A Major Project Report on

AI BASED CHATBOT FOR HEALTH CARE INFORMATION

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

MALLEBOINA KRANTHI KUMAR (16BD1A057J)

MIRUPALA VISHNU VARDHAN (16BD1A057C)

MACHARLA PANDU PRANAY (16BD1A057D)

LAKAVATH VENU GOPAL (16BD1A056Y)

Under the esteemed guidance of

Mr.Samuel Chepuri

Assistant Professor

Department of CSE

Department of Computer Science and Engineering

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

Approved by AICTE, Affiliated to JNTUH

3-5-1206, Narayanaguda, Hyderabad - 500029

2019 – 2020

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

Approved by AICTE, Affiliated to JNTU, Hyderabad

3-5-1206, Narayanaguda, Hyderabad - 500029.

CERTIFICATE

This is to certify that the project entitled “**AI BASED CHATBOT FOR HEALTH CARE INFORMATION**” being submitted by **Malleboina Kranthi Kumar (16BD1A057J), Mirupala Vishnu Vardhan (16BD1A057C), Macharla Pandu Pranay (16BD1A057D), Lakavath Venu Gopal (16BD1A056Y)** students of **Keshav Memorial Institute of Technology, JNTUH** in partial fulfillment of requirements of the award of the Degree of **Bachelor of Technology in Computer Science and Engineering** as a specialization is a record of bonafide work carried out by them under my guidance and supervision in the academic year 2019 – 2020.

GUIDE	**HOD**
Mr.Samuel Chepuri	**Dr. S. Padmaja**
Assistant Professor	**CSE DEPARTMENT**
CSE DEPARTMENT	

Submitted for the Project Viva Voce examination held on …………….

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project report entitled **"AI BASED CHATBOT FOR HEALTH CARE INFORMATION"** is done in the partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering affiliated to Jawaharlal Nehru Technological University, Hyderabad. This project has not been submitted anywhere else.

MALLEBOINA KRANTHI KUMAR (16BD1A057J)

MIRUPALA VISHNU VARDHAN (16BD1A057C)

MACHARLA PANDU PRANAY (16BD1A057D)

LAKAVATH VENU GOPAL (16BD1A056Y)

ACKNOWLEDGEMENT

We take this opportunity to thank all the people who have rendered their full support to our project work.

We render our thanks to **Dr. Maheshwar Dutta**, M.Tech., Ph.D.,Principal who encouraged us to do the Project.

We are grateful to **Mr. Neil Gogte**, Director for facilitating all the amenities required for carrying out this project.

We express our sincere gratitude to **Mr. S. Nitin**, Director and **Mrs. Deepa Ganu**, Dean Academic for providing an excellent environment in the college.

We are also thankful to **Dr. S. Padmaja**, Head, Department of CSE for providing us with both time and amenities to make this project a success within the given schedule.

We are also thankful to our guide **Samuel Chepuri**, for his valuable guidance and encouragement given to us throughout the project work.

We would further like to thank the entire CSE Department faculty, who helped us directly and indirectly in the completion of the project.

We sincerely thank our friends and family for their constant motivation during the project work.

MALLEBOINA KRANTHI KUMAR (16BD1A057J)

MIRUPALA VISHNU VARDHAN (16BD1A057C)

MACHARLA PANDU PRANAY (16BD1A057D)

LAKAVATH VENU GOPAL (16BD1A056Y)

CONTENTS

ABSTRACT

Project statement :

use of AI based chat bot for providing health related information.

Description:

Chatbots in health care may have the potential to provide patients with access to immediate medical information. Health care chatbots could help patients better manage their own health, improve access and timeliness to care.

In this project, the chatbot uses AI(Artificial Intelligence) to make a note of symptoms entered by patient through text and analyze it for detailed understanding. Matching the query against large repository with medical data, the chatbot answers the patient queries.It acts as a symptom checker.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1) AI - Artificial Intelligence
2) MIT - Massachusetts Institute of Technology
3) P2P - Peer-to-Peer
4) UX - User Experience
5) FAQ - Frequently Aksed Question
6) SRS - Software Requirement Specification
7) GUI - Graphical User Interface
8) OS - Operating System
9) RAM - Random Access Memory
10) NLTK - Natural Language Tool Kit
11) DNN - Deep Neural Network
12) UML - Unified Modelling Language
13) OMT - Object Modeling Technique
14) OOSE - Object Oriented Software Engineering

CHAPTER-1

1. INTRODUCTION

Chatbots are trending right now but the idea isn't new. If a patient said “my head hurts”, Eliza would respond, “why do you say your head hurts?”. But Eliza was not a doctor, or even a person. It was a program – 200 lines of experimental code written 50 years ago at MIT. This program was proof that it's possible to simulate human conversation using software.

Eliza was the first chatbot ever created. Chatbots are a new type of user interface which takes advantage of artificial intelligence software that can answer questions, converse, and assist us in a way that mimics human conversation.

History of chatbots – 1966 to the early 2000s

Eliza was named after Eliza Doolittle, a working-class character in Pygmalion (a play by G.B. Shaw) who is taught to speak with an upper-class accent. But Eliza the program was taught to simulate a Rogerian psychotherapist.The software wasn’t nearly as sophisticated as current chatbots, but the 200 lines of code were enough to create the illusion of talking to a psychotherapist. Ultimately, however, Eliza wasn’t capable of passing the Turing Test.

If you can't say whether you're talking to a person or a machine, the machine passes the test. You can talk to Eliza for a minute right here to see why it failed. Apparently the illusion works best when you talk about yourself (which is what you would be doing in a Rogerian psychotherapists office).

The first chatbot to pass the Turing Test was Parry, first implemented in 1972 by psychiatrist Kenneth Colby. Unlike Eliza, Parry imitated a paranoid individual and questioned everything it was told. Parry passed the Turing test in an experiment in which human interrogators questioned him and were unable to tell whether they were talking to a person or a machine.

