

V.S. B ENGINEERING COLLEGE, KARUR

PERSONAL ASSISTANCE FOR SENIORS WHO ARE SELF RELIANT

PROJECT REPORT

TEAM ID: PNT2022TMID33538

Submitted by

KEERTHANA S	922519106075
MAHALAKSHMI R	922519106086
PRIYANKA M	922519106119
RAJESWARI K	922519106122

TABLE OF CONTENTS:

S.NO	TITLE
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 requirements
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
6.3	Reports from JIRA
7	CODING & SOLUTIONING
7.1	Feature 1
7.2	Feature 2

7.3	Database Schema
8	TESTING
8.1	Test Cases
8.2	User Acceptance Testing
9	RESULTS
9.1	Performance Metrics
10	ADVANTAGES & DISADVANTAGES
11	CONCLUSION
12	FUTURE SCOPE
13	APPENDIX

1. INTRODUCTION

Project Overview

- ✓ Sometimes elderly people forget to take their medicine at the correct time.
- ✓ They also forget which medicine He / She should take at that particular time.
- ✓ And it is difficult for doctors/caretakers to monitor the patients around the clock. To avoid this problem, this medicine reminder system is developed.
- ✓ An app is built for the user (caretaker) which enables him to set the desired time and medicine. These details will be stored in the IBM Cloudant DB.
- ✓ If the medicine time arrives the web application will send the medicine name to the IoT Device through the IBM IOT platform.
- ✓ The device will receive the medicine name and notify the user with voice command

Purpose

The Purpose of our Project is

- ✓ It allows the patients to record whether the medication was taken and at what time it was taken, providing a tool for the patients to better control their medication adherence.
- ✓ Medication reminders serve as a good way to stay on track and uphold an appropriate schedule.
- ✓ This is an Android-based application in which an automatic alarm ringing system is implemented.
- ✓ It focuses on doctor and patient interaction.

It helps remind the patient when to take the necessary medications, yet it also serves to dispense the medication time.

ABSTRACT

Elder people find it difficult to do their work independently without others help. They must be given opportunities to live their life reliably and remove their hesitation in asking or depending on others. IoT(Internet Of Things) is the greatest deal in today's life which helps people a lot to lead a happy and prosperous life. IoT plays a major role in minimizing human work and improving their lively ethics. This made us integrate IOT with the need of helping elderly people. This gives an idea to build devices that work to help the elderly people in remembering their tablet or medicine timings since many people find it difficult to remember their medicine timings and also people surrounding do not remember it all the time. This has been developed using IoT. The model has been designed with the device efficiency, low cost, and low power consumption aim. This may be designed with voice assistance or a small alarm that would pick up ringing automatically and remember the elderly people. The device will be compact and portable in various locations.

KEY WORDS

Remember tablet timings, IoT(Internet of Things), voice assistance, and reminder alarms.

INTRODUCTION

Nearly 40% of old age people suffer from memory impairment. They also have many health issues like diabetes, blood pressure, etc., [\[Elderly health\]](#) for which they are under continuous medication. This medicating phase is not an easy one. They depend on other people to help them. They do not find it easy also they hesitate to obtain help from others. This makes them skip the tablet hours. This risks their health and safety. Sometimes this leads to deaths even. This simple reason could not cause such big effects. Old age people must have cared a lot. This is eternally important. These kinds of things and their intense feeling that no one is there to care for them will cause lots of depression and stress. Thus as a motive to help the elderly, this project has been done to remind them of their tablet timings which would help them have a better old age.

LITERATURE SURVEY

- Author 1 says, Compared to their grandparents living in the 1970s, our more self-reliant and empowered American aging baby boomers will be better able to access the goods, services, and care they need to age in place autonomously in their current dwellings. The emergence of the Internet economy and the prospects of gerontechnological advances will only increase this

connectivity divide. However, one unintended consequence of increased connectivity may be a generation of more socially isolated older baby boomers.

- Author 2 says, An estimated 8.4 million adults with disabilities have children under age 18 living with them. Despite the large number of adults with disabilities engaged in parenting, studies of parents with disabilities have been relatively scarce, though the number is growing. This article reviews the literature on parents with disabilities as a whole and elaborates three themes relevant to parental support and care: parental capacity, “young careers,” and social networks. Also discussed are key concepts from the feminist and disability studies literature that can form the basis for a theoretical framework to guide research on parents with disabilities.
- Author 3 says, Developing self-reliance in community plays an important role to enhance the well-being and participation of its members. This study aimed to understand the conditions which contribute to the success of community well-being and the strategies to develop healthy communities. Qualitative research methodology was employed. The data collection strategy utilized in-depth interviews of thirteen key informants from the core group leaders and local residents of the community and from community development networks in Bangkok, Thailand. Data analysis was conducted through content analysis methods. The results showed that two conditions contributed to the success in community well-being. Firstly, it included human potential capital such as having outstanding leadership capabilities, for example, being patient, team oriented, well respected, and the potential of the elder leader’s core group in contributing their experience. Secondly it included the ability to foster community development; cultural capital, namely, having a traditional community with close family ties; and natural resource capital with some plants as an economic crop. In addition, the community need to collaborate with network partners. Those strategies in implementing community well-being operations included, 1) fostering community participation in a “healthy space”; 2) creating exposure to a wide range of organizational networks; 3) the implementation of a mentoring system; and 4) continuous development of the “healthy space” and opening it for public use, along with fostering cooperation with neighboring communities. Practical implications for sustainable development in the area of community well-being are discussed.

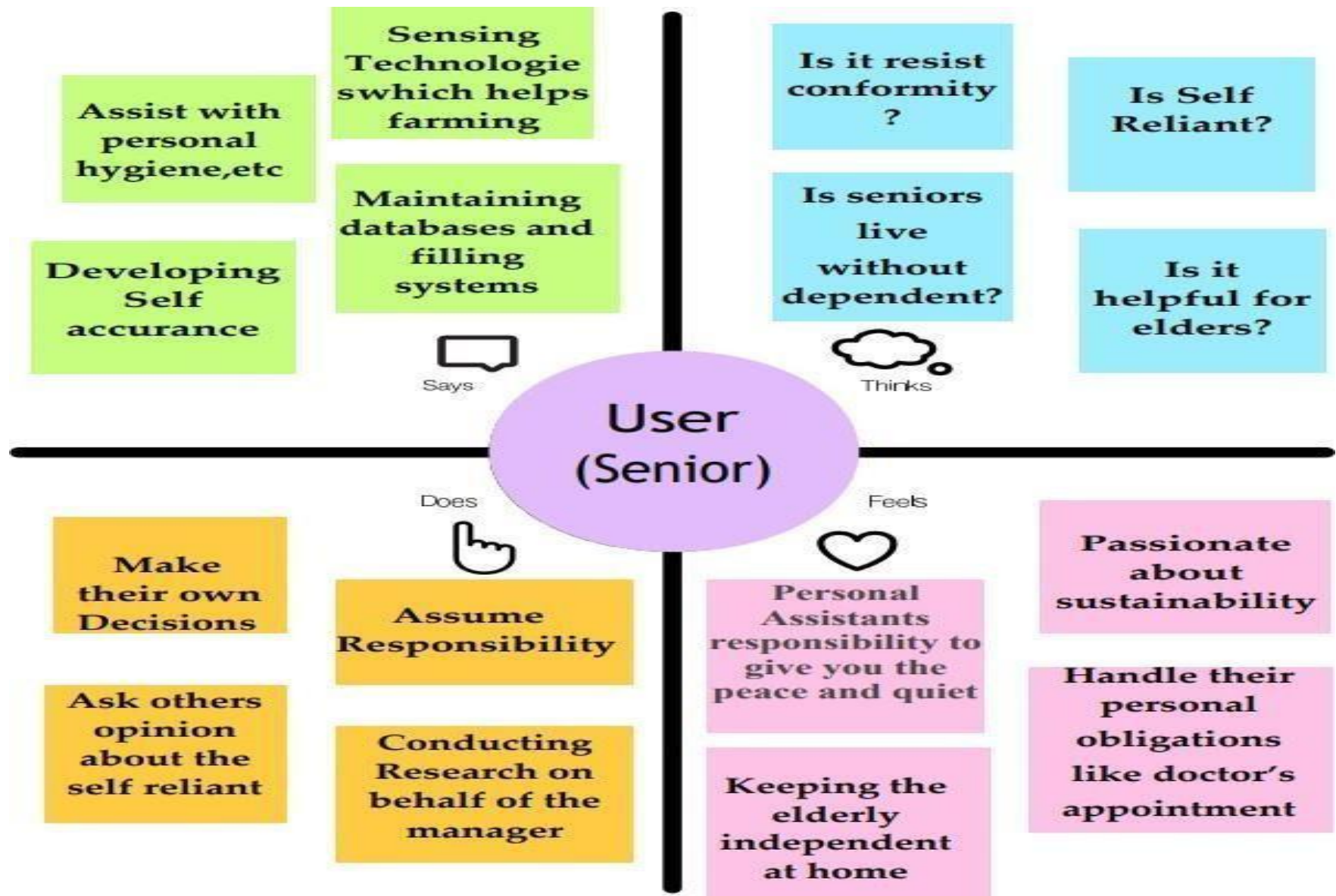
REFERENCE

1. Golant, S. M. (2017). Self-reliant older baby boomers are now better connected to goods, services, and care. *Generations*, 41(2), 79-87.
2. Rivera Drew, J. A. (2009). Disability and the self-reliant family: Revisiting the literature on parents with disabilities. *Marriage & family review*, 45(5), 431-447.
3. Thaithae, S., Chiangkhong, A., & Thanaboonpuang, P. (2021). Self-Reliant Community Development in a Semi-Urban Area of Bangkok: A Case Study of Community Well- Being. *The Journal of Behavioral Science*, 16(2), 114-126.

EMPATHY MAP

Problem Statement

A senior living in the family home **can upset the family dynamics**. Being a family caregiver is a demanding role. The added caregiving stress can take an emotional, physical, and mental toll on family members. Family members are torn between taking care of their own families and the senior.



IDEATION PHASE

PROBLEM STATEMENT

A senior living in the family home **can upset the family dynamics**. Being a family caregiver is a demanding role. The added caregiving stress can take an emotional, physical, and mental toll on family members. Family members are torn between taking care of their own families and the senior.

