PROJECT REPORT

HEALTH CARE CHAT BOT

Submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING IN
COMPUTER SCIENCE & ENGINEERING

Submitted to:
Ms. Jasneet Kaur

Submitted By:
Amit Kumar [19BCS1409]
Arun Gosain [19BCS1413]
Kishan Kumar [19BCS1421]
Preeti [19BCS1423]

Mentor Signature: Ms. Jasneet Kaur

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Chandigarh University, Gharua

CERTIFICATE

This is to certify that the work embodied in this Project Report entitled **“ HEALTH CARE CHAT BOT ”** being submitted by **“AMIT KUMAR – 19BCS1409”** , **“ARUN GOSAIN – 19BCS1413”** , **“KISHAN KUMAR – 19BCS1421”** , and **“PREETI – 19BCS1423” , 5th** Semester for partial fulfilment of the requirement for the degree of **“ Bachelor of Engineering in Computer Science & Engineering ”** discipline in **“Chandigarh University”** during the academic session AUG 2021 - DEC 2021 is a record of debonair piece of work, carried out by student under my supervision and guidance in the **“ Department of Computer Science & Engineering ”, Chandigarh University.**

The report has been approved as it satisfies the academic requirements in respect of mini-project work prescribed for the course.

DECLARATION

Me and my team, are the students of **Bachelor of Engineering** in **Computer Science & Engineering, 5th Semester**, **session: Aug 2021 – Dec 2021, Chandigarh University,** hereby declare that the work presented in this Project Report entitled **"Health Care Chat Bot"** is the outcome of my own work, is debonair and correct to the best of my knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe any patented work and has not been submitted to any other university or anywhere else for the award of any degree or any professional diploma.

Student details and Signature

Sr. No.	UID	Name	Signature
1	19BCS1409	**Amit Kumar**	[illegible]
2	19BCS1413	**Arun Gosain**	[illegible]
3	19BCS1421	**Kishan Kumar**	Kishan
4	19BCS1423	**Preeti**	Preeti

TABLE OF CONTENT

ACKNOWLEDGEMENT

No project is ever complete without the guidance of those expert who have already traded this past before and hence become master of it and as a result, our leader. So, we would like to take this opportunity to take all those individuals who have helped us in visualizing this project.

We would take this opportunity to thank our project teacher **Ms. Jasneet kaur** and other staff for their guidance in selecting this project and also for providing timely assistant to our query and provide guidance of this project.

We are really thankful to all our Professors from **Chandigarh University** for their valuable inside and tip during the designing of the project. Their contributions have been valuable in so many ways that we find it difficult to acknowledge of them individual.

Thanking You.

ABSTRACT

The main aim of project —**HEALTH CARE CHATBOT**, is to help you better visualize the presentation of mined data (information). It deals with all the health care issues which will really benefit stakeholders in the health care space.

Basically, chat-bot is a computer program that pretends chat with humans through natural language. On any platform like mobile, website and desktop application, this system can interact with the humans. While interacting with the human, chat-bot simulates as a human being. Human being only interacts with one human at a time, the chat-bot interacts and communicates with hundreds and thousands of persons simultaneously. It works and responds without considering how many persons are interacting and what time of the day and night it is.

Chat-bots in the health care sector can play an important role. Chat-bot algorithms can be trained on massive healthcare data using disease symptoms, diagnostics, markers, and available treatments. Chat-bots can be updated continuously using Public datasets, such as COVID-19 for COVID-19, and Wisconsin Breast Cancer Diagnosis (WBCD).

Conversational chat-bots with different intelligence levels can understand the questions from the users and provide answers based on pre - defined labels in the training data.

INTRODUCTION

Through chat bots one can communicate with text or voice interface and get reply through artificial intelligence. Typically, a chat bot will communicate with a real person. Chat bots are used in applications such as E-commerce customer service, call centers and Internet gaming.

Chat bots are programs built to automatically engage with received messages.

Chat bots can be programmed to respond the same way each time, to respond differently to messages containing certain keywords and even to use machine learning to adapt their responses to fit the situation. A developing number of hospitals, nursing homes, and even private centers, presently utilize online Chat bots for human services on their sites. These bots connect with potential patients visiting the site, helping them discover specialists, booking their appointments, and getting them access to the correct treatment.

