A PROJECT REPORT ON

“CHATBOT FOR HEALTHCARE SYSTEM USING AI”

By-

ADITI R PILLAI (4BD19CS006)

INDEX

ABSTRACT

AI is now a days increasingly being used in healthcare. Here AI-based chatbot system can act as automated chatbot system,capable of abet health, providing education and potentially promoting behaviour change.

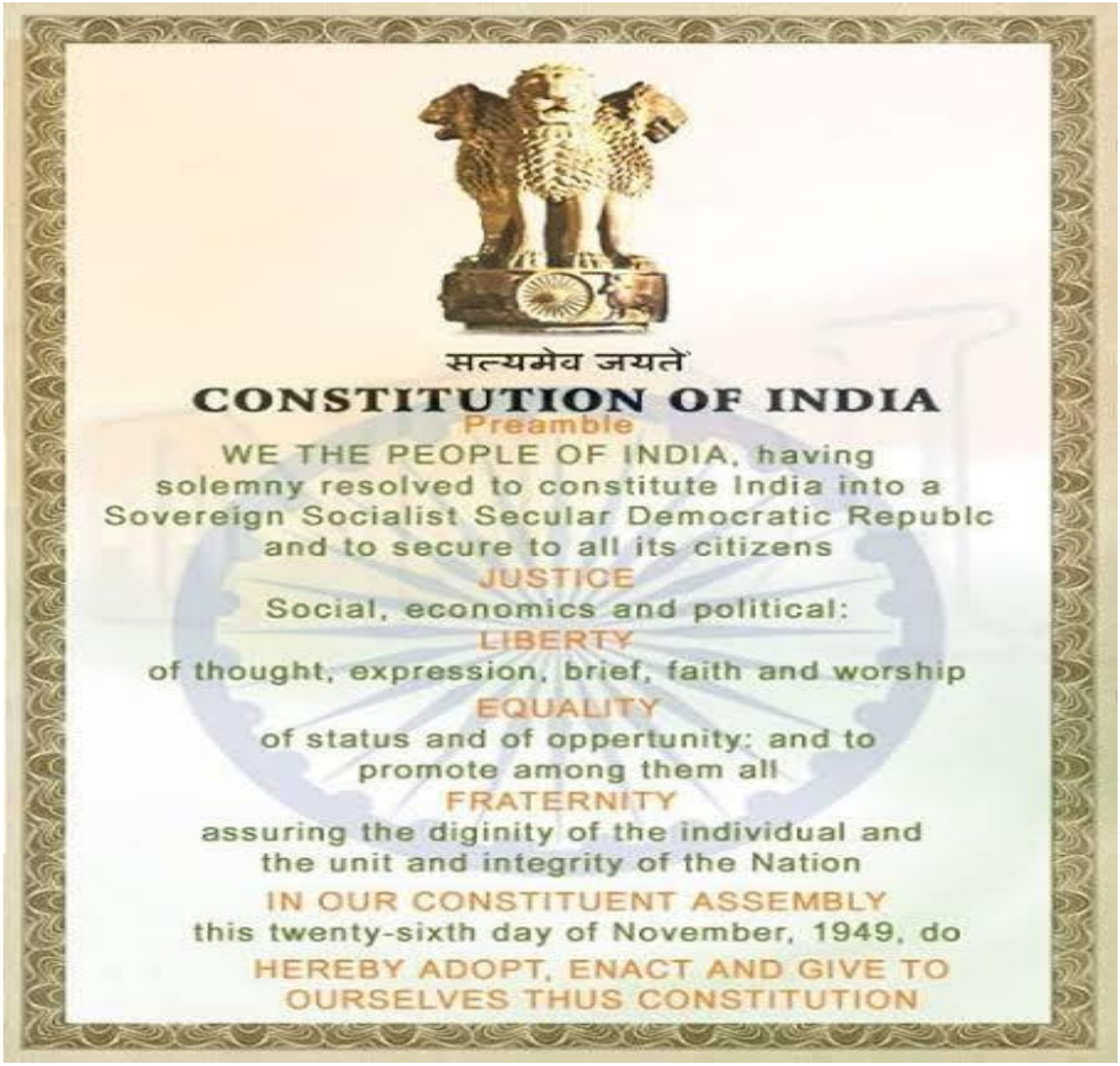
Through chatbots one can communicate with text or voice interface and get reply through AI. Typically, a chatbot will communicate will communicate with a real person. Chatbot are used in app such as eCommerce customer service call centres and Internet Gaming.

