY MAP CANVAS	4
	3.2 IDEATION AND BRAINSTORMING	5
	3.3 PROPOSED SOLUTION	8
	3.4 PROBLEM-SOLUTION FIT	9
4	**REQUIREMENT ANALYSIS**	**12**
	4.1 FUNCTIONAL REQUIREMENT	12
	4.2 NON- FUNCTIONAL REQUIREMENT	13
5	**PROJECT DESIGN**	**14**
	5.1 DATA FLOW DIAGRAM	14
	5.2 SOLUTION AND TECHNOLOGY ARCHITECTURE	**15**

5.3 USER STORIES 17

6 PROJECT PLANNING AND SCHEDULING 18

6.1 SPRINT PLANNING AND ESTIMATION 18

6.2 SPRINT DELIVERY SCHEDULE 19

6.3 REPORT FROM JIRA 19

7 CODING AND SOLUTIONS 21

7.1 FEATURE 1 21

7.2 FEATURE 2 24

7.3 DATABASE SCHEMA 30

8 TESTING 32

8.1 TEST CASES 32

8.2 USER ACCEPTANCE TESTING 33

9 RESULT 34

9.1 PERFORMANCE METRICS 34

10 ADVANTAGES AND DISADVANTAGES 35

11 CONCLUSION 36

12 FUTURE SCOPE 37

13 APPENDIX 37

13.1 SOURCE CODE 37

13.2 GITHUB & PROJECT DEMO LINK 61

CHAPTER- 1

INTRODUCTION

1.1 PROJECT OVERVIEW

In day-to-day life, most people need to take medicines which were not there in the past couple of years and the reason behind this is diseases are increasing in a large amount. So sooner or later many people encounter these diseases. Some diseases are temporary while many are permanent life-threatening diseases. Life-threatening diseases get mixed with the human body in such a way that they can't leave the body ever and they increase in rapid time. The life span of humans became less because of such diseases and to overcome or to live a better life we need to take medicines regularly and also in the large amount. We need to be on the advice of a doctor who tells us to take desired pills in the desired way so that patients face problems like forgetting pills to take at right time and when the Doctor changes the prescription of medicine patients have to remember the new schedule of medicine. This problem of forgetting to take pills at right time, taking the wrong medicines and accidentally taking expired medicine causes health issues for the patient and this leads to suffering from unhealthy life. Our project is to make a software-based helping system, which connects the caretaker of the patient with the patient, to send timely SMS alerts to them at the specified time and with the specified note set by the caretaker. The patient can be duly monitored by the caretaker and hence his/her health can be monitored better with this software.

1.2 PURPOSE

The purpose of this project is to keep people fit and safe from health threatening diseases. The sole purpose of medicines is to treat the patients and control their metabolisms properly so that the health risk can be reduced and thus the patient can get a cure for the illness and can live a longer life.

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.This is an Android-based application in which an automatic alarm ringing system is implemented. It focuses on doctor and patient interaction. Patients need not remember their medicine dosage timings as they can set an alarm on their dosage timings.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

The remarkable problem is that patients forget to take the proper medicines in proper proportion and in proper time. Medication adherence, which refers to the degree or extent to which a patient takes the right medication at the right time according to a doctor's prescription, has recently emerged as a serious issue because many studies have reported that non-adherence may critically affect the patient, thereby raising medical costs. Medication nonadherence is a common,complex, and costly problem that contributes to poor treatment outcomes and consumes health care resources.

Author and Year	Technique/ Methodology	Limitations/ Drawback	Advantages	Applications
Huai-Kuei Wu1, 2015	Matrix Bar Code	Costly	Alerts if wrong medicine taken	Home
Hiba Zeidan, 2018	Duplicating the Electrical Components	Works for only one type of medicine	checks medicine availability	Home, Hospital
Benbin Chen, 2019	Docker	Need stable internet	Data Storage	Home, Hospital
Obaidulla-Al-Mahud, 2020	IoT	Costly	Easy Communication	Hospital
Aakash Mharadwaj, 2017	IoT	Need stable internet	Connects Pharmacies	Home, Hospital, Pharmacies
R Al-Shammary, 2018	Software	Cannot collect kit data	Easy to use	Home

Table 2.1.Existing problem

2.2 REFERENCES

1) H Luoma-Halkola, L Häikiö - Published on Ageing & Society, 2022 – Cambridge.

2) Hiba Zeidan, Khalil Karam, Roy Abi Zed Daou, Ali Hayek, Josef Bolercsoek (2018)'Smart Medicine Box System', IEEE International Multidisciplinary Conference on Engineering Technology, DOI:10.1109/IMCET.2018.8603031

3) Obaidulla-Al-Mahmud1, Md.Kausar Khan, Rajdeep Roy, and Fakir and Mashuque Alamgir (2020) 'IoT based Smart Health Care Medical Box for Elderly People', International Conference for Emerging Technology, DOI:10.1109/INCET49848.2020.9153994.

4) R Al-Shammary, D.Mousa, S.E.Esmaeili (2018) 'The Design of a Smart Medicine Box', 26th Iranian Conference on Electrical Engineering, DOI:10.1109/ICEE.2018.8472586.

5) Aakash Bharadwaj, Divyank Yarravarapu, Sadiparala Charan Kumar Reddy, Thirumalaraju Prudhvi, KSP Sandeep and Obulam Siva Dheeraj Reddy (2017) 'Enhancing Healthcare using m-Care Box(Monitoring non Compliance of Medication)', International Conference for Innovative Mechanisms for Industry Applications, DOI:10.1109/ICIMIA.2017.7975594

2.3 PROBLEM STATEMENT DEFINITION

The Customer Problem Statement template helps you focus on what matters to create experiences people will love. 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.

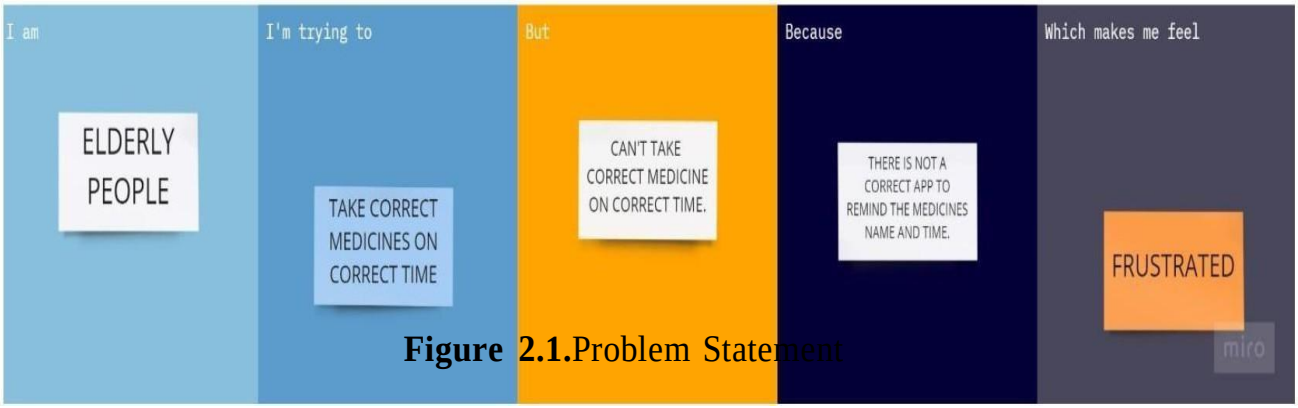


Figure 2.1. Problem Statement

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1.EMPATHY MAP CANVAS

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community. An Empathy Map consists of four quadrants. The four quadrants reflect four key traits, which the user demonstrated/possessed during the observation/research stage. The four quadrants refer to what the user: Said, Did, Thought, and Felt.

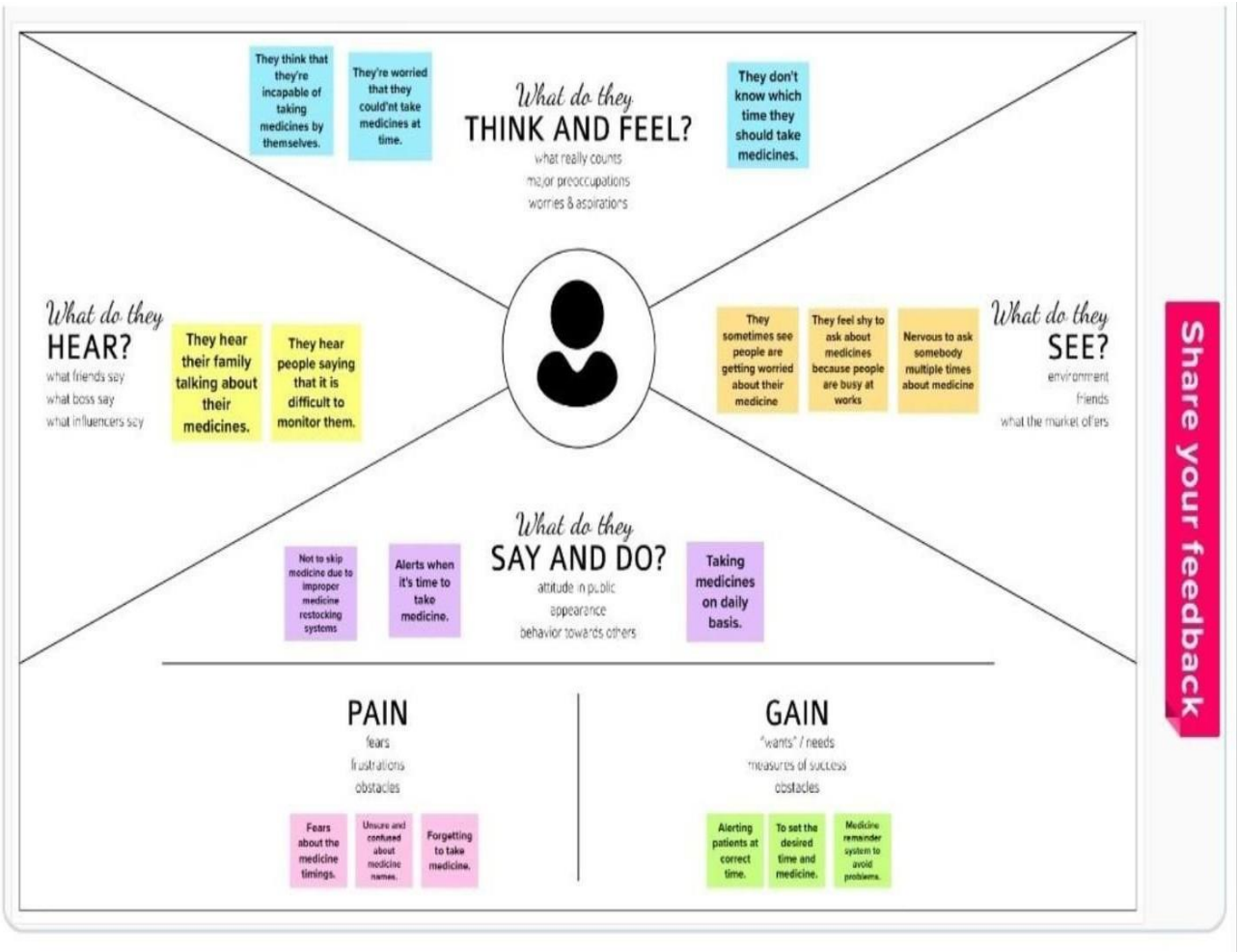


Figure.3.1 Empathy map

3.2 IDEATION & BRAIN STROMING :

Ideation is the process where you generate ideas and solutions through sessions such as Sketching, Prototyping, Brainstorming, Brainwriting, Worst Possible Idea, and a wealth of other ideation techniques. Ideation is also the third stage in the Design Thinking process.

Brainstorming is a method of generating ideas and sharing knowledge to solve a particular commercial or technical problem, in which participants are encouraged to think without interruption. Brainstorming is a group activity where each participant shares their ideas as soon as they come to mind.