After that, chatbots remained under the radar of mainstream technology news for some time. But work on new ones didn’t stop, such as the one developed in 1994 by the founder of Lycos, Inc. (first known as “Julia”) to compete for the Loebner Prize.Another example was an intelligent, chatbot-like customer service solution from 1999. It was the result of a partnership between FaceTime (then the leading provider of P2P interaction services) and Big Science Company (creators of the first application server to generate AI-empowered graphical assistants – Klones).

However, the technology that was then widely available was too immature for widespread adoption of chatbots. The markets weren't ready for them.

Recent history of chatbots – explosion in popularity

Currently artificial intelligence has developed to a point where programs can learn and effectively mimic human conversations. Accelerating technological progress has placed internet-enabled machines in every institution, company, home, and eventually pocket (who doesn’t have a smartphone nowadays?).In this environment, chatbots have become increasingly popular as useful tools for companies and institutions. One of the best known examples of chatbots in recent history is Siri – the AI assistant that is part of Apple's standard software for its products. Siri took chatbots mainstream in 2011.

Since then brands in every industry have started to use them, eventually sparking a new trend – conversational UX. This refers to a User Experience in which your interaction with a company or service is automated based on your prior behavior (like working together with someone who is getting to know you).

If you're a programmer, you can take advantage of software like Alexa, which enables the use of voice to control devices.On the other hand, if you are a consumer, you can already interact with chatbots on popular messaging platforms such as Facebook Messenger, iMessage, Skype, Snapchat, etc. According to Business Insider, about 60% of US millennials and Gen X adults have already done so!

Healthcare and chatbots

The use of chatbots has spread from consumer customer service to matters of life and death. Chatbots are entering the healthcare industry and can help solve many of its problems.

Health and fitness chatbots have begun to attract a market. Last year Facebook has started allowing companies to create Messenger chatbots to communicate with users. A great example is HealthTap – the first company to release a health bot on the Messenger app. It allows users to ask medical questions and receive answers from doctors.

Currently, there are dozens of health and fitness chatbots available online. Fitness bots dominate this category, but there are plenty of medical bots worth attention too. Chatbots are far from widespread adoption in healthcare, but they are on the rise. Here are 13 notable examples of niche market chatbots:

- Behavioral change: *GoCarrot*
- Diabetes support: *Brook.ai*
- Fitness: *FitCircle, GymBot*
- Health assistant: *Betterise*
- Medical second opinion: *iCliniq*
- Medication adherence: *Sense.ly*
- Mental health support: *X2, Koko, Joy*

- Nutrition coaching: *Forksy*
- Oncology: *Mendel Health*
- Patient engagement: *LifeLink*
- Symptom checkers: *Babylon Health, Gyant, FlorenceChat, HealthTap, MedWhat, Melody, Symptomate, Your.MD*
- Weight loss coaching: *Lark*
- Tools for navigating healthcare: *HealthJoy*

1.1 Purpose of Project

ChatBot can answer to general user queries related to diseases.They help to self-triage patients' disease based upon the symptoms they enter.

The AI implementations in healthcare shows us how the chatbots are improving the state of healthcare and will be going very far with its use in more number of tasks. It will increase reliability and cost effectiveness to the current scenario of health which proves chatbots to be a boon to mankind.

They provide information fast when there is not a moment to lose and are available 24x7.

1.2 Problems with Existing System

The problem with these chatbots is that they only provide answers for general healthcare FAQs. That is, these systems are unable to provide a natural communication with the user just as a doctor can.

1.3 Proposed System

In this project, the chatbot uses AI(Artificial Intelligence) to make a note of symptoms entered by patient through text and analyze it for detailed understanding. Matching the query against large repository with medical data, the chatbot answers the patient queries.It acts as a symptom checker.

1.4 Scope of the Project

This project can be used by both doctors and patients.If the doctor is not sure of the disease,this chatbot helps him to make a decision.Patient can also analyze his symptoms and medical state of disease.This can reduce time of consulting a doctor.

It helps a lot in terms of analyzing general sickness which doesn't need the attention of a doctor.Thereby reducing the cost.

1.5 ARCHITECTURE DIAGRAM

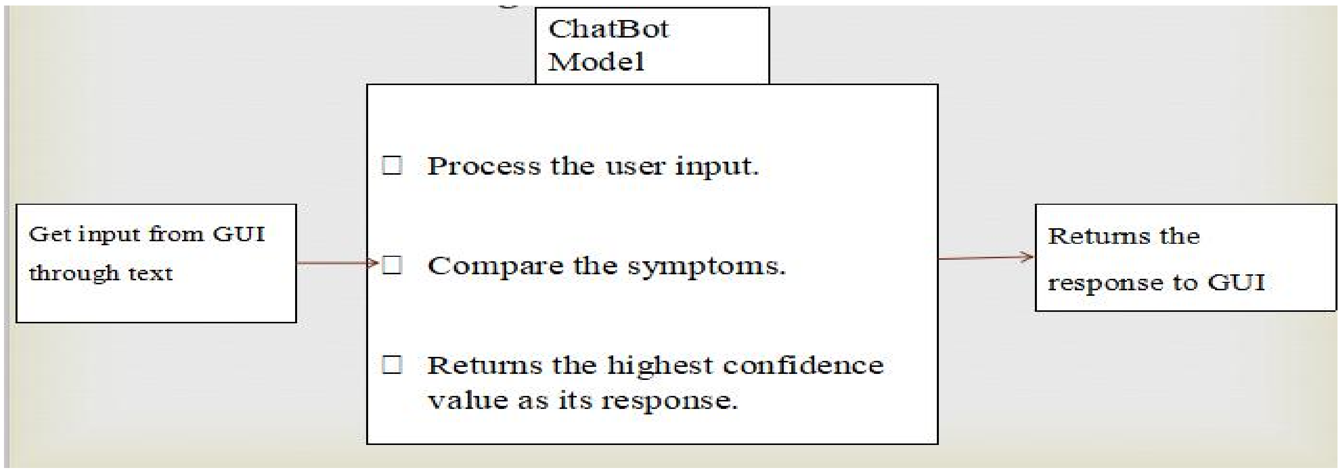

ChatBot Model
Get input from GUI through text
□ Process the user input.
□ Compare the symptoms.
□ Returns the highest confidence value as its response.
Returns the response to GUI

Figure : Architecture Diagram

.