Chatbots are programs built to automatically engage with received messages. They can be programmed to respond the same way each time, to respond differently to message containing certain keyword and even to use machine learning to adapt their responses to fit the situation.

These bots connect with potential patients visiting the site, helping them discover specialists, booking, their appointments and getting them access to the correct treatment.

Thus, Healthcare Chatbots System will help hospitals to provide healthcare support online 24x7, it answers deep as well as general questions.

PREAMBLE


सत्यमेव जयते
CONSTITUTION OF INDIA
Preamble
WE THE PEOPLE OF INDIA, having
solemny resolved to constitute India into a
Sovereign Socialist Secular Democratic Republc
and to secure to all its citizens
JUSTICE
Social, economics and political:
LIBERTY
of thought, expression, brief, faith and worship
EQUALITY
of status and of oppertunity: and to
promote among them all
FRATERNITY
assuring the diginity of the individual and
the unit and integrity of the Nation
IN OUR CONSTITUENT ASSEMBLY
this twenty-sixth day of November, 1949, do
HEREBY ADOPT, ENACT AND GIVE TO
OURSELVES THUS CONSTITUTION

INTRODUCTION

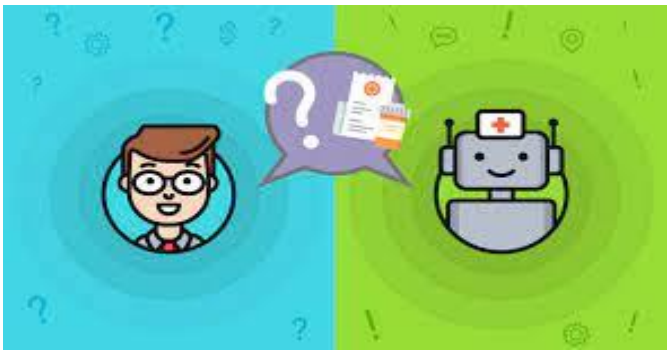
Chatbots, as a part of AI devices, are natural language processing structures performing as a digital conversational agent mimicking human interactions. While this generation remains in its developmental phase, fitness chatbots may want to probably growth get right of entry to to healthcare, enhance doctor–affected person and clinic–affected person verbal exchange, or assist to control the growing call for for fitness offerings consisting of through faraway testing, remedy adherence tracking or teleconsultations. The chatbot generation permits for such sports as precise fitness surveys, putting in place nonpublic fitness-associated reminders, verbal exchange with medical teams, reserving appointments, retrieving and analysing fitness facts or the interpretation of diagnostic styles considering behavioural signs consisting of bodily activity, sleep or nutrition. Such generation may want to probably modify the shipping of healthcare structures, growing uptake.

Computers give us information; they engage us and help us in a lot of manners. A chatbot is a program intended to counterfeit smart communication on a text or speech. These systems can learn themselves and restore their knowledge using human assistance or using web resources. This application is incredibly fundamental since knowledge is stored in advance. The system application uses the question and answer protocol in the form of a chatbot to answer user queries. This system is developed to reduce the healthcare cost and time of the users, as it is not possible for the users to visit the doctors or experts when immediately needed. The response to the question will be replied based on the user query and knowledge base. The significant keywords are fetched from the sentence and answer to those sentences. If the match is discovered or the significant, answer will be given or similar answers will be displayed.This chatbot aims to make a conversation between human and machine. Here the system stores the knowledge database to identify the sentence and making a decision to answer the question. The input sentence will get the similarity score of input sentences using bigram. The chatbot knowledge is stored in RDBMS.

A voice recognition chat-bot is developed in this paper. The questions asked to the bot are not understood are further processed using the expert-system of third parties. The webbots are created as web-friends based on text, a user entertainer. If the program is not only text-based, but also voice-based equipped, they concentrated on the improved system here. Here, a two-part process of capturing and analysing an input signal is required for voice recognition. Recognition of data from the server response and processing of information. The server used here is a black box

approach based on SOAP. Using an expert system makes it possible to improve unlimited and autonomous intelligence.

The chatbot implemented using pattern comparison in which the order of the sentence is recognized and saved response pattern. Here the author describes the implementation of the chatbot Operating system, software, programming language, database and how results of input and output are stored. Here the input is taken using text() function and other punctuation is removed using trim() function and random() function is used to choose a response from the database. The chatbot is used for an entertainment purpose.