STEP-1 TEAM GATHERING ,COLLABORATION AND SELECTING THE PROBLEM STATEMENT

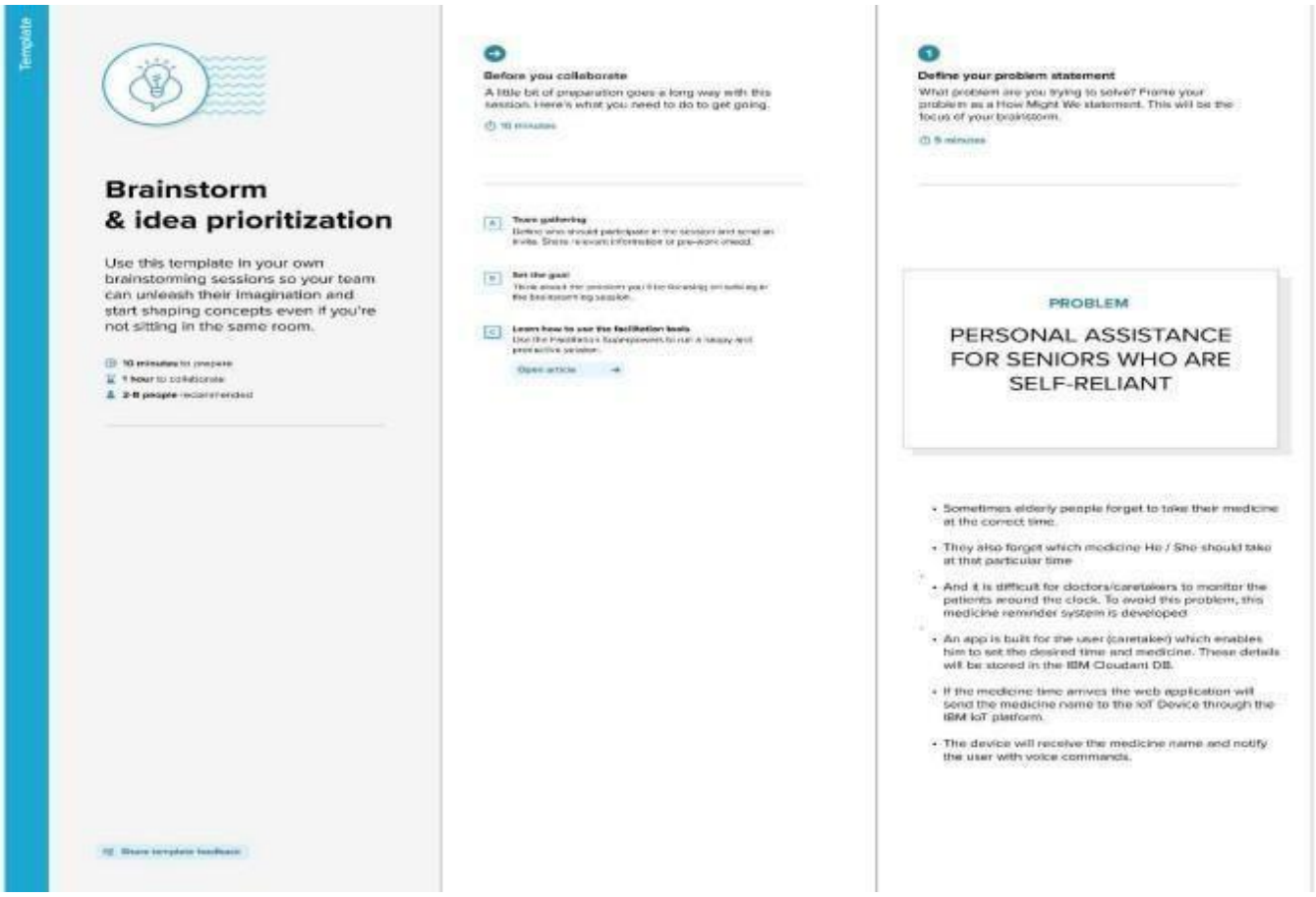


Figure 3.2. Ideation And Brainstorming

STEP 2:BRAINSTORM , IDEA LISTING AND GROUPING

This step of ideation includes the listing of individual ideas by teammates to help with the problem statement framed. All the individual ideas have been valued and made individual clusters.

Then discussed as a team and finally made an ideation Cluster A and concluded with the most voted ideas from all the clusters together and Cluster B with the least needed ideas.

STEP 3.IDEA PRIORITIZATION

This step includes the process of listing necessary components to come up with the working solution and making a hierarchy chart by prioritizing the components based on importance, say from the higher being backend and lower being the user interfacing components.

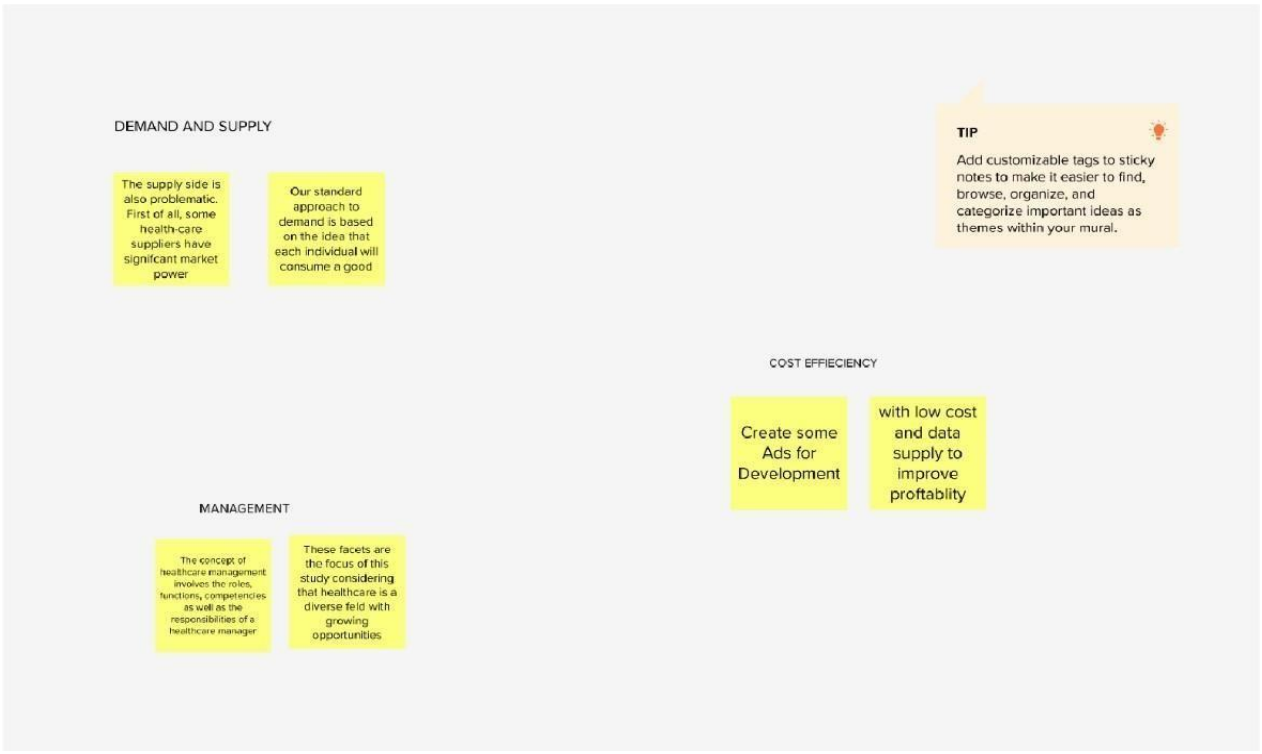


Figure 3.3.Idea Grouping

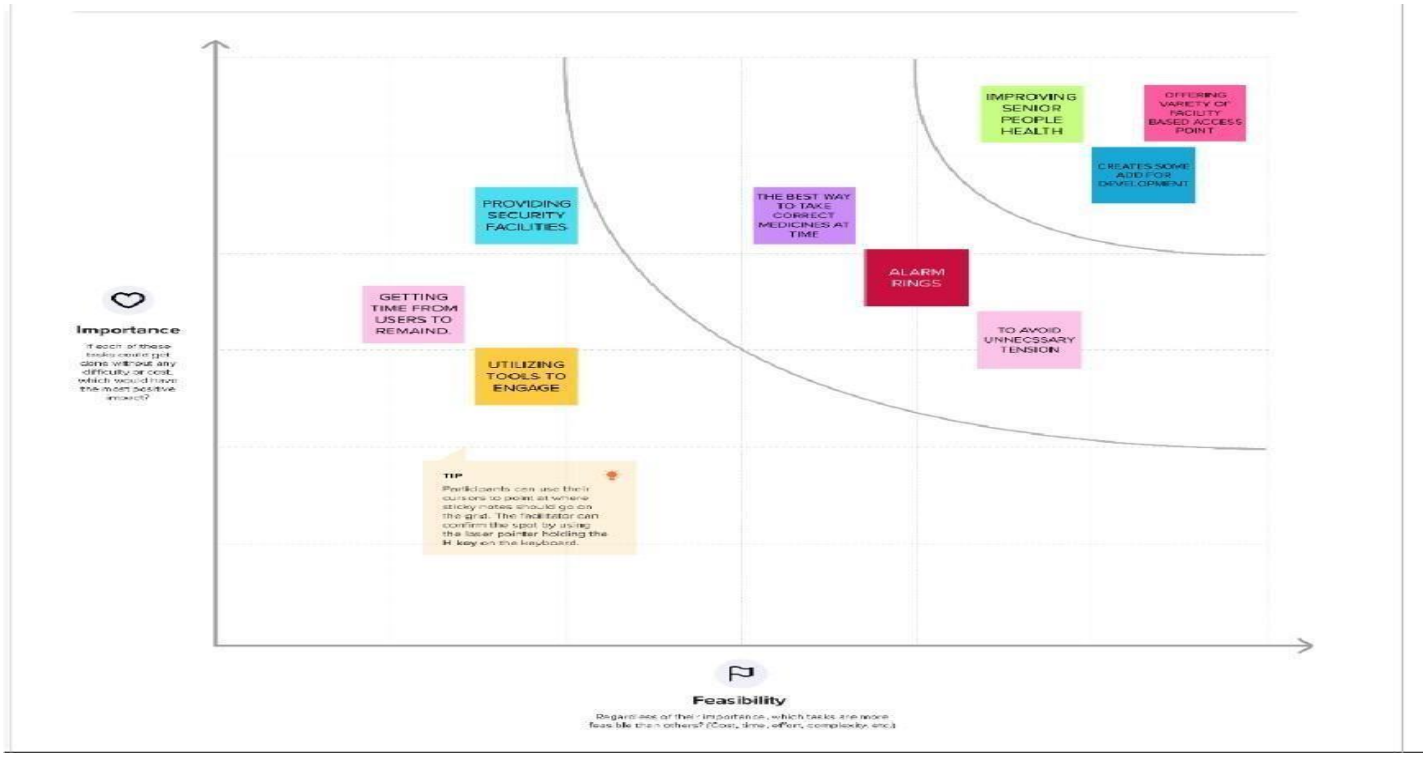


Figure 3.4. Idea Prioritization.

3.3 PROPOSED SOLUTION

Your proposed solution should relate the current situation to a desired result and describe the benefits that will accrue when the desired result is achieved. So, begin your proposed solution by briefly describing this desired result.

Sl.No	Parameter	Descripion
1.	Problem Statement (Problem to be solved)	Sometimes elderly people forgot to take their medicine at the correct time. They also forgot which medicine should they take at that particular time. And it is difficult for doctors/caretakers to monitor the patients around the clock.
2.	Idea / Solution description	To avoid this problem, this medicine reminder system is developed. An app built for the user which enables to set desired time and medicine. If the medicine arrives the web application will send the medicine name to the IOT Device through the IBM IOT platform.
3.	Novelty / Uniqueness	The device will receive the medicine name and notify the user with voice commands.
4.	Social Impact / Customer Satisfaction	It is life saving , unobtrustive and effective. It allows an ageing population to keep their independence , and

		gives friends and family peace of mind.
5.	Business Model (Revenue Model)	Revenue can be generated from advertisement .For additional features subscriptions can be provided which is payable.
6.	Scalability of the Solution	It provides alarm functionality. Data privacy and security. Monitored and handled by closed ones.

Table 3.1 Proposed Solution Fit

3.4 PROBLEM SOLUTION FIT

Problem-solution fit is a term used to describe the point validating that the base problem resulting in a business idea really exists and the proposed solution actually solves that problem. It helps entrepreneurs, marketers and corporate innovators identify behavioural patterns and recognize what would work and why.

1. CUSTOMER SEGMENT:Older adults who have difficulty with such daily activities such as bathing, grooming, cooking, eating or just getting to the bathroom often end up in hospitals or nursing homes, spending a disproportionately huge number of healthcare dollars. To lower their burdens and to change their stressful environment, this application has to be installed in each and every home in which self reliant people live. Older adults Older aged who have difficult with such daily activities as bathing ,grooming, cooking, eating , or just getting to the bathroom often end up in hospitals or nursing homes, spend a lot of money in order to just take of them Older aged people who have difficulty with such daily activity such as bathing, cooking, grooming, eating or just getting to the bathroom often end up in hospital or nursing homes, spending a disproportionately large amount of money just to take care of their aged beloved people in taking their daily medicines on time. This application has to be installed in each and every house in which self reliant aged people live. Making their stress fall to the lowest and thereby increasing their odds of survival.

2.JOBS-TO-BE-DONE/PROBLEM: There are no such applications that are as effective as this There are no effective solution for this kind of problem application. The job that is to be done is provide the database correctly and clearly which consists of the user name, the time in which the medicine has to be taken and also the name of the specific medicines. Once the databases along with the datasets are provided, the application is now ready to use. Installation of the system is a easier way.

3.TRIGGERS: In a population of 100 percentage,there are nearly 79 percentage of seniors out of them in which 40 percentage of senior people are self reliant. In such case, this application if installed in one house, the neighbourhood people will see that the self reliant people are stress free and enjoying their life with this application. If there is also senior people who are self reliant in their house means,, surely they will get triggered to install this application in their house.

4. EMOTIONS: BEFORE / AFTER: Coming to the emotion part, before the installation of the application the user who needs to be pampered always,feels it as a burden. one may find it difficult to do the same thing all the day long and for weeks,months and even years.It even makes the family members to get into a stressful situation. Even the client may sometimes find it hard to rely to the same person for even small things. But after the installation of this application, the family members may find it easy and out of stress. They can do their job in a most comfortable way without worrying about the aged people in their home. The emotions were darker before the installation of this application and became very lighter and colourful after the installation of this application.

5.AVAILABLE SOLUTIONS: Even though there are more various ways and methods to take care of the aged people. No other application is as efffec effective as this application. It remembers the user with an alarm that its time to take the medicine and a voice saying the name of the medicine that the user has to take to avoid confusion and also to avoid taking the wrong medicines.

6.CUSTOMER CONSTRAINTS: This application is not so expensive. Once installed and the databases are provided, it will perform its work at the fullest. Aged people will feel more comfortable in taking their daily medicines on time

without failing and without relying on other person's help. It has a vast network managing more than one customer.

7.BEHAVIOUR: This application behaves in a user friendly manner. It has been provided with a alarm and a voice that also remembers the user about the time to take the medicine It also remembers the user about the name of the medicine that has to be taken in that specific time.

8.CHANNELS OF BEHAVIOUR :

8.1 ONLINE: It can neither be used in online or either in offline too. it is a double mode operating system. Which is designed to be a user friendly.

8.2 OFFLINE: In offline mode it remembers the user with an specified alarm which even remembers the user that its time to take the medicine.To avoid the confusion on which medicine has to be taken in that particular specific type,the application even alarms with the name of the medicines too.

9.PROBLEM ROOT CAUSE: The main root cause of identifying this application is the emerging old aged homes in which people leave their elderly people because they can no longer take care of them anymore. This application surely reduces the odds and percentage of sending aged people to old age homes. If this application is installed, the old aged people can take care of themselves without other persons help or nurse help.