An ML model has to be created wherein we could give any text input and on the basis of training data it must analyze the symptoms. A Supervised Logistic Regression machine learning algorithm can be implemented to train the model with data sets containing various diseases CSV files. The goal is to compare outputs of various models and suggest the best model that can be used for symptoms in real-world inputs. Data set contains CSV file having all diseases compiled together. The logistic regression algorithm in ML allows us to process the data efficiently. The goal here is to model the underlying structure or distribution of the data in order to learn more from the training set.

In any case, the utilization of artificial intelligence in an industry where individuals' lives could be in question, still starts misgivings in individuals. It brings up issues about whether the task mentioned above ought to be assigned to human staff. This healthcare chat bot system will help hospitals to provide healthcare support online 24 x 7, it answers deep as well as general questions. It also helps to generate leads and automatically delivers the information of leads to sales. By asking the questions in series it helps patients by guiding what exactly he/she is looking for.

1.1. Purpose and Scope

Almost everyone kept on hold while operators connect you to a customer care executive. On an average people spend around 7 minutes until they are assigned to a person. Gone are the frustrating days of waiting in a queue for the next available operative. They are replacing live chat and other forms of slower contact methods such as E-mails and phone calls. Since chat bots are basically virtual robots they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break.

1.2. Problem Statement

Through chat bots one can communicate with text or voice interface and get reply through artificial intelligence. Typically, a chat bot will communicate with a real person. Chat bots are used in applications such as E-commerce customer service, call centres and Internet gaming.

Chat bots are programs built to automatically engage with received messages.

Chat bots can be programmed to respond the same way each time, to respond differently to messages containing certain keywords and even to use machine learning to adapt their responses to fit the situation. A developing number of hospitals, nursing homes, and even private centers, presently utilize online Chat bots for human services on their sites. These bots connect with potential patients visiting the site, helping them discover specialists, booking their appointments, and getting them access to the correct treatment.

An ML model has to be created wherein we could give any text input and on the basis of training data it must analyze the symptoms. A Supervised Logistic Regression machine learning algorithm can be implemented to train the model with data sets containing various diseases CSV files. The goal is to compare outputs of various models and suggest the best model that can be used for symptoms in real-world inputs. Data set contains CSV file having all diseases compiled together. The logistic regression algorithm in ML allows us to process the data efficiently. The goal here is to model the underlying structure or distribution of the data in order to learn more from the training set.

In any case, the utilization of artificial intelligence in an industry where individuals' lives could be in question, still starts misgivings in individuals. It brings up issues about whether the task mentioned above ought to be assigned to human staff. This healthcare chat bot system will help hospitals to provide healthcare support online 24 x 7, it answers deep as well as general questions. It also helps to generate leads and automatically delivers the information of leads to sales. By asking the questions in series it helps patients by guiding what exactly he/she is looking for.