PROPOSED IDEAS

We find that working with older adults on their own goals while making small changes to the home environment is powerful medicine," says Szanton, who launched the program in Baltimore, MD and has since seen it piloted in Michigan among lower-income older adults on Medicaid and Medicare. Her study, Preliminary Data from Community Aging in Place, Advancing Better Living for Elders, a Patient-Directed, Team-Based Intervention to Improve Physical Function and Decrease.

Nursing Home Utilization: The First 100 Individuals to Complete a Centers for Medicare and Medicaid Services Innovation Project, appears in The Journal of the American Geriatrics Society.

“We provide pretty much whatever they need,” says Jennings. “If they need errands done, if they want to go shopping, we provide that. It just depends upon the individual”. For many, something as simple as a weekly trip to the mall can mean the difference between living on their own and having to move into an assisted living facility. It also greatly impacts their quality of life.

“I feel like when they live in their home, they still have that independence. They still have that desire to go on and live,” says Jennings. “I also think it’s very important that they are able to keep their pets, because a lot of times their pets are the only thing they have left. Household maintenance. Keeping a household running smoothly takes a lot of work. If you’re finding it hard to keep up, you can look into laundry, shopping, gardening, housekeeping, and handyman services. If you’re having trouble staying on top of bills and appointments, financial and healthcare management may also be helpful.

Transportation: Transportation is a key issue for older adults. Maybe you’re finding it hard to drive or don’t like to drive at night. Having access to trains, buses, rideshare apps, reduced fare taxis, and senior transportation services can help prolong your independence and maintain your social network.

Project Design Phase-I Proposed Solution**Proposed Solution**

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Elderly people forget to take their medicine at the correct time. They also forget which medicine He or She should take at that particular time and it is difficult for doctors and caretakers to monitor the patients around the clock.
2.	Idea / Solution description	An app is built for the user (care taker) which enables him to set the desired time and medicine .
3.	Novelty / Uniqueness	It serves as a good way to stay on track and uphold an appropriate schedule.
4.	Social Impact / Customer Satisfaction	Medication reminders help in decreasing medication dispensing errors and wrong dosages.
5.	Business Model (Revenue Model)	Medication reminder tool that uses an alarm cue to prompt users to take medication.
6.	Scalability of the Solution	Elderly people take their medicine correctly and the problem can be solved.

Project Design phase – IProblem Solution fitVSB Engineering College,karur-639111Project Design phase – IProblem Solution fit

Project name : Personal Assistance for seniors who are self reliant
 Team Id : PNT2022TMID33538

<u>1.Customer segments:-</u> <ul style="list-style-type: none"> Users who are not remind to take the medicine while they use the medicine remainder app to take the medicine at desired time. 	<u>6.Customer constrains:-</u> <ul style="list-style-type: none"> Constraint that customer will face about the safety of the device. 	<u>5.Available solutions</u> <ul style="list-style-type: none"> Built an app for the user which enables to set the desired time and medicine. We synced the calenders and added reminders according to the dosage.
<u>2.Jobs to be done :-</u> <ul style="list-style-type: none"> Help them continue their regular activities Encourage exercise Keep the brain busy Support their social life 	<u>9.Problem route cause:-</u> <ul style="list-style-type: none"> Communication problem Inadequate information flow Inadequate patient assessment 	<u>7.Behavior:-</u> <ul style="list-style-type: none"> Care about the patient and send the medicine name to lot devices at desired time .
<u>3.Triggers:-</u> <ul style="list-style-type: none"> Effective edication remainder program Back to your healthy self <u>4.Emotions:-</u> <ul style="list-style-type: none"> Immediate positive impact Touching,feeling,etc 	<u>10.Solution:-</u> <ul style="list-style-type: none"> We synced the calenders and added reminders according to the dosage. Built an app for the user which enables to set the desired time and medicine. 	<u>8.Channels of behavior:-</u> <ul style="list-style-type: none"> Online users know the usage of web application of medicine remainder and ask help from experts.

Project Design Phase-II

Solution Requirements (Functional & Non-functional)

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via EmailConfirmation via OTP
FR-3	Registered patient module	The doctor can view all the patients registered with him withall of his details.
FR-4	Appointment schedule module	The doctor can view the appointment schedule and can set new appointmentsaccordingly.
FR-5	Reply mode module	The reply mode module allows thepatient to ask some questions related to the prescribed pills, medicine schedule, and other queries.
FR-6	Database	Every patient has some necessary data like phone number, their first and last name,personal health number, postal code, country, address, city,'patient's ID number, etc.

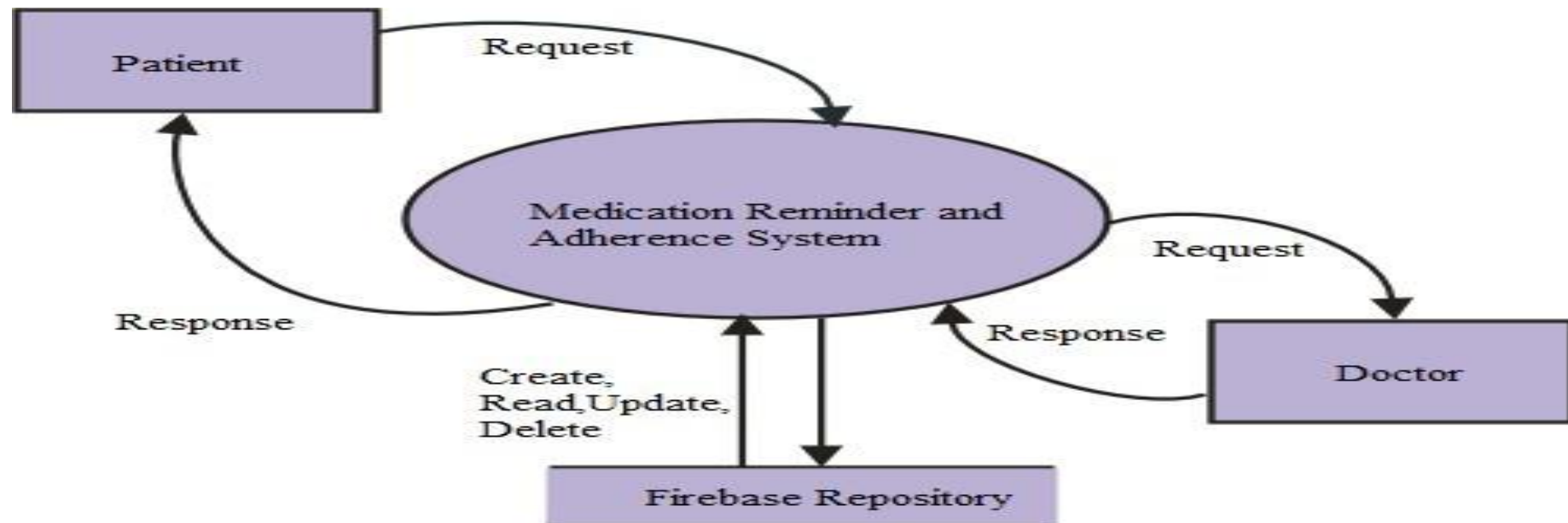
Non-functional Requirements

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	We aimed to examine the usability and feasibility of the medication reminder application in adults with diarrhoea-predominant.
NFR-2	Security	The system needs the patient to recognize herself or himself using the phone. Any users who make use of the system need to hold a Logon ID and password.
NFR-3	Reliability	Low-cost reminder devices did not improve adherence among nonadherent patients
NFR-4	Performance	The system provides acknowledgment in just one second once the patient's information is checked.
NFR-5	Availability	The medicine reminder app is available all the time.
NFR-6	Scalability	Elder people take their medicine correctly and the problem can be solved.

Project Design Phase-II Data Flow Diagram & User Stories

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. Android based medication reminder and adherence system is a system in which an alarm and notification system is implemented and also provides a platform for doctors, healthcare givers and patients' interaction. Patients can set a customized alarm tone in their local language or select from a list of default tones.



*FLOW DIAGRAM OF MEDICINE
REMINDER APP*

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, and password, and confirming my password.	I can access my account/dashboard	High	Sprint-1
		USN-2	As a user, I will receive a confirmation email once I have registered for the application	I can receive a confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering my email & password		High	Sprint-1
	Dashboard	USN-6	As a user, dashboards to quickly gain insights into the most important aspects of their data		High	Sprint -1
Customer (Webuser)	User Account	USN-1	As a user, Registration process, Password change or restoration after forgetting the password Customization of the user setting	I can access my account/dashboard	High	Sprint-1
Customer Care Executive	Reply to customer queries	USN-1	Maintaining various registers and files in department and those required for key quality indicators.		Medium	Sprint-2
Administrator	Feedback	USN-1	One of the recommendations to reduce medication errors and harm is to use.		Low	Sprint -2

Project Design Phase-I

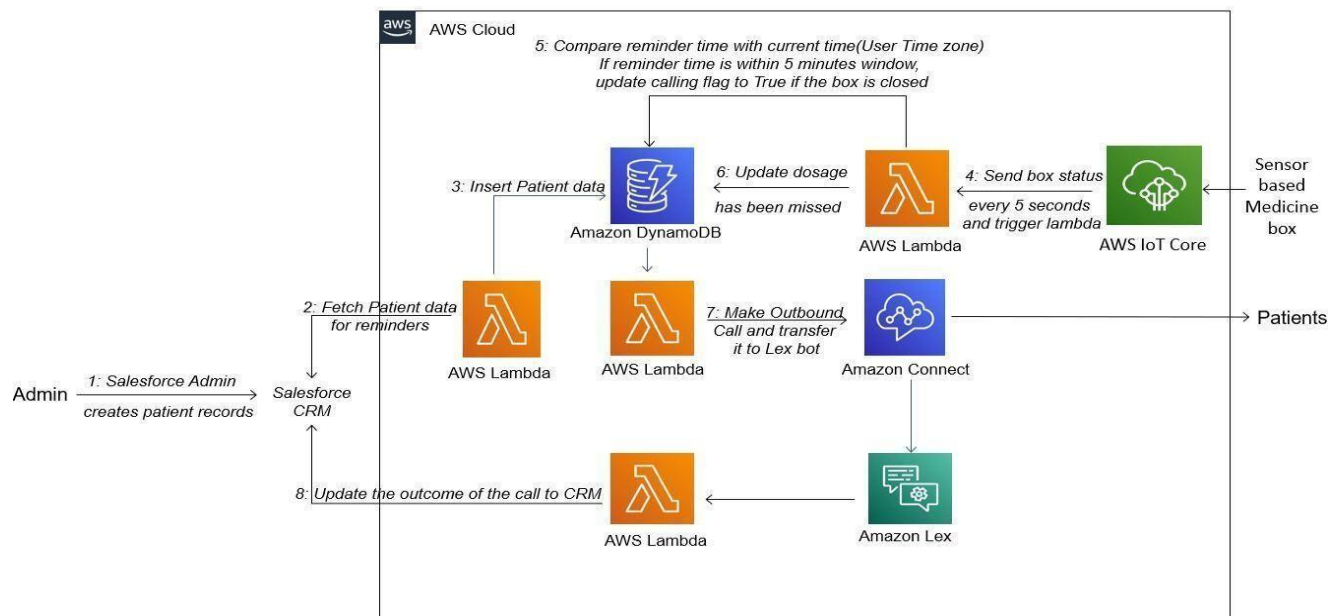
Solution Architecture

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

- Find the best tech solution to solve existing problems for seniors who are self-reliant.
- Device to help the elderly people in reminding their tablet or medicine timings that work based on IoT.
- The device comprises an alarm system to help people remember to take their medicines on time. It consists of timer units.