PROBLEM STATEMENT

Artificial intelligence chatbot is a technology that makes interactions between man and machines using natural language possible. From literature, we found out that in general, chatbot are functions like a typical search engine. Although chatbot just produced only one output instead of multiple outputs/results, the basic process flow is the same where each time an input is entered, the new search will be done. Nothing related to previous output. This research is focused on enabling chatbot to become a search engine that can process the next search with the relation to the previous search output. In chatbot context, this functionality will enhance the capability of chatbot's input processing.

SYSTEM DESIGN

The below Figure proceeds with the user login where the users' details will be saved in the database. Then the user can start their conversation with the chatbot and it will be saved in the database for future reference. The chatbot will clarify the user symptoms with serious of queries and also the symptom conformation will be done. The disease will be classified as minor and major disease. Chatbot will reply if it's a major or minor disease. If it's a major one user will be recommended with the doctor details for further treatment.

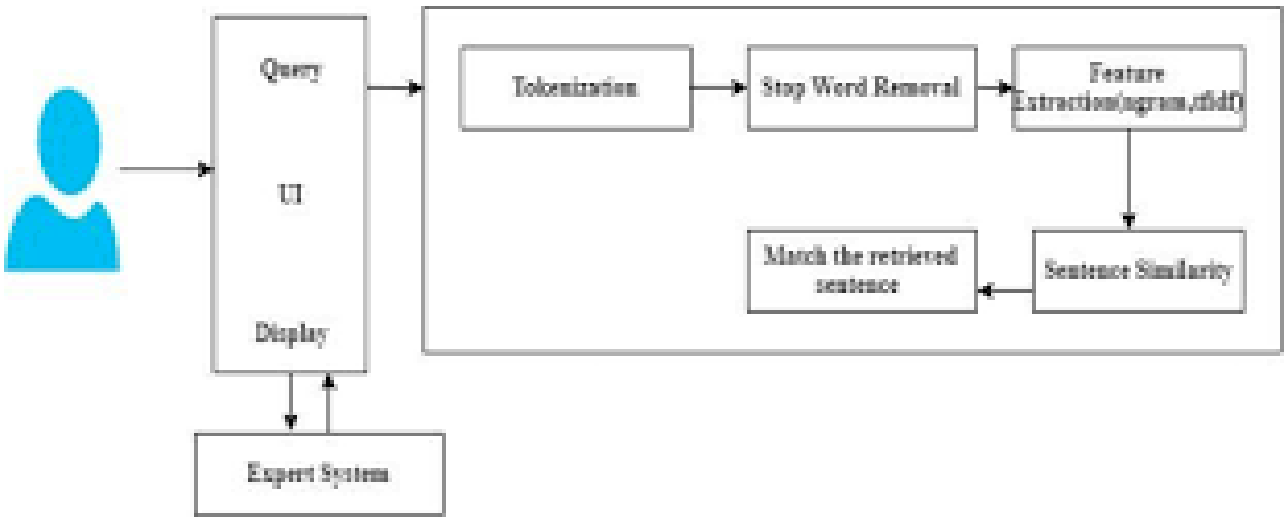


Fig. 1: System architecture

SOFTWARE REQUIREMENT SPECIFICATION

❖ FUNCTIONAL REQUIREMENTS

Hardware Requirement:

- i3 Processor Based Computer or higher
- Memory: 1 GB
- Hard Drive: 50 GB
- Monitor
- Internet Connection

❖ Non-Functional Requirements:

Software Quality Attributes

- Robustness
- Reliability
- Better learning methods
- Acquiring good accuracy results

❖ Software Requirement:

- Windows 7 or higher
- WAMP Server
- Notepad++
- My SQL 5.6
- Google Chrome Browser

IMPLEMENTATION

AES:- The Advanced Encryption Standard (AES) is a symmetric block cipher chosen by the U.S. government to protect classified information. AES is implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, cyber security and electronic data protection. AES has been adopted by the U.S. government and is now used worldwide. It supersedes the Data Encryption Standard(DES), which was published in 1977. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. AES is based on a design principle known as a substitution–permutation network, and is efficient in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael, with a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, Rijndael per se is specified with block and key sizes that may be any multiple of 32 bits, with a minimum of 128 and a maximum of 256 bits. II. Hashing and mapping:- A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. The values are used to index a fixed-size table called a hash table. Use of a hash function to index a hash table is called hashing or scatter storage addressing. A map is a symbolic depiction emphasizing relationships between elements of some space, such as objects, regions, or themes. Many maps are static, fixed to paper or some other durable medium, while others are dynamic or interactive. Although most commonly used to depict geography, maps may represent any space, real or fictional, without regard to context or scale, such as in brain mapping, DNA mapping, or computer network topology mapping. The space being mapped may be two dimensional, such as the surface of the earth, three dimensional, such as the interior of the earth, or even more abstract spaces of any dimension, such as arise in modeling phenomena having many independent variables.