10. YOUR SOLUTION: An intervention called CAPABLE - for Community Aging in Place, Advancing Better Living for Elders involves home visits with an occupational therapists, a registered nurse, and a handyman to work together with older adults to identify mobility and self care issues in their homes and fix or modify them.As a part of this,by making small adjustments ,from installing such application systems,it helps the client remember to take medicines at the proper time each and every day.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions.

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

Table 4.1 Functional Requirements

4.2 NON-FUNCTIONAL REQUIREMENT

Non functional requirements are requirements that define 'how' the app must perform a certain function. In essence, they are the quality attributes of an app that define the user experience of the app. They are also known as non-behavioral requirements and are to be implemented according to their priority to the app function. This makes them flexible to an extent, making it possible to skip a few in case of time, budget or technology constraints.

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	App can be used by anyone who has operational knowledge about internet and computer.
NFR-2	**Security**	For security, TFA is enabled and biometrics are alsoadded for user safety.
NFR-3	**Reliability**	Highly reliable since, It uses Trusted cloud services like IBM
NFR-4	**Performance**	Performance is better compared to other market products.
NFR-5	**Availability**	Available on mobile app. Web version is getting ready for next release.
NFR-6	**Scalability**	Using Cloud services, makes the scalability higher the using traditional database.

Table 4.2 Non Functional Requirements

CHAPTER 5
PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

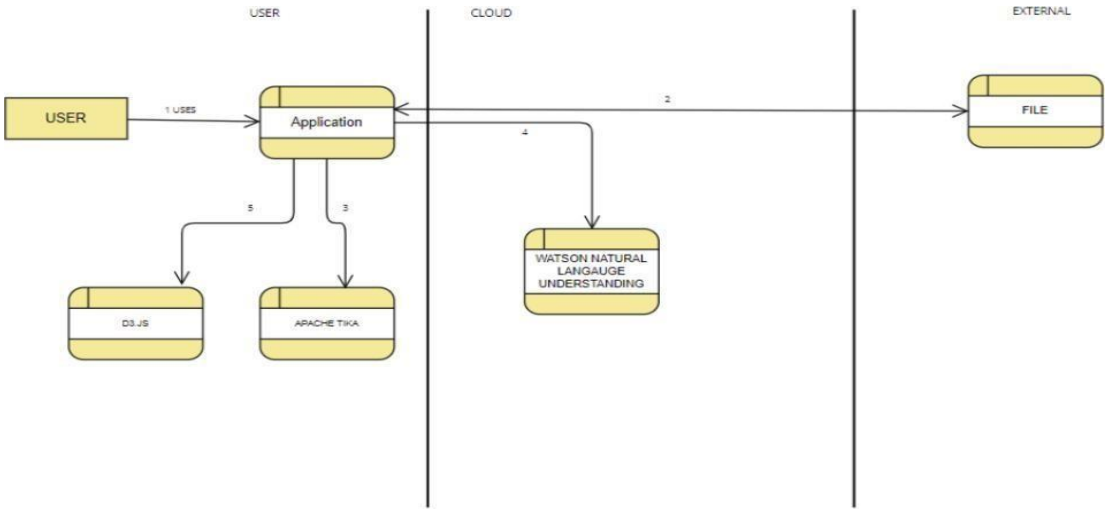


Figure.5.1. Dataflow Diagram

FLOWCHART

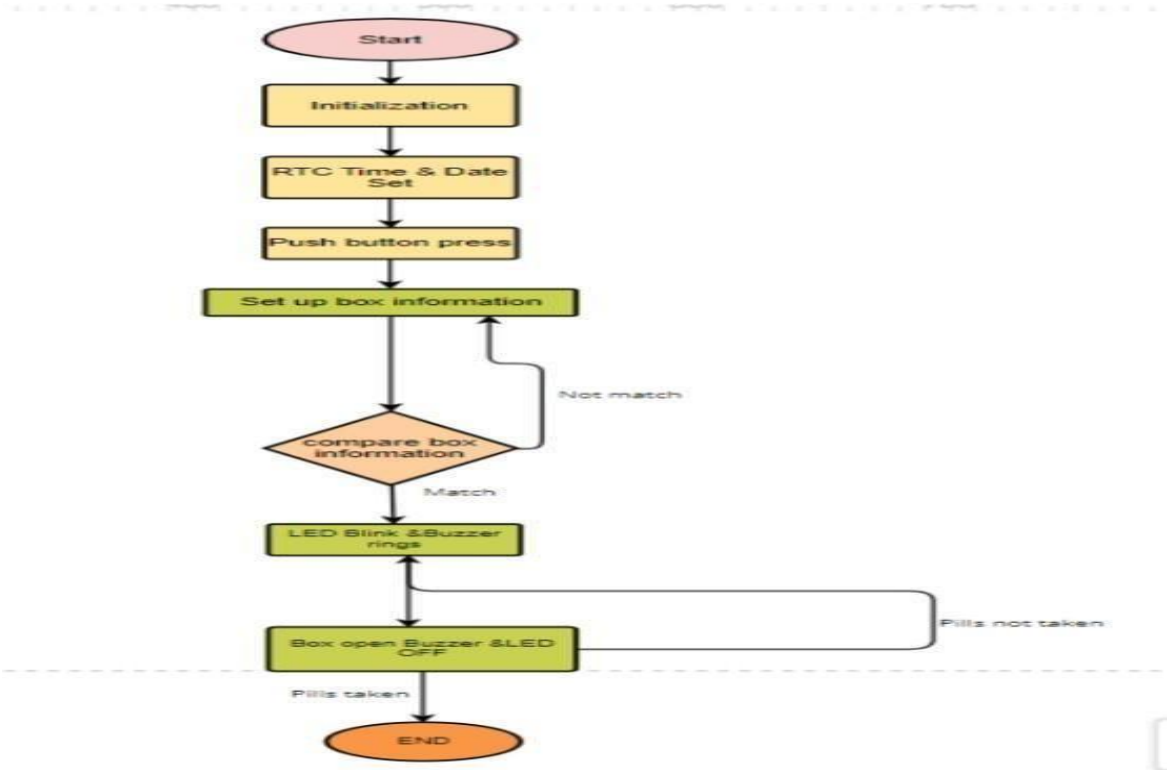


Figure 5.2. FlowChart

5.2 SOLUTION & TECHNICAL ARCHITECTURE

As people become older, people generally will experience a health decline such as becomes weak, suspectable to disease, decreased vision ability, etc. Therefore, special health a end on is needed for the elderly people, especially from the family member or personal doctors / nurses. On the other hand, the number of elderly people in the world is rapidly increase so there's more people will need special a end on. Therefore, this research try to develop an application on mobile phone that could help elderly people and their family member to supervise and monitor the health of the elderly.

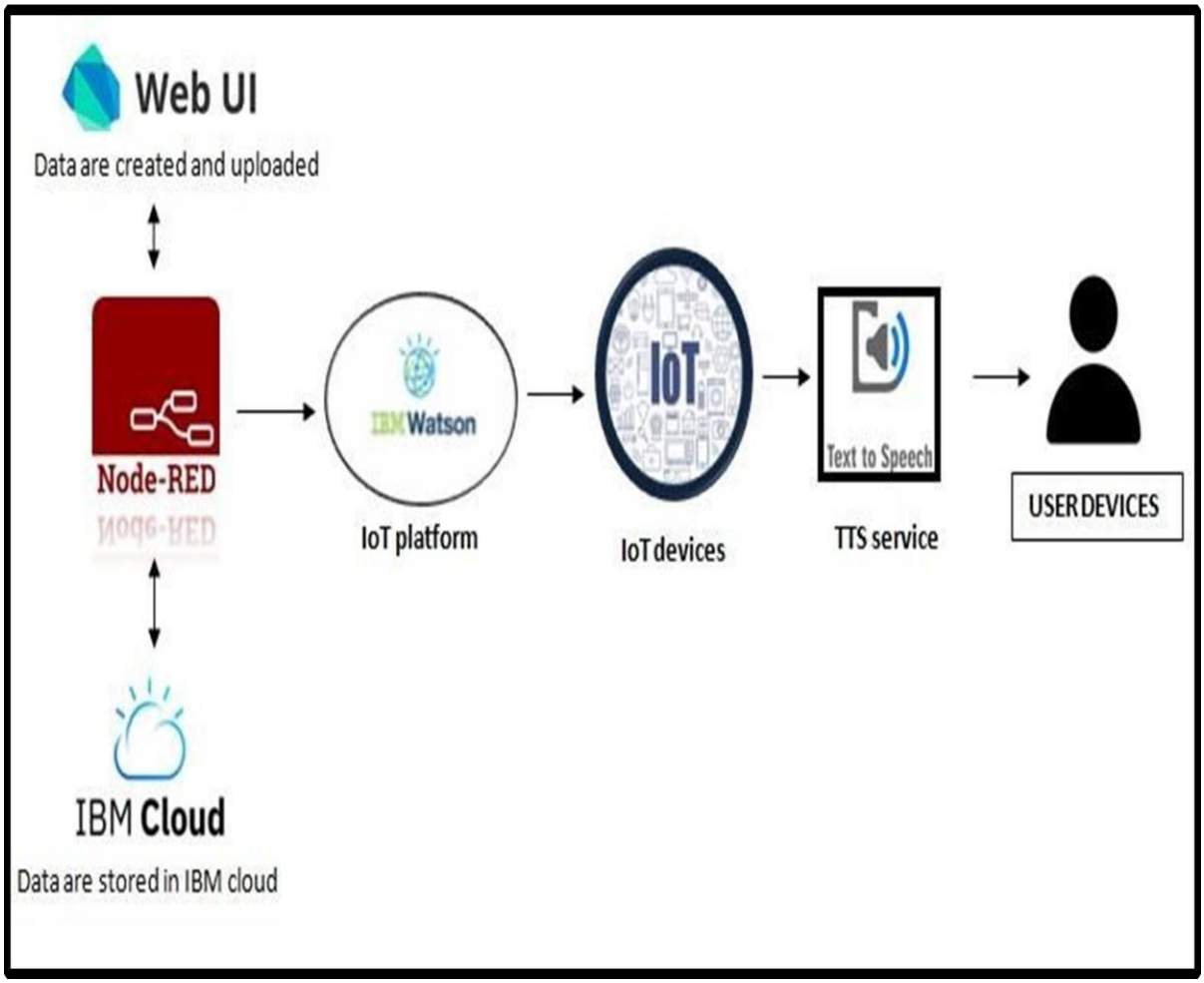


Figure 5.3 Technical Architecture

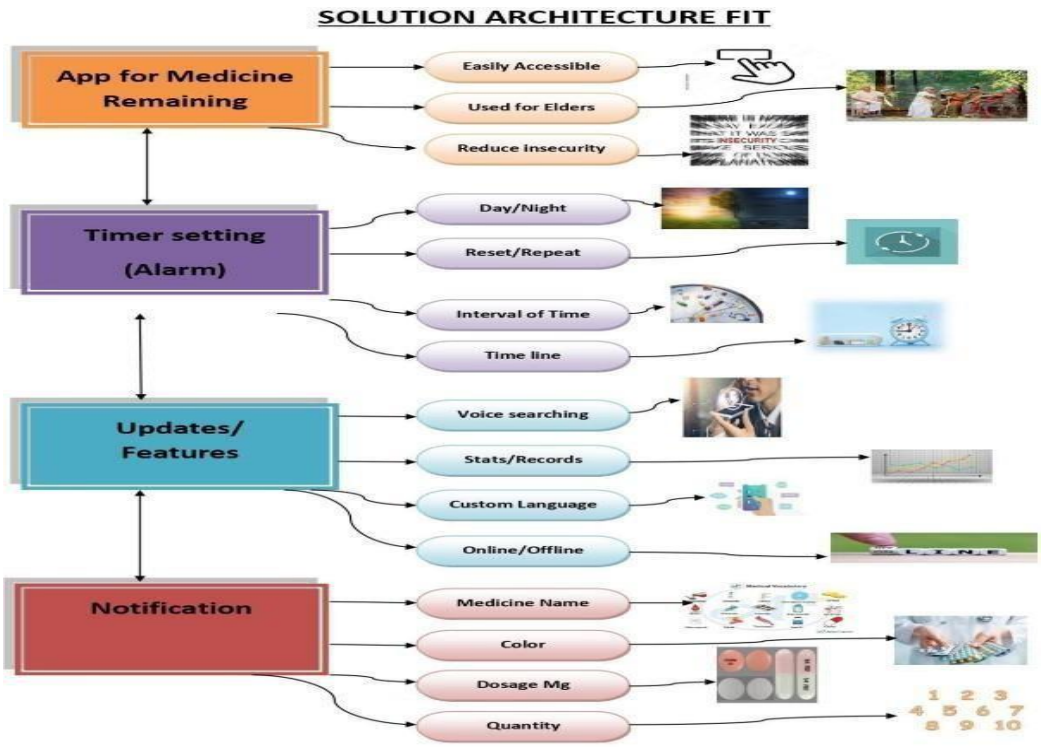


Figure 5.4 Solution Architecture

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Mobile application	HTML, CSS
2.	Application Logic-1	Enter the details of the medicines in a weekly basis	Javascript / Python
3.	Application Logic-2	Application gets the medicine data from the database	IBM Watson IoT API Call data
4.	Application Logic-3	Converts the text into speech format to pronounce for the user	IBM Watson Assistant
5.	Database	Medicine name, medicine time on daily basis	MySQL
6.	Cloud Database	Call the data IBM Cloudant is used and user login credentials	IBM DB2, IBM Cloudant
7.	File Storage	App code and IoT credentials are stored and also the API keys	IBM Block Storage
8.	External API-1	To get the medicine box status whether it is open or not	IBM box status API.
9.	External API-2	To get the login credentials in IBM DB2	User name and Password API
10.	Machine Learning Model	To convert the text into speech for voice command the medicine details	Text to speech.
11.	Infrastructure (Server / Cloud)	To host the server and application	Cloud Foundry, Node Red

