GOOGLE SUMMER OF CODE 2024

PROJECT PROPOSAL

Personal Details:

Name:	Farhan Ikbal
Organization Details:	University: Jadavpur University Course: B.E. in Computer Science and Engineering Current Semester: 6th Semester Expected Graduation Completion: May,2025
Contact Details:	Phone No: +918967463602 Email ID: farhanikbal07@gmail.com
Current Timezone:	Indian Standard Time (IST)

Project Proposal for Open HealthCare Network

Project No: 10
Project Title: WhatsApp bot for CARE

Project Overview:

Open HealthCare Network has been always been a frontline open-source organization actively taking part in sector of healthcare delivery and management. The open-source community has thus developed a healthcare service management system named **CARE**.

CARE is a online website platform that can be used by frontline medical officers and people in charge of hospital or medical facilities to keep track of the asset and availability of resources and also keep a record of the incoming patient flows in various healthcare facilities across the country. It also digitizes patient records and help underserved and backward citizens to gain access to TeleICU services through **CARE.**

There has been a website (care.coronasafe.in) made available for users (not patients) to gain access to the data of available resources and patient records through the various functionalities incorporated in it but there is no platform available for patients to access their patient records or personal information. In order to make an attempt to solve this, the idea for the deployment of WhatsApp bot becomes an easy solution as it can use the existing functionalities of the care backend and provide all the necessary information that can be made available to the WhatsApp user after proper authentication of credentials.

Project Features:

- Build a wrapper on care that IM bots can use to get information or send a notification. The idea behind this feature is to ease and improve the user experience.

- Build a WhatsApp bot which can be used by both patients and hospital staff

  For Patients

  1. Should fetch their patient records
  2. Should fetch their current medications
  3. Should fetch their procedures

     (More options can be added later.)

  For Users (Hospital Staff)

  1. Should fetch their schedules (to be done after scheduling)
  2. Option to fetch asset status
  3. Option to fetch inventory data

     (More options can be added later.)

- Features that are related for the above implementation:

  1. Proper Authentication for user. Check required to segregate hospital staff users from other user-types.

  2. Creation of an admin access separately for WhatsApp bot since there is no API available for directly accessing patient information by using patient credentials. Other solution to this might be a new API creation that can provide us information from the **Serializer** embedded in the backend without compromising confidentiality of the patients.

  3. Handling of edge cases while searching for information from the database in case of API calls, thus creating new filters if required.

- Features I found to be intriguing:

  1. **Documentation Excellence:**
     The project boasts well-maintained documentation, providing comprehensive guidance for developers at every stage.

  2. **Seamless UI Experience:**
     Users enjoy a smooth and intuitive interface, enhancing their interaction with the system and ensuring a positive user experience.

3. **Effortless Local Setup:**
   Setting up both care and care_fe locally was a breeze, streamlining the onboarding process for new contributors and developers.

4. **Logical and Modular Structure:**
   The codebase is organized with a clear and modular structure, facilitating easy comprehension and maintenance for all team members.

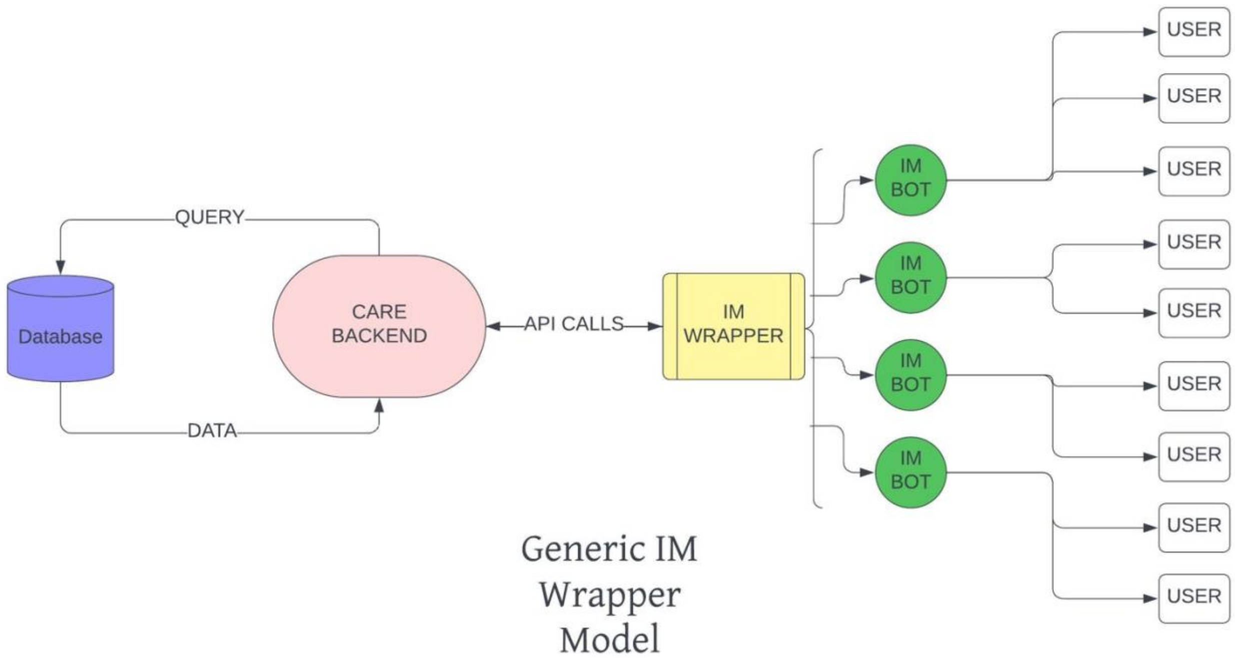
5. **Simple API Integration:**
   The project offers straightforward API integration capabilities, allowing seamless interaction with external services and platforms.

Flowchart diagrams:

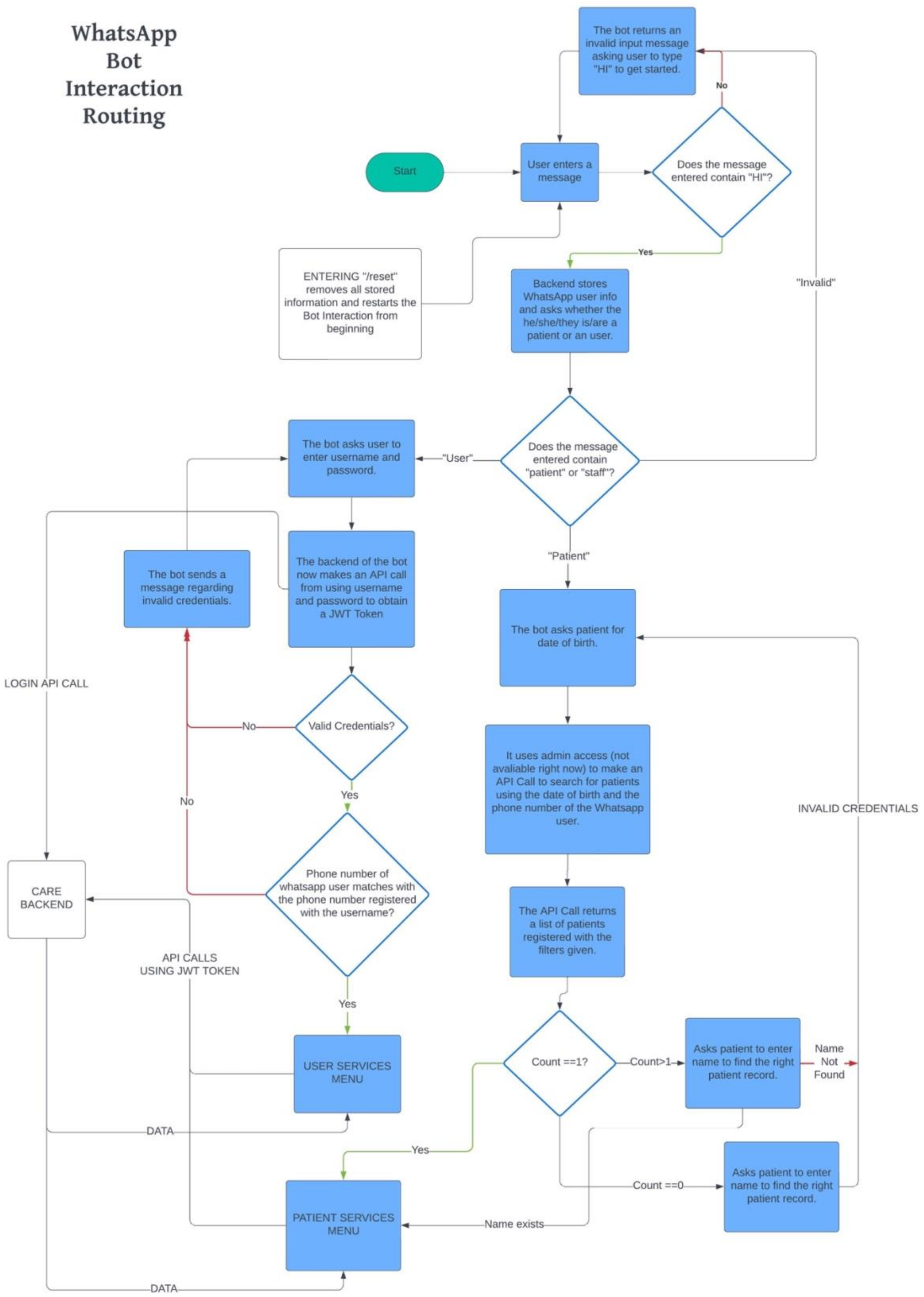
Two major tasks are there:

- Building a wrapper for IM bots
- Building the backend for the WhatsApp Bot

{PLEASE READ IMPORTANT CLARIFICATION ON PAGE 16}



Generic IM Wrapper Model


WhatsApp Bot Interaction Routing
Start
User enters a message
Does the message entered contain "HI"?
No
The bot returns an invalid input message asking user to type "HI" to get started.
Yes
Backend stores WhatsApp user info and asks whether the he/she/they is/are a patient or an user.
ENTERING "/reset" removes all stored information and restarts the Bot Interaction from beginning
"Invalid"
Does the message entered contain "patient" or "staff"?
"User"
The bot asks user to enter username and password.
The backend of the bot now makes an API call from using username and password to obtain a JWT Token
The bot sends a message regarding invalid credentials.
Valid Credentials?
No
Yes
No
Phone number of whatsapp user matches with the phone number registered with the username?
Yes
USER SERVICES MENU
LOGIN API CALL
CARE BACKEND
API CALLS USING JWT TOKEN
DATA
"Patient"
The bot asks patient for date of birth.
It uses admin access (not avaliable right now) to make an API Call to search for patients using the date of birth and the phone number of the Whatsapp user.
The API Call returns a list of patients registered with the filters given.
Count ==1?
Count>1
Asks patient to enter name to find the right patient record.
Name Not Found
INVALID CREDENTIALS
Count ==0
Asks patient to enter name to find the right patient record.
Yes
PATIENT SERVICES MENU
Name exists
DATA

Technical Skills and Relevant Experience:

Programming Languages I have used until now:

- Python
- Java
- Kotlin
- C
- C++
- JavaScript
- TypeScript

List of Technologies I have used in Course Work and Projects:

1. Web Development:
    - HTML
    - CSS
    - Node.js
    - Express.js
    - React.js
    - Django
    - SQL
    - MongoDB
    - Python
    - Json

2. Android Development
    - Kotlin

- Gradle
- Android Studio

3. Deep Learning
   - Image Classification using Tensorflow
   - Object Detection using Tensorflow, Pytorch and YOLO.

4. Relevant Projects:
   - Chat Message Web Application built using React and implementing Websocket with functionalities of Private and Public Groups.
   - Student Repository System built using Node.js and Express.js.

Implementation Skills and Milestones:

- Timeline:

  I would like to discuss that I have invested a considerable amount of time delving into the technicalities of your repository and comprehending the underlying codebase for CARE backend. In addition to studying the code, I have actively utilized the provided APIs, exploring their capabilities and assessing their integration within the project context. Furthermore, I have forked and git cloned the backend and deployed my own dummy server to prototype a WhatsApp bot, enabling me to experiment with various features and evaluate their feasibility. The WhatsApp bot after verifying whether a staff or a patient, provides a list of services accordingly after authentication like fetching Patient records: {Personal information, Current medical status, Medical history, Facility details, etc.}, medications, procedures. This is done by calling necessary APIs required to fetch information based on filters and then provide information to the WhatsApp user as per his/her/their requirement.