SOFTWARE TESTING

AI-enabled chatbot allows you to input your symptoms and get the most likely diagnoses. Trained with machine learning models that enable the app to give accurate or near-accurate diagnoses. The user will have to be in a text to text communication with the chatbot and get the particular disease and users can also get their previous chat history through their details which are stored in the data.

CODE

```
from tkinter import *
from tkinter import messagebox
import os
import webbrowser

import numpy as np
import pandas as pd

class HyperlinkManager:
    def __init__(self, text):
self.text = text
self.text.tag_config("hyper", foreground="blue", underline=1)
self.text.tag_bind("hyper", "<Enter>", self._enter)
self.text.tag_bind("hyper", "<Leave>", self._leave)
self.text.tag_bind("hyper", "<Button-1>", self._click)

self.reset()

    def reset(self):
self.links = {}

    def add(self, action):
        # add an action to the manager.  returns tags to use in
        # associated text widget
        tag = "hyper-%d" % len(self.links)
self.links[tag] = action
        return "hyper", tag
```

```
    def _enter(self, event):
self.text.config(cursor="hand2")

    def _leave(self, event):
self.text.config(cursor="")

    def _click(self, event):
        for tag in self.text.tag_names(CURRENT):
            if tag[:6] == "hyper-":
self.links[tag]()
                Return

# Importing the dataset
training_dataset = pd.read_csv('Training.csv')
test_dataset = pd.read_csv('Testing.csv')

# Slicing and Dicing the dataset to separate features from predictions
X = training_dataset.iloc[:, 0:132].values
Y = training_dataset.iloc[:, -1].values

# Dimensionality Reduction for removing redundancies
dimensionality_reduction = training_dataset.groupby(training_dataset['prognosis']).max()

# Encoding String values to integer constants
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
```

```
y = labelencoder.fit_transform(Y)

# Splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Implementing the Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

# Saving the information of columns
cols     = training_dataset.columns
cols     = cols[:-1]

# Checking the Important features
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
features = cols

# Implementing the Visual Tree
from sklearn.tree import _tree

# Method to simulate the working of a Chatbot by extracting and formulating questions
def print_disease(node):
        #print(node)
        node = node[0]
```

```
        #print(len(node))
val  =node.nonzero()
        #print(val)
        disease = labelencoder.inverse_transform(val[0])
        return disease
def recurse(node, depth):
            global val,ans
            global tree_,feature_name,symptoms_present
            indent = "  " * depth
            if tree_.feature[node] != _tree.TREE_UNDEFINED:
                name = feature_name[node]
                threshold = tree_.threshold[node]
                yield name + " ?"
#                 ans = input()
ans = ans.lower()
                if ans == 'yes':
val = 1
                else:
val = 0
if  val<= threshold:
                    yield from recurse(tree_.children_left[node], depth + 1)
                else:
symptoms_present.append(name)
                    yield from recurse(tree_.children_right[node], depth + 1)
            else:
strData=""
present_disease = print_disease(tree_.value[node])
```

```
#               print( "You may have " +  present_disease )
#               print()
strData="You may have :" +  str(present_disease)
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
red_cols = dimensionality_reduction.columns
symptoms_given = red_cols[dimensionality_reduction.loc[present_disease].values[0].nonzero()]
#               print("symptoms present  " + str(list(symptoms_present)))
#               print()
strData="symptoms present:  " + str(list(symptoms_present))
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
#               print("symptoms given "  +  str(list(symptoms_given)) )
#               print()
strData="symptoms given: "  +  str(list(symptoms_given))
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
#               print("confidence level is " + str(confidence_level))
#               print()
strData="confidence level is: " + str(confidence_level)
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
#               print('The model suggests:')
#               print()
strData='The model suggests:'
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
              row = doctors[doctors['disease'] == present_disease[0]]
#               print('Consult ', str(row['name'].values))
#               print()
strData='Consult '+ str(row['name'].values)
```

```
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
#                print('Visit ', str(row['link'].values))
                #print(present_disease[0])
                hyperlink = HyperlinkManager(QuestionDigonosis.objRef.txtDigonosis)
strData='Visit '+ str(row['link'].values[0])
                def click1():
webbrowser.open_new(str(row['link'].values[0]))