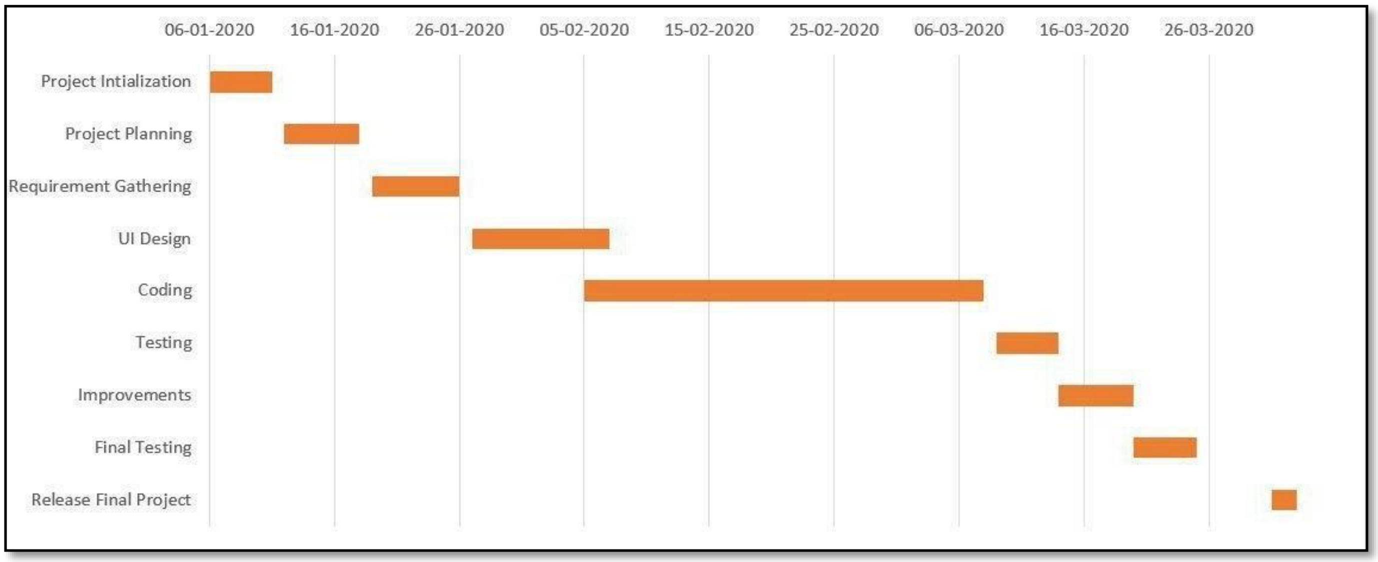
PROJECT ANALYSIS

2.1 Review of Literature

The main purpose of the scheme is to build the language gap between the user and health providers by giving immediate replies to the Questions asked by the user. Today's people are more likely addicted to the internet but they are not concerned about their personal health. They avoid going to hospital for small problems which may become a major disease in future.

Establishing question answer forums is becoming a simple way to answer those queries rather than browsing through the list of potentially relevant documents from the web. Many of the existing systems have some limitations such as there is no instant response given to the patients they have to wait for experts to acknowledge for a long time. Some of the processes may charge an amount to perform live chat or telephony communication with doctors online. The aim of this system is to replicate a person's discussion.