CHAPTER-2

2. SOFTWARE REQUIREMENT SPECIFICATION

What is SRS?

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)
The SRS phase consists of two basic activities:

Problem/Requirement Analysis:

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

Requirement Specification:

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

Role of SRS

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium though which the client and user needs are accurately specified. It forms the basis of software development.A good SRS should saatisfy all the parties in the system.

2.1.Requirements Specification Document

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application.

There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and nonfunctional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behaviour of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS.

This section introduces the requirement specification document for use of AI based ChatBot for Health related Information which enlists functional as well as non-functional requirements.

2.1.1.Functional Requirements

For documenting the functional requirements, the set of functionalities supported by the system are to be specified. A function can be specified by identifying the state at which data is to be input to the system, its input data domain, the output domain, and the type of processing to be carried on the input data to obtain the output data.

Functional requirements define specific behaviour or function of the application. Following are the functional requirements:

- Load data
- Data analysis
- Data preprocessing
- Model building
- Combining model with GUI

2.1.2.Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. Especially these are the constraints the system must work within. Following are the non-functional requirements:

- 24 X 7 availability.
- Better component design to get better performance at peak time

- Flexible service based architecture will be highly desirable for future extension

2.2 Software Requirements

- OS: Windows or Linux
- Jupyter Notebook Required
- Setup tools and pip to be installed for 3.6
- Language : Python

2.3 Hardware Requirements

- RAM: 4GB and Higher
- Processor: Intel i3 and above
- Hard Disk: 500GB: Minimum

CHAPTER-3

3. LITERATURE SURVEY

ChatBots are automated systems which replicate users behavior on one side of the chatting communication. They are mimic systems which imitate the conversations between two individuals. They provide a simulating platform for effective and smart communications with the user on the other end. They copy marketers, sales person, counsellors and other mediators and work to provide services that the above mentioned people provide. There are wide ranges of chatbots catering in many domains some of them are as follows: business, market, stock, customer care, healthcare, counselling, recommendation systems, support system, entertainment, brokering, journalism, online food and accessory shopping, travel chatbots, banking chatbots, recipe guides, etc.

The most famous chatbots like Alexa or Google assistant are the best examples that can be given for smart communicating chatbots. These are general purpose chatbots that provide services for all domains and are not restricted to a specific domain. There are also domain-specific chatbots which provide functionalities to the above-mentioned domains. Some of them are as follows: Botsify is a chatbot which helps developers to create smart Facebook Messenger Chatbots and is used to collect information from Facebook users. Imperson is a chatbot which helps developers to create business chatbots and provide customer care services. NBC is a chatbot which helps the newsreaders to navigate quickly through top headlines.

The above mentioned chatbots were the systems for business or market domains. The aim of this project is to discuss the need and usage of chatbots in the healthcare domain. There are a lot of existing chatbots for healthcare domain serving different functionalities. Endurance is a chatbot which deals with users suffering from Dementia (disease). The chatbot Casper helps people suffering from insomnia to pass their nights which are sleepless due to loneliness. MedWhat is a question-answering chatbot which gives answers to basic healthcare FAQs and also provides information about various diseases and its symptoms. The problem with these chatbots is that they just provide monotonous answers to users questions. They are not capable of establishing a smart communication with the user just as a doctor does. These systems are also not able to predict the problems (diseases) faced by the user. To know what the patient is suffering from, the system should be friendly to the user, so that the user can communicate all the problems faced by him to the system. The project aims at proposing a chatbot system, which is capable of establishing a smart communication. This can be achieved by implementing a smart and a responsive chatbot which is able to communicate with the user just as another human can communicate.

To make conventional chatbots function like virtual friends, techniques of NLTK and AI require to be incorporated into the system. These techniques make the system more communicative in the natural language, proves fruitful for counselling, and can also be modelled for prediction of diseases.

The chatbots used by the telecom and marketing sectors for customer service are scripted types of chatbots. They help the customers on some predefined customer care questions. Research is being carried out in making the conventional monotonous chatbots to be communicative, responsive and carry out the communication in a natural (conversational) language. This requires the inclusion of NLP and AI techniques in the system. There are a number of ways to do so. Selection of an appropriate method is based on the domain of the chatbot, the functionalities it intends to provide, the language of communication, the end user, etc. All the above-mentioned issues need to be considered while implementing a chatbot system.

The operating model of a chatbot is always the same, whatever its scope, its theme and its level:

- Users formulate their queries in natural language text interface.
- The chatbot receives the request and the Model interprets it to understand it.
- The chatbot provides a unique and qualified answer to the user's query. The answer may be generic, contextualized or cusomized.

3.1.Language Used

Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

3.1.1. Libraries used in python:

Nltk:

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

LancasterStemmer:

Lancaster stemming algorithm is the most aggressive stemming algorithm of the bunch. However, if you use the stemmer in NLTK, you can add your own custom rules to this algorithm very easily. It's a good choice for that. One complaint around this stemming algorithm though is that it sometimes is overly aggressive and can really transform words into strange stems.

Numpy:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Tflearn:

TFlearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

Tensorflow:

Tensorflow is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

Random:

Random module can generate random numbers. These are pseudo-random number as the sequence of number generated depends on the seed. If the seeding value is same, the sequence will be the same.

Json:

In Python, the json module provides an API similar to convert in-memory Python objects to a serialized representation known as JavaScript Object Notation (JSON) and vice-a-versa.