Week by Week Scheduling:

- Week 1:
  - Goals:
    - Define the scope and requirements of the WhatsApp bot for patients and hospital staff.
    - Set up the development environment and necessary tools.

  - Tasks:
    - Discuss with the mentor to finalize the scope and requirements.

        - Create a detailed document outlining the features and functionality of the WhatsApp bot.
        - Set up development environments, including version control systems and project management tools.

    - Deliverables:
        - Requirements document for the WhatsApp bot.
        - Setup of development environments.

- Week 2:
    - Goals:
        - Build the generic wrapper that any IM bot can use.
        - Setup the WhatsApp Business API for usage.

    - Tasks:
        - Find the formatting of how various IM bots interact with the backend.
        - Create the interface as generic as possible for later usage of different APIs to interact with the wrapper.
        - Build functions for WhatsApp API for message passing.

    - Deliverables:
        - Generic IM Wrapper.
        - Initial setup of WhatsApp Bot.

- Week 3:
    - Goals:
        - Implement authentication mechanisms for users(hospital staff).
        - Development of backend services for hospital staff data.

    - Tasks:
        - Implement user authentication functionality using secure authentication protocols.

    - Complete backend services to fetch patient records, medications, and procedures.
    - Share progress with the mentor and get feedback.

  - Deliverables:
    - Basic user authentication system.
    - Initial backend services for fetching staff data.

- Week 4:
  - Goals:
    - Implement features for fetching schedules, asset status, and inventory data.
    - Test the already implemented functions regarding user services.
  - Tasks:
    - Develop backend services for fetching hospital staff schedules, asset status, and inventory data through API Calls.
    - Integrate these new features into the WhatsApp bot.
    - Conduct initial testing of authentication and newly implemented features.

  - Deliverables:
    - Implemented features for fetching user based necessary information.
    - Round 1 of testing and bug fixing.

- Week 5:
  - Goals:
    - Implement Admin Access for WhatsApp API.
    - Investigate options for securely accessing patient information without compromising confidentiality.

  - Tasks:
    - Develop and implement admin access features for the WhatsApp bot.

        - Decide on the best approach for accessing patient information securely.
        - Seek mentor's guidance on implementing admin access and secure data retrieval.

    - Deliverables:
        - Implemented admin access for the WhatsApp bot.
        - Secure access solution for patient information.

- Week 6:
    - Goals:
        - Implement features for fetching patient records, medications, and procedures.
        - Test the already implemented functions regarding patient services.

    - Tasks:
        - Develop backend services for fetching patient records, medications and procedures through API Calls.
        - Integrate these new features into the WhatsApp bot.
        - Conduct testing of admin access and newly implemented features.

    - Deliverables:
        - Implemented features for fetching patient necessary information.
        - Round 2 of testing and bug fixing.

- Week 7:
    - Goals:
        - Finalize documentation and prepare for deployment.

    - Tasks:
        - Create user manuals and technical documentation for the WhatsApp bot.

    - Prepare deployment scripts and procedures.

  - Deliverables:
    - Implemented features for fetching patient necessary information.
    - Round 2 of testing and bug fixing.

- Week 8:
  - Goals:
    - Deploy the WhatsApp bot to production.
    - Conduct final checks and handover to the mentor.

  - Tasks:
    - Deploy the WhatsApp bot to the production environment.
    - Perform final checks to ensure everything is functioning correctly.
    - Merge Pull Request.

  - Deliverables:
    - Resolved any final testing issues.
    - Deployed WhatsApp bot in the production environment.

Important Clarification:

The GSOC idea for WhatsApp bot talks about a Generic IM Wrapper that can handle any kind of IM bot API and since it does **API Calls**, it needs to be **hosted separately**. The advantage for this design is that the functions in the code are more readable and easily debuggable in case of any kind of error and newer methods and application of newer filters is very modular and easy to implement.

However, if the necessary requirement is to properly integrate the entire wrapper file in the care backend, I can also do that. It is tedious, a bit time taking but doable. I need to read the entire codebase for understanding how to use **Serializers** and extract information based on what is asked. I need to use the pre-existing **functions** instead of **API Calls** in the information extraction functions I have written for the dummy bot. In that case, the **Week by Week scheduling** gets modified a bit. All the API Calls that I have mentioned gets replaced by appropriate queryset and validate functions and I successfully integrate the wrapper into the backend by the end of Week 4.

Summary About Me:

Hi Mentors, I am Farhan Ikbal, currently pursuing B.E. in Computer Science and Engineering at Jadavpur University, Kolkata. I am completed 5 semesters with the 6th semester getting completed by mid-May. I am an active core-member of the Jadavpur University Code Club which takes part in organizing seminars, tutorial classes, hackathons, contests, etc. I have also gained knowledge about Machine Learning and Deep Learning under the guidance of one of our professors. A week ago, I and two more members of Code Club organized a reverse engineering contest named “UnCode” at our annual techno management fest “Srijan”. I also invest time in solving coding problems from various online platforms and try to solve it in the most minimalist way possible.

Talking about the project, I was eagerly looking for contributing into open source since my first year of College but I found my skill-set not up to the mark to contribute to the repositories that might be valuable. Last year, I came to know about GSOC and understood what it offers. So, it was last year that I got information about GSOC but it was too late to apply. This year, I was prepared to apply for a single proposal to a single organization that fulfils my requirement of contribution for a greater cause. Open HealthCare Network is a great match for me as it works in the sector of medical facilities to the underserved patients. Making the WhatsApp bot will be a breakthrough as it brings information to the users at their own hands with simple authentications and choice selection. Easy to use and operate, WhatsApp will be the new information notebook for both staff and patients. I believe I have all necessary requirements fulfilled to contribute and be selected for this project idea.

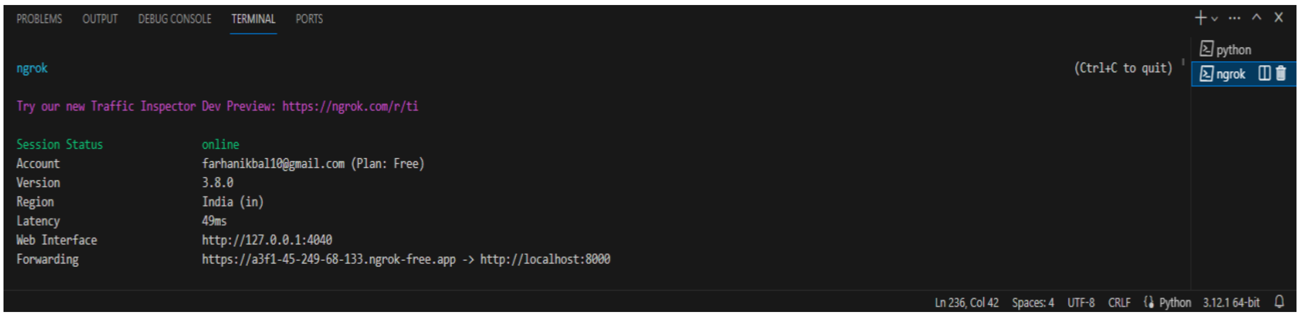
Availability and Commitment

I have summer holiday break from mid-May after my year end semesters end. So, I am completely available from that time (The exam schedule hasn't been yet declared so I cannot give a specific date.). I have no other commitment during my entire 2-month summer break ranging up to mid-July. I am only looking forward to work with OHCN if I get the chance.

I can easily devote 5-7 hours of work every day for the sake of this project and at any time, I am available to attend calls, meetings or give reports through telecommunication or video communication from 9 AM IST or 3:30 AM UTC (morning) to 1 AM IST or 7:30 PM UTC (night). When my college reopens in August, I can still work for 5 hours a day beside my daily routine for the project idea mentioned above.

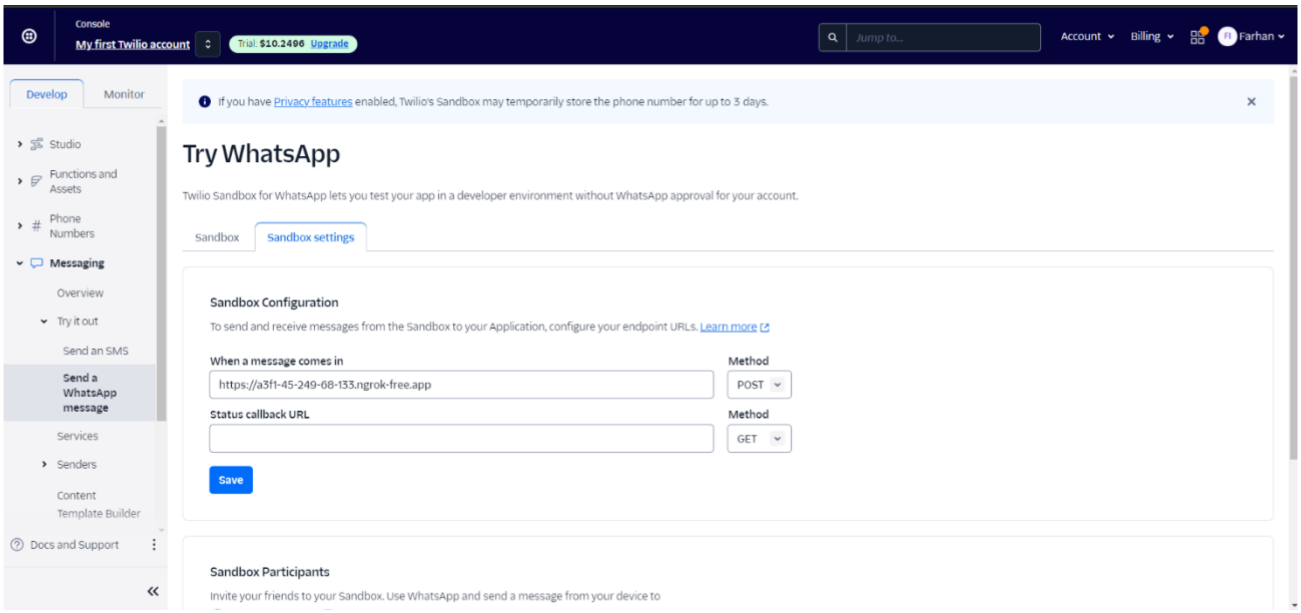
Background Work :

In order to setup a dummy WhatsApp bot for Open HealthCare Network, I used Twilio API for sending and receiving messages and hosted my dummy server through a http tunnel using ngrok and making sure that whenever a message is sent to the bot over WhatsApp, it makes a POST method call to the server that has temporarily hosted my dummy server through the tunnel.



```
ngrok                                                                   (Ctrl+C to quit)

Try our new Traffic Inspector Dev Preview: https://ngrok.com/r/ti

Session Status                online
Account                       farhanikbal10@gmail.com (Plan: Free)
Version                       3.8.0
Region                        India (in)
Latency                       49ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://a3f1-45-249-68-133.ngrok-free.app -> http://localhost:8000
```

ngrok



If you have Privacy features enabled, Twilio's Sandbox may temporarily store the phone number for up to 3 days.

Try WhatsApp

Twilio Sandbox for WhatsApp lets you test your app in a developer environment without WhatsApp approval for your account.

Sandbox Configuration

To send and receive messages from the Sandbox to your Application, configure your endpoint URLs. Learn more

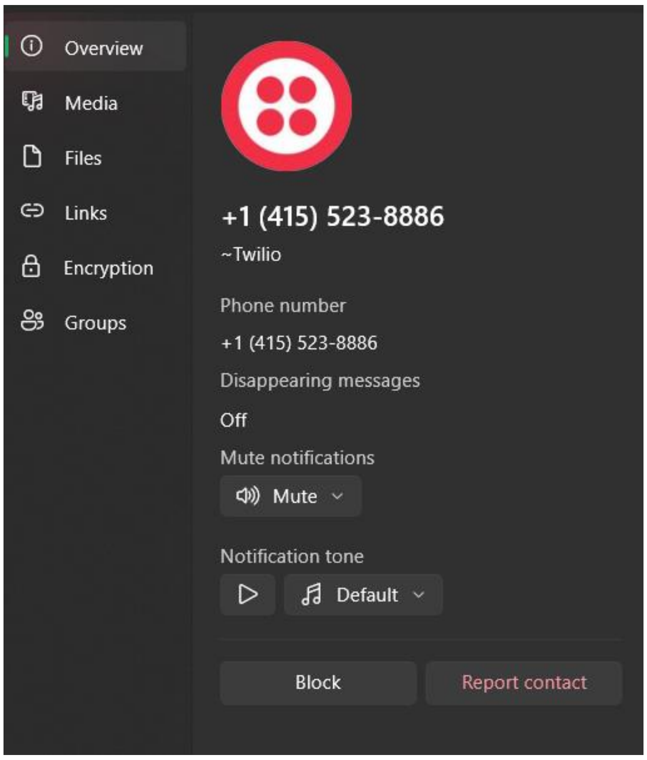
When a message comes in: https://a3f1-45-249-68-133.ngrok-free.app — Method: POST

Status callback URL: — Method: GET

Sandbox Participants

Invite your friends to your Sandbox. Use WhatsApp and send a message from your device to

Twilio Console



+1 (415) 523-8886
~Twilio
Phone number
+1 (415) 523-8886
Disappearing messages
Off
Mute notifications
Notification tone

This Phone Number was provided by Twilio where I interacted with my server by message passing.