QuestionDigonosis.objRef.txtDigonosis.insert(INSERT, strData, hyperlink.add(click1))
                #QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
                yield strData
def tree_to_code(tree, feature_names):
        global tree_,feature_name,symptoms_present
        tree_ = tree.tree_
        #print(tree_)
feature_name = [
feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
            for i in tree_.feature
        ]
        #print("def tree({}):".format(", ".join(feature_names)))
symptoms_present = []
#        recurse(0, 1)


def execute_bot():
#    print("Please reply with yes/Yes or no/No for the following symptoms")
tree_to_code(classifier,cols)
```

```
# This section of code to be run after scraping the data


doc_dataset = pd.read_csv('doctors_dataset.csv', names = ['Name', 'Description'])



diseases = dimensionality_reduction.index

diseases = pd.DataFrame(diseases)


doctors = pd.DataFrame()

doctors['name'] = np.nan

doctors['link'] = np.nan

doctors['disease'] = np.nan


doctors['disease'] = diseases['prognosis']



doctors['name'] = doc_dataset['Name']

doctors['link'] = doc_dataset['Description']


record = doctors[doctors['disease'] == 'AIDS']

record['name']

record['link']
```

```
# Execute the bot and see it in Action
#execute_bot()



class QuestionDigonosis(Frame):
objIter=None
objRef=None
    def __init__(self,master=None):
master.title("Question")
        # root.iconbitmap("")
master.state("z")
#        master.minsize(700,350)
QuestionDigonosis.objRef=self
        super().__init__(master=master)
self["bg"]="light blue"
self.createWidget()
self.iterObj=None


    def createWidget(self):
self.lblQuestion=Label(self,text="Question",width=12,bg="bisque")
self.lblQuestion.grid(row=0,column=0,rowspan=4)


self.lblDigonosis = Label(self, text="Digonosis",width=12,bg="bisque")
self.lblDigonosis.grid(row=4, column=0,sticky="n",pady=5)


        # self.varQuestion=StringVar()
self.txtQuestion = Text(self, width=100,height=4)
```

```
self.txtQuestion.grid(row=0, column=1,rowspan=4,columnspan=20)

self.varDiagonosis=StringVar()
self.txtDigonosis =Text(self, width=100,height=14)
self.txtDigonosis.grid(row=4, column=1,columnspan=20,rowspan=20,pady=5)

self.btnNo=Button(self,text="No",width=12,bg="bisque", command=self.btnNo_Click)
self.btnNo.grid(row=25,column=0)
self.btnYes = Button(self, text="Yes",width=12,bg="bisque", command=self.btnYes_Click)
self.btnYes.grid(row=25, column=1,columnspan=20,sticky="e")

self.btnClear = Button(self, text="Clear",width=12,bg="bisque", command=self.btnClear_Click)
self.btnClear.grid(row=27, column=0)
self.btnStart = Button(self, text="Start",width=12,bg="bisque", command=self.btnStart_Click)
self.btnStart.grid(row=27, column=1,columnspan=20,sticky="e")
    def btnNo_Click(self):
        global val,ans
        global val,ans
ans='no'
str1=QuestionDigonosis.objIter.__next__()
self.txtQuestion.delete(0.0,END)
self.txtQuestion.insert(END,str1+"\n")
    def btnYes_Click(self):
        global val,ans
ans='yes'
self.txtDigonosis.delete(0.0,END)
str1=QuestionDigonosis.objIter.__next__()
```

```
#       self.txtDigonosis.insert(END,str1+"\n")
    def btnClear_Click(self):
self.txtDigonosis.delete(0.0,END)
self.txtQuestion.delete(0.0,END)
    def btnStart_Click(self):
execute_bot()
self.txtDigonosis.delete(0.0,END)
self.txtQuestion.delete(0.0,END)
self.txtDigonosis.insert(END,"Please Click on Yes or No for the Above symptoms in Question")
QuestionDigonosis.objIter=recurse(0, 1)
str1=QuestionDigonosis.objIter.__next__()
self.txtQuestion.insert(END,str1+"\n")



class MainForm(Frame):
main_Root = None
    def destroyPackWidget(self, parent):
        for e in parent.pack_slaves():
e.destroy()
    def __init__(self, master=None):
MainForm.main_Root = master
        super().__init__(master=master)
master.geometry("300x250")
master.title("Account Login")
self.createWidget()
    def createWidget(self):
self.lblMsg=Label(self, text="Health Care Chatbot", bg="PeachPuff2", width="300", height="2",
font=("Calibri", 13))
```

```
self.lblMsg.pack()

self.btnLogin=Button(self, text="Login", height="2", width="300", command = self.lblLogin_Click)

self.btnLogin.pack()

self.btnRegister=Button(self, text="Register", height="2", width="300", command = self.btnRegister_Click)

self.btnRegister.pack()

self.lblTeam=Label(self, text="Made by:", bg="slateblue4", width = "250", height = "1", font=("Calibri", 13))

self.lblTeam.pack()

self.lblTeam1=Label(self, text="Aditi R Pillai", bg="RoyalBlue1", width = "250", height = "1", font=("Calibri", 13))

self.lblTeam1.pack()

self.lblTeam2=Label(self, text="Soundarya P", bg="RoyalBlue2", width = "250", height = "1", font=("Calibri", 13))

self.lblTeam2.pack()

self.lblTeam3=Label(self, text="T Ruby", bg="RoyalBlue3", width = "250", height = "1", font=("Calibri", 13))

self.lblTeam3.pack()

    def lblLogin_Click(self):

self.destroyPackWidget(MainForm.main_Root)

frmLogin=Login(MainForm.main_Root)

frmLogin.pack()

    def btnRegister_Click(self):

self.destroyPackWidget(MainForm.main_Root)

frmSignUp = SignUp(MainForm.main_Root)

frmSignUp.pack()
```