Pickle:

Python pickle module is used for serializing and de-serializing a Python object structure. ... Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

Tkinter:

Tkinter is the most commonly used method for developing GUI(Graphical User Interface). It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications.

3.2 Model studied and used

DNN(Deep Neural Network):

A Deep Neural Network is nothing but an Artificial Intelligence(AI) Neural Network with multiple layers between the input and the output. At each layer, the network calculates how probable each output is. A DNN will model complex non-linear relationships when it needs to. With extra layers, we can carry out the composition of features from lower layers.

Typically, a DNN is a feedforward network that observes the flow of data from input to output. It never loops back. Deep Neural Network creates a map of virtual neurons and assigns weights to the connections that hold them together. It multiplies the weights to the inputs to produce a value between 0 and 1. When it doesn't accurately recognize a value, it adjusts the weights. This is to make parameters more influential with an ulterior motive to determine the correct mathematical manipulation so we can fully process the data.

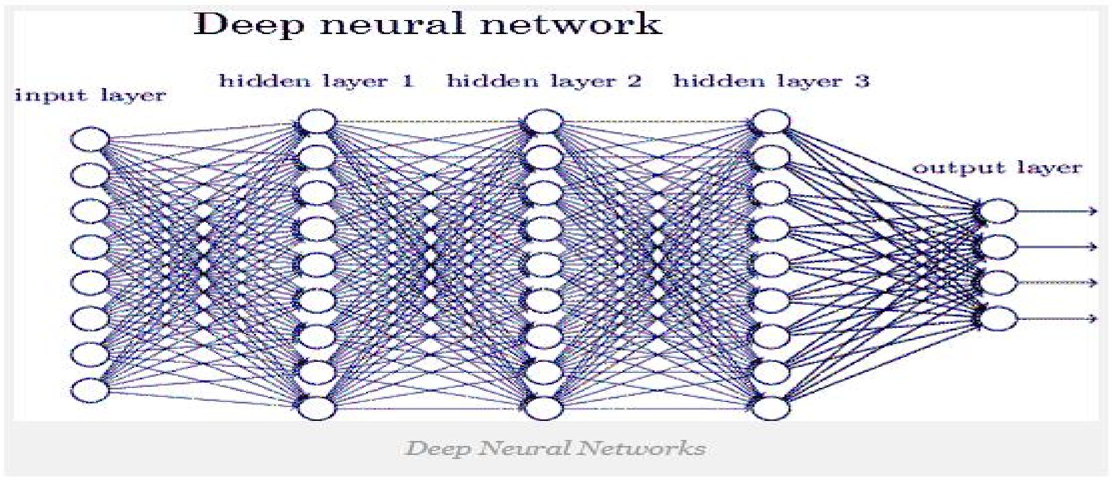


Figure : DNN

Conclusion

There are many examples of healthcare chatbots, and a wide range of applications. Many countries and health stakeholders have understood the importance of this technology in meeting the challenges of the sector in the years to come.

Fears have been expressed that chatbots could replace the physician or interfere with the patient-physician relationship, but this has not occurred in the experiments conducted to date. As we have seen above, these tools can, on the contrary, help improve healthcare, such as by supporting diagnosis or patient referral, which would leave the physician more time to attend to the patient.

The future of healthcare chatbots will depend on the perception of health professionals and patients and the extent to which they adopt them.

CHAPTER-4

4. SYSTEM DESIGN

4.1 Introduction to UML

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows:

1. User Model View

i. This view represents the system from the users' perspective.

ii. The analysis representation describes a usage scenario from the end-users' perspective.

2. Structural Model View

i. In this model, the data and functionality are arrived from inside the system.

ii. This model view models the static structures.

3. Behavioural Model View

It represents the dynamic of behavioural as parts of the system, depicting he interactions of collection between various structural elements described in the user model and structural model view.

4. Implementation Model View

In this view, the structural and behavioural as parts of the system are represented as they are to be built.

5. Environmental Model View

In this view, the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

4.2 UML Diagrams

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- actors
- business processes
- (logical) components
- activities
- programming language statements
- database schemas, and
- Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