Solution Architecture Diagram

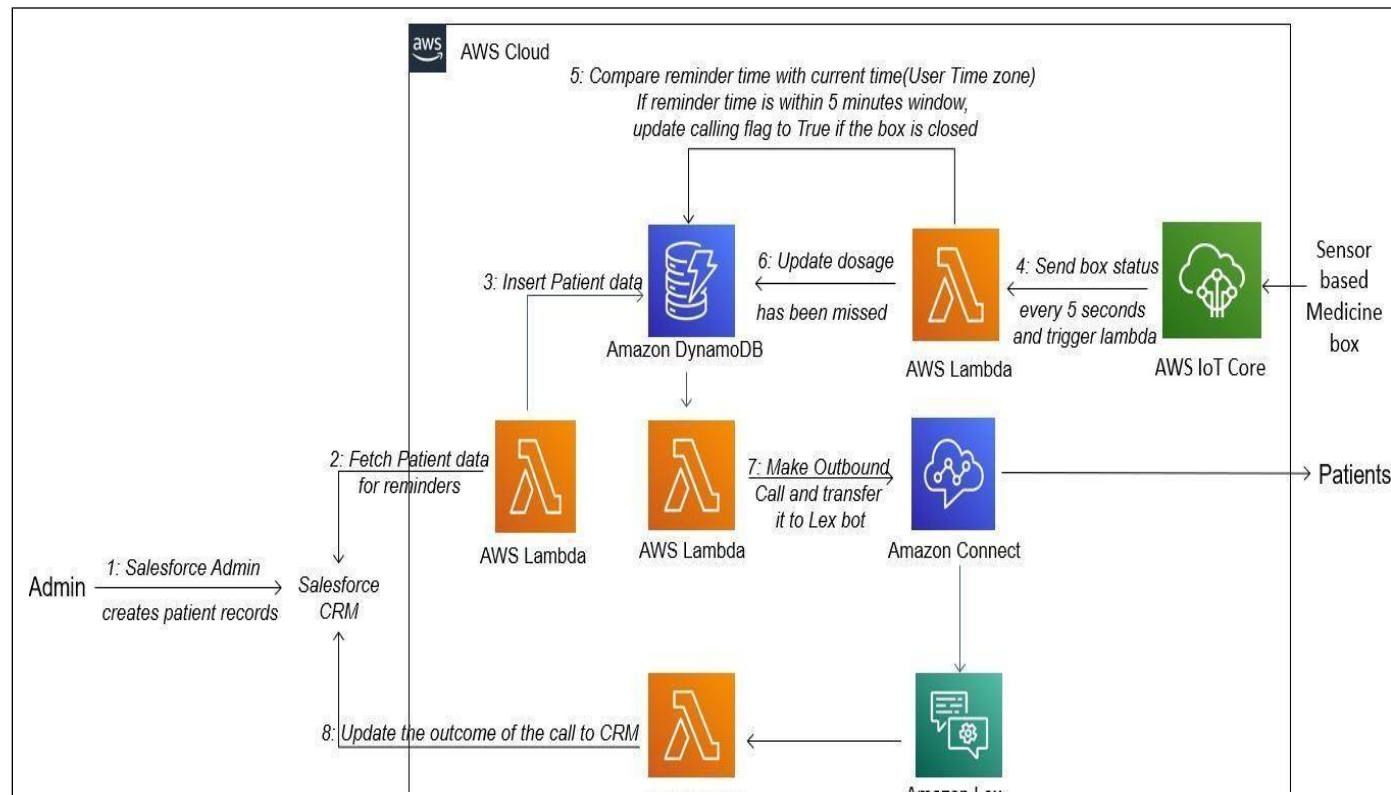
<https://aws.amazon.com/blogs/contact-center/build-a-drug-reminder-service-with-aws-iot-amazon-lex-and-amazon-connect>



Project Design Phase-II

Technology Stack (Architecture & Stack)

Technical Architecture:



1.	User Interface	Mobile App.	HTML, CSS, JavaScript
2.	Application Logic-1	Mobile App to enter the MedicineDetails weekly.	Python
3.	Application Logic-2	Gets the medication data from database.	IBM Watson IOT API call data.
4.	Application Logic-3	Converts the text to speech topronunciation for the user.	IBM Watson Assistant
5.	Database	Medication time and tablets name on daily .	MySQL.
6.	Cloud Database	Call the data IBM Cloud and is used and user login credentials.	IBM DB2, IBM Cloudant etc.
7.	File Storage	App code and IoT credentials arestored and API keys.	IBM Block Storage
8.	External API-1	To get the medicine box status Open or not.	IBM box status API
9.	External API-2	To get the login credentials in IBMDB2.	Username and Password
10.	Machine Learning Model	To convert the text into speech for voice command the tablet details.	Text to Speech
11.	Infrastructure (Server /Cloud)	To host the server and application.	Cloud Foundry, Node Red

Table-2: Application Characteristics

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	To develop the application interface, we use MIT App Inventor.	Technology of Opensource framework
2.	Security Implementations	To secure the user's login credentials and personal information.	SHA-256, OWASP, etc.
3.	Scalable Architecture	To scale the application database	IBM Autoscaling.
4.	Availability	To make use of the application and date area available 24/7	IBM Cloud load balancer
5.	Performance	To increase the performance the application is hosted in the high-performance instance.	IBM instance

References:

<https://aws.amazon.com/blogs/contact-center/build-a-drug-reminder-service-with-aws-iot-amazon-lex-and-amazon-connect/>

Design Phase-II Technology Stack (Architecture & Stack)

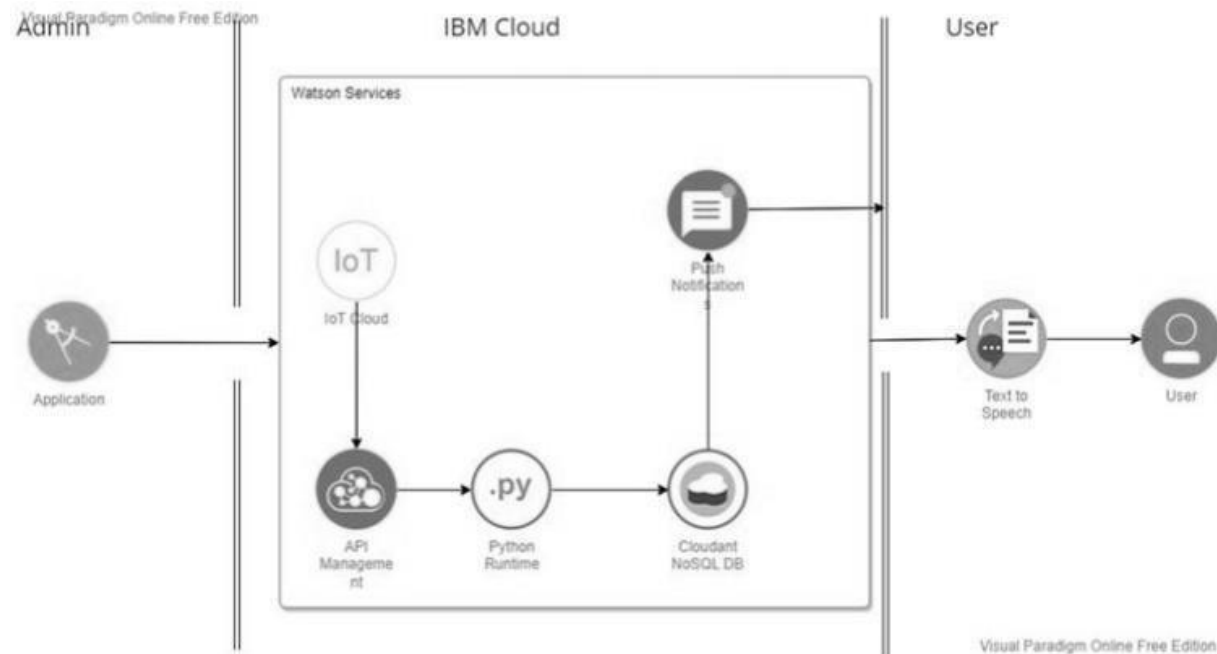


Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Mobile App	HTML, CSS, JavaScript
2.	Application Logic-1	Mobile App to enter the Medicine Details weekly	Python
3.	Application Logic-2	Gets the medication data from the database	IBM Watson IoT Calls data
4.	Application Logic-3	Converts the text-to-speech to pronunciation for the user	IBM Watson Assistant
5.	Database	Medication time and tablets name on daily and	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 API keys	IBM Block Storage
8.	External API-1	To get the medicine box status Open or not	IBM box status API
9.	External API-2	To get the login credentials in IBM DB2	Username and Password API
10.	Machine Learning Model	To convert the text into speech for voice command the tablet details	Text to speech
11.	Infrastructure (Server / Cloud)	To host the server and application	Cloud Foundry, Node Red

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	To develop the application interface, we use MITApp Inventor	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 24/7	IBM Cloud load balancer
5.	Performance	To increase the performance the application is hosted in the high-performance instance	IBM instance

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

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, password, and confirming my password.	2	High	S.Keerthana and R.Mahalakshmi
Sprint-2	Authorization	USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	K.Rajeswari and M.Priyanka
Sprint-3	User interface	USN-3	Using Mobile application it is easy to receive an alert when the medicine is missed to take and also give the correct medicines at correct time.	2	Medium	K.Rajeswari and M.Priyanka
Sprint-4	System Design	USN-4	Uses a cloud database to store medicinal reports. Connecting API to the cloud and mobile application. Connecting an IoT device to the cloud	2	High	S.Keerthana and R.Mahalaksmi

Project Tracker, Velocity & Burndown Chart:

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	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	15	4 Days	31 Oct 2022	05 Nov 2022	15	31 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	07 Nov 2022
Sprint-4	20	10 Days	14 Nov 2022	19 Nov 2022	20	14 Nov 2022

Velocity:

Imagine we have a 10-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).