Twilio provided me with an Account SID and an Authentication Token for creating a client and using its methods in Python.

```
account_sid = 'AC19d32cc92a35f05f4c4af29e45a58884'
auth_token = '860272559073fa7797a3d400c8ebc636'
client = Client(account_sid, auth_token)
```

According to the outline provided in the ideation for Project 10 hosted on github at https://github.com/coronasafe/care/issues/1977 , I have created a user schema module which contains data members and getter-setter method functions in order to access the information for either an user or a patient respectively.

The class definitions for User and Patient are as follows:

```python
class User:
    def __init__(self, jwttoken={}, username="NULL", password="NULL",
is_user=False, uinfo={},verified=False,last_login=0.0):
        self.jwttoken = jwttoken
        self.username = username
        self.password = password
        self.is_user = is_user
        self.uinfo = uinfo
        self.verified=verified
        self.last_login=last_login

    def reset(self):
        self.jwttoken = {}
        self.username = "NULL"
        self.password = "NULL"
        self.is_user = False
        self.uinfo = {}
        self.verified=False
        self.last_login=0.0

    def set_username_value(self, value):
        self.username = value

    def get_username_value(self):
        return self.username

    def set_password_value(self, value):
        self.password = value

    def get_password_value(self):
        return self.password

    def set_jwttoken_value(self, value):
        self.jwttoken = value

    def get_jwttoken_value(self):
        return self.jwttoken

    def set_user_info_value(self, value):
        self.uinfo = value

    def get_user_info_value(self):
        return self.uinfo

    def set_is_user(self, value):
        self.is_user = value
    def get_is_user(self):
        return self.is_user

    def set_is_authenticated(self,value):
        self.verified=value

    def get_is_authenticated(self):
        return self.verified

    def set_last_login(self,value):
        self.last_login=value

    def get_last_login(self):
        return self.last_login

class Patient:
    def __init__(self, demo_jwttoken={}, pextid="NULL", is_patient=False,
psearchinfo={}, pinfo={},verified=False,last_login=0.0):
        self.demo_jwttoken = demo_jwttoken
        self.pextid = pextid
        self.is_patient = is_patient
        self.psearchinfo = psearchinfo
        self.pinfo = pinfo
        self.verified = verified
        self.last_login=last_login


    def reset(self):
        self.demo_jwttoken={}
        self.pextid="NULL"
        self.is_patient=False
        self.psearchinfo={}
        self.pinfo={}
        self.verified=False
        self.last_login=0.0

    def set_demo_jwttoken_value(self, value):
        self.demo_jwttoken = value

    def get_demo_jwttoken_value(self):
        return self.demo_jwttoken

    def set_is_patient(self, value):
        self.is_patient = value

    def get_is_patient(self):
        return self.is_patient
    def set_patient_search_info_value(self, value):
        self.psearchinfo = value

    def get_patient_search_info_value(self):
        return self.psearchinfo

    def set_patient_info_value(self, value):
        self.pinfo = value

    def get_patient_info_value(self):
        return self.pinfo

    def set_patient_ext_id(self, value):
        self.pextid = value

    def get_patient_ext_id(self):
        return self.pextid

    def set_is_authenticated(self,value):
        self.verified=value

    def get_is_authenticated(self):
        return self.verified

    def set_last_login(self,value):
        self.last_login=value

    def get_last_login(self):
        return self.last_login
```

In addition to these definitions, I have created another class definition of a context handler so that I can route the required incoming message received from the POST method to the required function expecting a message as argument.

```
class Response_Handler():
    def __init__(self,response_handler=0):
        self.response_handler=response_handler

    def set_response_handler(self, value):
        self.response_handler = value

    def get_response_handler(self):
        return self.response_handler
```

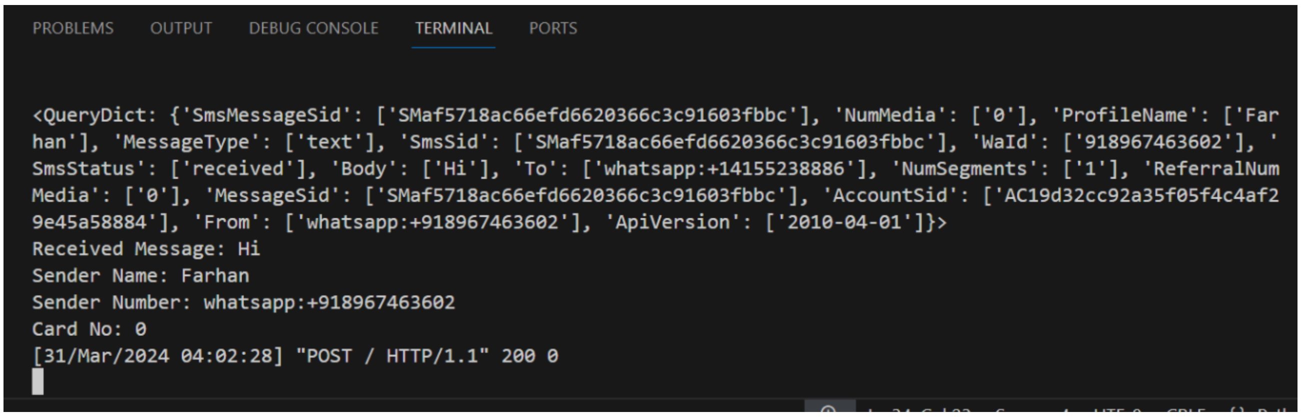
Objects for all the classes are called in the main python file where the incoming messages is received as a request.

```
handler = Response_Handler(0)
user = User()
patient = Patient()
```

The next function takes the request as an argument and calls an appropriate function to handle the message received in request.POST

```
from django.views.decorators.csrf import csrf_exempt
from django.http import HttpResponse
import requests

#ENTRY POINT

@csrf_exempt
def bot(request):

    print(request.POST)
    # data = JSONParser().parse(request)
    # print(data)
    message = request.POST.get('Body', '')
    sender_name = request.POST.get('ProfileName', '')
    sender_no = request.POST.get('From', '')

    print("Received Message:", message)
    print("Sender Name:", sender_name)
    print("Sender Number:", sender_no)

    handle_message(message,sender_name,sender_no)

    return HttpResponse()
```

The print statements are just for the purpose of debugging.



```
<QueryDict: {'SmsMessageSid': ['SMaf5718ac66efd6620366c3c91603fbbc'], 'NumMedia': ['0'], 'ProfileName': ['Far
han'], 'MessageType': ['text'], 'SmsSid': ['SMaf5718ac66efd6620366c3c91603fbbc'], 'WaId': ['918967463602'], '
SmsStatus': ['received'], 'Body': ['Hi'], 'To': ['whatsapp:+14155238886'], 'NumSegments': ['1'], 'ReferralNum
Media': ['0'], 'MessageSid': ['SMaf5718ac66efd6620366c3c91603fbbc'], 'AccountSid': ['AC19d32cc92a35f05f4c4af2
9e45a58884'], 'From': ['whatsapp:+918967463602'], 'ApiVersion': ['2010-04-01']}>
Received Message: Hi
Sender Name: Farhan
Sender Number: whatsapp:+918967463602
Card No: 0
[31/Mar/2024 04:02:28] "POST / HTTP/1.1" 200 0
```

The bot method extracts the necessary required information from the QueryDict and sends it to the message handler method called ahead.

```python
## MESSAGE ROUTER

def handle_message(message, sender_name, sender_no):

    actions = {
        0:  lambda: send_first_message(sender_name, sender_no,message),
        1:  lambda: decide_userorpatient(sender_name, sender_no,message),


        102:  lambda: get_username(sender_name,sender_no,message),
        103:  lambda: get_password(sender_name,sender_no,message),
        104:  lambda: user_services(sender_name,sender_no),


        202:  lambda: get_dateofbirth(sender_name,sender_no,message),
        203:  lambda: get_patient_info_single(sender_name,sender_no,message),
        204:  lambda: patient_services(sender_name,sender_no,message),


        2041: lambda: get_patientrecords(sender_name,sender_no,message),
        # 2042: lambda: get_patientmedications(sender_name,sender_no,message),
        # 2043: lambda: get_patientprocedures(sender_name,sender_no,message),


        # 20411: lambda: get_patientpersonalinfo(sender_name,sender_no,message),
        # 20412: lambda: get_patientmedicalstatus(sender_name,sender_no,message),


        -1:   lambda: send_default_message()
        -2:   lambda: send_relogin_message(sender_no)
```

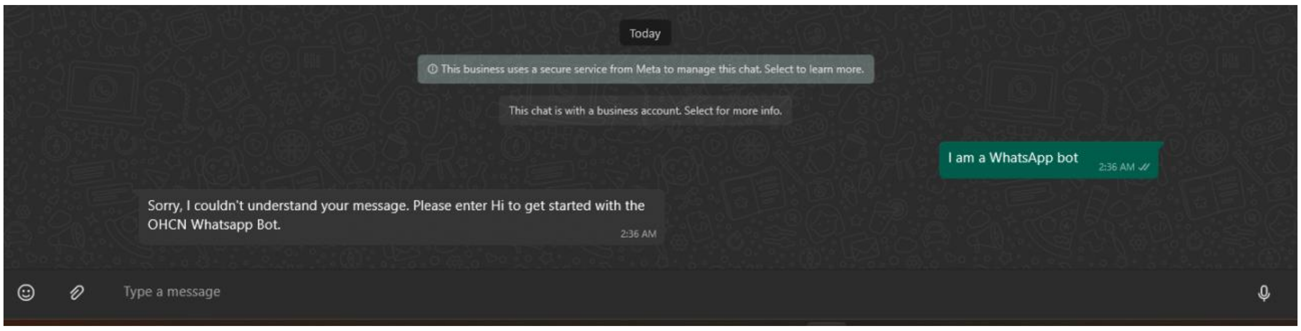
```
     }


    print("Card No: {}".format(handler.get_response_handler()))
    if user.get_is_user() == True:
        print("yes")
        if user.get_is_authenticated() == True and (time.time() -
user.get_last_login() > LOGIN_EXPIRATION_TIME*60):
            print("yes")
            user.reset()
            handler.set_response_handler(-2)
        else:
            user.set_last_login(time.time())
    if patient.get_is_patient() == True:
        if patient.get_is_authenticated() == True and (time.time() -
patient.get_last_login() > LOGIN_EXPIRATION_TIME*60):
            patient.reset()
            handler.set_response_handler(-2)
        else:
            patient.set_last_login((time.time()))

    if message.lower() == "/reset":

        user.reset()
        patient.reset()
        handler.set_response_handler(0)

        send_first_message(sender_name, sender_no,"hi")
    else:
        actions[handler.get_response_handler()]()
```

A dictionary is maintained based on the response handler on which function should be called based on the incoming message and the present value of the handler. Only few functions have been implemented that are related to the project idea in order to check whether the bot is actually functionable or not and can provide accurate information based on the user/patient credentials.

A '/reset' message is kept which when sent in the request.POST body resets all the information stored in that sessions and begins a new interaction with the user.This can later be implemented as a button at the button of every message sent to the user as a reply.

```
## STARTING MESSAGE

def send_first_message(sender_name, sender_no,message):

    if message.lower() != "hi":
        send_default_message(sender_no)
    else:
        if user.get_is_authenticated()==False and
patient.get_is_authenticated()==False:
            send_hi_message(sender_name,sender_no)
            handler.set_response_handler(1)
        else:
            if user.get_is_authenticated()==True:
                print_user_services_menu(sender_no)
                handler.set_response_handler(102)
            elif patient.get_is_authenticated()==True:
                print_patient_services_menu(sender_no)
                handler.set_response_handler(202)


def send_default_message(sender_no):
    client.messages.create(
        from_='whatsapp:+14155238886',
        body='Sorry, I couldn\'t understand your message. Please enter Hi to get
started with the OHCN Whatsapp Bot.',
        to=sender_no
    )
```

The default message that is provided when any message other than "Hi" is provided at the beginning of the interaction.