4.2.1.Use Case Diagram

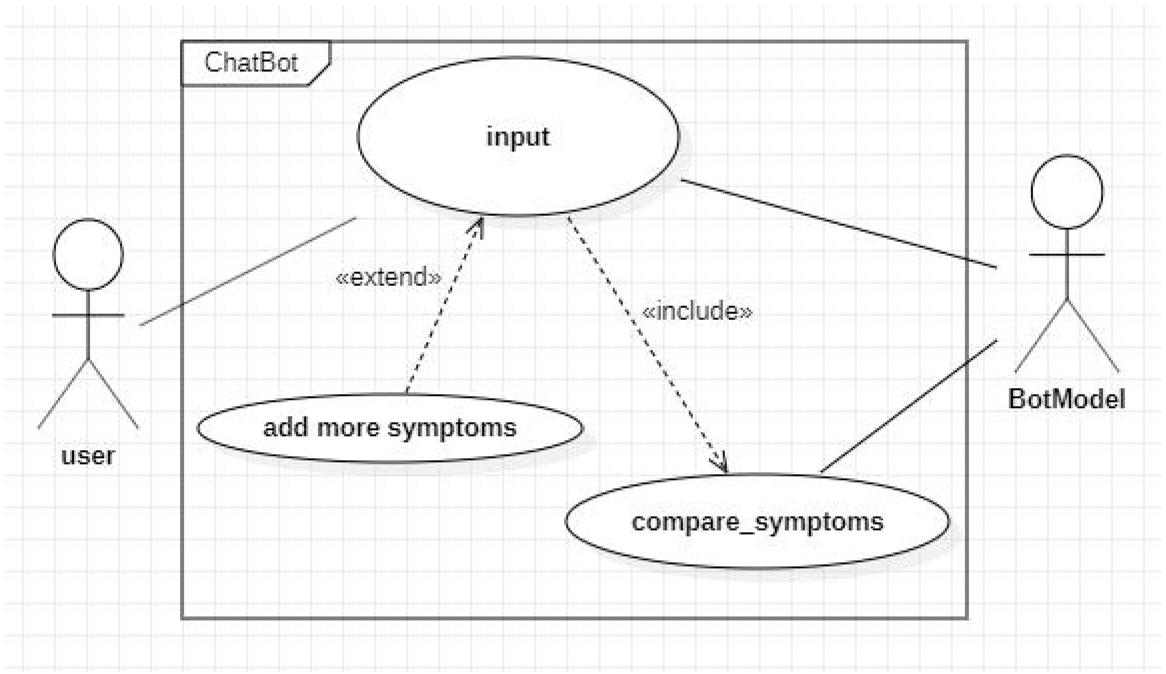


Figure : Use Case Diagram

4.2.2.Class Diagram:

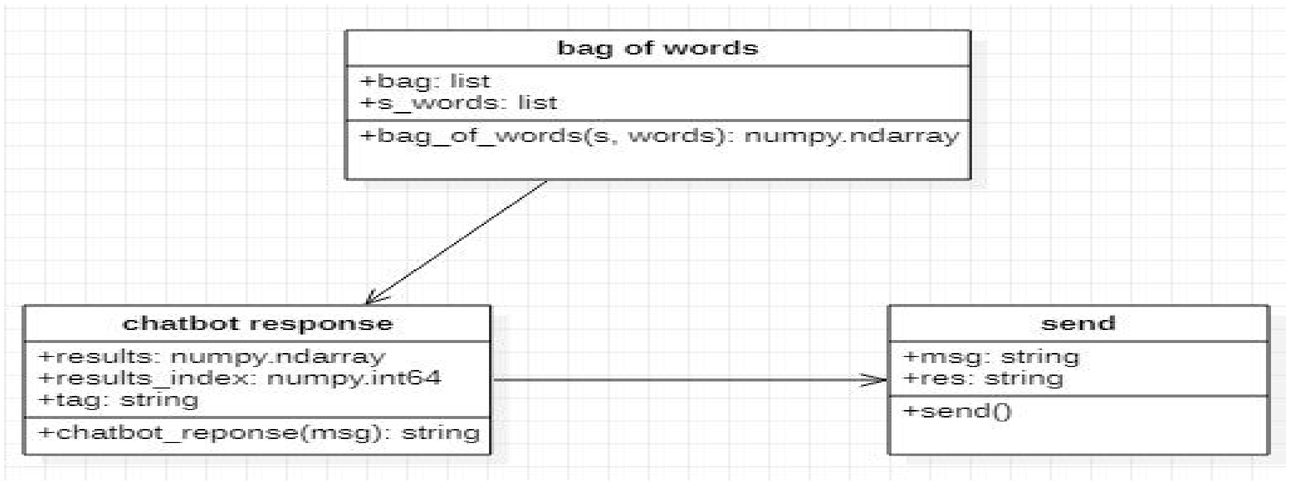


Figure : Class Diagram

4.2.3.Sequence diagram:

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

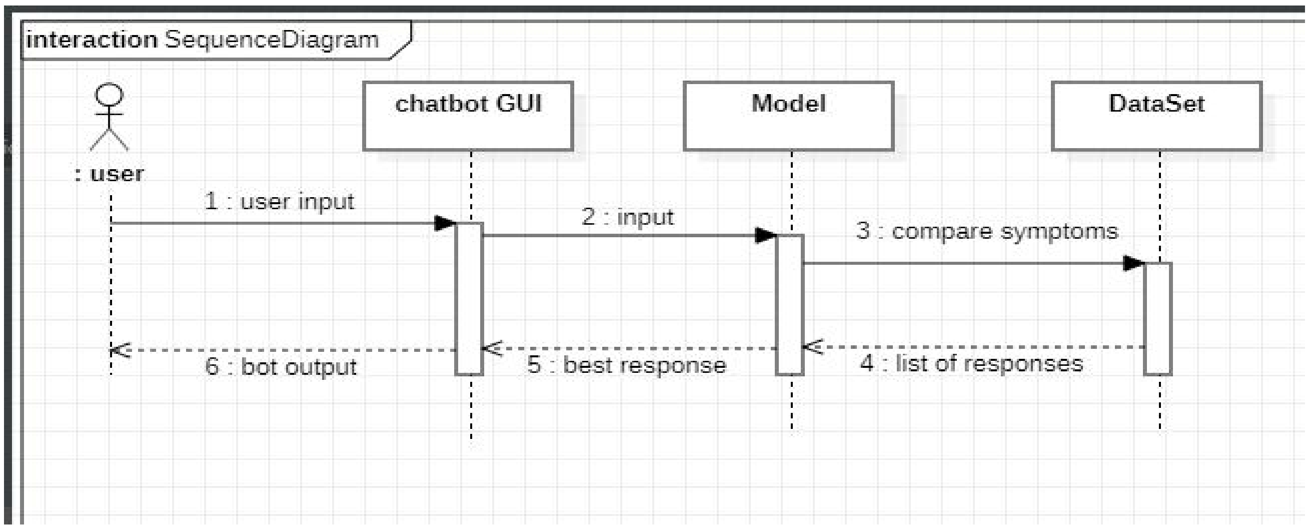


Figure : Sequence Diagram

4.2.4.Activity diagram:

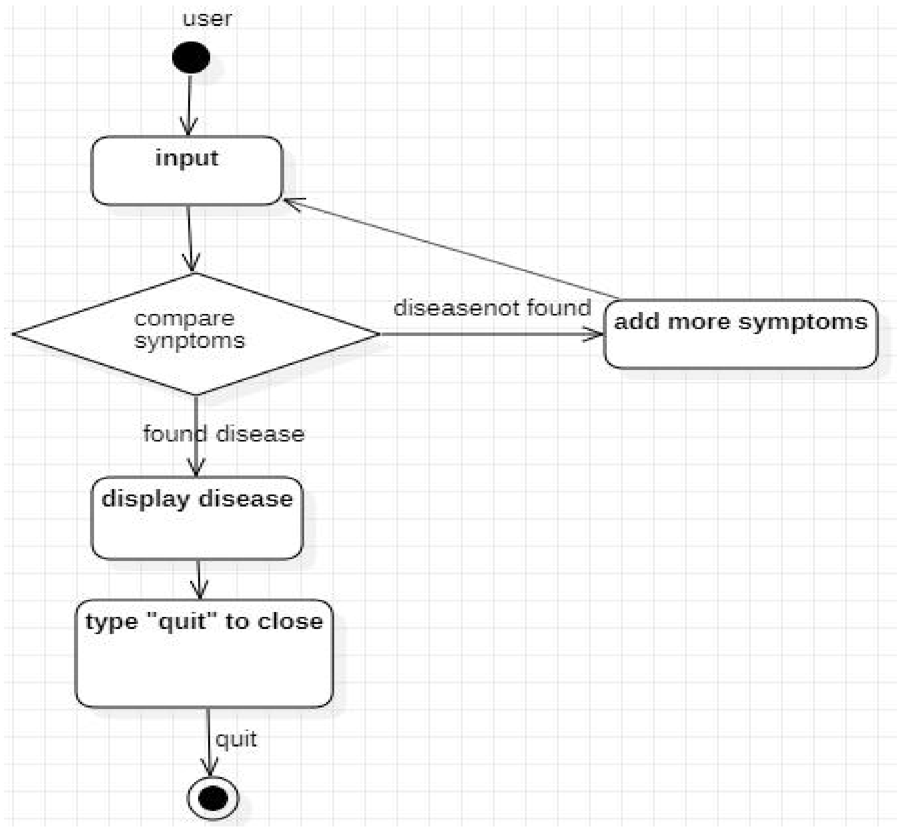


Figure : Activity Diagram

CHAPTER-5

5. IMPLEMENTATION

5.1 Pseudo Code

Step 1:

Import the necessary packages.

Step 2:

Load the data set and Pre-process it.

Step 3:

Create the Model.

Step 4:

Fit and Save the Model.

Step 5:

Create function to give input to the Model.

Step 6:

Create GUI Chat window and other necessary components with tkinter.

Step 7:

Chat with Bot.

Step 8:

Type “quit” to Stop.

Step 9:

Open “anvil.works/build#” website and create a web app for DocBot.

Step 10:

Import anvil server in Jupyter notebook and connect it with the server.

Step 11:

Create anvil callable function in the Jupyter notebook.

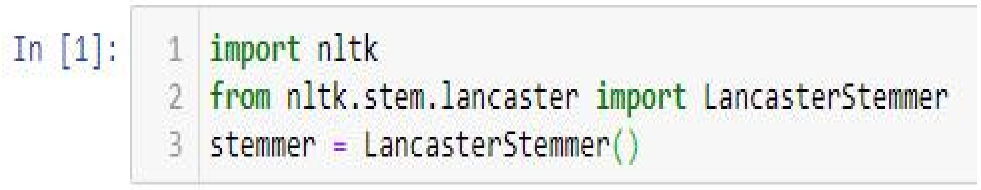
Step 12:

Open “anvil.works/build#” website and publish the DocBot web app in public domain with the following URL:

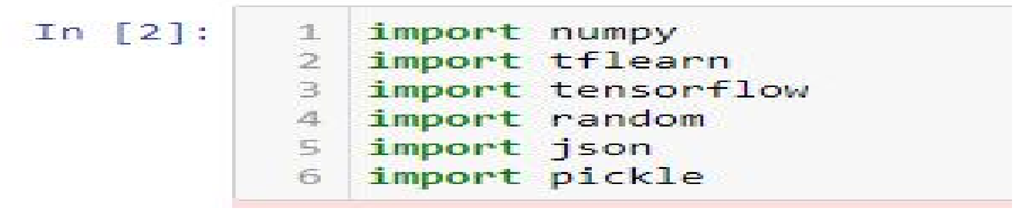
https://docbot.anvil.app/

5.2 Code Snippets

5.2.1 Packages Imported

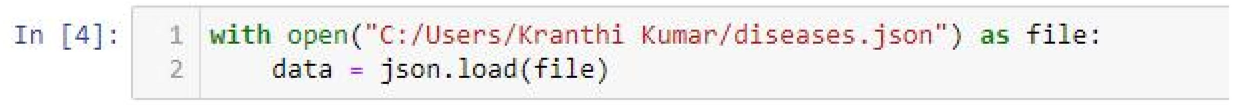


```
In [1]:
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()
```