Sprint 1

Average velocity = $20/5 = 0.4$

Sprint 2

Average velocity = $15/4 = 3.7$

Sprint 3

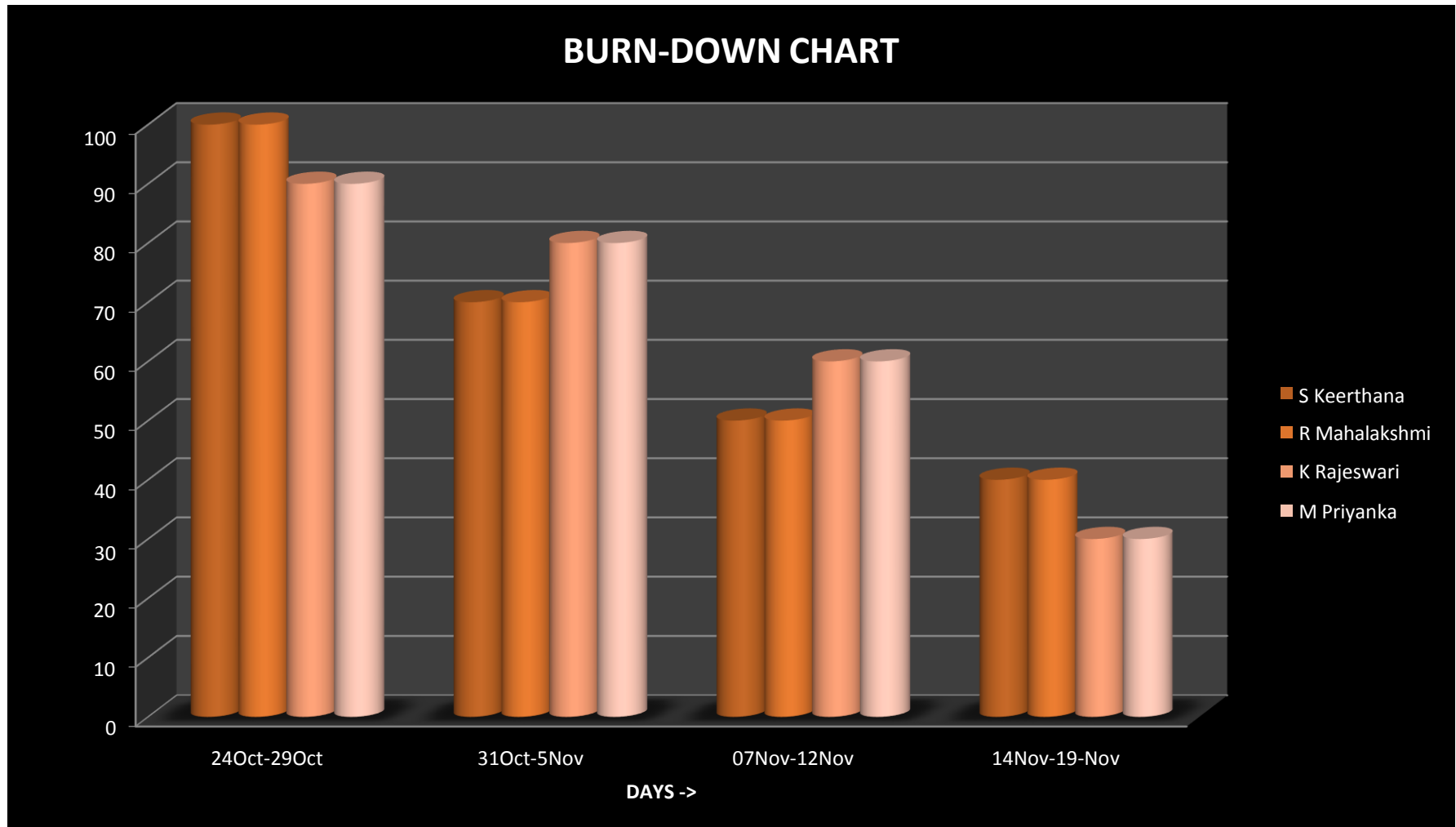
Average velocity = $20 / 6 = 3.3$

Sprint 4

Average velocity = $20 / 10 = 2$

Burn-down Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



Welcome to Project! Delighted to x IBM IBM-EPBL/IBM-Project-41585-16 IBM Watson IoT Platform

xf3l19.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform 922519106075@smartinternz.com ID: xf3l19

Browse Action Device Types Interfaces Add Device +

Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID Device Simulator

<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added
You don't have any devices.					

Type here to search

Raining now 5:36 PM 11/11/2022

IBM Watson IoT Platform

922519106075@smartinternz.com
ID: xf3l19

Browse Action Device Types Interfaces

Add Device +


Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator ☐

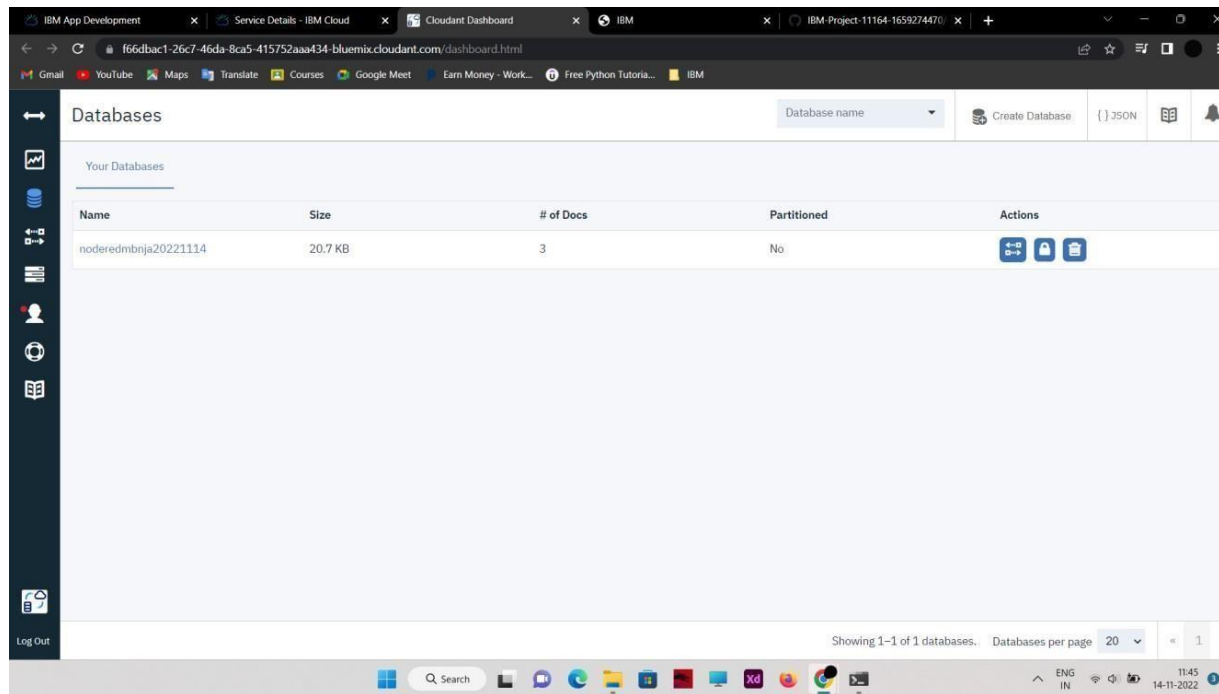
<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added
 <p>You don't have any devices.</p>					