Table 5.1 Components And Technologies.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	To develop the application interface	MIT APP INVENTOR
2.	Security Implementations	To secure the users login credentials and personal information	. SHA-256, OWASP
3.	Scalable Architecture	To scale the application database	IBM Auto scaling
4.	Availability	To make use the application and data are available for 24/7	IBM Cloud load balancer
5.	Performance	To increase the performance of the application inhosted in the high performance instance	IBM instance

Table 5.2 Application Characteristics

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Senior citizen)	Caretaker	USN-1	As a user, I want to take Medicines on time and monitor my health	I want to Take Medicines On time	High	Sprint-1
Customer (Alzheime r patient)	Smart medicine box	USN-2	As a user, I want to take my tablets on time by voice command	I want to take my tablets on time by voice command	High	Sprint-1
Custome r (Mentally idled patient)	Caretaker	USN-3	As a user, my patient needs to take medicines on time and monitoring the activity	My patient needs to take medicines on time	Medium	Sprint-2
Custome r (Disable d people's)	Smart medicine box	USN-4	As a user, I need to take my medicine in nearby places with light notification	I need to take my medicine in nearby places with light notification	Medium	Sprint-3
Custome r (Coma patient)	Caretaker	USN-5	As a user, my patient medication time and prescription should load in database for upcoming week	My patient medication time and prescription should be in database list	High	Sprint-4
Customer(diabe tes)	Caretaker		As a user,take medicines at time	Take medicine at time	High	Sprint 4

Table 5.3 User Stories

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & 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, and password, and confirming user password.	4	High	Lavanya.R
Sprint-1	Sign up	USN-2	As a user, I can sign up by giving the details.	4	High	Keerthana.A
Sprint-1	Authentication	USN-3	As a user, I can register for the application	4	Medium	Keerthana.A Lavanya.R
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	4	High	Akshaya.k
Sprint-2	Dashboard	USN-5	As a user, I need to be able to view the functions that I can perform	4	High	Monikaa.R
Sprint-2	Add medicine	USN-1	As a user, I should be able to add medicine names	10	High	Keerthana.A Akshaya.K
Sprint-2	Add prescriptions	USN-2	As a user, I can add the prescriptions	10	Medium	Lavanya.R
Sprint-2	Schedule	USN-3,1	I should be able to schedule as a user	6	Low	Monikaa.R
Sprint-2	Medicines,prescription	USN-1,4	As a user I can view the medicines and prescription	7	Medium	Keerthana.A
Sprint-3	IBM Watson Iot device creation	USN-5,2	Creating the ibm iot device.	7	High	Lavanya.R Keerthana.A
Sprint-3	Text to speech service	USN-1,4	Converting text to speech and remind the medicines using ibm text to speech service.	6	High	Akshaya.K Monikaa.R
Sprint-3	Workflow for iot using node red.	USN-2,3,5	Creating medicine remainder form using node red.	7	High	Lavanya.R Akshaya.K Keerthana.A Monikaa.R
Sprint 4	Mit app inventor	USN-1,4	Application using mit app	7	High	Keerthana A Lavanya R
Sprint 4	Alarm remainder	USN 2,3,5	Alarm remainder based on medication time	6	High	Akshaya K Monikaa R

Table 6.1 Sprint planning and evaluation

6.2 SPRINT DELIVERY SCHEDULE

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	5 Days	28 october 2022	1 November 2022	20	1 Nov 2022
Sprint-2	20	5 Days	2 November 2022	6 November 2022	20	6 Nov 2022
Sprint-3	20	5 Days	7 November 2022	11 November 2022	20	11 Nov 2022
Sprint-4	20	5 Days	12 November 2022	16 November 2022	20	16 Nov 2022

Table 6.2 Sprint Delivery Schedule

Velocity:

Imagine we have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

AV = SPRINT DURATION / VELOCITY

=20/10

=2

6.3 REPORTS FROM JIRA

Burndown Chart:

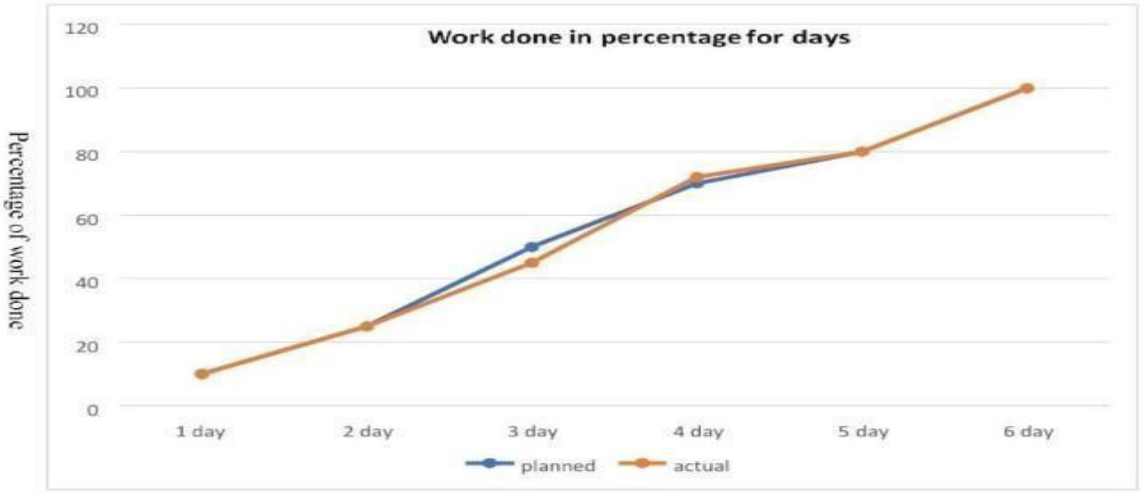


Figure 6.1 Burndown Chart

ROAD MAP:

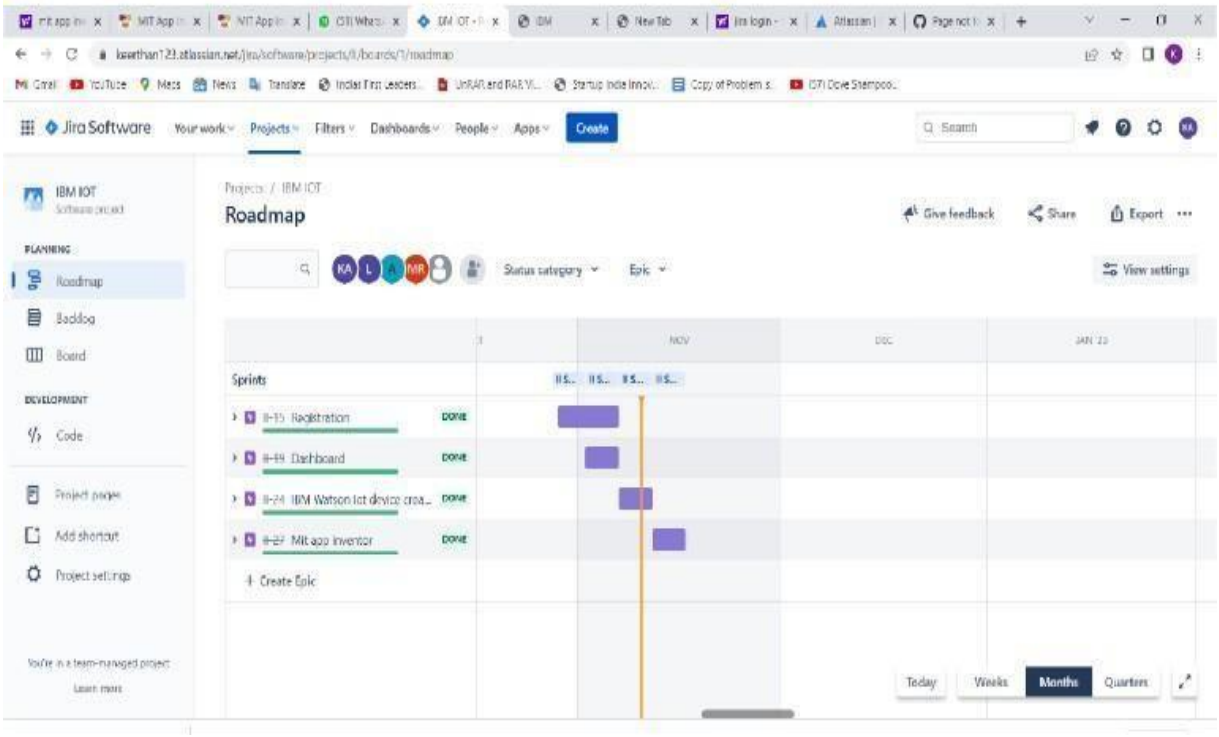


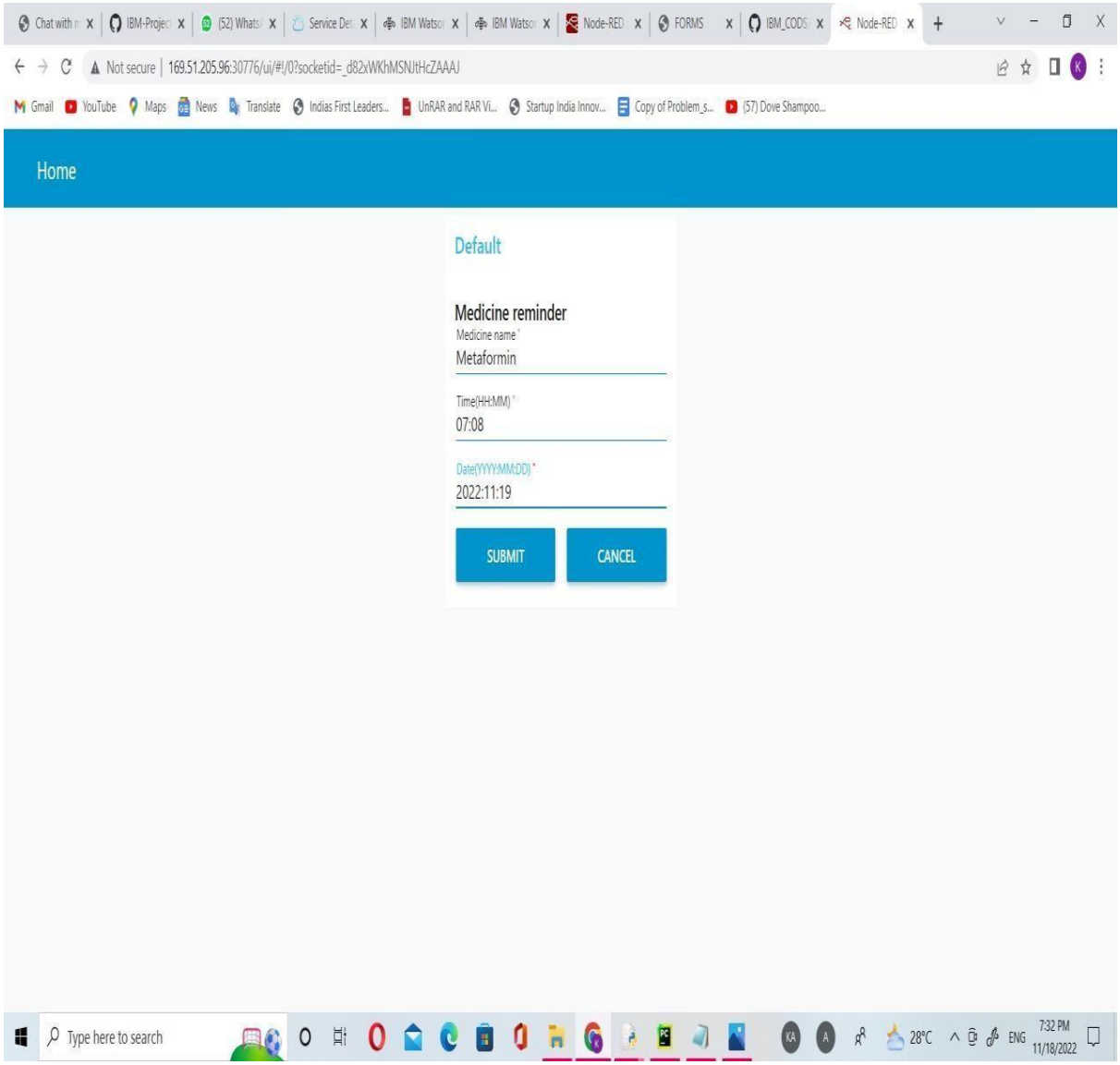
Figure 6.2 Road Map

CHAPTER 7

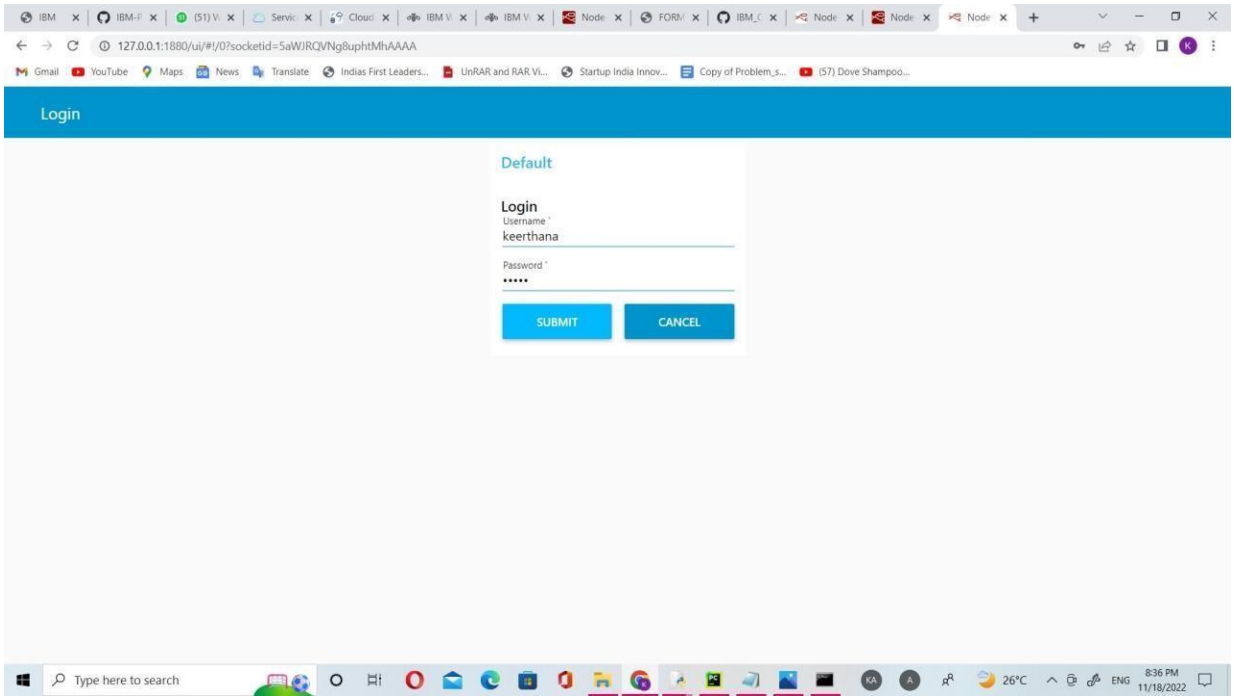
CODING & SOLUTIONING

7.1 Feature 1: NODE-RED

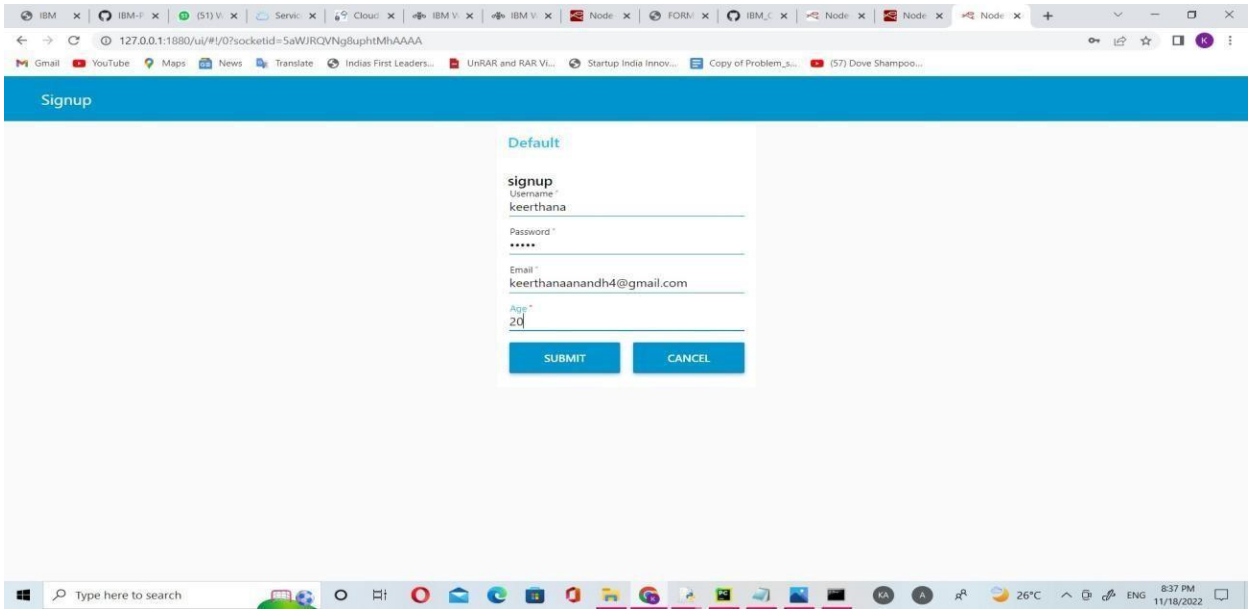
Web UI to enter medicine name,date and time for reminder:



Login page using node-red:

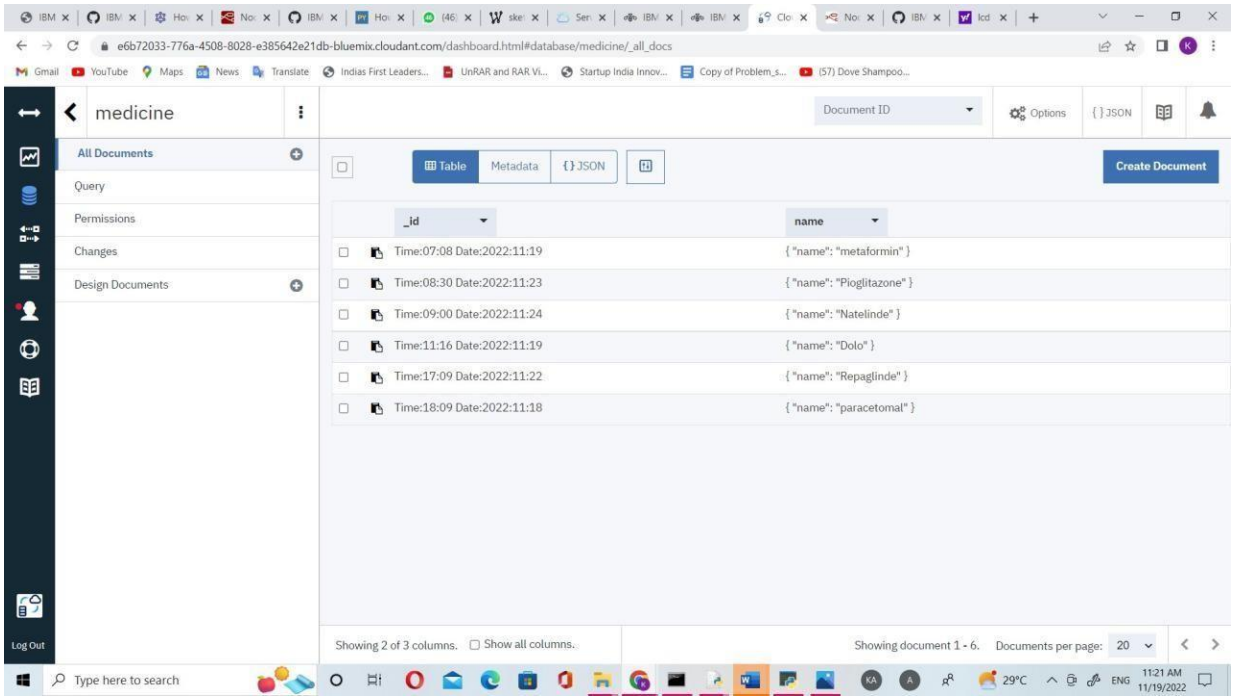


Signup page using node red:

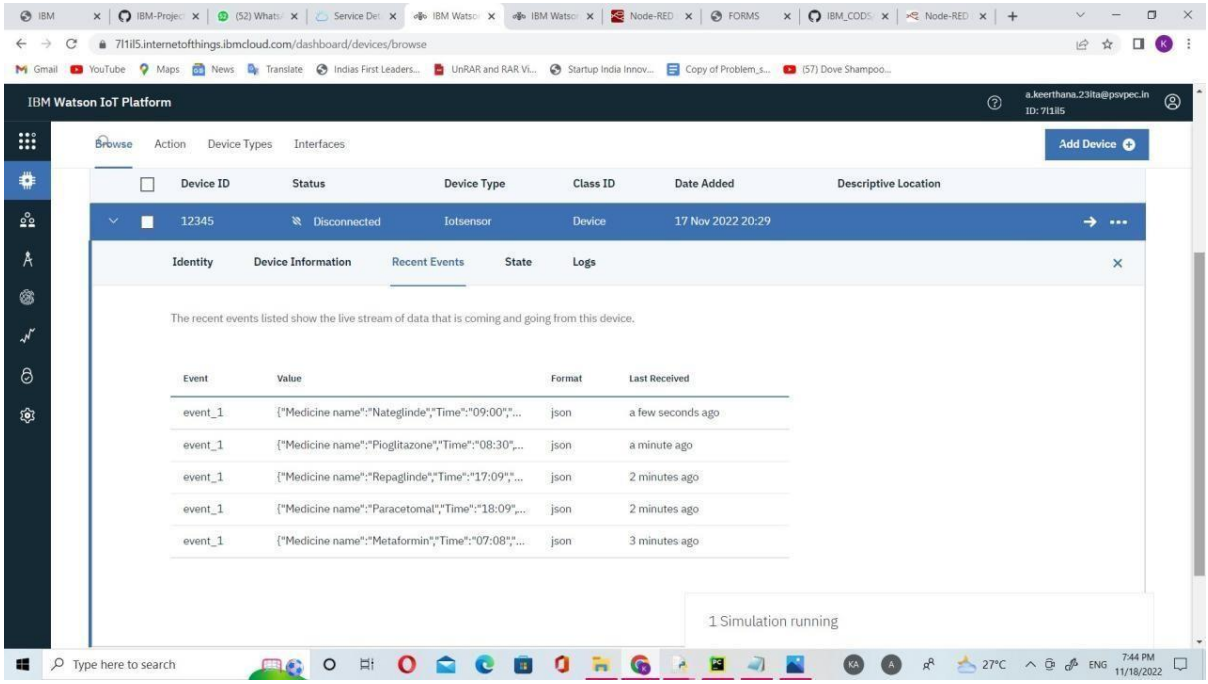


The user's input will be stored in the cloundant database.

CLOUDANT DB:

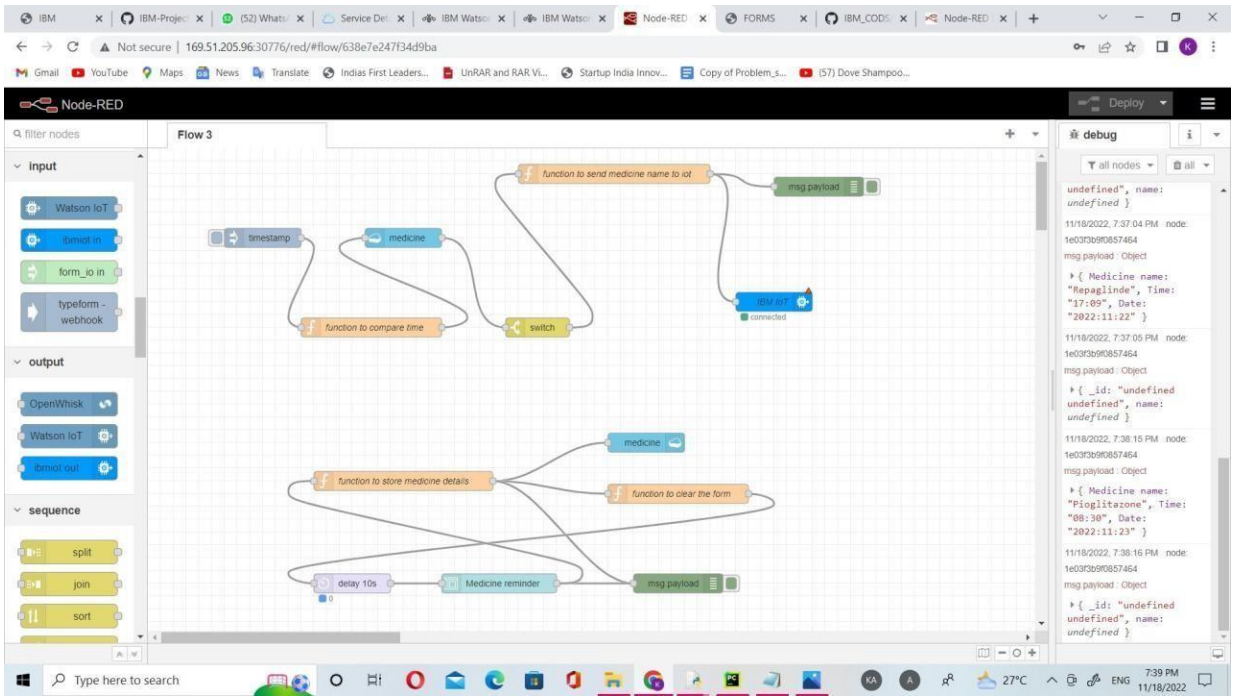


The user will be notified to take medicine when time arrives.



FLOW:

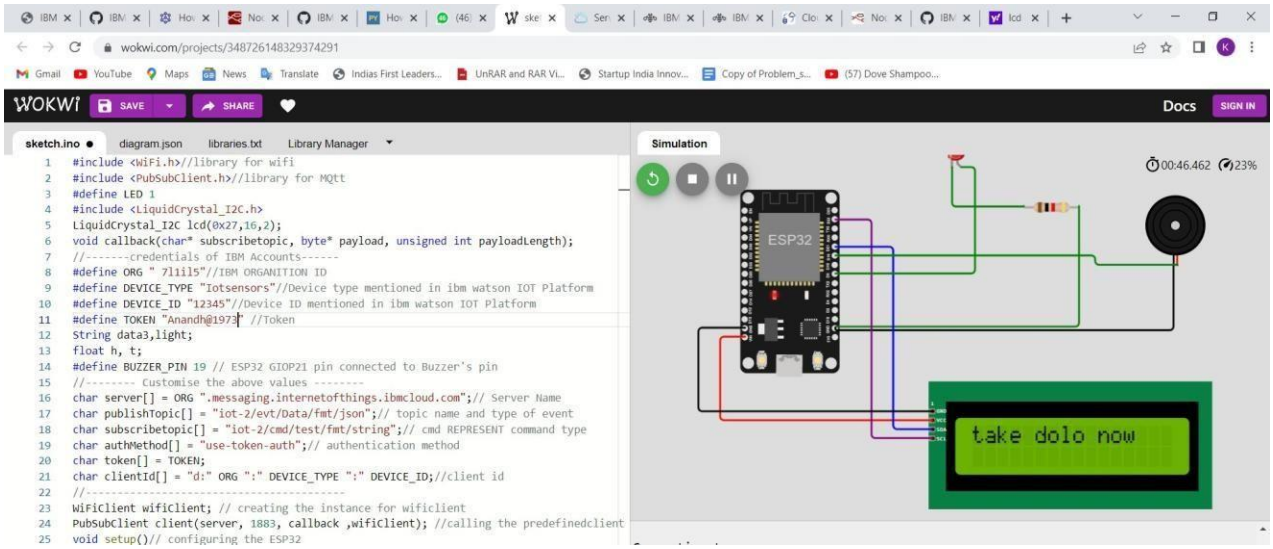
The medicine name, date and time is displayed in the debug window.