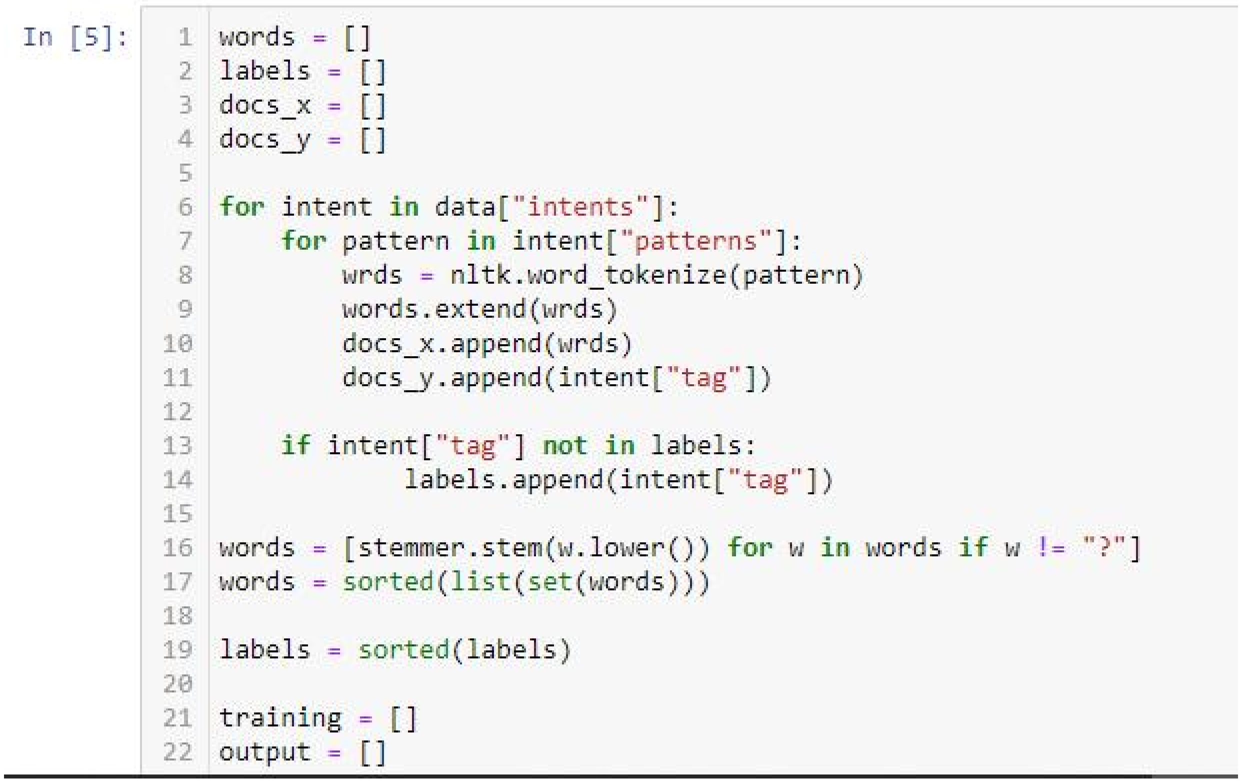


```
In [2]:
import numpy
import tflearn
import tensorflow
import random
import json
import pickle
```

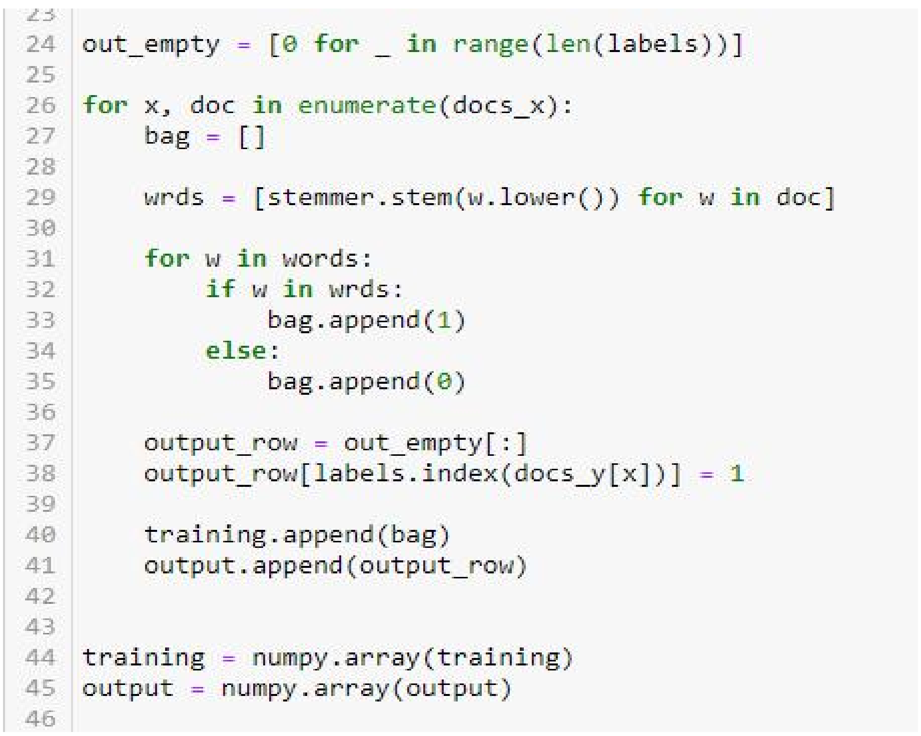
5.2.2 Load the data and Pre-Process the data.



```
In [4]:
with open("C:/Users/Kranthi Kumar/diseases.json") as file:
    data = json.load(file)
```

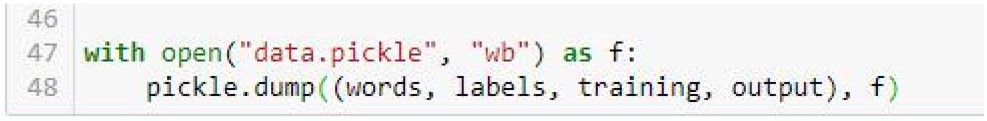


```
In [5]:
words = []
labels = []
docs_x = []
docs_y = []

for intent in data["intents"]:
    for pattern in intent["patterns"]:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds)
        docs_x.append(wrds)
        docs_y.append(intent["tag"])

    if intent["tag"] not in labels:
            labels.append(intent["tag"])

words = [stemmer.stem(w.lower()) for w in words if w != "?"]
words = sorted(list(set(words)))

labels = sorted(labels)

training = []
output = []
```