Today

This business uses a secure service from Meta to manage this chat. Select to learn more.

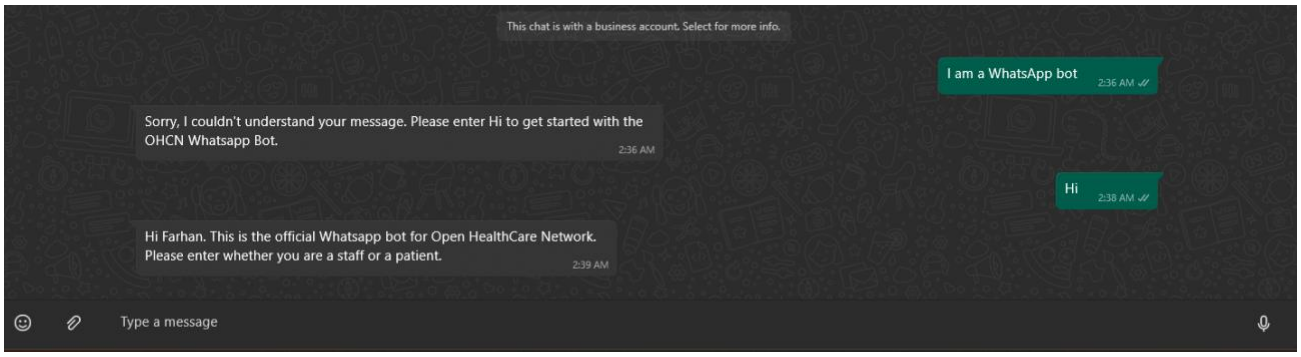
This chat is with a business account. Select for more info.

I am a WhatsApp bot 2:36 AM

Sorry, I couldn't understand your message. Please enter Hi to get started with the OHCN Whatsapp Bot. 2:36 AM

```python
def send_hi_message(sender_name,sender_no):
    client.messages.create(
        from_='whatsapp:+14155238886',
        body='Hi {}. This is the official Whatsapp bot for Open HealthCare 
Network.\nPlease enter whether you are a staff or a patient.'.format(sender_name),
        to=sender_no
    )
```

If a "Hi" is encountered, the handler is set to the next method where it decides whether the current user is a patient or a staff.


This chat is with a business account. Select for more info.

I am a WhatsApp bot 2:36 AM

Sorry, I couldn't understand your message. Please enter Hi to get started with the OHCN Whatsapp Bot. 2:36 AM

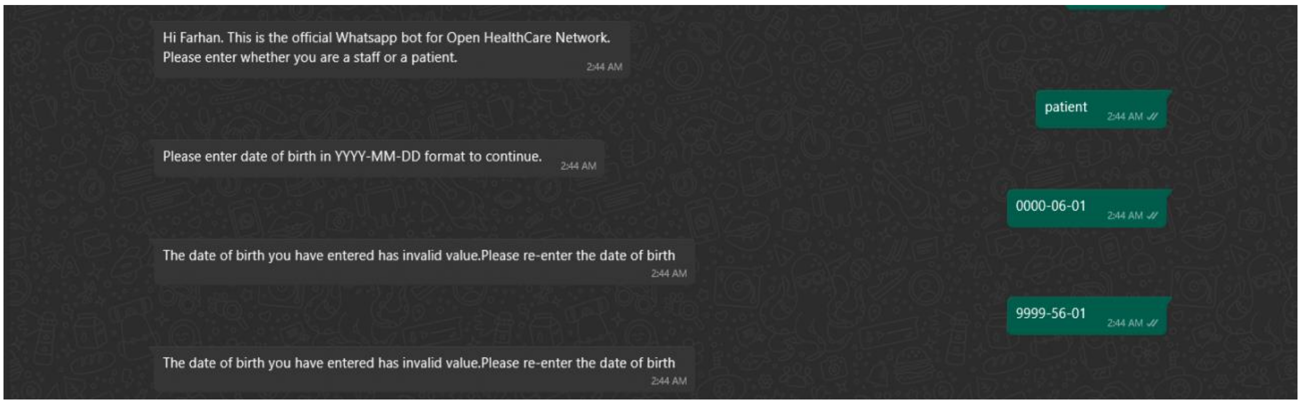
Hi 2:38 AM

Hi Farhan. This is the official Whatsapp bot for Open HealthCare Network. Please enter whether you are a staff or a patient. 2:39 AM

```python
## USER OR PATIENT RECOGNITION

def decide_userorpatient(sender_name, sender_no,message):

    if "patient" in message.lower():
        patient.set_is_patient(True)
        user.set_is_user(False)
        send_message(sender_no,'Please enter date of birth in YYYY-MM-DD format to 
continue.')
        handler.set_response_handler(202)
    elif "staff" in message.lower():
        patient.set_is_patient(False)
        user.set_is_user(True)
        send_message(sender_no,'Please enter your username to login.')
        handler.set_response_handler(102)
    else:
        handler.set_response_handler(0)
        send_default_message(sender_no)
```

Based on the reply from the user, it sets the handler to the required the action number and sends the next message required for authentication.

The Patient Route:


Hi Farhan. This is the official Whatsapp bot for Open HealthCare Network.
Please enter whether you are a staff or a patient.
2:44 AM

patient
2:44 AM

Please enter date of birth in YYYY-MM-DD format to continue.
2:44 AM

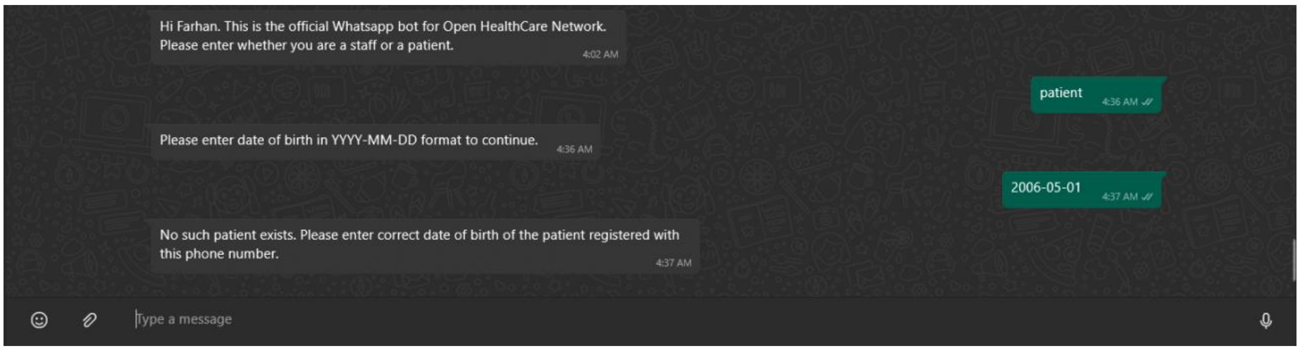
0000-06-01
2:44 AM

The date of birth you have entered has invalid value.Please re-enter the date of birth
2:44 AM

9999-56-01
2:44 AM

The date of birth you have entered has invalid value.Please re-enter the date of birth
2:44 AM

After selecting ‘patient’, it asks the user the date-of-birth in the required format for finding out the patient information with the filters date of birth and phone number of the WhatsApp user. **It makes sure that only the WhatsApp account logged in with the patient’s number can access the information and thus establishes PPI privacy for the user end.**


Hi Farhan. This is the official Whatsapp bot for Open HealthCare Network.
Please enter whether you are a staff or a patient.
4:02 AM

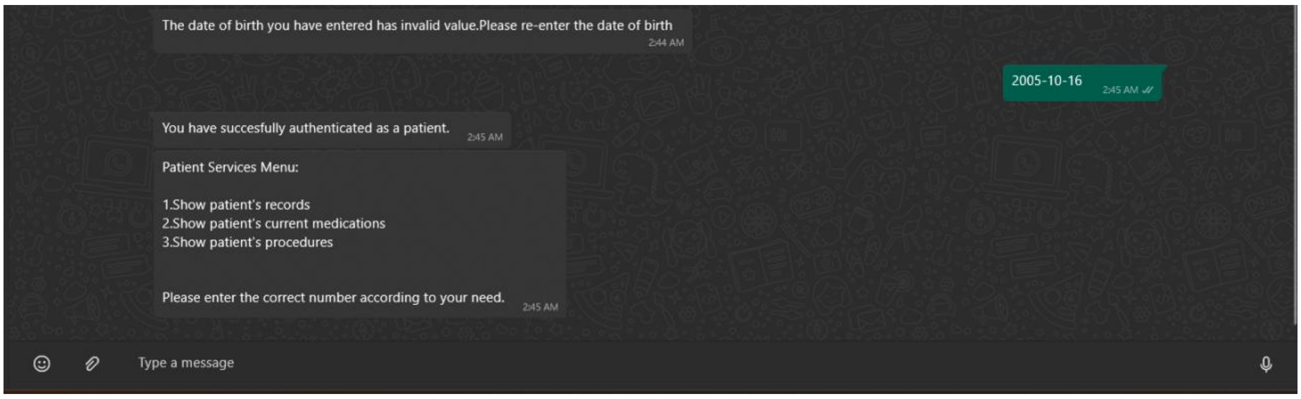
patient
4:36 AM

Please enter date of birth in YYYY-MM-DD format to continue.
4:36 AM

2006-05-01
4:37 AM

No such patient exists. Please enter correct date of birth of the patient registered with this phone number.
4:37 AM

I then changed the dummy data for Facility.patientregistration in the facility.json changing the first patient’s number to my own number and re-load the data and put in the same date of birth, it replies me with a successful authentication.



The date of birth you have entered has invalid value.Please re-enter the date of birth 2:44 AM

2005-10-16 2:45 AM

You have succesfully authenticated as a patient. 2:45 AM

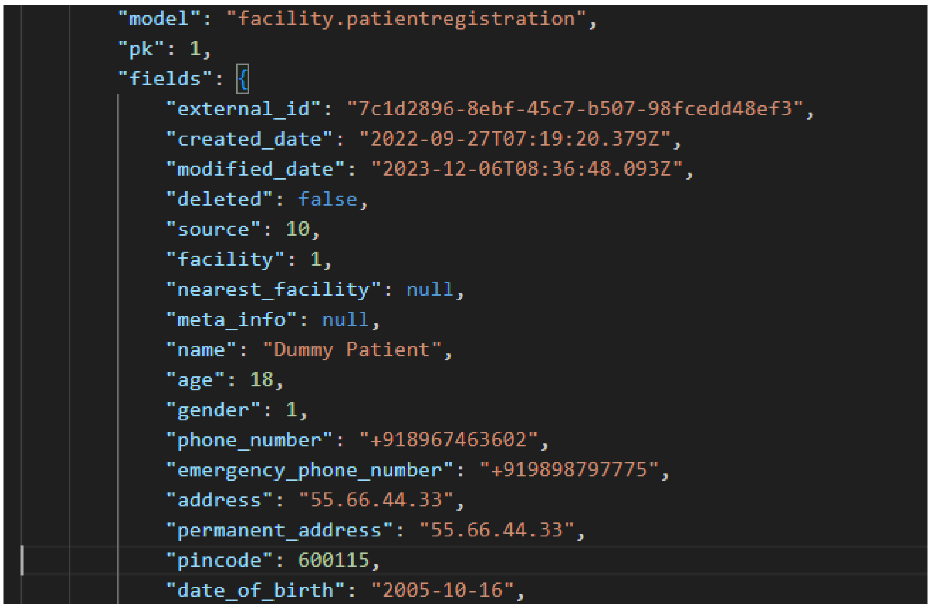
Patient Services Menu:

1.Show patient's records
2.Show patient's current medications
3.Show patient's procedures

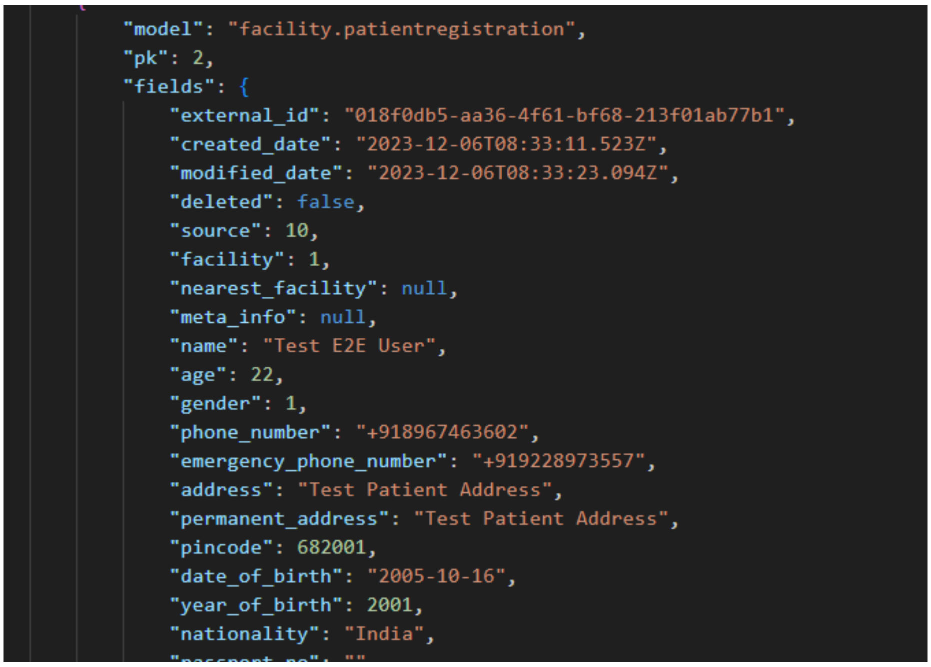
Please enter the correct number according to your need. 2:45 AM

I also thought of an edge case: What if twins are registered with the same phone number of the father, in that case, name can be the deciding factor.