Type here to search

Raining now 5:36 PM 11/11/2022

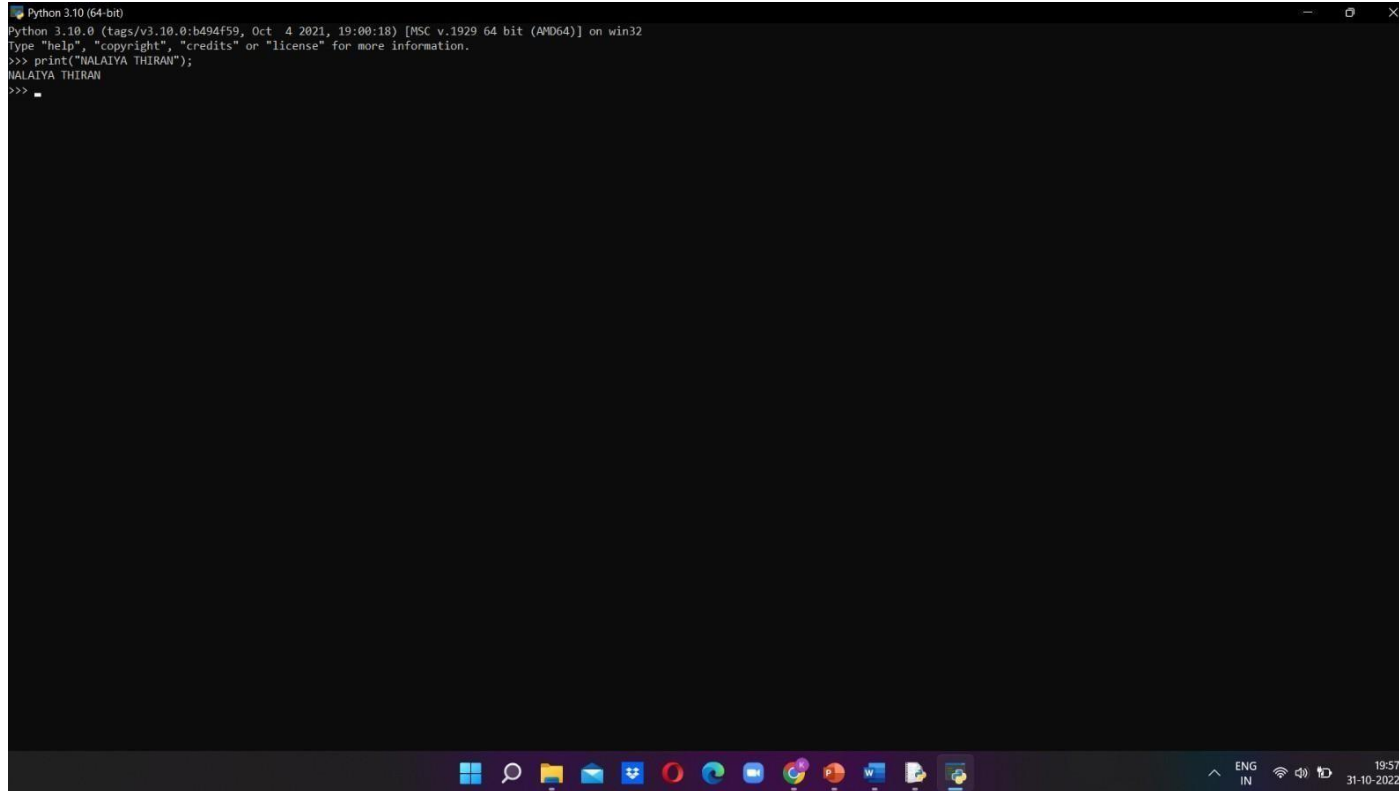
Create a database in cloudant DB

1. Open cloudant DB and launch the dashboard
2. Create the database



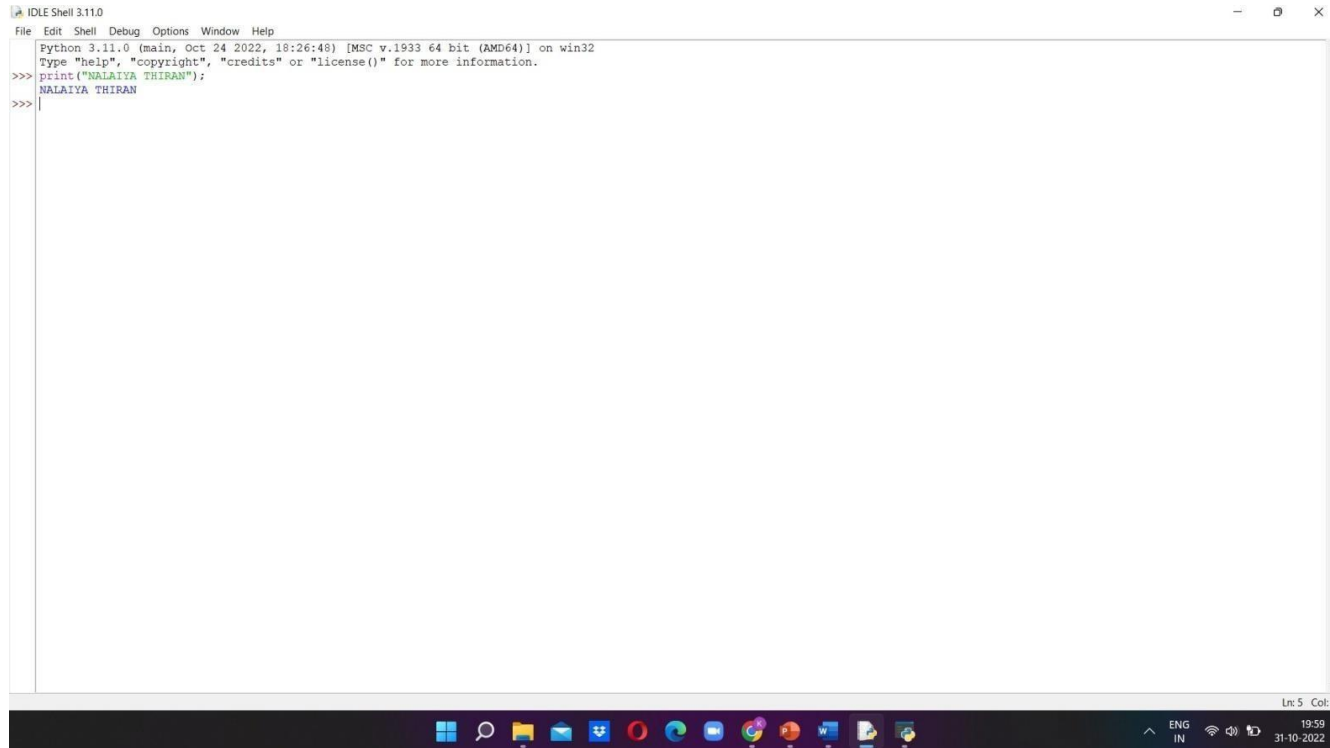
SOFTWARE

PYTHON 3.10 INSTALLATION:



```
Python 3.10 (64-bit)
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("NALAIYA THIRAN");
NALAIYA THIRAN
>>>
```


PYTHON IDLE:



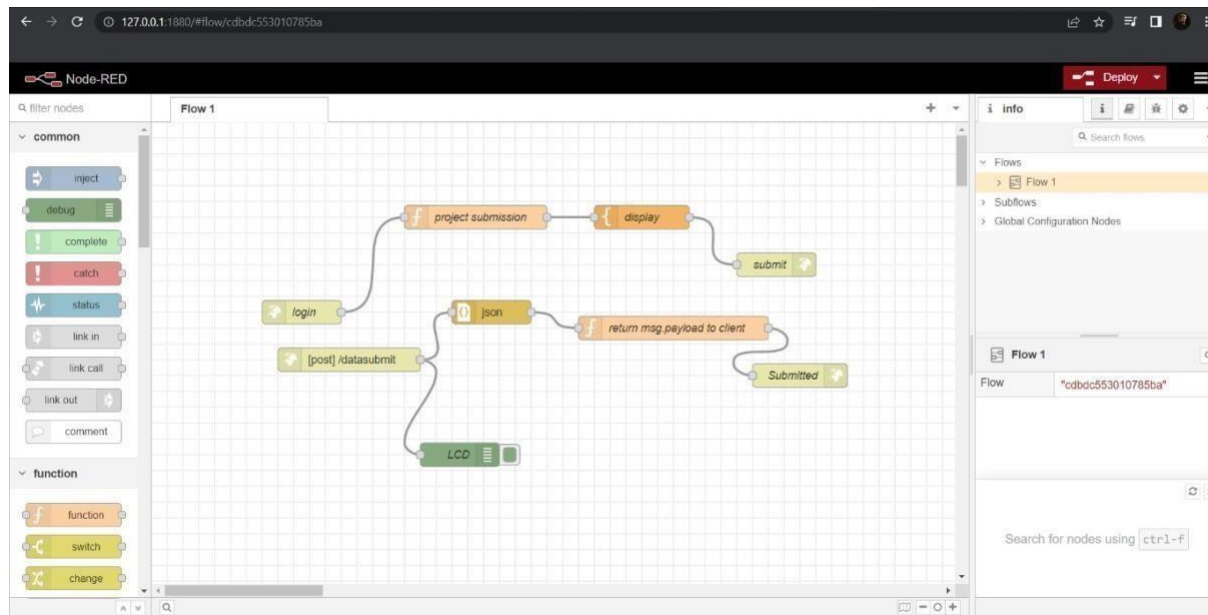
The screenshot shows the Python IDLE 3.11.0 Shell window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell displays the following text:

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("NALAIYA THIRAN");
NALAIYA THIRAN
>>>
```

The status bar at the bottom right indicates "Ln: 5 Col: 0". The Windows taskbar is visible at the bottom of the screen, showing various application icons and the system clock displaying 19:59 on 31-10-2022.

CREATE A FORM

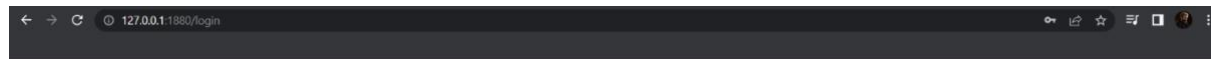
Connecting nodes using node-red:



After deploying the nodes:

The screenshot shows a web browser at the address 127.0.0.1:1880/login. The page displays a login form with the following elements:

- A heading 'LOGIN' in bold.
- A 'Username:' label followed by a text input field.
- A 'Password:' label followed by a text input field.
- A 'submit' button below the password field.



LOGIN

Username: Medicare

Password: *****

submit

After clicking on Submit button:



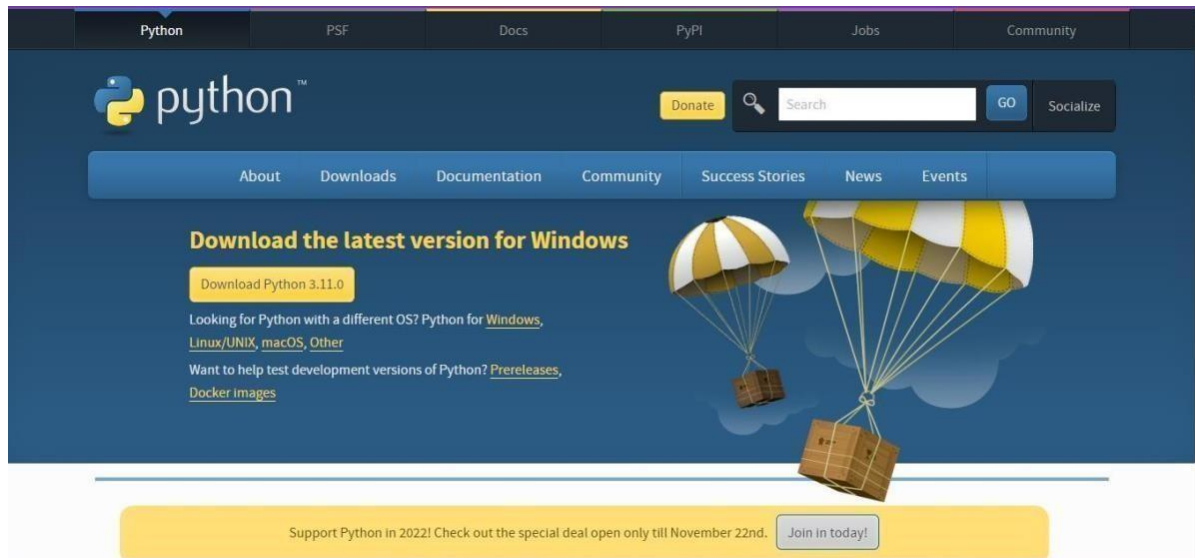
the following data was submitted {'username': 'Medicare', 'password': 'medicare'}

CREATION OF NODE-REDSERVICE

1. Node.js was downloaded and installed node-RED service.
2. An account was created in node-RED and features are viewed.



IBM PYTHON SCRIPT :



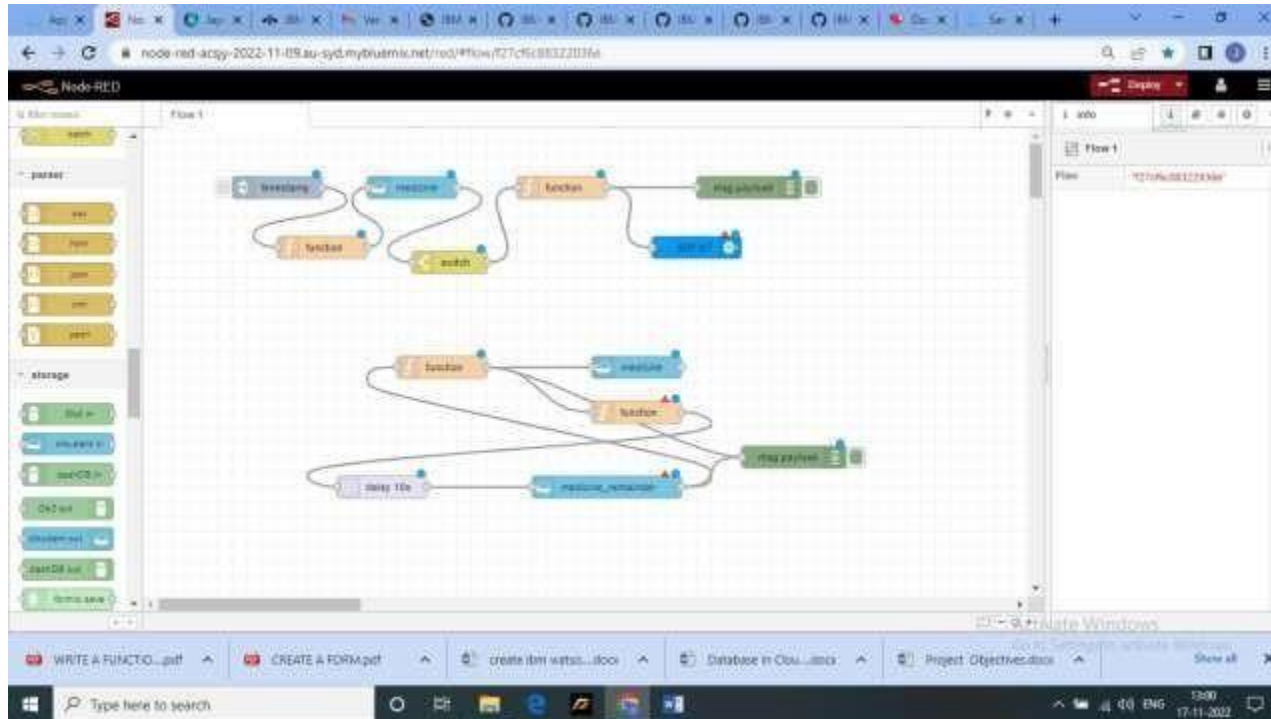
TEXT TO SPEECH SYNTHESIS

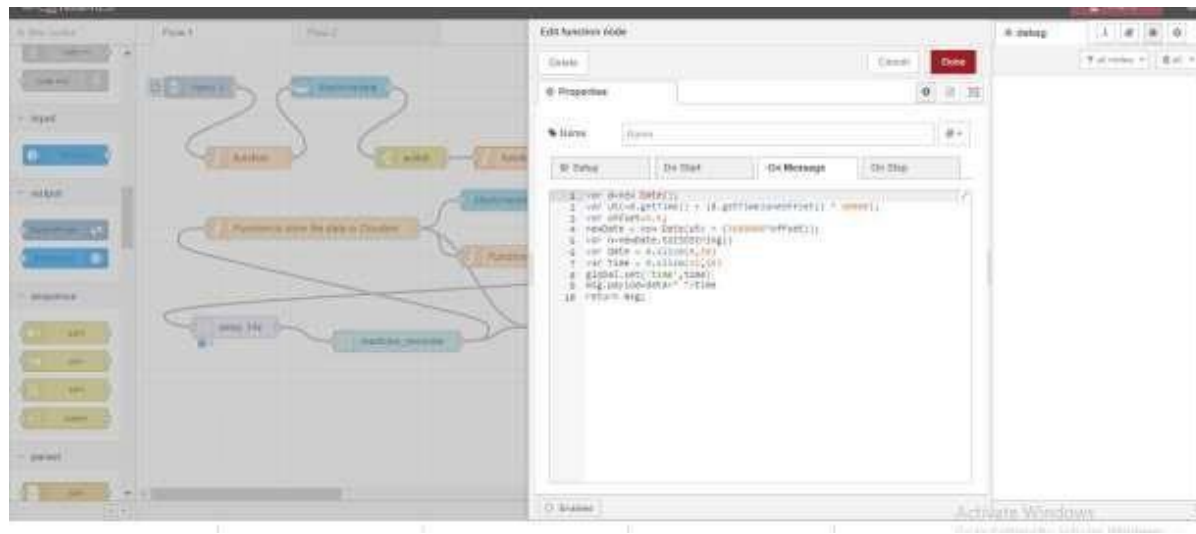
The screenshot shows the IBM Cloud console interface. At the top, there's a browser window with the URL `cloud.ibm.com/services/text-to-speech/crn%3Av1%3Abluemix%3Apublic%3Atext-to-speech%3Aau-syd%3Aa%2Ffac9c76ed3d44212b4c73e86f1befb48%3A0c90a2...`. The IBM Cloud header includes a search bar, navigation links (Catalog, Manage), and the user account 'KEERTHANA S's Account'. The main content area is titled 'Text to Speech-q0' and shows it is 'Active'. A sidebar on the left lists navigation options: Manage, Getting started, Service credentials (highlighted), Plan, and Connections. The 'Service credentials' section contains a description and a 'New credential' button. Below this is a table of existing credentials.

Key name	Date created
Auto-generated service credentials	2022-11-12 7:08 AM

The Windows taskbar at the bottom shows the system clock as 7:08 AM on 11/12/2022, with weather information (22°C Cloudy) and various system icons.

WRITE A FUNCTION TO COMPARE THE TIME





CODES AND SOLUTIONS

Urls.py :

```
urlpatterns = [  
    path("", views.college, name="college"), path("notice/<int:myid>/", views.notice,  
    name="notice"),  
    path("application_form/", views.application_form, name="application_form"), path("edit_application/",  
    views.edit_application, name="edit_application"), path("status/", views.status, name="status"),  
    # Authentication  
    path("register/", views.register, name="register"), path("login/", views.loggedin,  
    name="login"), path("logout/", views.loggedout, name="logout"),  
    # Admin  
    path("handle_admin/", views.handle_admin, name="handle_admin"), path("users/", views.users,  
    name="users"),  
    path("student_application/<int:myid>/", views.student_application, name="student_application"), path("application_status/<int:pk>/",  
    UpdatePostView.as_view(), name="application_status"), path("approved_applications/", views.approved_applications,  
    name="approved_applications"), path("pending_applications/", views.pending_applications, name="pending_applications"),  
    path("rejected_applications/", views.rejected_applications, name="rejected_applications"),  
]
```

Models.py

```
from django.db import models

from django.contrib.auth.models import User
from django.utils.timezone import now

from django.urls import reverse

class Application(models.Model):
    COURSES = (
        ('Computer Science Engineering', 'Computer Science Engineering'),
        ('Information Technology Engineering', 'Information Technology Engineering'),
        ('Electronics and Telecommunication Engineering', 'Electronics and Telecommunication Engineering'),
        ('Electronics Engineering', 'Electronics Engineering'),
    )

    STATUS = (
        ('Approved', 'Approved'),
        ('Pending', 'Pending'),
        ('Rejected', 'Rejected'),
    )

    user = models.OneToOneField(User, on_delete=models.CASCADE, blank=True, null=True)
    course = models.CharField(max_length=100, choices= COURSES)
    name = models.CharField(max_length=200)
    email = models.CharField(max_length=200)
    phone_no =
```

```
models.CharField(max_length=200)address = models.TextField(max_length=200)
student_profile = models.ImageField(upload_to="images")
ssc_percentage = models.DecimalField(max_digits=4, decimal_places=2, null=True)ssc_marksheet =
models.ImageField(upload_to="images", null=True)
```

```
ssc_passing_certificate = models.ImageField(upload_to="images", null=True) ssc_leaving_certificate =
models.ImageField(upload_to="images", null=True) hsc_percentage = models.DecimalField(max_digits=4,
decimal_places=2, null=True)hsc_marksheet = models.ImageField(upload_to="images", null=True)
hsc_passing_certificate = models.ImageField(upload_to="images", null=True) hsc_leaving_certificate =
models.ImageField(upload_to="images", null=True) cet_percentile = models.DecimalField(max_digits=5,
decimal_places=3, null=True) cet_scorecard = models.ImageField(upload_to="images", null=True)
jee_percentile = models.DecimalField(max_digits=5, decimal_places=3, null=True)jee_scorecard =
models.ImageField(upload_to="images", null=True)
```

```
Application_Status = models.TextField(max_length=100, choices=STATUS, default="Pending")message =
models.TextField(max_length=100, default="")
```

```
def str(self): return self.name
```

```
def get_absolute_url(self):return reverse('users')
```

```
class Notice(models.Model):
    title = models.CharField(max_length=200)

    def str(self): return self.title

class Detail(models.Model):
    title = models.ForeignKey(Notice, on_delete=models.CASCADE)notice =
    models.CharField(max_length=200)

    def str(self):
        return self.notice
```

1.For the home page, all the notice for different year students will be shown (college.html):

```
<div class="container mt-4">
    <h1>Important Notice</h1>
<div class="row mt-4">
    {% for i in notice %}
        <div class="col-sm-6">
            <div class="card">
                <div class="card-body">
                    <h5 class="card-title">{{ i.title }}</h5>
```

```
<p class="card-text"><a href="/notice/{{ i.id }}">View all recent updates.</a></p>
```