2.2 Project Timeline

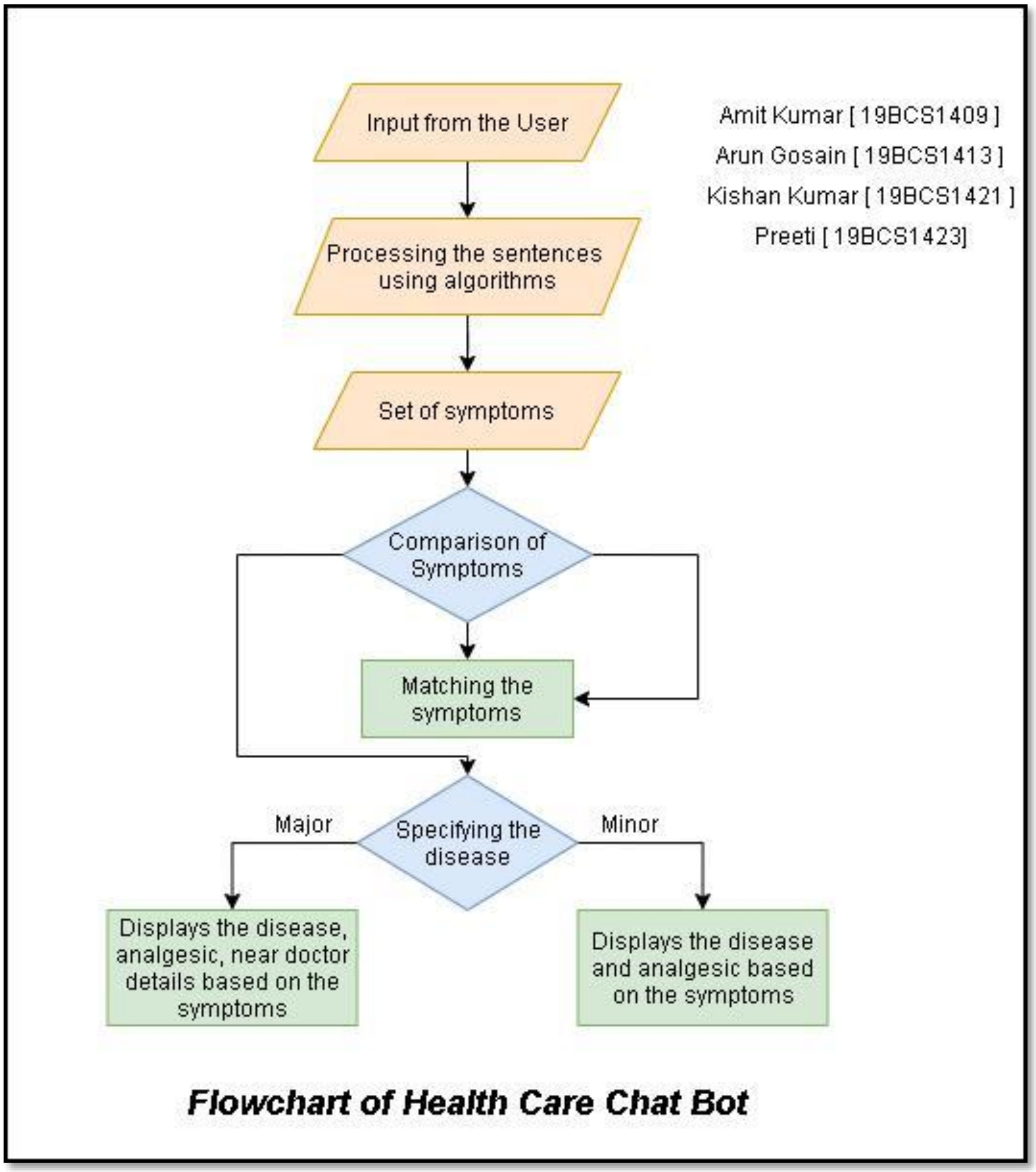


2.3 Data Set Details

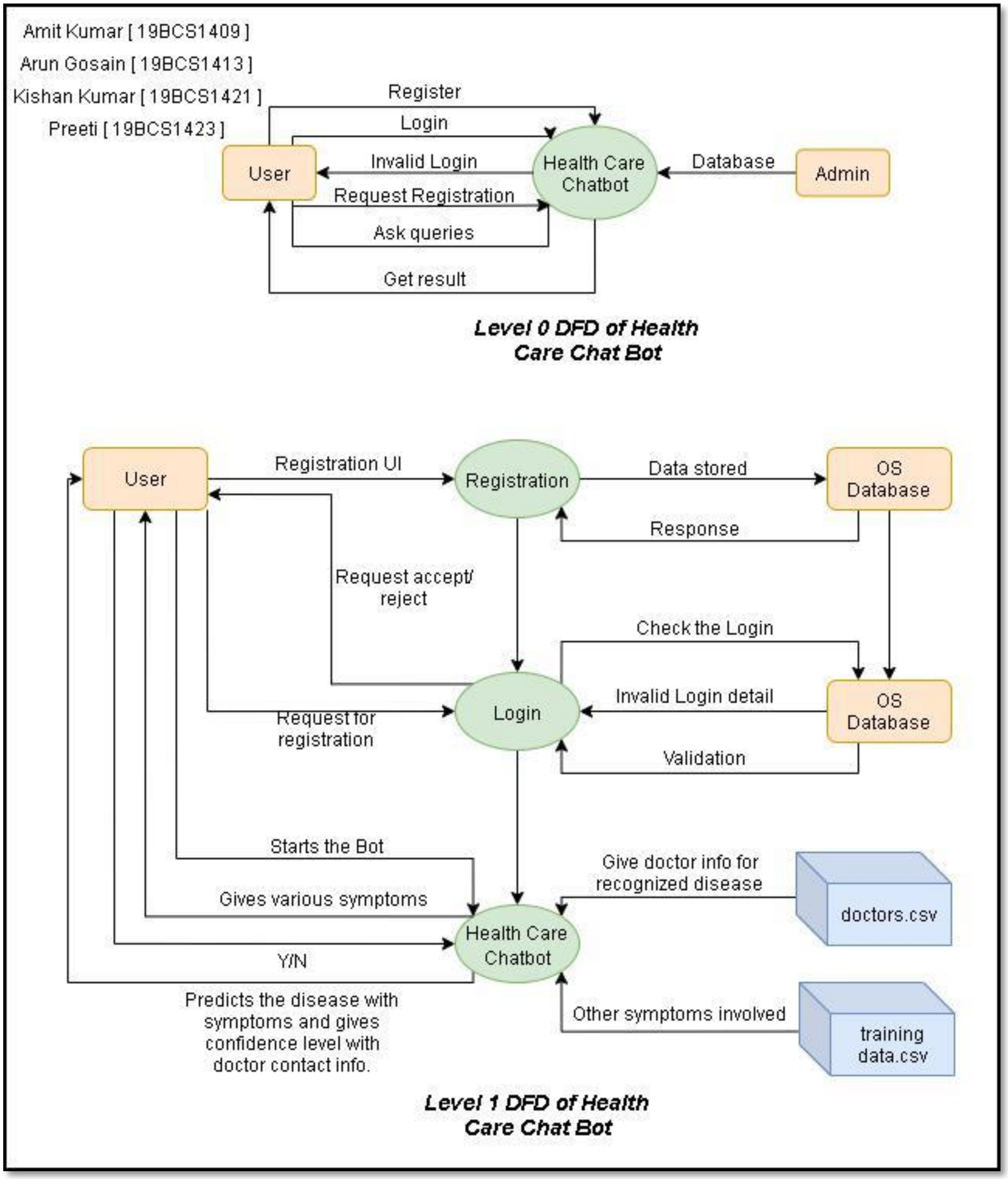
Data Set contains description of different types of diseases. There are different sets of different types of diseases. These sets consists of descriptions of a single disease with different doctors, hospitals, etc. A Data Set has been created by recording sequences from over 133 number of diseases and doctors and hospitals.