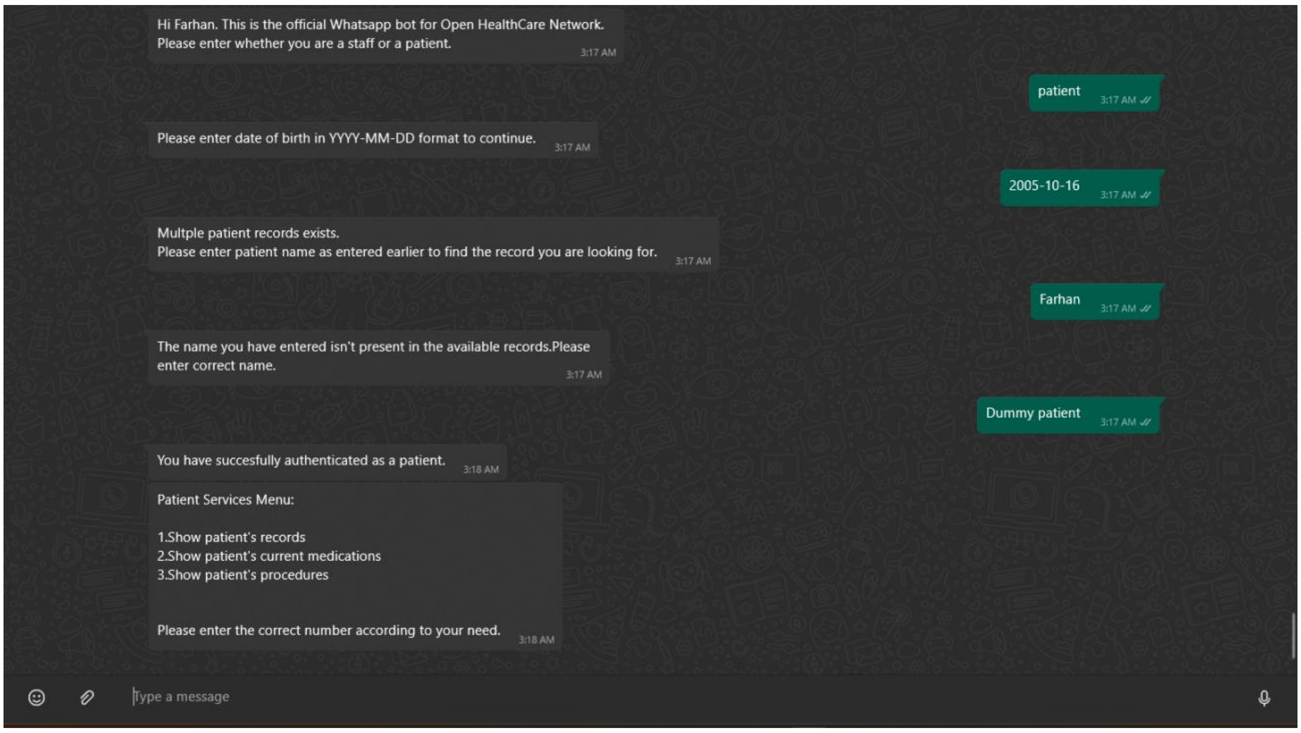
In order to check this, I manipulated two patient records making both their d.o.b and phone number same.



```
        "model": "facility.patientregistration",
        "pk": 1,
        "fields": {
            "external_id": "7c1d2896-8ebf-45c7-b507-98fcedd48ef3",
            "created_date": "2022-09-27T07:19:20.379Z",
            "modified_date": "2023-12-06T08:36:48.093Z",
            "deleted": false,
            "source": 10,
            "facility": 1,
            "nearest_facility": null,
            "meta_info": null,
            "name": "Dummy Patient",
            "age": 18,
            "gender": 1,
            "phone_number": "+918967463602",
            "emergency_phone_number": "+919898797775",
            "address": "55.66.44.33",
            "permanent_address": "55.66.44.33",
            "pincode": 600115,
            "date_of_birth": "2005-10-16",
```



```json
        "model": "facility.patientregistration",
        "pk": 2,
        "fields": {
            "external_id": "018f0db5-aa36-4f61-bf68-213f01ab77b1",
            "created_date": "2023-12-06T08:33:11.523Z",
            "modified_date": "2023-12-06T08:33:23.094Z",
            "deleted": false,
            "source": 10,
            "facility": 1,
            "nearest_facility": null,
            "meta_info": null,
            "name": "Test E2E User",
            "age": 22,
            "gender": 1,
            "phone_number": "+918967463602",
            "emergency_phone_number": "+919228973557",
            "address": "Test Patient Address",
            "permanent_address": "Test Patient Address",
            "pincode": 682001,
            "date_of_birth": "2005-10-16",
            "year_of_birth": 2001,
            "nationality": "India",
            "passport_no": "",
```



Hi Farhan. This is the official Whatsapp bot for Open HealthCare Network.
Please enter whether you are a staff or a patient. 3:17 AM

patient 3:17 AM

Please enter date of birth in YYYY-MM-DD format to continue. 3:17 AM

2005-10-16 3:17 AM

Multple patient records exists.
Please enter patient name as entered earlier to find the record you are looking for. 3:17 AM

Farhan 3:17 AM

The name you have entered isn't present in the available records.Please enter correct name. 3:17 AM

Dummy patient 3:17 AM

You have succesfully authenticated as a patient. 3:18 AM

Patient Services Menu:

1.Show patient's records
2.Show patient's current medications
3.Show patient's procedures

Please enter the correct number according to your need. 3:18 AM

If name entered isn't present, it asks user to enter correct name. [Type Case is ignored.]

This is implemented as:

```python
## PATIENT INFO


def get_dateofbirth(sender_name,sender_no,message):

    if(re.match(r'^\d{4}-\d{2}-\d{2}$',message)):
        if re.match(r'^(19[2-9]\d|20[0-9][0-9])-(0[1-9]|1[0-2])-(0[1-9]|[12]\d|3[01])$',message):

            user.set_username_value('devdistrictadmin')
            user.set_password_value('Coronasafe@123')

            api_status=user_login_api_call()

            if api_status == -1:
                send_message(sender_no,'Server Error.\nTry Later')
            elif api_status != 200:
                send_message(sender_no,'Wrong API Call.\nTry Later')

            api_status_patient_info=search_patient_api(message,sender_no.split(':')[1],patient.get_demo_jwttoken_value()['access']) #2005-10-16 #+919987455444

            if api_status_patient_info == -1:
                send_message(sender_no,'Server Error.\nTry Later')
            elif api_status_patient_info != 200:
                send_message(sender_no,'Wrong API Call.\nTry Later')
            else:
                if patient.get_patient_search_info_value()['count'] == 1:
                    send_message(sender_no,'You have succesfully authenticated as a patient.')
                    print_patient_services_menu(sender_no)
                    patient.set_is_authenticated(True)


                    patient.set_patient_info_value(patient.get_patient_search_info_value()['results'][0])
                    patient.set_patient_ext_id(patient.get_patient_search_info_value()['results'][0]['patient_id'])
                    handler.set_response_handler(204)

                elif patient.get_patient_search_info_value()['count'] > 1:
                    send_message(sender_no,'Multple patient records exists.\nPlease enter patient name as entered earlier to find the record you are looking for.')
                    handler.set_response_handler(203)
                    # To be implemented check with name and accordingly filter
         else:

                    handler.set_response_handler(202)
                    send_message(sender_no,'No such patient exists. Please enter
correct date of birth of the patient registered with this phone number.')

        else:
            handler.set_response_handler(202)
            send_message(sender_no,'The date of birth you have entered has invalid
value.Please re-enter the date of birth')
    else:
        handler.set_response_handler(202)
        send_message('The date of birth you have entered has incorrect
format.Please re-enter the date of birth in correct format.')

def get_patient_info_single(sender_name,sender_no,message):
    found_patient = "NULL"
    results=patient.get_patient_search_info_value()['results']
    for pat in results:
        if pat['name'].lower() == message.lower():
            found_patient = pat
            send_message(sender_no,'You have succesfully authenticated as a
patient.')
            print_patient_services_menu(sender_no)
            patient.set_is_authenticated(True)

            patient.set_patient_info_value(patient.get_patient_search_info_value()[
'results'][0])
            patient.set_patient_ext_id(patient.get_patient_search_info_value()['res
ults'][0]['patient_id'])
            handler.set_response_handler(204)
            break  # Exit the loop once a match is found

    if found_patient=="NULL":
        handler.set_response_handler(203)
        send_message(sender_no,'The name you have entered isn\'t present in the
available records.Please enter correct name.')
```

Since admin access isn't available, I had to obtain a JWT token to call the necessary API to get the information from the care backend. So I had to take a demo JWT Token variable and store it in order to use it for authentication and further information extraction.

The function calling the API is:

```
def search_patient_api(date_of_birth, ph_no,authorization_token):

    encoded_date_of_birth = quote(date_of_birth)   #Encoding parameters
    encoded_ph_no = quote(ph_no)
    print(encoded_date_of_birth)
    print(encoded_ph_no)

    headers = {
        'accept': 'application/json',
        'Authorization': 'Bearer {}'.format(authorization_token)
    }

    url =
f"http://localhost:9000/api/v1/patient/search/?date_of_birth={encoded_date_of_birth
}&phone_number={encoded_ph_no}" #date_of_birth={encoded_date_of_birth}&

    try:
        response = requests.get(url, headers=headers)

        if response.status_code == 200:
            patient.set_patient_search_info_value(response.json())
            print(patient.get_patient_search_info_value())
            return response.status_code
        else:
            print("Error: Failed to fetch patient information.")
            return response.status_code
    except requests.exceptions.RequestException as e:
        print("Error making API call:", e)
        return -1
```

The API call returns a directory of results with a count variable that informs how many similar patient records have been found according to the filter provided during the API Call.

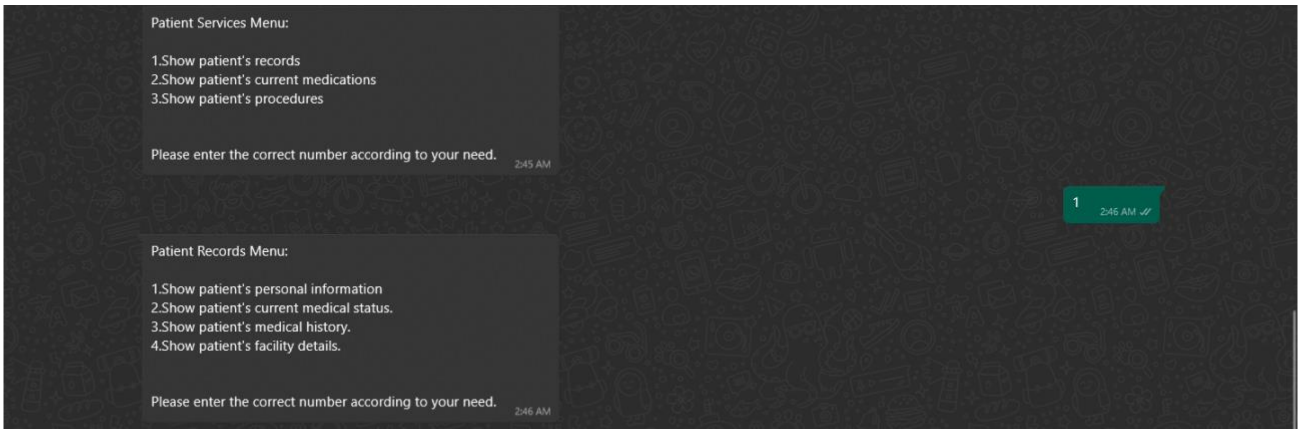
Next, the user if the count is greater than 1, is asked to provide the name of the patient to find the exact record. Function patient_info_single() is called for the rescue at the above instance.