7.2 Feature 2

A IOT device using ESP32 which notifies the user when the medicine time arrives.

The device consists of ESP32 which is used for connecting with ibm cloud to publish and subscribe data.The led glows ,buzzer rings and medicine name will be displayed in lcd when the time arrives.



The user can enter the medicine name,date and time which is stored in cloudant db and node red checks in cloudant database if any medicine has to be taken,it issues a command to IOT device through IBM IOT watson platform.When time arrives it shows medicine name in display,led glows and buzzer rings.

MIT APP INVENTOR:

An app is designed using the mit app inventor to

1.Remind medicines

2.Buy medicines

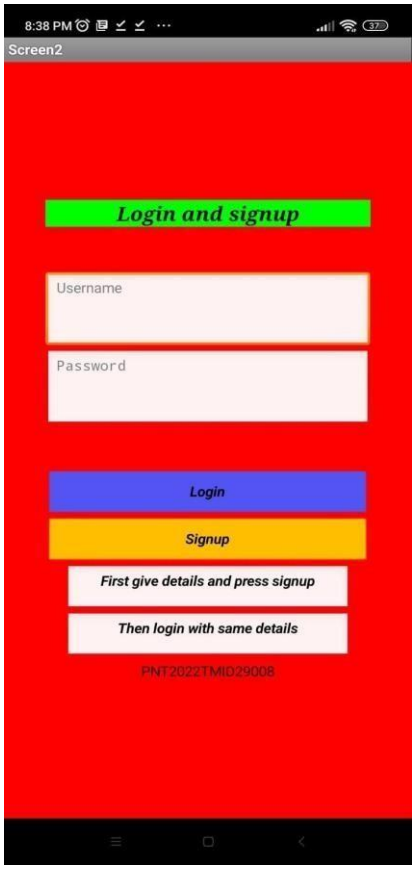
3.Chat with caretaker

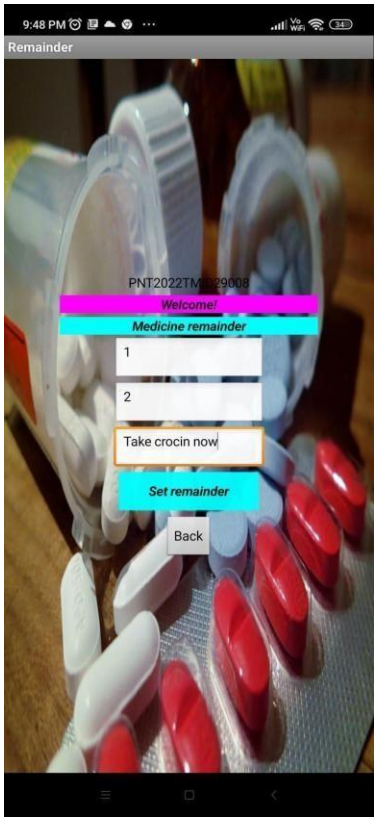
4.Assistant

5.Check health info


8:24 PM
Screen1
Personal assistance for seniors who are self reliant
Start here
IBM PROJECT
IBM-Project-11164-1659274470
TEAM ID PNT2022TMID29008
By: Keerthana,Lavanya,Akshaya,Monikaa

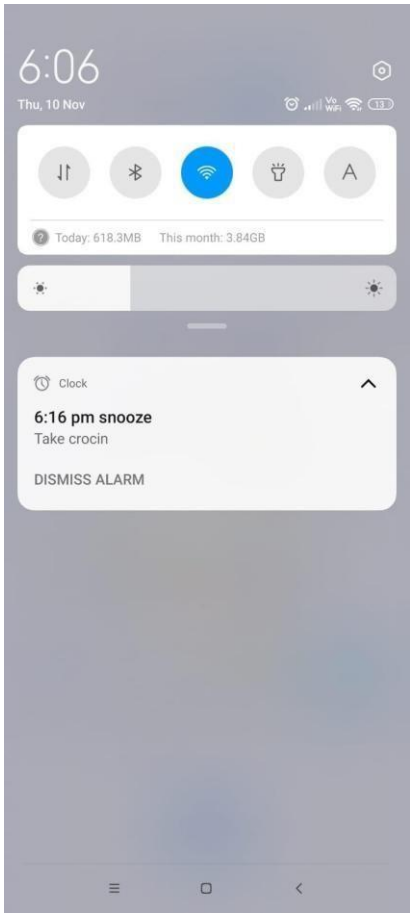

9:52 PM
Screen2
PNT2022TMID29008
Set remainder
Health info
Buy medicines
Assistant
Chat with caretaker
Call and share message

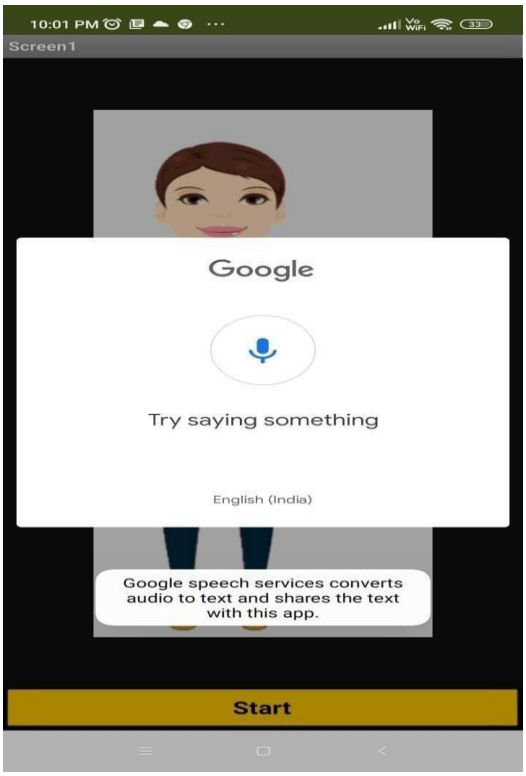

8:38 PM
Screen2
Login and signup
Username
Password
Login
Signup
First give details and press signup
Then login with same details
PNT2022TMID29008


9:48 PM
Remainder
PNT2022TMID29008
Welcome!
Medicine remainder
1
2
Take crocin now
Set remainder
Back


10:08 PM
Call_and_message
Do call and share message
PNT2022TMID29008
12345
call
Hello
Share

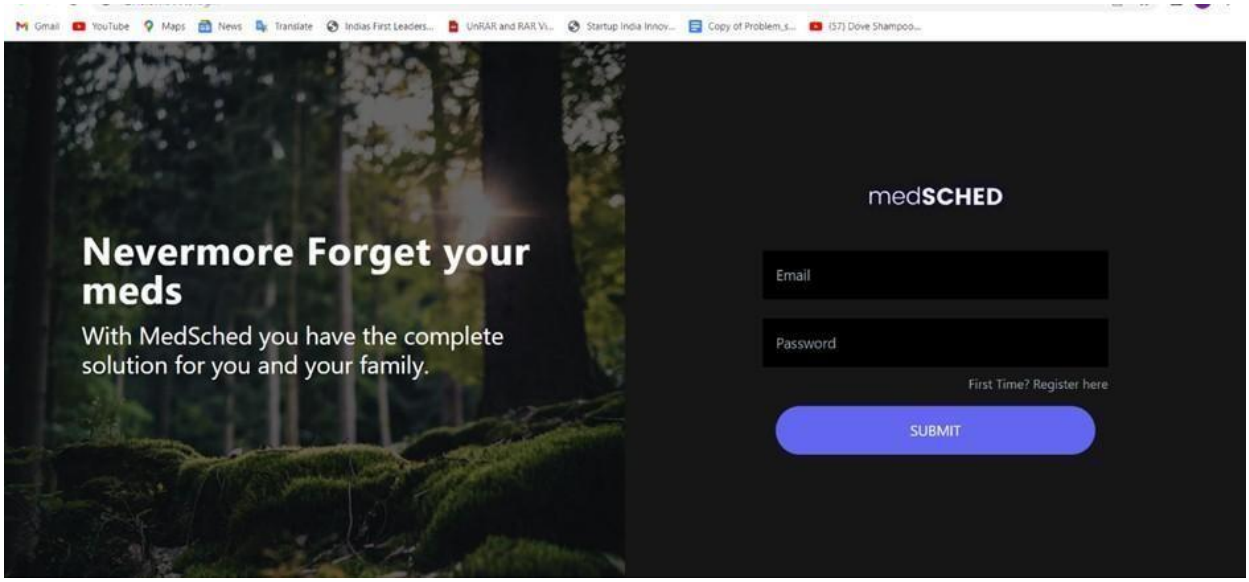

10:04 PM
Chat
Keerthana
Save
How are you
Send
Delete all the messages
CHAT MESSAGES
Hello-Keerthana
How are you-Keerthana

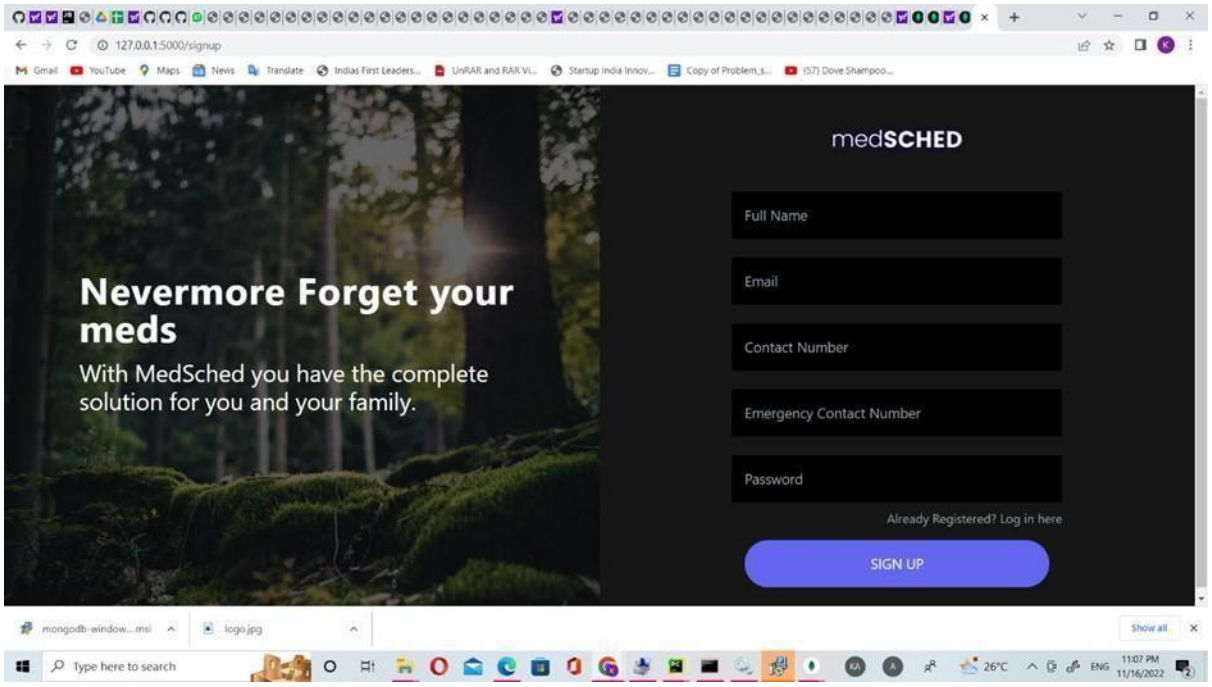


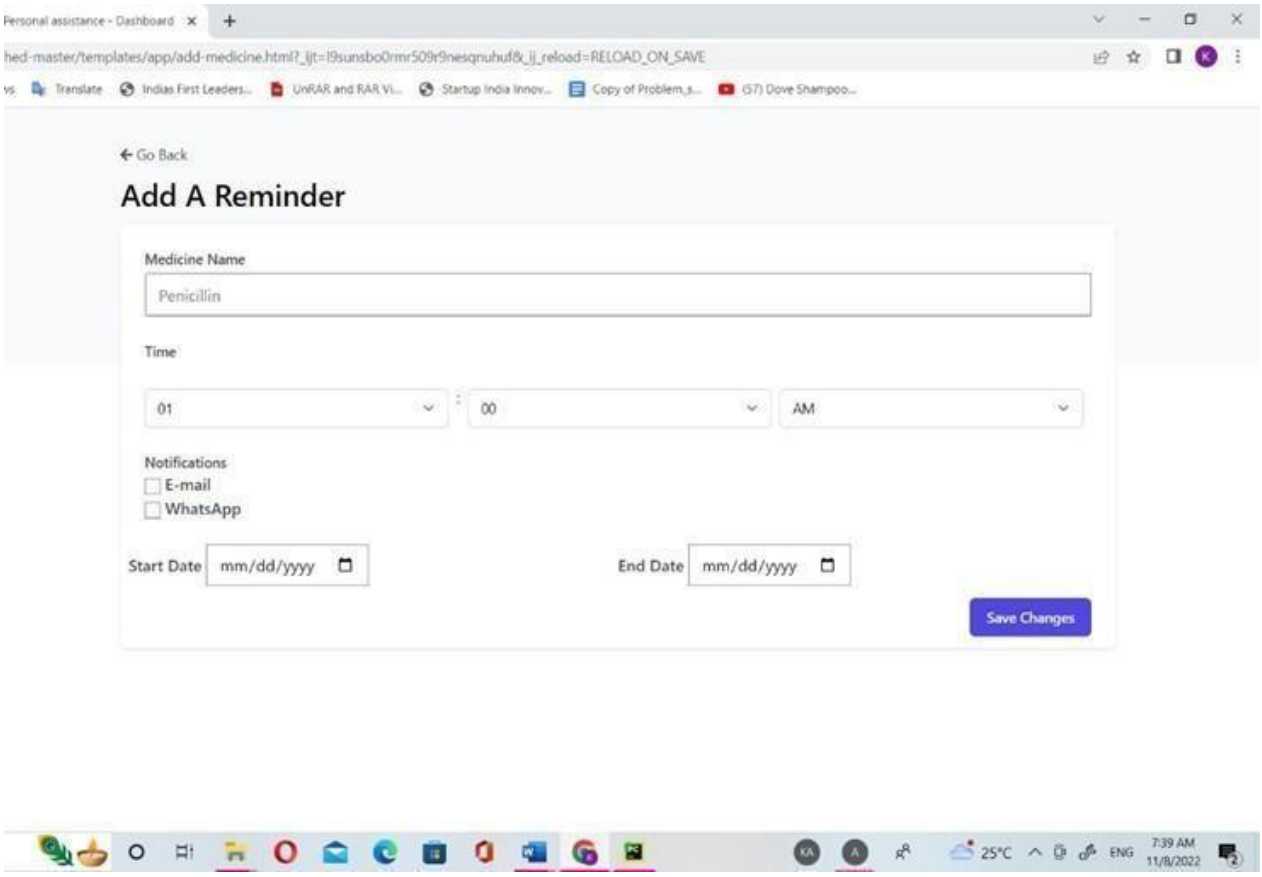


The medicine will be reminded at time with an alarm and message.