PROJECT DESIGN

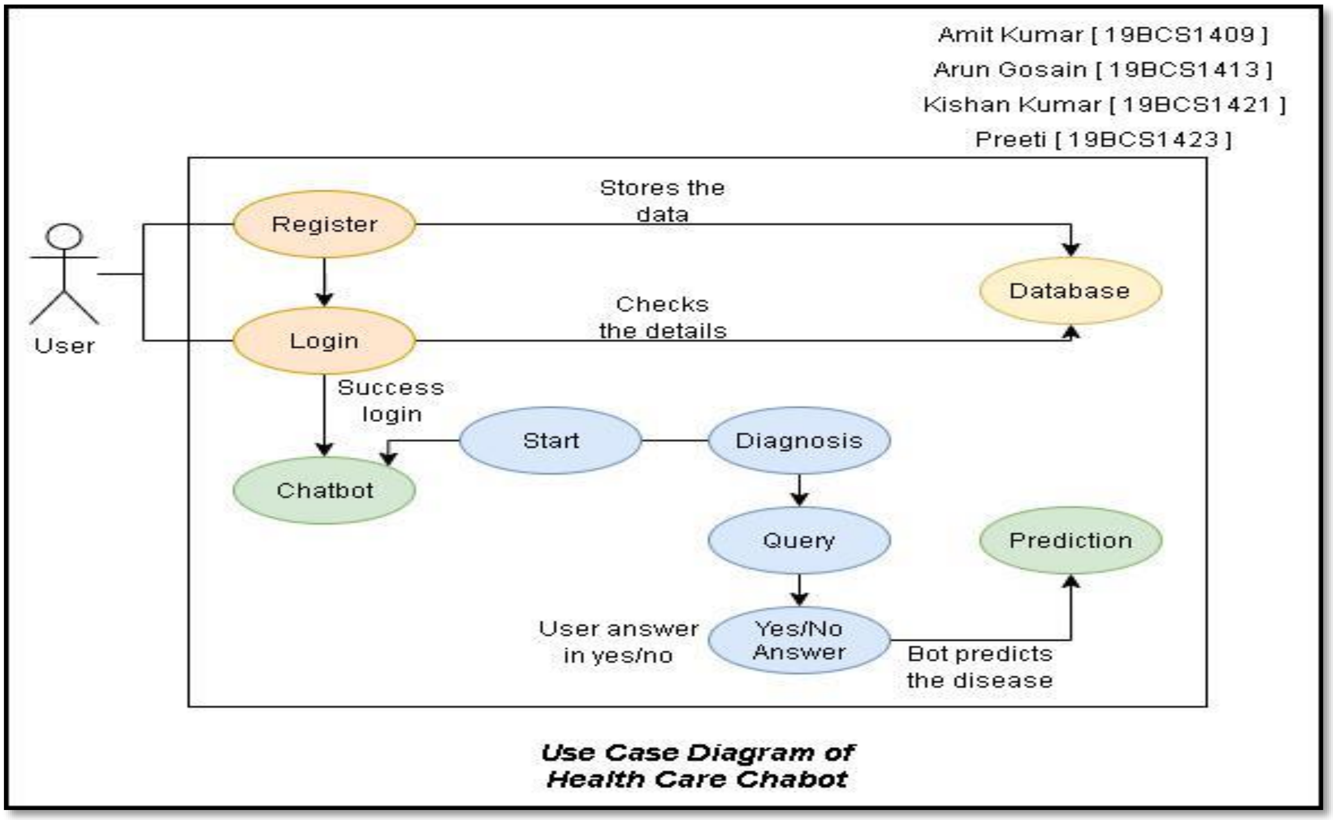
3.1. Flowchart


Input from the User
Processing the sentences using algorithms
Set of symptoms
Comparison of Symptoms
Matching the symptoms
Specifying the disease
Major
Minor
Displays the disease, analgesic, near doctor details based on the symptoms
Displays the disease and analgesic based on the symptoms
Amit Kumar [19BCS1409]
Arun Gosain [19BCS1413]
Kishan Kumar [19BCS1421]
Preeti [19BCS1423]
Flowchart of Health Care Chat Bot

3.2. Data Flow Diagram



3.3. Use Case Diagram



IMPLEMENTATION

4.1. Project Implementation Technology

In machine learning, **support-vector machines** (**SVMs**, also **support-vector networks**) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high- dimensional feature spaces.

4.2. Hardware Requirements

In recent years, a great variety of hardware solutions for real-time TSR has been proposed. These include conventional (general purpose) computers, custom ASIC (application-specific integrated circuit) chips, field programmable gate arrays (FPGAs), digital sign processors (DSPs) and also graphic processing units

4.3. Software Requirements

In a software-based solution running on a Linux or window system with a 2.4-GHz dual core CPU is presented.

4.4. Experimental Setup

The main purpose of the scheme is to build the language gap between the user and health providers by giving immediate replies to the Questions asked by the user. Today’s people are more likely addicted to the internet but they are not concerned about their personal health. They avoid going to hospital for small problems which may become a major disease in future. Establishing question answer forums is becoming a simple way to answer those queries rather than browsing through the list of potentially relevant documents from the web. Many of the existing systems have some limitations such as there is no instant response given to the patients they have to wait for experts to acknowledge for a long time. Some of the processes may charge an amount to perform live chat or telephony communication with doctors online. The aim of this system is to replicate a person’s discussion.

CODING/ IMPLEMENTATION

```
# Importing the libraries
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-m
        # 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
```

```python
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].non
zero()]
#                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.maxsize(900,360)
        master.minsize(900,315)
        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=3)
        self.txtQuestion.grid(row=0,
column=1,rowspan=4,columnspan=20)

        self.varDiagonosis=StringVar()
        self.txtDigonosis =Text(self, width=100,height=12)
        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.btnClear = Button(self,
text="Clear",width=12,bg="bisque", command=self.btnClear_Click)
        self.btnClear.grid(row=25,
column=1,columnspan=20,sticky="e")

        self.btnYes = Button(self, text="Yes ",width=12,bg="bisque",
command=self.btnYes_Click)
        self.btnYes.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("300x260")
        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="Amit Kumar",
bg="RoyalBlue1", width = "250", height = "1", font=("Calibri", 13))
        self.lblTeam1.pack()
        self.lblTeam2=Label(self, text="Arun Gosain",
bg="RoyalBlue2", width = "250", height = "1", font=("Calibri", 13))
        self.lblTeam2.pack()
        self.lblTeam3=Label(self, text="Kishan Kumar",
bg="RoyalBlue3", width = "250", height = "1", font=("Calibri", 13))
        self.lblTeam3.pack()
        self.lblTeam3 = Label(self, text="Preeti", bg="RoyalBlue4",
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("300x260")
        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()
    def __init__(self, master=None):
        SignUp.main_Root=master
        master.title("Register")
        super().__init__(master=master)
        master.title("Register")
        master.geometry("300x260")
        self.createWidget()
    def createWidget(self):
        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 = StringVar()
        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()
```

5.1. Testing

Without a well-thought testing effort, the project will undoubtedly fail overall and will impact the entire operational performance of the solution. With a poorly tested solution, the support and maintenance cost will escalate exponentially, and the reliability of the solution will be poor.