```
</div>
```

```
{% endfor %}
```

```
</div>
```

```
</div> Views.py :
```

```
def college(request)
```

```
notice = Notice.objects.all()
```

```
    return render(request, "college.html", {'notice':notice})
```

Code Explanation:

On the first page of the project all the notices will be displayed by using the for loop from the Notice model. Students can see the notice by clicking on the title regarding their year or branch

1 . A form is created for the user registration (register.html):

```
<form action="/register/" method="POST"> { % csrf_token % }  
  
  <div class="container mt-5">  
    <div class="mb-3">  
      <label for="username" class="form-label">Username</label>  
      <input type="text" class="form-control" id="username" name="username">  
    </div>  
    <div class="mb-3">  
      <label for="first_nam" class="form-label">First Name</label>  
      <input type="text" class="form-control" id="first_name" name="first_name">  
    </div>  
    <div class="mb-3">  
      <label for="last_name" class="form-label">Last Name</label>  
      <input type="text" class="form-control" id="last_name" name="last_name">  
    </div>  
    <div class="mb-3">  
      <label for="email" class="form-label">Email address</label>  
      <input type="email" class="form-control" id="email" name="email">  
    </div>  
    <div class="mb-3">
```

```
<label for="password1" class="form-label">Password</label>
<input type="password" class="form-control" id="password1" name="password1">
</div>
<div class="mb-3">
  <label for="password2" class="form-label">Confirm Password</label>
  <input type="password" class="form-control" id="password2" name="password2">
</div>
<button type="submit" class="btn btn-primary">Submit</button>
</div>
</form> Views.py :
```

```
def register(request):
    if request.method=="POST":
        username = request.POST['username'] email = request.POST['email']
        first_name=request.POST['first_name']
        last_name=request.POST['last_name'] password1 =
        request.POST['password1']password2 = request.POST['password2']

        if password1 != password2:
            messages.error(request, "Passwords do not match.")return redirect('/register')

        user = User.objects.create_user(username, email, password1)user.first_name = first_name
```

```
user.last_name = last_name
user.save()

return render(request, 'login.html')
return render(request,
"register.html")
```

Code Explanation:

We create a form with a post request to take the required details from the user. These are then stored in the Django User model. These details will be further required to identify the user while logging in.

1. Then a form is created for the user login (login.html):

```
<form action="/login/" method="POST"> {% csrf_token %}

<div class="container mt-5">

<div class="mb-3">

<label for="username" class="form-label">Username</label>

<input type="text" class="form-control" id="username" name="username">

</div>

<div class="mb-3">

<label for="password" class="form-label">Password</label>

<input type="password" class="form-control" id="password" name="password">

</div>

<br>

<button type="submit" class="btn btn-primary">Submit</button>

</div>

</form> Views.py :
```



```
def loggedin(request):
    if request.user.is_authenticated: return redirect("/")
    else:
        if request.method=="POST":
            username = request.POST['username'] password = request.POST['password']

            user = authenticate(username=username, password=password)

            if user is not None: login(request,
                user)messages.success(request, "Successfully Logged
                In")return redirect("/")
            else:
                messages.error(request, "Invalid Credentials")return render(request,
                'college.html')
    return render(request, "login.html")
```

Code Explanation:

If the user wants to fill the application form, then the user has to register and then login. While login if the username and password is wrong then “Invalid Credentials” message will be shown else the user will get successfully logged in.

After login the student can fill the application form and if the student has already filled the application form, then the student can view and edit the form. (application_form.html):

```
{% if hide.exists %}

<div class="container mt-4">

    <div class="row">

        <h1 style="text-align: center;">Personal Details</h1>

        <hr>

        <div class="col-md-4">

            <div class="profile-img">

                {% if user.application.student_profile.url %}

                {% endif %}

            </div>

        </div>

        <div class="col-md-8">

            <div class="profile-tab">

                <div class="tab-pane">

                    <br><br>

                    <div class="row">

                        <div class="col-md-6">

                            <label>Course :</label>

                        </div>

                        <div class="col-md-6">
```

```
<p>{{ user.application.course }}</p>
</div>
</div>
<div class="row">
  <div class="col-md-6">
<label>Full Name :</label>
    </div>
    <div class="col-md-6">
      <p>{{ user.application.name }}</p>
    </div>
  </div>
  <div class="row">
    <div class="col-md-6">
      <label>Email ID :</label>
    </div>
    <div class="col-md-6">
      <p>{{ user.email }}</p>
    </div>
  </div>
  {% if user.application.phone_no %}
  <div class="row">
    <div class="col-md-6">
```

```
<label>Phone Number :</label>

</div>

<div class="col-md-6">

  <p>{{ user.application.phone_no }}</p>

</div>

</div>

{% endif %}

<div class="row">

  <div class="col-md-6">

    <label>Address :</label>

    </div>

    <div class="col-md-6">

      <p>{{ user.application.address }}</p>

    </div>

  </div>

</div>

</div>

</div>

</div>

<hr>

<div class="container mt-4">
```

```
<div class="row">

<h1 style="text-align: center;">Educational Details</h1>

  <hr>

  <div class="col-md-4">

    <div class="profile-img">

      <h3>10th Std Details</h3>

    </div>

  </div>

  <div class="col-md-8">

    <div class="profile-tab">

      <div class="tab-pane">

        <br><br>

        <div class="row">

          <div class="col-md-6">

            <label>SSC Percentage :</label>

          </div>

          <div class="col-md-6">

            <p>{{user.application.ssc_percentage}}%</p>

          </div>

        </div>

      </div>

    </div>

  </div>

</div>
```

```
<div class="col-md-6">
```

```
  <label>SSC Marksheet :</label>
```

```
</div>
```

```
<div class="col-md-6">
```

```
  <p><a href="{{ user.application.ssc_marksheet.url }}">View SSC Marksheet</a></p>
```

```
</div>
```

```
</div>
```

```
<div class="row">
```

```
  <div class="col-md-6">
```

```
    <label>SSC Passing Certificate :</label>
```

```
  </div>
```

```
  <div class="col-md-6">
```

```
<p><a href="{{ user.application.ssc_passing_certificate.url }}"> View SSC Passing Certificate</a></p>
```

```
</div>
```

```
</div>
```

```
{% if user.application.phone_no %}
```

```
<div class="row">
```

```
  <div class="col-md-6">
```

```
    <label>SSC Leaving Certificate :</label>
```

```
  </div>
```

```
  <div class="col-md-6">
```

<div class="col-md-6">

```
<label>HSC Percentage :</label>

</div>

<div class="col-md-6">

  <p>{{ user.application.hsc_percentage }} %</p>

</div>

</div>

<div class="row">

  <div class="col-md-6">

    <label>HSC Marksheet :</label>

    </div>

    <div class="col-md-6">

      <p><a href="{{ user.application.hsc_marksheet.url }}">View HSC Marksheet</a></p>

    </div>

  </div>

  <div class="row">

    <div class="col-md-6">

      <label>HSC Passing Certificate :</label>

    </div>

    <div class="col-md-6">

      <p><a href="{{ user.application.hsc_passing_certificate.url }}">View HSCMarksheet</a></p>

    </div>

  </div>

</div>
```



```
</div>

{% if user.application.phone_no %}

<div class="row">

  <div class="col-md-6">

    <label>HSC Leaving Certificate :</label>

  </div>

  <div class="col-md-6">

    <p><a href="{{ user.application.hsc_leaving_certificate.url }}">View HSC LeavingCertificate</a></p>

  </div>

</div>

{% endif %}

</div>

</div>

</div>

<hr

iv class="container mt-4">

  <div class="row">

    <div class="col-md-4">

      <div class="profile-img">

        <h3>CET and JEE Scorecard</h3>
```

```
</div>
```

```
</div>
```

```
<div class="col-md-8">
```

```
<div class="profile-tab">
```

```
<div class="tab-pane">
```

```
<br><br>
```

```
<div class="row">
```

```
<div class="col-md-6">
```

```
<label>CET Percentile :</label>
```

```
</div>
```

```
<div class="col-md-6">
```

```
<p>{{user.application.cet_percentile}}</p>
```

```
</div>
```

```
</div>
```

```
<div class="row">
```

```
<div class="col-md-6">
```

```
<label>CET Scorecard :</label>
```

```
</div>
```

```
<div class="col-md-6">
```

```
<p><a href="{{user.application.cet_scorecard.url}}">View CET Scorecard</a></p>
```

```
</div>
```

```
</div>
```

```
<div class="row">
```

```
  <div class="col-md-6">
```

```
    <label>JEE Percentile :</label>
```

```
  </div>
```

```
<div class="col-md-6">
```

```
  <p>{{ user.application.jee_percentile }}</p>
```

```
</div>
```

```
</div>
```

```
{% if user.application.phone_no %}
```

```
<div class="row">
```

```
  <div class="col-md-6">
```

```
    <label>JEE Scorecard :</label>
```

```
  </div>
```

```
<div class="col-md-6">
```

```
  <p><a href="{{ user.application.jee_scorecard.url }}">View JEE Scorecard</a></p>
```

```
</div>
```

```
</div>
```

```
{% endif %}
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<hr>
```

```
<h4>Click here to edit the application</h4><a href="/edit_application/" class="btn btn-secondary" style="width: 20rem;">Edit Application</a>
```

```
<br><br>
```

```
{ % else % }
```

```
<div class="container mt-4">
```

```
    <form action="/admission_form/" method="POST" enctype="multipart/form-data"> { % csrf_token
% }
    { form.as_p }
```

```
    <button type="submit">Submit</button>
```

```
</form>
```

```
</div>
```

```
{ % endif % } Views.py :
```

```
def application_form(request):
```

```
    hide = Application.objects.filter(user=request.user)if request.method=="POST":
```

```
    form = ApplicationForm(request.POST, request.FILES)if form.is_valid():
```

```
        application = form.save() application.user = request.user
```

```
        application.save()
```

```

    return render(request, "application_form.html")else:
    form=ApplicationForm()
    return render(request, "application_form.html", { 'form':form,'hide':hide})Code Explanation:

```

After the student is logged in, he/she can fill the application form on python college admission system and upload all the original documents asked in the application form.