After successful identification of the patient, the patient specific services menu shows up and asks for further input.

```
#PATIENT SERVICES

def patient_services(sender_name,sender_no,message):
    if message == "1":
        handler.set_response_handler(2041)
        print_patient_records_menu(sender_no)
    elif message == "2":
        handler.set_response_handler(2042)
    elif message == "3":
        handler.set_response_handler(2043)
    else:
        send_message(sender_no,'Invalid input provided.Please enter suitable number
to access information')
        print_patient_services_menu(sender_no)
        handler.set_response_handler(204)

def print_patient_services_menu(sender_no):
    client.messages.create(
        from_='whatsapp:+14155238886',
        body='Patient Services Menu:\n\n1.Show patient\'s records\n2.Show
patient\'s current medications\n3.Show patient\'s procedures\n\n\nPlease enter the
correct number according to your need.',
        to=sender_no
    )
```

The patient services menu has currently three options to extract information. Just for the sake of checking whether the information extraction is alright, I implemented the first option for “patient records” in the services menu that further opens up a “patient records menu” that has several options as follows:

```
def get_patientrecords(sender_name,sender_no,message):
    patient_full_info=patient_info_api(patient.get_demo_jwttoken_value()['access'])
     if message == "1":
        handler.set_response_handler(204)
        get_patientpersonalinfo(sender_name,sender_no,patient_full_info)
        print_patient_services_menu(sender_no)
    elif message == "2":
        handler.set_response_handler(204)
        get_patientmedicalstatus(sender_name,sender_no,patient_full_info)
        print_patient_services_menu(sender_no)
    elif message == "3":
        handler.set_response_handler(204)
        #Need to use the last_consultation key for extracting.
    elif message == "4":
        handler.set_response_handler(204)
        get_patientfacilitydetails(sender_name,sender_no,patient_full_info)
        print_patient_services_menu(sender_no)
    else:
        send_message(sender_no,'Invalid input provided.Please enter suitable number
to access information')
        print_patient_records_menu(sender_no)
        handler.set_response_handler(2041)


def print_patient_records_menu(sender_no):
    client.messages.create(
        from_='whatsapp:+14155238886',
        body='Patient Records Menu:\n\n1.Show patient\'s personal
information\n2.Show patient\'s current medical status.\n3.Show patient\'s medical
history.\n4.Show patient\'s facility details.\n\n\nPlease enter the correct number
according to your need.',
        to=sender_no
    )
```


Patient Services Menu:

1.Show patient's records
2.Show patient's current medications
3.Show patient's procedures

Please enter the correct number according to your need. 2:45 AM

1 2:46 AM

Patient Records Menu:

1.Show patient's personal information
2.Show patient's current medical status.
3.Show patient's medical history.
4.Show patient's facility details.

Please enter the correct number according to your need. 2:46 AM

Options 1,2 and 4 here have been successfully implemented by calling an API call for fetching patient information and then on user choice, provide the required details.

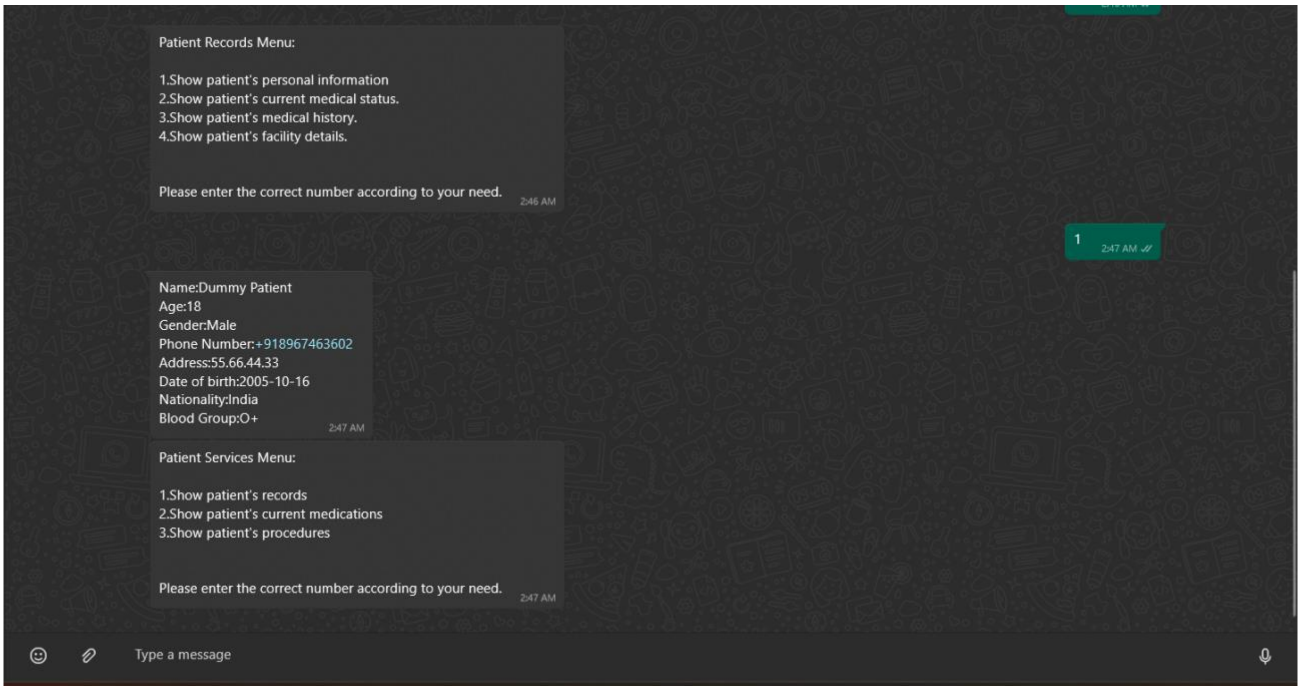
The get_patientrecords() menu calls a patient_info_api() function that returns all the necessary information regarding a patient using his/her/their external id which is uuid string of 32 bits.

The function making the API call is as below:

```python
def patient_info_api(authorization_token):
    url = f'http://localhost:9000/api/v1/patient/{patient.get_patient_ext_id()}/'

    headers = {
        'accept': 'application/json',
        'Authorization': 'Bearer {}'.format(authorization_token)
    }
    try:
        response = requests.get(url, headers=headers)

        if response.status_code == 200:
            return response.json()
        else:
            print("Error:", response.status_code)
            print("Response Body:", response.text)
            return None

    except requests.exceptions.RequestException as e:
        print("Error making API call:", e)
        return None
```

Now after we have reached the Patient Records Menu.

The entire patient information is stored in a patient info variable of the patient object in form a json file and then values is extracted from the json file using the respective key as mentioned in the patient information schema.



Patient Records Menu:

1.Show patient's personal information
2.Show patient's current medical status.
3.Show patient's medical history.
4.Show patient's facility details.

Please enter the correct number according to your need. 2:46 AM

1 2:47 AM

Name:Dummy Patient
Age:18
Gender:Male
Phone Number:+918967463602
Address:55.66.44.33
Date of birth:2005-10-16
Nationality:India
Blood Group:O+ 2:47 AM

Patient Services Menu:

1.Show patient's records
2.Show patient's current medications
3.Show patient's procedures

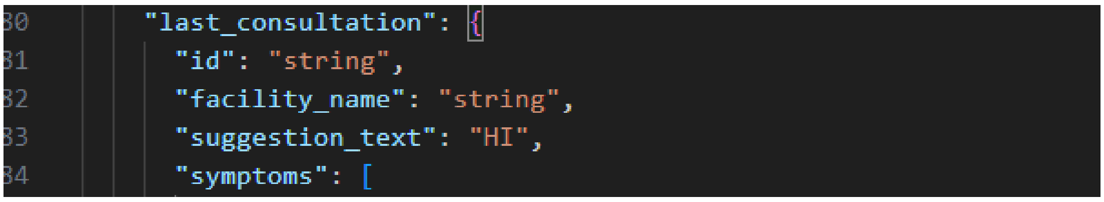
Please enter the correct number according to your need. 2:47 AM

After the information is replied as a message, it goes back to the services menu so that user can again interact with the bot and find out about something else.

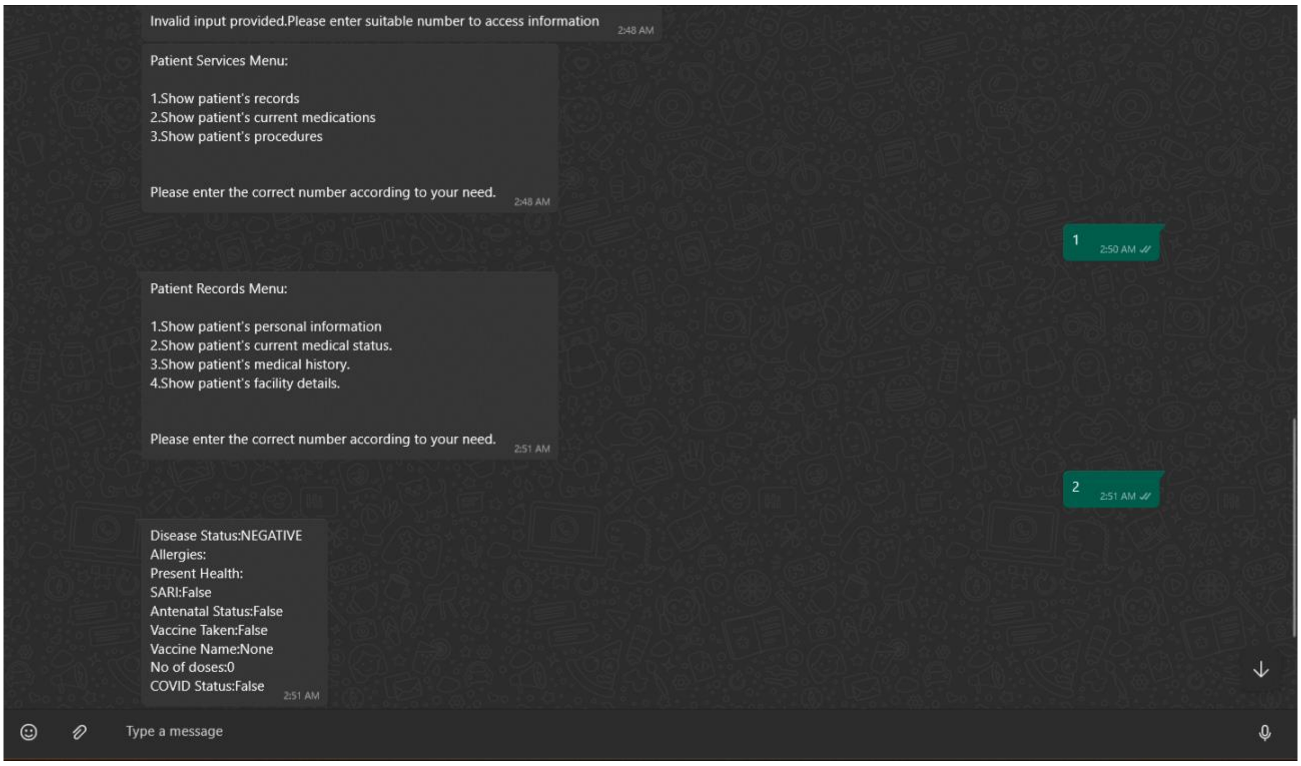
This routing back to services menu can be changed to using buttons present on the message with option to exit the bot interaction or to go back to services menu to use any other unused option.[Three* buttons to do to different tasks][The previously mentioned reset button exists.]

The replies for other two options are shown as below:

Option 3 hasn't been implemented but it can be done by using the last_consultation object present in the json file returned.



```
"last_consultation": {
  "id": "string",
  "facility_name": "string",
  "suggestion_text": "HI",
  "symptoms": [
```



Invalid input provided.Please enter suitable number to access information 2:48 AM

Patient Services Menu:

1.Show patient's records
2.Show patient's current medications
3.Show patient's procedures

Please enter the correct number according to your need. 2:48 AM

1 2:50 AM

Patient Records Menu:

1.Show patient's personal information
2.Show patient's current medical status.
3.Show patient's medical history.
4.Show patient's facility details.