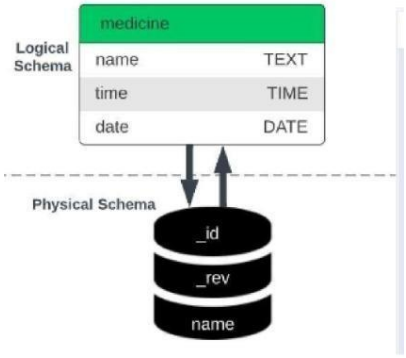
WEB APP:

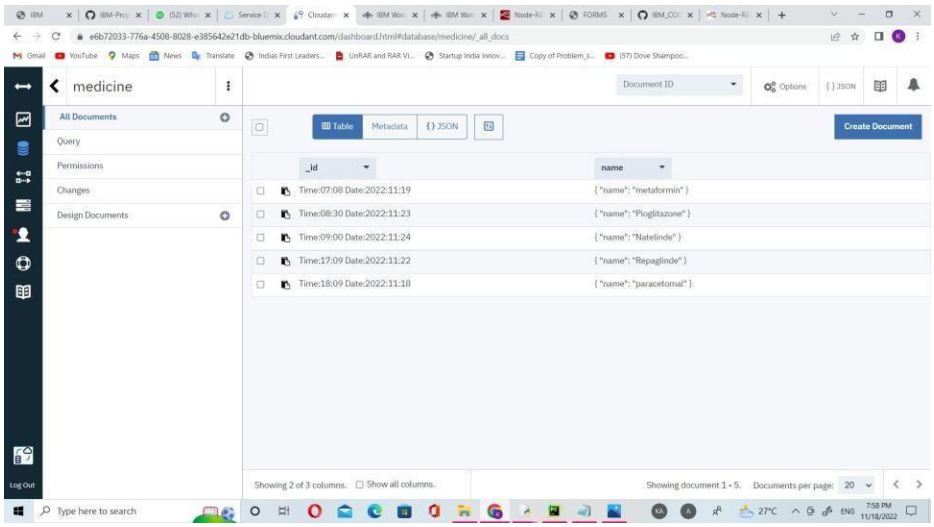

Nevermore Forget your meds
With MedSched you have the complete solution for you and your family.
medSCHED
Email
Password
First Time? Register here
SUBMIT

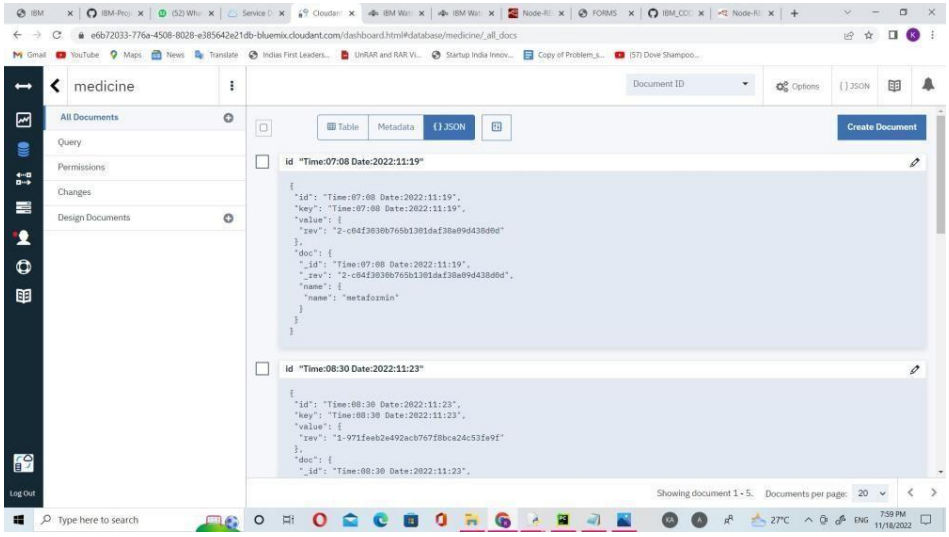

127.0.0.1:5000/signup
Nevermore Forget your meds
With MedSched you have the complete solution for you and your family.
medSCHED
Full Name
Email
Contact Number
Emergency Contact Number
Password
Already Registered? Log in here
SIGN UP
Type here to search



7.3 DATABASE SCHEMA :




medicine
All Documents
Query
Permissions
Changes
Design Documents
Table
Metadata
{ } JSON
Create Document
_id
name
Time:07:08 Date:2022:11:19
{ "name": "metaformin" }
Time:08:30 Date:2022:11:23
{ "name": "Pioglitazone" }
Time:09:00 Date:2022:11:24
{ "name": "Natelinde" }
Time:17:09 Date:2022:11:22
{ "name": "Repaglinde" }
Time:18:09 Date:2022:11:18
{ "name": "paracetomal" }
Showing 2 of 3 columns.
Show all columns.
Showing document 1 - 5.
Documents per page: 20
Log Out
Type here to search


medicine
All Documents
Query
Permissions
Changes
Design Documents
Table
Metadata
{ } JSON
Create Document
id "Time:07:08 Date:2022:11:19"
id "Time:08:30 Date:2022:11:23"
Showing document 1 - 5.
Documents per page: 20
Log Out
Type here to search

CHAPTER 8

TESTING

8.1 Test Cases

1					Date	3-Nov-22				
2					Team ID	PNT2022TMID29008				
3					Project Name	Personal assistant for senior who are self r				
4					Maximum Marks	4 marks				
5	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
6	LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup displayed or not	https://drive.google.com/drive/folders/12Ue5Wnx7KEkZ7-pM9cKrWo6J2oJ_tZEp	Login/Signup popup should display	Working as expected	Pass
7	LoginPage_TC_002	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	https://drive.google.com/drive/folders/12Ue5Wnx7KEkZ7-pM9cKrWo6J2oJ_tZEp	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	pass
8	LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: lavanya02 password: Testing123	User should navigate to user account homepage	Working as expected	pass
9	LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: keerthana05 password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	pass
10	LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username:monikaa26 password: Testing123678686786876	Application should show 'Incorrect email or password ' validation message.	Working as expected	pass
11	LoginPage_TC_005	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: akshaya06 password: Testing123678686786876876	Application should show 'Incorrect email or password ' validation message.	Working as expected	pass
12	Reminder page_TC_00_6	Functional	Reminder page	page should display the details of the medicine		1.set the medicine details 2.set the e and date 3.set the alarm the submit button for reminding	medicine name:crocin time:every 6 hrs	Alarm along with medicine details	working as expected	pass

Figure 8.1 Test Cases

8.2.USER ACCEPTANCE TESTING

The purpose of this document is to briefly explain the test coverage and open issues of the Medicine reminder project at the time of the release to User Acceptance Testing (UAT).

1. DEFECT ANALYSIS

This report shows the number of resolved or closed bugs at each severity level ,and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

Table 8.1 Defect Analysis

2. TEST CASE ANALYSIS

This report shows the number of test cases that have passed,failed and untested.

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	2	0	0	2
Client Application	2	0	0	2
Security	1	0	0	1
Outsource Shipping	1	0	0	1
Exception Reporting	2	0	0	2
Final Report Output	1	0	0	1
Version Control	1	0	0	1

Table 8.2 Test Case Analysis

CHAPTER 9

RESULTS

9.1 Performance Metrics

NFT - Risk Assessment

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Voluem Changes	Risk Score	Justification
1	Personal Assistance for Seniors Who Are Self-Reliant	New	Low	Moderate	Moderate	Low	>5 to 10%	ORANGE	As we have seen the changes

NFT - Detailed Test Plan

S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
1	Medicine Reminder Web -UI	Stress	App Crash/ Developer team/ Site Down	Approved
2	Medicine Reminder Web -UI	Endurance	App Crash/ Site Down	Approved
3	Medicine Reminder Web -UI	Load	Server Crash/ Developer team/ Server Down	Approved

End Of Test Report

S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Medicine Reminder Web -UI	Stress	Performance	CPU -01	GO	High Performance server	Closed	Approved
2	Medicine Reminder Web -UI	Load	Scalability	DB Storage - 01	NO-GO	MongoDB Instance for free	Closed	Approved
3	Medicine Reminder Web -UI	Endurance	Connectivity	Connection	GO	High Performance	Closed	Approved

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES

1) It avoids the difficulties for doctors (or) caretakers to monitor the patients around the clock.

2) Patient can easily take the medicines at correct time.

3) The software is very user-friendly; the need not install any external app by the patient, economic for the caretaker too.

4) The details of the time scheduled, and patients' intake is stored in the database for future reference easily.

5) The overall stress of patients and caretakers is reduced and maintained under control by the software.

DIS-ADVANTAGES

1) If seniors/patients who are physically disabled (like deaf). They can't hear the voice command.

2) If seniors/patients who are visually challenged (like cataracts)and illiterate, they can't read the medicines name properly.

CHAPTER 11

CONCLUSION

The elderly population is large in general and growing due to advancement of health care education. These people are faced with numerous physical, psychological and social role changes that challenge their sense of self and capacity to live happily. Many people experience loneliness and depression in old age, either as a result of living alone or due to lack of close family ties and reduced connections with their family members, which results in an inability to take care of themselves. So if they are provided with an application that acts as a self assistance will help them in taking care of themselves very well. This application will also remember them the correct medicine to consume at the correct time. It will assist them with a voice over to avoid confusion on what medicine has to be consumed at what time. Thus it will improve their living standards and both their physical and mental health. Even the illiterate senior citizens will also find it easy to use this application and will get benefited. Installation and usage of this application is very much easy. Care takers can guide the elderly people even at a far distance through this application. Medicines that are out of stock can also be purchased via this application. It directs the user to the particular pharmacy page directly and thus the user can buy all the medicines he/she needed without stepping out from the home. It also has a assistance feature to guide the users.

CHAPTER 12

FUTURE SCOPE

The project can be enhanced with many other features that can serve senior citizens even better. The product currently is a simple basic version which can only send messages alerts on time. Some other additional features that are planned to be incorporated with this existing product are listed below:

1) A button on the device , on pressed can send an alert call/text to the caregiver if the user is ill or needs help.
2) A button ,when not pressed after the notification,send a message to the user and the caregiver an alert prompt "MEDICINE NOT TAKEN.PLEASE TAKE."
3) The system can be enhanced with a smartwatch or health devices so that the health conditions can be continuously connected with the hospitals, and doctors to supervise and help them during emergencies.