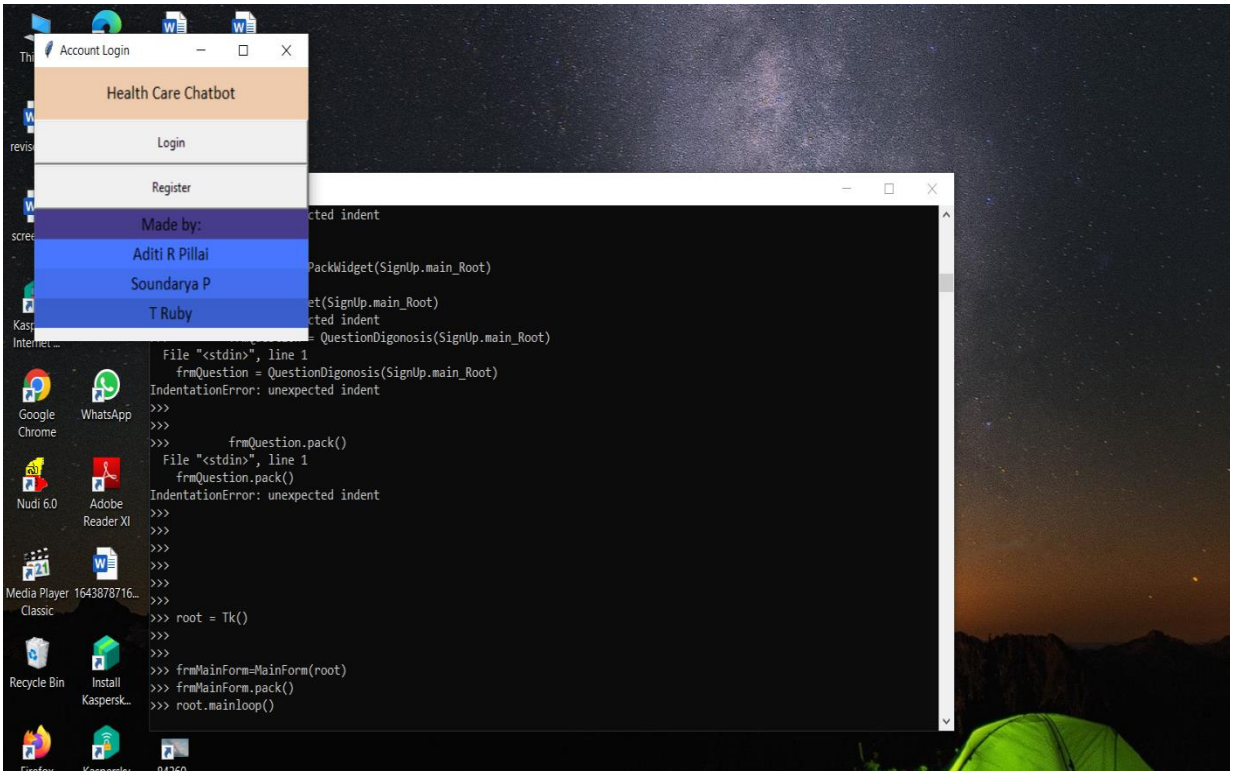
```
class Login(Frame):
main_Root=None
    def destroyPackWidget(self,parent):
        for e in parent.pack_slaves():
e.destroy()
    def __init__(self, master=None):
Login.main_Root=master
        super().__init__(master=master)
master.title("Login")
master.geometry("300x250")
self.createWidget()
    def createWidget(self):
self.lblMsg=Label(self, text="Please enter details below to login",bg="blue")
self.lblMsg.pack()
self.username=Label(self, text="Username * ")
self.username.pack()
self.username_verify = StringVar()
self.username_login_entry = Entry(self, textvariable=self.username_verify)
self.username_login_entry.pack()
self.password=Label(self, text="Password * ")
self.password.pack()
self.password_verify = StringVar()
self.password_login_entry = Entry(self, textvariable=self.password_verify, show='*')
self.password_login_entry.pack()
self.btnLogin=Button(self, text="Login", width=10, height=1, command=self.btnLogin_Click)
self.btnLogin.pack()
    def btnLogin_Click(self):
```