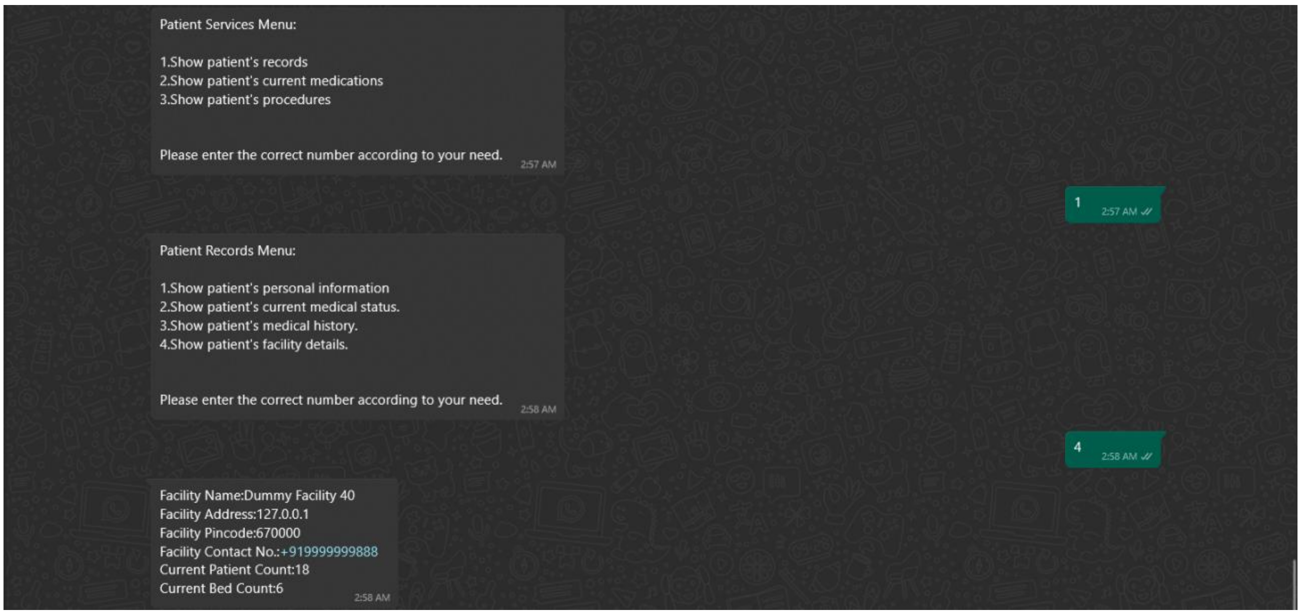
Please enter the correct number according to your need. 2:51 AM

2 2:51 AM

Disease Status:NEGATIVE
Allergies:
Present Health:
SARI:False
Antenatal Status:False
Vaccine Taken:False
Vaccine Name:None
No of doses:0
COVID Status:False 2:51 AM

Type a message

Option for current medical status



Patient Services Menu:

1.Show patient's records
2.Show patient's current medications
3.Show patient's procedures

Please enter the correct number according to your need. 2:57 AM

1 2:57 AM

Patient Records Menu:

1.Show patient's personal information
2.Show patient's current medical status.
3.Show patient's medical history.
4.Show patient's facility details.

Please enter the correct number according to your need. 2:58 AM

4 2:58 AM

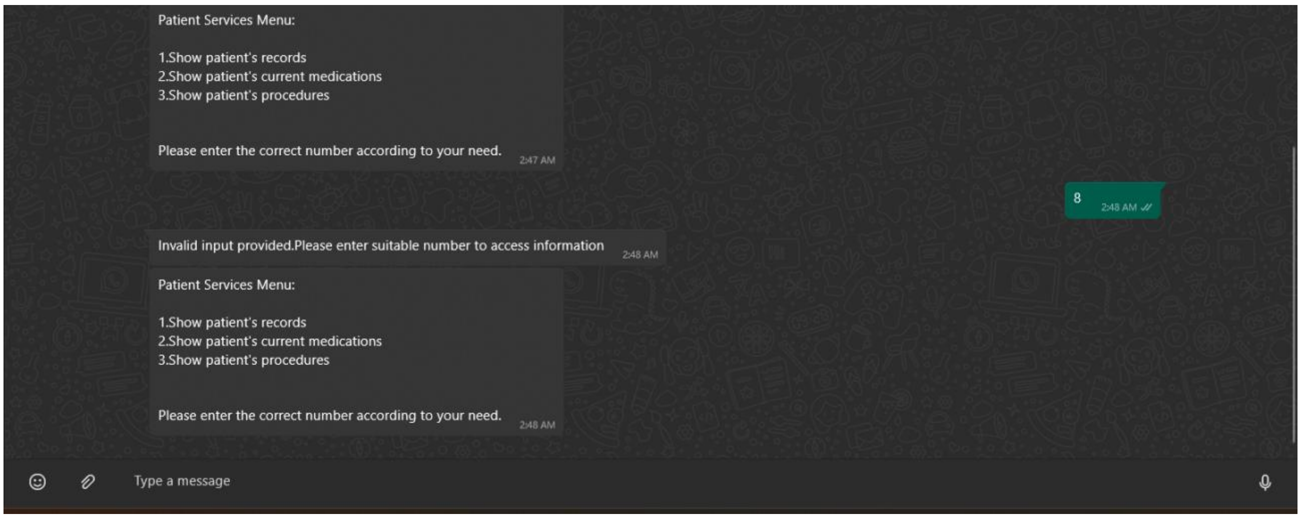
Facility Name:Dummy Facility 40
Facility Address:127.0.0.1
Facility Pincode:670000
Facility Contact No.:+919999999888
Current Patient Count:18
Current Bed Count:6 2:58 AM

Option for facility details.

The functions sending these messages to the user back are as follows:

```
def get_patientpersonalinfo(sender_name,sender_no,patient_full_info):
    gender = "Male" if patient_full_info['gender'] == 1 else ("Female" if
patient_full_info['gender'] == 2 else "Other")
    send_message(sender_no,f'Name:{patient_full_info['name']}\nAge:{patient_full_in
fo['age']}\nGender:{gender}\nPhone
Number:{patient_full_info['phone_number']}\nAddress:{patient_full_info['address']}\
nDate of
birth:{patient_full_info['date_of_birth']}\nNationality:{patient_full_info['nationa
lity']}\nBlood Group:{patient_full_info['blood_group']}\n')


def get_patientmedicalstatus(sender_name,sender_no,patient_full_info):
    send_message(sender_no,f'Disease
Status:{patient_full_info['disease_status']}\nAllergies:{patient_full_info['allergi
es']}\nPresent
Health:{patient_full_info['present_health']}\nSARI:{patient_full_info['has_SARI']}\
nAntenatal Status:{patient_full_info['is_antenatal']}\nVaccine
Taken:{patient_full_info['is_vaccinated']}\nVaccine
Name:{patient_full_info['vaccine_name']}\nNo of
doses:{patient_full_info['number_of_doses']}\nCOVID
Status:{patient_full_info['is_declared_positive']}\n')


def get_patientfacilitydetails(sender_name,sender_no,patient_full_info):
    send_message(sender_no,f'Facility
Name:{patient_full_info['facility_object']['name']}\nFacility
Address:{patient_full_info['facility_object']['address']}\nFacility
Pincode:{patient_full_info['facility_object']['pincode']}\nFacility Contact
No.:{patient_full_info['facility_object']['phone_number']}\nCurrent Patient
Count:{patient_full_info['facility_object']['patient_count']}\nCurrent Bed
Count:{patient_full_info['facility_object']['bed_count']}\n')
```

If the user gives an invalid input in any of menus. It shows an invalid input message and asks to user to retry in order to get a correct acceptable choice input.



Patient Services Menu:

1.Show patient's records
2.Show patient's current medications
3.Show patient's procedures

Please enter the correct number according to your need. 2:47 AM

8 2:48 AM

Invalid input provided.Please enter suitable number to access information 2:48 AM

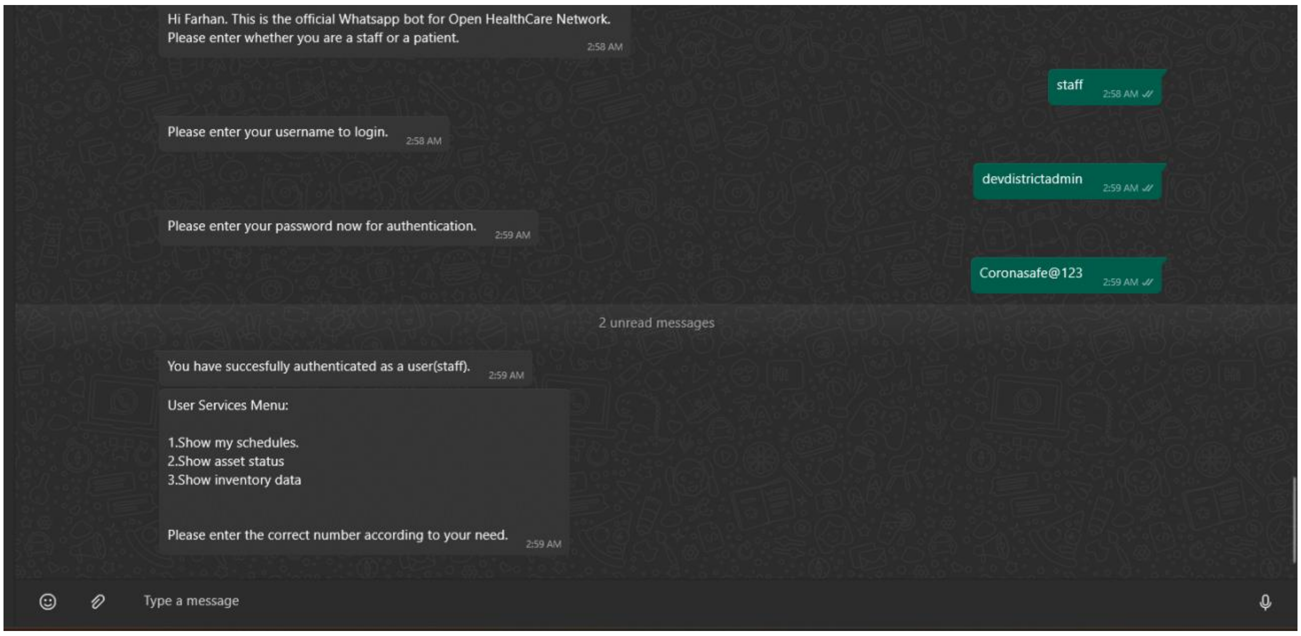
Patient Services Menu:

1.Show patient's records
2.Show patient's current medications
3.Show patient's procedures

Please enter the correct number according to your need. 2:48 AM

This was all I did for Patient Route in the dummy server in order to check if the things are functional and usable or not so that I can host/integrate them later.

The User Route:



Hi Farhan. This is the official Whatsapp bot for Open HealthCare Network.
Please enter whether you are a staff or a patient. 2:58 AM

staff 2:58 AM

Please enter your username to login. 2:58 AM

devdistrictadmin 2:59 AM

Please enter your password now for authentication. 2:59 AM

Coronasafe@123 2:59 AM

2 unread messages

You have succesfully authenticated as a user(staff). 2:59 AM

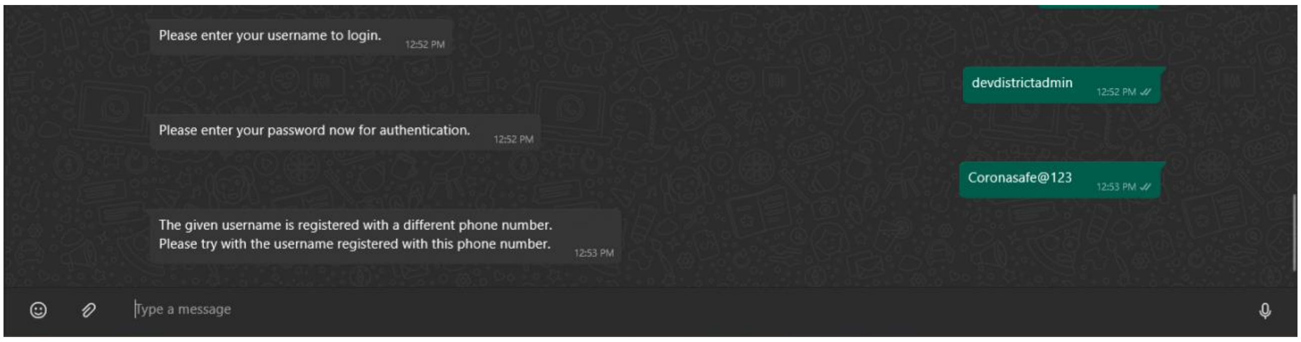
User Services Menu:

1.Show my schedules.
2.Show asset status
3.Show inventory data

Please enter the correct number according to your need. 2:59 AM

After selecting 'staff', it asks the user the username and password for the user he wants to register as. Take care that **password** is case-sensitive and thus needs to be exact same. After getting the required user credentials, it tries for an auth login API call to get the JWT Token. **It makes sure that only the WhatsApp account logged in with the user's number can access the information and thus establishes PPI privacy for the user end.**

In order to check the PPI of the user, I changed the phone number in the users.json file in the care backend and changed the 'devdistrictadmin' phone number with my personal number and ran the above shown conversation and later reverted it back to its original one so that I can show that **privacy** is maintained.