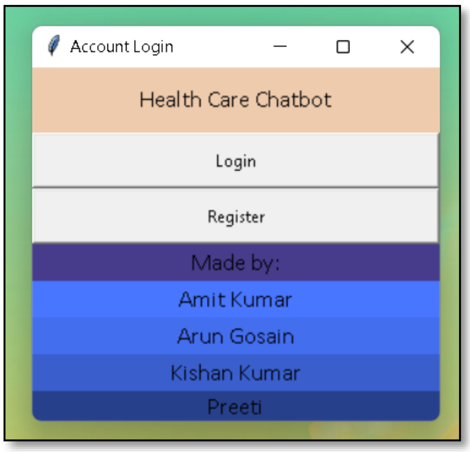
Therefore, project managers need to realize that the testing effort is a necessity, not merely as an ad hoc task that is the last hurdle before deployment.

The project manager should pay specific attention to developing a complete testing plan and schedule. At this stage, the project manager should have realized that this effort would have to be accommodated within the project budget, as many of the testing resources will be designing, testing, and validating the solution throughout the entire project life cycle—and this consumes work-hours and resources.

The testing effort begins at the initial project phase (i.e. preparing test plans) and continues throughout until the closure phase.

RESULTS

6.1. Screenshot of the Output



Account Login

Health Care Chatbot

Login

Register

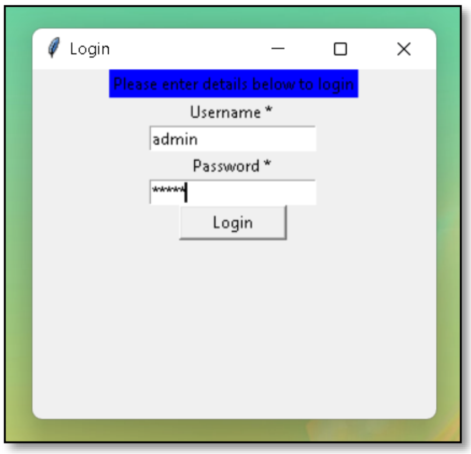
Made by:

Amit Kumar

Arun Gosain

Kishan Kumar

Preeti



Login

Please enter details below to login

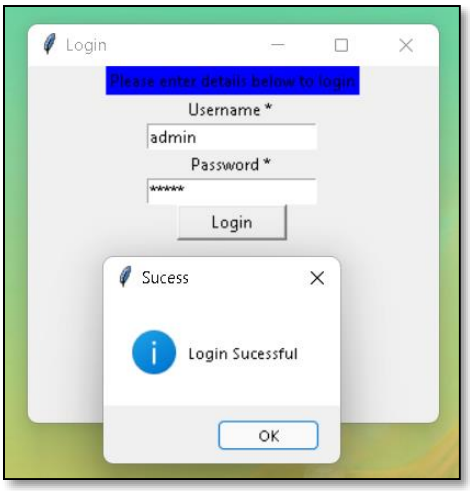
Username *

admin

Password *

******

Login



Login

Please enter details below to login

Username *

admin

Password *

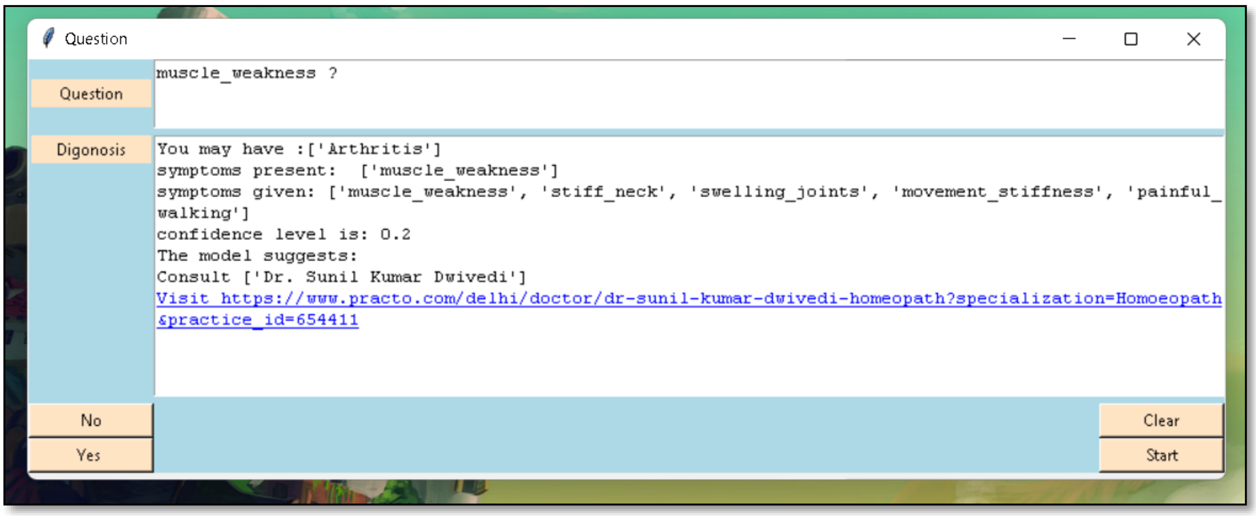
******

Login

Sucess

Login Sucessful

OK



Question

Question

```
muscle_weakness ?
```

Digonosis

```
You may have :['Arthritis']
symptoms present:  ['muscle_weakness']
symptoms given: ['muscle_weakness', 'stiff_neck', 'swelling_joints', 'movement_stiffness', 'painful_
walking']
confidence level is: 0.2
The model suggests:
Consult ['Dr. Sunil Kumar Dwivedi']
Visit https://www.practo.com/delhi/doctor/dr-sunil-kumar-dwivedi-homeopath?specialization=Homoeopath
&practice_id=654411
```

ADVANTAGES & DISADVANTAGES OF THE MODEL

7.1. Advantages

- **Omani-capable**

  The chat bot converses seamlessly across multiple digital channels and retains data and context for a seamless experience. In best cases, even passing that information to a live agent if needed.

- **Free to Explore**

  The chat bot can reach, consume, and process vast amounts of data– both structured and unstructured–to surface insights from any source - to gather relevant data to solve customer issues quickly.

- **Autonomous Reasoning**

  The chat bot can perform complex reasoning without human intervention. For example, a great Service chat bot should be able to infer solutions based on relevant case histories.

- **Pre-Trained**

  The chat bot is pre-trained to understand brand-specific or industry-specific knowledge and terms. Even better, it's pre-configured to resolve common customer requests of a particular industry.

- **Register/Log-in**

  To access this chat bot and individual needs to register and then use the registration ID to log in to access the features.

- **User Interface**

  A user friendly interface which is engaging and easy to access.

7.2. Disadvantages

- **Complex Interface**

  Chat bots are often seen to be complicated and require a lot of time to understand user's requirement. It is also the poor processing which is not able to filter results in time that can annoy people.

- **Inability to Understand**

  Due to fixed programs, chat bots can be stuck if an unsaved query is presented in front of them. This can lead to customer dissatisfaction and result in loss. It is also the multiple messaging that can be taxing for users and deteriorate the overall experience on the website.

- **Time-Consuming**

  Chat bots are installed with the motive to speed-up the response and improve customer interaction. However, due to limited data-availability and time required for self-updating, this process appears more time-taking and expensive. Therefore, in place of attending several customers at a time, chat bots appear confused about how to communicate with people.

- **Zero Decision Making**

  Chat bots are known for being infamous because of their inability to make decisions. A similar situation has landed big companies like Microsoft etc. in trouble when their chat bot went on making a racist rant. Therefore, it is critical to ensure proper programming of your chat bot to prevent any such incident which can hamper your brand.

- **Poor Memory**

  Chat bots are not able to memorize the past conversation which forces the user to type the same thing again & again. This can be cumbersome for the customer and annoy them because of the effort required. Thus, it is important to be careful while designing chat bots and make sure that the program is able to comprehend user queries and respond accordingly.

FUTURE SCOPE

Chat bots are a thing of the future which is yet to uncover its potential but with its rising popularity and craze among companies, they are bound to stay here for long. Machine learning has changed the way companies were communicating with their customers. With new platforms to build various types of chat bots being introduced, it is of great excitement to witness the growth of a new domain in technology while surpassing the previous threshold.

CONCLUSION

Thus, we can conclude that this system giving the accurate result. As we are using large data set which will ensures the better performance. Thus we build up a system which is useful for people to detect the disease by typing symptoms

REFERENCES

- *https://en.wikipedia.org/wiki/Chatbot*
- *https://en.wikipedia.org/wiki/Disease*
- *https://data-flair.training/blogs/python-chatbot-project/*
- *https://www.youtube.com/playlist?list=PLQVvvaa0QuDdc2k5dwtDTyT9aCja0on8j*