```
hide = Application.objects.filter(user=request.user)
```

We then check that the logged in students application already exists or not and save it in the “hide” variable. Then the “hide” variable is used in the templates as follows

```
{% if hide.exists %}
```

This means that the logged in student has already filled the application form so the whole application filled by the particular student will be displayed with an edit option. If this condition fails, that is the logged in student has not filled the application form, then the else statement will get executed, that is the entire form will be displayed on python college admission system.

1. Status of the application (status.html):

```
<div class="container w-50 mt-4 shadow">
```

```
<h3>Your application is {{ application.Application_Status }}</h3>
```

```
<br>
```

```
<h5>{{ application.message }}</h5>
```

```
</div> Views.py:
```

```
def status(request):
```

```
    application = Application.objects.get(user=request.user)
```

```
    return render(request, "status.html", {'application':application})
```

Code Explanation:

After successfully submitting the application form, the student can check the status of the form by navigating to the application status option given on the navigation bar. At first the status will be pending, as pending status is given as default in the models.py. After viewing the application form the admin can change the status of the application only. The admin can change the status from Pending to Approved or Rejected depending upon the student's application form. Lastly, the status will get displayed with a message if any

1.Admin's home page (handle_admin.html):

```
<div class="box">
```

```
    <h2><a href="/users/"> All Users ({{ users }}) </a></h2>
```

```
</div>
```

```
<div class="box">
```

```
    <h2><a href="/approved_applications/"> Approved Applications ({{ approve }}) </a></h2>
```

```
</div>
```

```
<div class="box">
```

```
    <h2><a href="/pending_applications/"> Pending Applications ({{ pending }}) </a></h2>
```

```
</div>
```

```
<div class="box">
```

```
    <h2><a href="/rejected_applications/"> Rejected Applications ({{ reject }}) </a></h2>
```

```
</div> Views.py:
```

```
def handle_admin(request):
    if not request.user.is_superuser: return redirect("/login")
    users = User.objects.all().count
    approve = Application.objects.filter(Application_Status='Approved').count
    reject = Application.objects.filter(Application_Status='Rejected').count
    pending = Application.objects.filter(Application_Status='Pending').count
    return render(request, "handle_admin.html", {'approve':approve, 'reject':reject, 'pending':pending, 'users':users })
```

Code Explanation:

On the admin's home page four options are created to view all the applications, the pending applications, the approved applications and the rejected applications respectively. The admin can basically check all the pending applications and then approve or reject it

```
approve = Application.objects.filter(Application_Status='Approved').count
```

The above line basically fetches the number of applications from the Application model filtering the application status to be approved. Similarly, the number of applications is stored for all users, pending and rejected applications.

1. Admin can view and change the status of the application (student_application.html):

```
<div class="container mt-4">
    <div class="row">
        <h1 style="text-align: center;">Personal Details</h1>
        <hr>
        <div class="col-md-4">
            <div class="profile-img">
```

```
{ % if user.application.student_profile.url % }  
  
{ % endif % }  
</div>  
</div>  
<div class="col-md-8">  
  <div class="profile-tab">  
    <div class="tab-pane">  
      <br><br>  
      <div class="row">  
        <div class="col-md-6">  
          <label>Course :</label>  
        </div>  
        <div class="col-md-6">  
          <p>{ { user.application.course } }</p>  
        </div>  
      </div>  
      <div class="row">  
        <div class="col-md-6">  
          <label>Full Name :</label>  
        </div>
```



```
<div class="col-md-6">
  <p>{{ user.application.name }}</p>
</div>
</div>
<div class="row">
  <div class="col-md-6">
    <label>Email ID :</label>
  </div>
  <div class="col-md-6">
    <p>{{ user.email }}</p>
  </div>
</div>
{% if user.application.phone_no %}
<div class="row">
  <div class="col-md-6">
    <label>Phone Number :</label>
  </div>
  <div class="col-md-6">
    <p>{{ user.application.phone_no }}</p>
  </div>
</div>
```

```
{% endif %}

<div class="row"
div class="col-md-6">
    <label>Address :</label>

</div>

<div class="col-md-6">
    <p>{{ user.application.address }}</p>
</div>
</div>
</div>
</div>
</div>
</div>

<div class="container mt-4">

    <div class="row">

<h1 style="text-align: center;">Educational Details</h1>
    <hr>
<div class="col-md-4">
    <div class="profile-img">
```

```
<h3>10th Std Details</h3>
```

```
</div>
```

```
</div>
```

```
<div class="col-md-8">
```

```
<div class="profile-tab">
```

```
<div class="tab-pane">
```

```
<br><br>
```

```
<div class="row">
```

```
<div class="col-md-6">
```

```
<label>SSC Percentage :</label>
```

```
/div>
```

```
<div class="col-md-6">
```

```
<p>{{ user.application.ssc_percentage }} %</p>
```

```
</div>
```

```
</div>
```

```
<div class="row">
```

```
<div class="col-md-6">
```

```
<label>SSC Marksheet :</label>
```

```
</div>
```

```
<div class="col-md-6">
```

```
<p><a href="{{ user.application.ssc_marksheet.url }}">View SSC Marksheet</a></p>
```

```
</div>

</div>

<div class="row">

  <div class="col-md-6">

    <label>SSC Passing Certificate :</label>

  </div>

  <div class="col-md-6">

    <p><a href="{ { user.application.ssc_passing_certificate.url } }"> View SSC PassingCertificate</a></p>

  </div>

</div>

{% if user.application.phone_no % }

<div class="row">

  <div class="col-md-6">

    <label>SSC Leaving Certificate :</label>

  </div>

  <div class="col-md-6">

    <p><a href="{ { user.application.ssc_leaving_certificate.url } }">View SSC LeavingCertificate</a></p>

  </div>

</div>

{% endif % }

</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<hr>
```

```
<div class="container mt-4">
```

```
    <div class="row">
```

```
        <div class="col-md-4">
```

```
            <div class="profile-img">
```

```
                <h3>12th Std Details</h3>
```

```
            </div>
```

```
        </div>
```

```
        <div class="col-md-8">
```

```
            <div class="profile-tab">
```

```
                <div class="tab-pane">
```

```
                    <br><br>
```

```
                    <div class="row">
```

```
                        <div class="col-md-6">
```

```
                            <label>HSC Percentage :</label>
```

```
                        </div>
```

```
                        <div class="col-md-6">
```

```
<p>{{ user.application.hsc_percentage }} %</p>
</div>
</div>
<div class="row">
  <div class="col-md-6">
    <label>HSC Marksheet :</label>
  </div>
  <div class="col-md-6">
    <p><a href="{{ user.application.hsc_marksheet.url }}">View HSC Marksheet</a></p>
  </div>
</div>
<div class="row">
  <div class="col-md-6">
    <label>HSC Passing Certificate :</label>
  </div>
  <div class="col-md-6">
    <p><a href="{{ user.application.hsc_passing_certificate.url }}">View HSCMarksheet</a></p>
  </div>
</div>
{ % if user.application.phone_no % }
<div class="row">
```

```
<div class="col-md-6">
    <label>HSC Leaving Certificate :</label>
</div>
<div class="col-md-6">
    <p><a href="{{ user.application.hsc_leaving_certificate.url }}">View HSC LeavingCertificate</a></p></div>
</div>
{% endif % }
</div>
</div>
</div>
</div>
<hr>

<div class="container mt-4">
    <div class="row">
        <div class="col-md-4">
            <div class="profile-img">
                <h3>CET and JEE Scorecard</h3>
            </div>
        </div>
        <div class="col-md-8">
            <div class="profile-tab">
```

```
<div class="tab-pane">
```

```
    <br><br>
```

```
    <div class="row">
```

```
        <div class="col-md-6">
```

```
            <label>CET Percentile :</label>
```

```
        </div>
```

```
        <div class="col-md-6">
```

```
            <p>{{ user.application.cet_percentile }}</p>
```

```
        </div>
```

```
    </div>
```

```
    <div class="row">
```

```
        <div class="col-md-6">
```

```
            <label>CET Scorecard :</label>
```

```
        </div>
```

```
        <div class="col-md-6">
```

```
            <p><a href="{{ user.application.cet_scorecard.url }}">View CET Scorecard</a></p>
```

```
        </div>
```

```
    </div>
```

```
    <div class="row">
```

```
        <div class="col-md-6">
```

```
            <label>JEE Percentile :</label>
```



```
</div>

<div class="col-md-6">

    <p>{{ user.application.jee_percentile }}</p>

</div>

</div>

{% if user.application.phone_no %}

<div class="row">

    <div class="col-md-6">

        <label>JEE Scorecard :</label>

    </div>

    <div class="col-md-6">

        <p><a href="{{ user.application.jee_scorecard.url }}">View JEE Scorecard</a></p>

    </div>

</div>

{% endif %}

</div>

</div>

</div>

</div>

<hr>

<h4>Click here to edit the status</h4><a href="/application_status/{{ application.id }}" class="btn btn-secondary" style="width:
20rem;">Edit Status</a>
```

Views.py

```
def student_application(request, myid):  
    if not request.user.is_superuser: return redirect("/login")  
    application = Application.objects.filter(id=myid)  
    return render(request, "student_application.html", {'application': application[0]})
```

Code Explanation:

After clicking on one of the 4 options from all students list, approved application list, pending application list and rejected application list the list will be displayed in the form of a table. On clicking the view application button the admin can view the particular student's application on python college admission system application. Then the admin can go through the entire application, check all the submitted documents and then change the status of the application.

Python Online College Admission System Output: College Admission Home without student logged in:

college admission home

Register:

python college registration

Application Form:
college admission application form

College Admission Admin Home page:

python college admin home

Summary

So here we come to an end of this project and we have successfully developed an online college admission management system in Django. With this project in Django, we have developed an efficient, time-saving and easy to use system for the hectic admission process.

Advantages

This project has the following advantages

- Helps seniors to get reminded of their medicine time easily.
- Lower expenses.
- Maintenance of the device is easy.
- Fewer mistakes.

Disadvantages

This project has the following disadvantages

- Accidental failures.
- All the users aren't aware of the technology.

Conclusion

This project provides a conclusion that the seniors can be benefitted highly using this personal assistance which makes to them have better health and longevity.

Future Scope

This project is expected to reach the maximum of elders and their medicine problems can be rectified drastically.

Appendix

A- Android- based application
B- -
C- Cloud, Cloudant DB
D- Data flow graph
E- Empathy map
F- Functional rrequirement
G- -
H- -
I- Ideation phase, IoT
J- Journey map
K- -
L- Literature survey
M- -
N- NodeRed service
O- -
P- Project Planning phase
Q- -
R- Requirements
S- Sprint
T- Technology architecture, Text to speech synthesis
U- User Stories
V- -
W- Watson platform
X- -
Y- -
Z- -