CHAPTER 13

APPENDIX

SOURCED

E

Sketch.ino:

```
#include<wifi.h>

#include <PubSubClient.h>//library for MQtt
```

```
#define LED 1

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);

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

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

#define ORG " 7l1il5"//IBM ORGANITION ID

#define DEVICE_TYPE "Iotsensors"//Device type mentioned in ibm watson IOT Platform

#define DEVICE_ID "12345"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN "Anandh@1973" //Token

String data3,light;

float h, t;

#define BUZZER_PIN 19 // ESP32 GIOP21 pin connected to Buzzer's pin

//-------- 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

char subscribetopic[] = "iot-2/cmd/test/fmt/string";// cmd REPRESENT command type

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 predefinedclient id by passing parameter like server id,portand wificredential

void setup()// configuring the ESP32

{

Serial.begin(115200);

Serial.begin(9600);

// dht.begin();

pinMode(LED,OUTPUT);

pinMode(BUZZER_PIN, OUTPUT);

delay(10);

lcd.init();

lcd.clear();

lcd.backlight();

Serial.println();

wificonnect();

mqttconnect();

}

void loop()// Recursive Function

{
```

```
 digitalWrite(BUZZER_PIN, HIGH);

delay(1000);

if (!client.loop()) {

mqttconnect();

}

}

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);

light=(char)payload[0];

for (int i = 1; i < payloadLength; i++) {

Serial.print((char)payload[i]);

data3 += (char)payload[i];

}

// Make sure backlight is on

Serial.println("data: "+ data3);

if(light=="n")

{

digitalWrite(BUZZER_PIN, HIGH);

Serial.println(data3);

digitalWrite(LED,HIGH);

// Print a message on both lines of the LCD.

lcd.setCursor(2,0); //Set cursor to character 2 on line 0

lcd.print("Take now");

lcd.setCursor(2,1); //Move cursor to character 2 on line 1
```

```
lcd.print(data3);

delay(3000);

digitalWrite(BUZZER_PIN, LOW);

digitalWrite(LED,LOW);

lcd.clear();

}

else

{

digitalWrite(BUZZER_PIN, LOW);

Serial.println(data3);

digitalWrite(LED,LOW);

lcd.clear();

}

data3="";

}
```

Texttospeechservice.py:

```
import time
import sys
import ibmiotf
```

```
import ibmiotf.device
import random
from ibm_watson import TextToSpeechV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator


#Provide your IBM Watson Device Credentials
organization = "7l1il5" # repalce it with organization ID
deviceType = "Iotsensor" #replace it with device type
deviceId = "12345" #repalce with device id
authMethod = "token"
authToken = "Anandh@1973"#repalce with token
authenticator = IAMAuthenticator('9aUkwzosUqF1y0gJfUA0Vq50z0dLa70tE1Ujbecq0p_L')
text_to_speech = TextToSpeechV1(
  authenticator=authenticator
)


text_to_speech.set_service_url('https://api.au-syd.text-to-speech.watson.cloud.ibm.com/instances/53de65a2-1737-4f34-9a7b-15ca2e0819bc')




def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)
    if cmd.data['n']!='undefined':
        print(cmd.data['n'])
        x="its time to take "+ cmd.data['n']
        with open('hello_world.wav', 'wb') as audio_file:

audio_file.write(text_to_speech.synthesize(x,voice='en-US_AllisonV3Voice',accept='audio/wav').get_result().content)
    else:
      print("LIGHT OFF")
```

```
try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method":
authMethod, "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #.....................................................

except Exception as e:
        print("Caught exception connecting device: %s" % str(e))
        sys.exit()


deviceCli.connect()


while True:

    time.sleep(1)

    deviceCli.commandCallback = myCommandCallback


# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

main.py : (Web app)

```
from flask import Flask, flash, request, session
from flask.templating import render_template
from werkzeug.utils import redirect, secure_filename
import bcrypt
from functions.users import check_existing_user, add_new_user, check_user_credentials
from functions.scheduler import add_medicine, fetch_user_schedule, card, no_data_check
from functions.dashboard import user_name, today_card, tomorrow_card, day_after_card
from functions.medicines import get_all_medicines, medicine_card
from functions.medscraper import get_medicines
```

```
from functions.prescription import add_prescription, add_prescription_record, fetch_prescriptions, prescription_card
from functions.notifications import *
from datetime import datetime, timedelta

import os

app = Flask(_name_)

app.secret_key =  'subhogay'

app.config['UPLOAD_FOLDER'] = 'uploads'

app.config['MAX_CONTENT_LENGTH'] = 20 * 1000 * 1000

@app.route('/')

def index():

    try:

        if session['login']:

            login = True

        else:

            login = False

    except KeyError:

        session['login'] = False

        login = session['login']

    return render_template('index.html', login=login)

# Authentication

@app.route('/signup')

def signup():

    return render_template('signup.html')
```

```
@app.route('/signup/verify', methods=['GET', 'POST'])
def signup_verify():
    if request.method == 'POST':
        data = request.form
        name = data['name']
        email = data['email']
        contact = data['contact']
        emergency_contact = data['emergency_contact']
        password = data['password'].encode()
        hashed = bcrypt.hashpw(password, bcrypt.gensalt())
        if not check_existing_user(email):
            add_new_user(name, email, hashed, contact, emergency_contact)
            flash('Registered')
            return redirect('/')
        else:
            flash('User with this email already exists.', 'error')
            return render_template('signup.html')
@app.route('/login')
def login():
    try:
        if session['login']:
            return redirect('/dashboard')
```

```
        else:
 return render_template('login.html')
    except:
        return render_template('login.html')
@app.route('/login/verify', methods=["GET", "POST"])
def login_verify():
    if request.method == "POST":
        data = request.form
        email = data['email']
        password = data['password']
        if check_user_credentials(email, password):
            session['user'] = email
            session['login'] = True
            flash('Logged in')
            return redirect('/dashboard')
        else:
            flash('Incorrect username or password!', 'error')
            return render_template('login.html')
@app.route('/logout')
def logout():
    session.clear()
    session['login'] = False
```

```
    session['user'] = None
return     redirect('/login')
@app.route('/dashboard')
def dashboard():
   try:
      if session['user']:
         if not no_data_check(session['user']):
            data = fetch_user_schedule(datetime.now(), session['user'])
            now = str(datetime.now()).split('.')[0][-8:-3]
            data['current_time'] = now
 upcoming = {}
 completed = {}
  data = dict(sorted(data.items(), key=lambda item: item[1]))
   upcoming_count = 0
   completed_count = 0
    checkpoint = False
    for medicine in data:
   if medicine == 'current_time':
   checkpoint = True
   elif checkpoint:
                    upcoming_count += 1
                    upcoming[medicine] = data[medicine]
```

```
            Else
             completed_count += 1
               completed[medicine] = data[medicine]
today_card_html = ''
          for medicine in upcoming:
            today_card_html += today_card(medicine, upcoming[medicine])
  first = 0
          for medicine in upcoming:
            first += 1
            if first < 2:
              first_medicine = medicine
              first_medicine_time = upcoming[medicine]
card1 = f"""<div class="md:p-7 p-4">
                <h2 class=" text-xl text-center text-primary-green-dark capitalize">Next Dose</h2>
                        <h3 class="text-sm text-primary-green-dark text-center">{first_medicine_time} -
{first_medicine}</h3>
              </div>"""
            card2 = f"""<div class="md:p-7 p-4">
              <h2 class="text-xl text-center text-primary-blue-dark capitalize">Today</h2>
                <h3 class="text-sm text-primary-blue-dark text-center">{upcoming_count + completed_count}
doses</h3>
              </div>"""
            card3 = f"""<div class="md:p-7 p-4">
```

```
            <h2 class="text-lg text-center text-primary-yellow-dark capitalize">
              <span>{first_medicine}</span>
            </h2>
            <h3 class="text-sm text-primary-yellow-dark text-center">{first_medicine_time}</h3>
          </div>"""
count = 0
tomorrow_card_html = ''
data = fetch_user_schedule(datetime.now() + timedelta(days=1), session['user'])
data = dict(sorted(data.items(), key=lambda item: item[1]))
for medicine in data:
    count += 1
    if count < 4:
        tomorrow_card_html += tomorrow_card(medicine, data[medicine])
count = 0
day_after_card_html = ''
data = fetch_user_schedule(datetime.now() + timedelta(days=2), session['user'])
data = dict(sorted(data.items(), key=lambda item: item[1]))
for medicine in data:
    count += 1
    if count < 4:
        day_after_card_html += day_after_card(medicine, data[medicine])
session['upcoming_count'] = upcoming_count
```

```
        if not upcoming and not completed:
            card1 = f"""<div class="md:p-7 p-4">
                                <h2 class=" text-xl text-center text-primary-green-dark capitalize">Add some data from scheduler</h2>
                </div>"""
            card2 = f"""<div class="md:p-7 p-4">
                                <h2 class="text-xl text-center text-primary-blue-dark capitalize">Today</h2>
                                        <h3 class="text-sm text-primary-blue-dark text-center">{upcoming_count + completed_count}  doses</h3>
                                </div>"""
            return render_template('app/dashboard.html', name=user_name(session['user']).title(), card1=card1,
                                card2=card2, upcoming_count=upcoming_count, today_card_html=today_card_html,
                                tomorrow_card_html=tomorrow_card_html,
                                day_after_card_html=day_after_card_html)
        elif not upcoming:
            card2 = f"""<div class="md:p-7 p-4">
                                <h2 class="text-xl text-center text-primary-blue-dark capitalize">Today</h2>
                                        <h3 class="text-sm text-primary-blue-dark text-center">{upcoming_count + completed_count}  doses</h3>
                                </div>"""
            return render_template('app/dashboard.html', name=user_name(session['user']).title(),
                                card2=card2, upcoming_count=upcoming_count,
                                today_card_html=today_card_html,
```

```
                                tomorrow_card_html=tomorrow_card_html,
                                day_after_card_html=day_after_card_html)
            else:
                return render_template('app/dashboard.html', name=user_name(session['user']).title(), card1=card1,
                                card2=card2,
                                card3=card3, upcoming_count=upcoming_count, today_card_html=today_card_html,
                                tomorrow_card_html=tomorrow_card_html,
                                day_after_card_html=day_after_card_html)
        else:
            card1 = f"""<div class="md:p-7 p-4">
                                <h2 class=" text-xl text-center text-primary-green-dark capitalize">Add some data from scheduler</h2>
                            </div>"""
            return render_template('app/dashboard.html', name=user_name(session['user']).title(), card1=card1,
                                upcoming_count=0)
    else:
        return redirect('/login')
except KeyError:
    return redirect('/login')
@app.route('/schedule')
def schedule():
    try:
        if session['user']:
```

```
data = fetch_user_schedule(datetime.now(), session['user'])

now = str(datetime.now()).split('.')[0][-8:-3]

data['current_time'] = now

upcoming = {}

completed = {}

data = dict(sorted(data.items(), key=lambda item: item[1]))

checkpoint = False

for medicine in data:

    if medicine == 'current_time':

        checkpoint = True

    elif checkpoint:

        upcoming[medicine] = data[medicine]

    else:

        completed[medicine] = data[medicine]

upcoming_count = 0

upcoming_card_html = ""

for medicine in upcoming:

   upcoming_count += 1

    upcoming_card_html += card(medicine.title(), upcoming[medicine])

completed_count = 0

completed_card_html = ""

for medicine in completed:
```

```
            completed_count += 1
            completed_card_html += card(medicine.title(), completed[medicine])
        session['upcoming_count'] = upcoming_count
        return render_template('app/schedule.html', completed_card_html=completed_card_html,
                               upcoming_card_html=upcoming_card_html, upcoming_count=upcoming_count,
                               completed_count=completed_count)
    else:
        return redirect('/login')
  except KeyError:
    return redirect('/login')
@app.route('/schedule/add', methods=["GET", "POST"])
def add_schedule():
  if session['user']:
    try:
      upcoming_count = session['upcoming_count']
      return render_template('app/add-medicine.html', upcoming_count=upcoming_count)
    except KeyError:
      return render_template('app/add-medicine.html')
  else:
    return  redirect('/login')
@app.route('/schedule/submit', methods=['GET', "POST"])
def submit_schedule():
```

```
data = request.form
data = data.copy()
print(data)
dose_time = datetime.strptime(f"{data['hours']}:{data['minutes']} {data['halftime']}", '%I:%M %p')
schedule_data = {
    'medicine_name': data['medicine-name'],
    'dose_time': dose_time.strftime("%H:%M:%S"),
    'start_date': datetime.strptime(data['start-date'].replace('-', ''), '%Y%m%d'),
    'end_date': datetime.strptime(data['end-date'].replace('-', ''), '%Y%m%d')
}
if schedule_data['start_date'] < schedule_data['end_date']:
    data['medicine-name'] = None
    data['hours'] = None
    data['minutes'] = None
    data['halftime'] = None
    data['start-date'] = None
    data['end-date'] = None
    days = []
    for key in data:
        if data[key]:
            days.append(key)
    schedule_data['notification'] = days
```

```
        add_medicine(session['user'], schedule_data)
        return redirect('/schedule')
    else:
        flash("Start date can't be before end date")
        return redirect('/schedule/add')
@app.route('/prescription')
def prescription_view():
    if session['user']:
        data, count = fetch_prescriptions(session['user'])
        prescription_card_html = ''
        for index in range(len(data)):
            prescription_card_html += prescription_card(data[index]['name'], data[index]['link'])
        try:
            upcoming_count = session['upcoming_count']
            return render_template('app/prescriptions.html', upcoming_count=upcoming_count, total=count,
                                   prescription_card_html=prescription_card_html)
        except KeyError:
                                                    return render_template('app/prescriptions.html', total=count,
prescription_card_html=prescription_card_html)
    else:
        return redirect('/login')
@app.route('/prescription/add', methods=["GET", "POST",'PUT'])
def add_prescription_page():
```

```
if request.method == 'POST':
    try:
        data = request.form
        prescription_title = data.get('prescription-name')
        file = request.files['file']
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        path = f"{app.config['UPLOAD_FOLDER']}/{filename}"
        session['filename'] = path
        link = add_prescription(path, session['user'], filename)
        add_prescription_record(session['user'], prescription_title.title(), link)
        return redirect('/prescription')
    except:
        try:
            os.remove(session['filename'])
        except KeyError:
            pass
        return redirect('/prescription/add')
try:
    upcoming_count = session['upcoming_count']
    return render_template('app/add-prescription.html', upcoming_count=upcoming_count)
except KeyError:
```

```
        return render_template('app/add-prescription.html')
@app.route('/medicines')
def medicines():
    try:
        if session['user']:
            medicines = get_all_medicines(session['user'])
            medicine_card_html = ''
            total = 0
            for medicine in medicines:
                results = get_medicines(medicine)
                count = 0
                if results:
                    for index in range(len(results['names'])):
                        count += 1
                        if count < 3:
                            total += 1
                            try:
                                medicine_card_html += medicine_card(results['names'][index], results['prices'][index],
                                                                    results['order links'][index])
                            except IndexError:
                                pass
            try:
```

```
            upcoming_count = session['upcoming_count']
            return render_template('app/medicines.html', medicine_card_html=medicine_card_html, total=total,
                                   upcoming_count=upcoming_count)
        except:
            return render_template('app/medicines.html', medicine_card_html=medicine_card_html, total=total)
    else:
        return redirect('/login')
except KeyError:
    return redirect('/login')
@app.errorhandler(404)
def error(error):
    return '<h1>Error 404</h1>'
@app.errorhandler(500)
def error(error):
    return '<h1>Error 500</h1>'
@app.errorhandler(502)
def error(error):
    return '<h1>error502</h1>'
if__name__== '_main_':
    app.run(debug=True)
```

13.1 Project Demo Link

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-32156-1660208384

YOUTUBE LINK: https://youtu.be/t3ezk8m8RqA