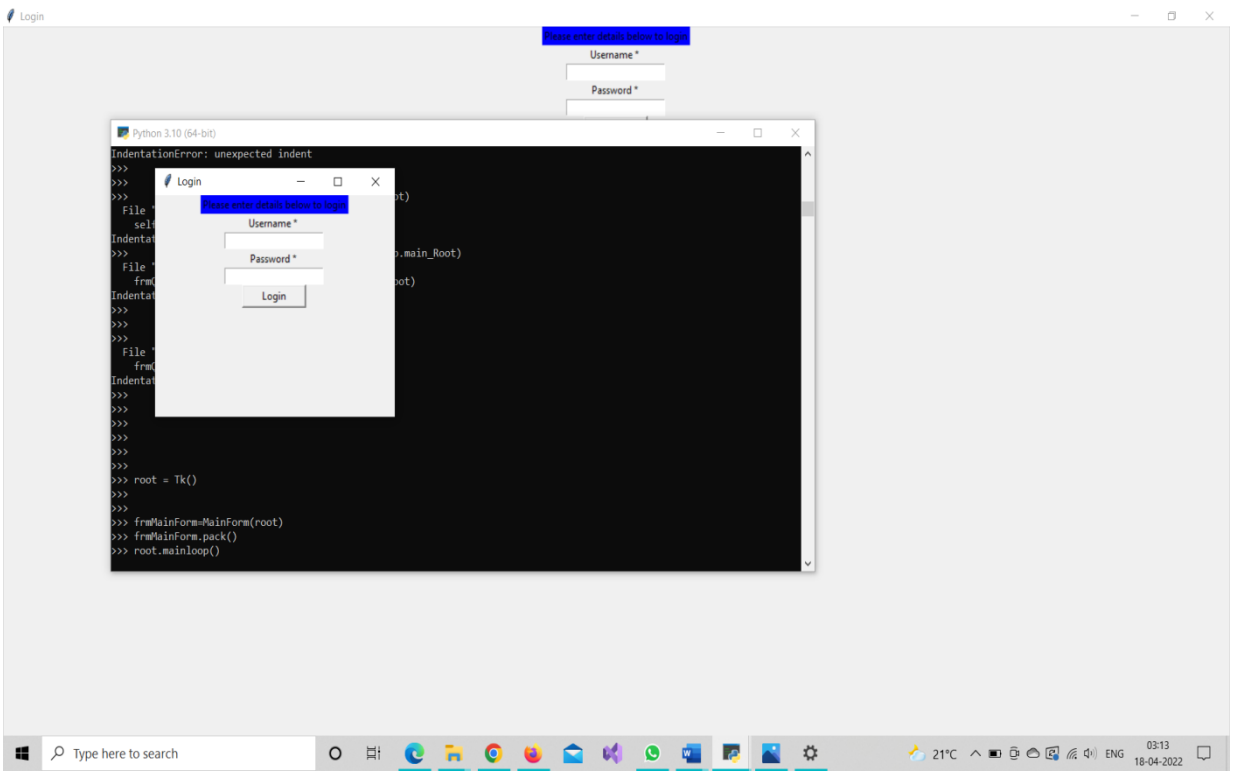
```
        username1 = self.username_login_entry.get()
        password1 = self.password_login_entry.get()
#        messagebox.showinfo("Failure", self.username1+":"+password1)
list_of_files = os.listdir()
        if username1 in list_of_files:
            file1 = open(username1, "r")
            verify = file1.read().splitlines()
            if password1 in verify:
messagebox.showinfo("Sucess","Login Sucessful")
self.destroyPackWidget(Login.main_Root)
frmQuestion = QuestionDigonosis(Login.main_Root)
frmQuestion.pack()
            else:
messagebox.showinfo("Failure", "Login Details are wrong try again")
        else:
messagebox.showinfo("Failure", "User not found try from another user\n or sign up for new user")



class SignUp(Frame):
main_Root=None
print("SignUp Class")
    def destroyPackWidget(self,parent):
        for e in parent.pack_slaves():
e.destroy()
SignUp.main_Root=master
self.lblMsg=Label(self, text="Please enter details below", bg="blue")
self.lblMsg.pack()
```

```
self.username_lable = Label(self, text="Username * ")
self.username_lable.pack()
self.username_entry = Entry(self, textvariable=self.username)
self.username_entry.pack()
self.password_lable = Label(self, text="Password * ")
self.password_lable.pack()
self.password = StringVar()
self.password_entry = Entry(self, textvariable=self.password, show='*')
self.password_entry.pack()
Self.btnRegister=Button(self,text="Register", width=10, height=1, bg="blue", command=self.register_user)
        self.btnRegister.pack()



    def register_user(self):
        file = open(self.username_entry.get(), "w")
        file.write(self.username_entry.get() + "\n")
        file.write(self.password_entry.get())
        file.close()

        self.destroyPackWidget(SignUp.main_Root)

        self.lblSucess=Label(root, text="Registration Success", fg="green", font=("calibri", 11))
        self.lblSucess.pack()

        self.btnSucess=Button(root, text="Click Here to proceed", command=self.btnSucess_Click)
        self.btnSucess.pack()
    def btnSucess_Click(self):
```

```
        self.destroyPackWidget(SignUp.main_Root)
        frmQuestion = QuestionDigonosis(SignUp.main_Root)

        frmQuestion.pack()



root = Tk()

frmMainForm=MainForm(root)
frmMainForm.pack()
root.mainloop()
```