Please enter your username to login. 12:52 PM

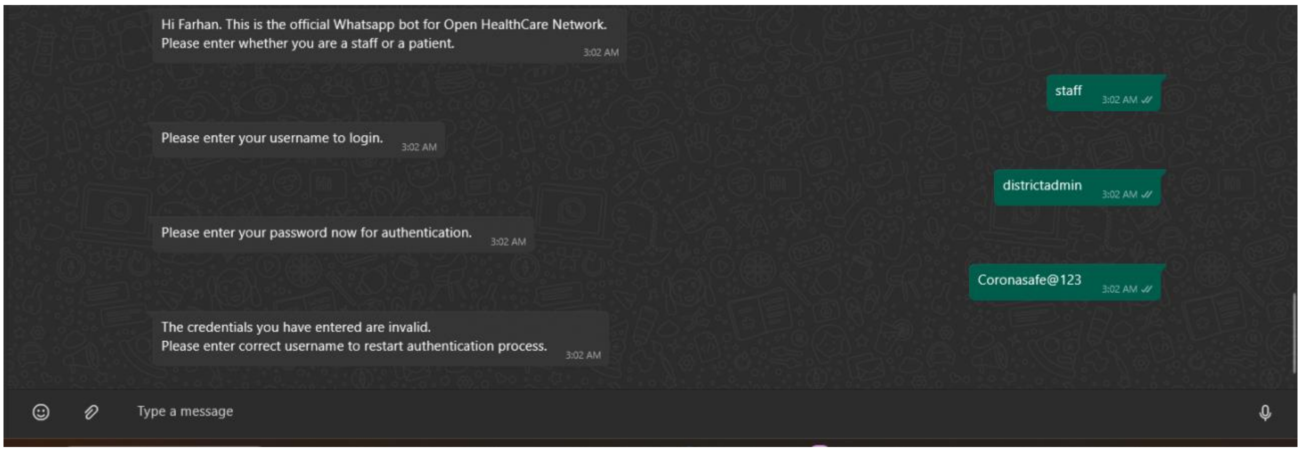
devdistrictadmin 12:52 PM

Please enter your password now for authentication. 12:52 PM

Coronasafe@123 12:53 PM

The given username is registered with a different phone number.
Please try with the username registered with this phone number. 12:53 PM

Wrong credentials provided to the bot might result in an invalid API call and an invalid credentials message is sent to the user to re-enter username and password correctly.


Hi Farhan. This is the official Whatsapp bot for Open HealthCare Network.
Please enter whether you are a staff or a patient. 3:02 AM

staff 3:02 AM

Please enter your username to login. 3:02 AM

districtadmin 3:02 AM

Please enter your password now for authentication. 3:02 AM

Coronasafe@123 3:02 AM

The credentials you have entered are invalid.
Please enter correct username to restart authentication process. 3:02 AM

The functions responsible for above authentication and further data fetch is as below:

```
## USER INFO

def get_username(sender_name,sender_no,message):
    user.set_username_value(message)
    handler.set_response_handler(103)
    send_message(sender_no,'Please enter your password now for authentication.')

def get_password(sender_name,sender_no,message):
    user.set_password_value(message)
    api_status=user_login_api_call()
    if api_status == -1:
        send_message(sender_no,'Server Error.\nTry Later')
    elif api_status == 401:
        handler.set_response_handler(102)
```

```
       send_message(sender_no,'The credentials you have entered are
invalid.\nPlease enter correct username to restart authentication process.')

    elif api_status!= 200:
        handler.set_response_handler(102)
        send_message(sender_no,'Wrong API Call.Try Later')
    else:
        api_status_user=get_user_info_api(user.get_jwttoken_value()['access'])
        if api_status_user == -1:
            send_message(sender_no,'Server Error.\nTry Later')
        elif api_status_user != 200:
            send_message(sender_no,'Wrong API Call.\nTry Later')
        else:
            if sender_no !=
'whatsapp:'+user.get_user_info_value()['phone_number']:  #change lhs of this
condition to sender_no later.

                send_message(sender_no,'The given username is registered with a
different phone number.\nPlease try with the username registered with this phone
number.')
                handler.set_response_handler(102)
            else:
                send_message(sender_no,'You have succesfully authenticated as a
user(staff).')
                user.set_is_authenticated(True)

                handler.set_response_handler(104)
                user_services(sender_name,sender_no)

def user_services(sender_name,sender_no):
    print_user_services_menu(sender_no)
```

Two functions are present here that are making API calls to the server in order to extract information.

- user_info_api_call()
- get_user_info_api(access token)

The first one is responsible for establishment of authentication while the second one fetches information based on the JWT token received after successful authentication on the Login POST API.

The functions calling the respective APIs are as follows:

```
## API CALLS

def user_login_api_call():

    url = "http://localhost:9000/api/v1/auth/login/"

    payload = {
        "username": user.get_username_value(),
        "password": user.get_password_value()
    }
    print(payload)

    payload_json = json.dumps(payload)

    try:
        response = requests.post(url, data=payload_json, headers={"Content-Type": 
"application/json"})

        print(response.status_code)
        # print(payload_json)

        if response.status_code == 200:

            if user.get_is_user() == True:
                # print("user")
                user.set_jwttoken_value(response.json())
            elif patient.get_is_patient() == True:
                # print("patient")
                patient.set_demo_jwttoken_value(response.json())
            return response.status_code

        else:
            print("error")
            return response.status_code

    except requests.exceptions.RequestException as e:
        print("Error making API call:", e)
        return -1
```

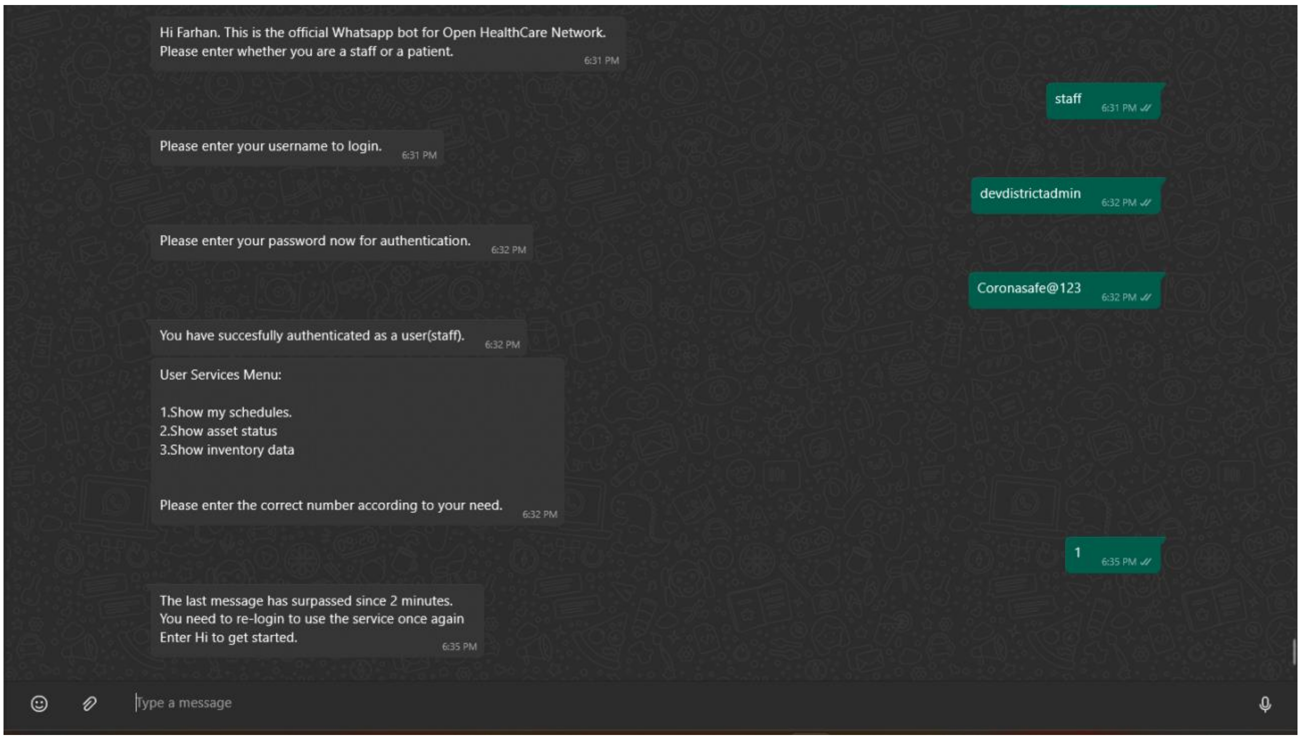
```python
def get_user_info_api(authorization_token):

    url = f'http://localhost:9000/api/v1/users/{user.get_username_value()}/'

    headers = {
        'accept': 'application/json',
        'Authorization': 'Bearer {}'.format(authorization_token)
    }
    try:
        response = requests.get(url, headers=headers)

        if response.status_code == 200:
            user.set_user_info_value(response.json())
        else:
            print("Error:", response.status_code)
            print("Response Body:", response.text)
        return response.status_code

    except requests.exceptions.RequestException as e:
       print("Error making API call:", e)
        return -1
```

The work I did for User route is less as compared to the Patient Route as I just needed to check if the APIs are working fine and I am able to call them correctly in order to get user specific information.

Timeout:

Another thing I have also implemented in the dummy server is the re-authentication of the WhatsApp user when a certain amount of time has passed since the user has authenticated and sent a message. The timer is initialized to the current time using time.time(). The user object/patient object has an attribute called last_login that stores the time in seconds when the last message was passed and when a new message arrives, it checks whether authentication has been done for the user and if the time difference between the last message and the incoming new message is more than the limit that has been set. If limit is exceeded, the objects are reset using a reset function and the user is asked to re-login by providing the credentials once again.

```
LOGIN_EXPIRATION_TIME=15

#Check for time-out

if user.get_is_user() == True:
        print("yes")
        if user.get_is_authenticated() == True and (time.time() -
user.get_last_login() > LOGIN_EXPIRATION_TIME*60):
            print("yes")
            user.reset()
            handler.set_response_handler(-2)
        else:
            user.set_last_login(time.time())
    if patient.get_is_patient() == True:
        if patient.get_is_authenticated() == True and (time.time() -
patient.get_last_login() > LOGIN_EXPIRATION_TIME*60):
            patient.reset()
            handler.set_response_handler(-2)
        else:
            patient.set_last_login((time.time()))
```



```
Hi Farhan. This is the official Whatsapp bot for Open HealthCare Network.
Please enter whether you are a staff or a patient.   6:31 PM
                                                   staff   6:31 PM
Please enter your username to login.   6:31 PM
                                          devdistrictadmin   6:32 PM
Please enter your password now for authentication.   6:32 PM
                                          Coronasafe@123   6:32 PM
You have succesfully authenticated as a user(staff).   6:32 PM
User Services Menu:

1.Show my schedules.
2.Show asset status
3.Show inventory data

Please enter the correct number according to your need.   6:32 PM
                                                       1   6:35 PM
The last message has surpassed since 2 minutes.
You need to re-login to use the service once again
Enter Hi to get started.   6:35 PM
```

Here, I set the time-out for 2 minutes and checked whether it works or not.

Thank You for your time to read this proposal. I have high hopes for this project idea and further work.

Date of submission: 31st March,2024.

Contributions:

I have no such existing PRs to any of the public repos that are hosted by your organization on your **GitHub** but meanwhile after I submit this proposal, I will look into the existing issues and create solutions for them in all the repos hosted by Open Healthcare Network.

I am providing a link to a google doc that will contain any updates I make regarding contributions in form of PR to the repositories in real-time.

https://docs.google.com/document/d/1KRhBEr2zTOUmUCECl7hO5vjfU9SQ3A621VJR9A_Lb3E/edit?usp=sharing