OUTPUT


Account Login
Health Care Chatbot
Login
Register
Made by:
Aditi R Pillai
Soundarya P
T Ruby
File "<stdin>", line 1
frmQuestion = QuestionDigonosis(SignUp.main_Root)
IndentationError: unexpected indent
>>>
>>>
>>> frmQuestion.pack()
File "<stdin>", line 1
frmQuestion.pack()
IndentationError: unexpected indent
>>>
>>>
>>>
>>>
>>>
>>>
>>> root = Tk()
>>>
>>>
>>> frmMainForm=MainForm(root)
>>> frmMainForm.pack()
>>> root.mainloop()


Login
Please enter details below to login
Username *
Password *
Python 3.10 (64-bit)
IndentationError: unexpected indent
Login
Please enter details below to login
Username *
Password *
Login
>>> root = Tk()
>>> frmMainForm=MainForm(root)
>>> frmMainForm.pack()
>>> root.mainloop()
Type here to search
21°C
ENG
03:13
18-04-2022

CONCLUSION

It determined that the modern chatbots perform at a very high standard to provide a reliable response to users compared to the traditional chatbots. Unlike existing chatbots which focused on various domains of healthcare. This is the best solution for peoplewho are busy with their job schedules. They do not need to wait in the queue for hours to get an appointment with a doctor every time instead they can chat with the bot.

The usage of Chatbot is user friendly and can be used by anyone who knows how to type in their own language in mobile app or desktop version. A medical chatbot provides personalized diagnoses based on symptoms. In the future, the bot's symptom recognition and diagnosis performance could be much improved by adding support for more medical features, for instance location, duration, and intensity of symptoms, and more detailed symptom description.

REFERENCES

[1] (Parker et al., 2001)Parker, P. A., Baile, W. F., De Moor, C., Lenzi, R., Kudelka, A. P., & Cohen, L. (2001). Breaking bad news about cancer: Patients' preferences for communication. Journal of Clinical Oncology, 19(7), 2049–2056. https://doi.org/10.1200/JCO.2001.19.7.2049.

[2] (Rarhi, Bhattacharya, Mishra, & Mandal, 2017)Cameron, G., Cameron, D., Megaw, G., Bond, R., Mulvenna, M., Neill, S. O., … McTear, M. (2018). Best Practices for Designing Chatbots in Mental Healthcare – A Case Study on iHelpr. Proceedings of British HCI 2018, 1–5

[3] Rarhi, K., Bhattacharya, A., Mishra, A., & Mandal, K. (2017). Automated Medical Chatbot.

SSRN Electronic Journal, 0–2. https://doi.org/10.2139/ssrn.3090881