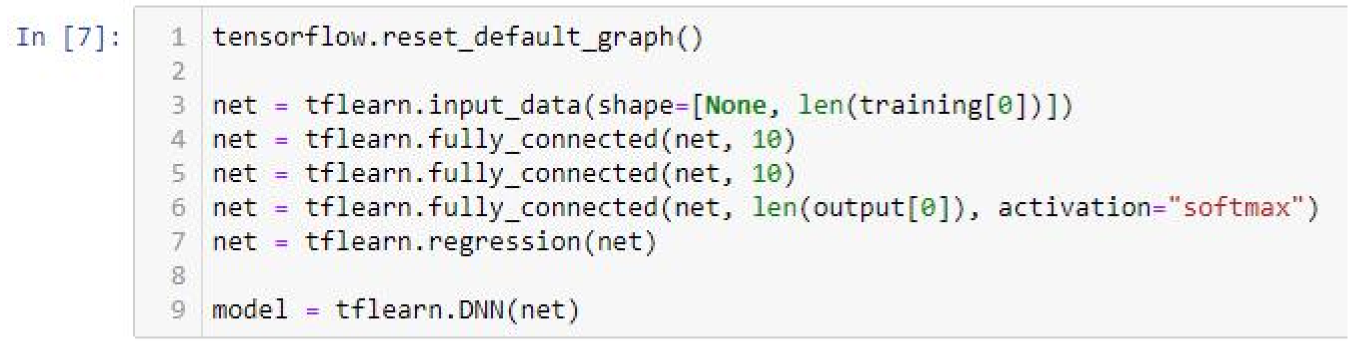
```
out_empty = [0 for _ in range(len(labels))]

for x, doc in enumerate(docs_x):
    bag = []

    wrds = [stemmer.stem(w.lower()) for w in doc]

    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)

    output_row = out_empty[:]
    output_row[labels.index(docs_y[x])] = 1

    training.append(bag)
    output.append(output_row)


training = numpy.array(training)
output = numpy.array(output)
```

Dump words, labels, training, output into data.pkl file.



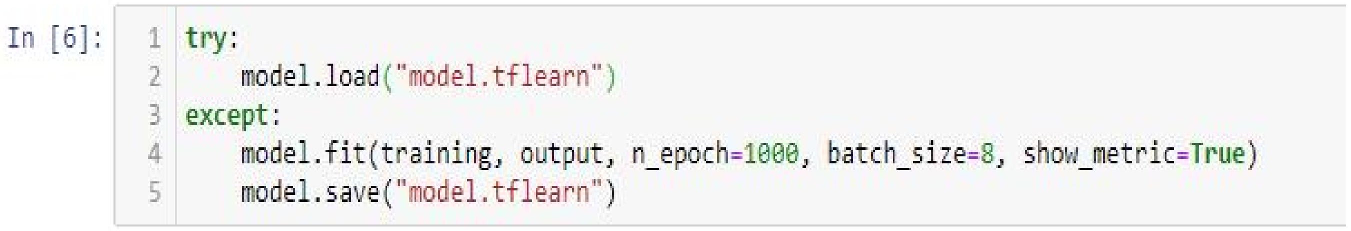
```
with open("data.pickle", "wb") as f:
    pickle.dump((words, labels, training, output), f)
```

5.2.3 Create a Model.



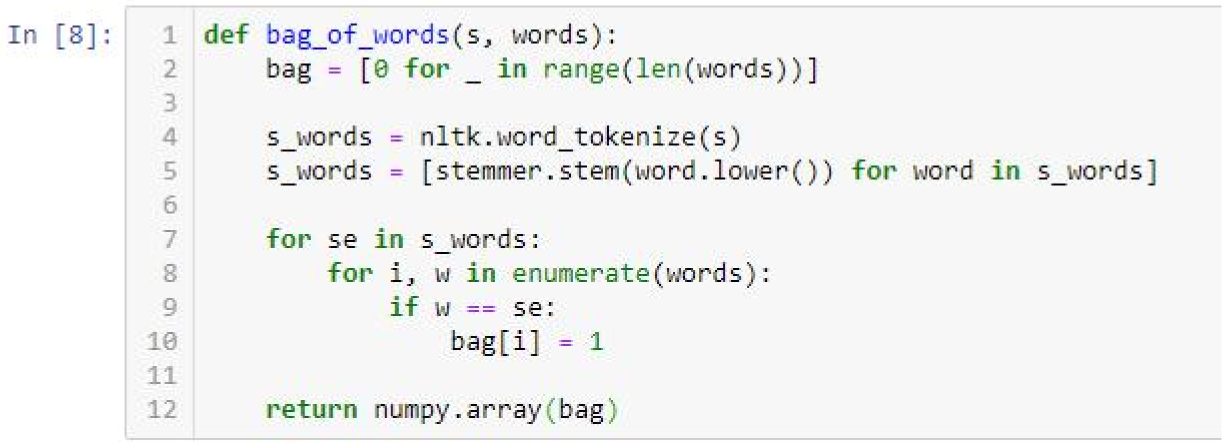
```
In [7]:
tensorflow.reset_default_graph()

net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
net = tflearn.regression(net)

model = tflearn.DNN(net)
```

5.2.4 Fit and save the Model.



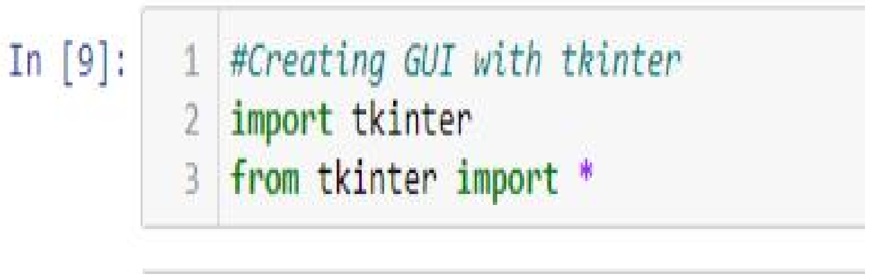
```
In [6]:
try:
    model.load("model.tflearn")
except:
    model.fit(training, output, n_epoch=1000, batch_size=8, show_metric=True)
    model.save("model.tflearn")
```

5.2.5 Create function to give input to the Model.



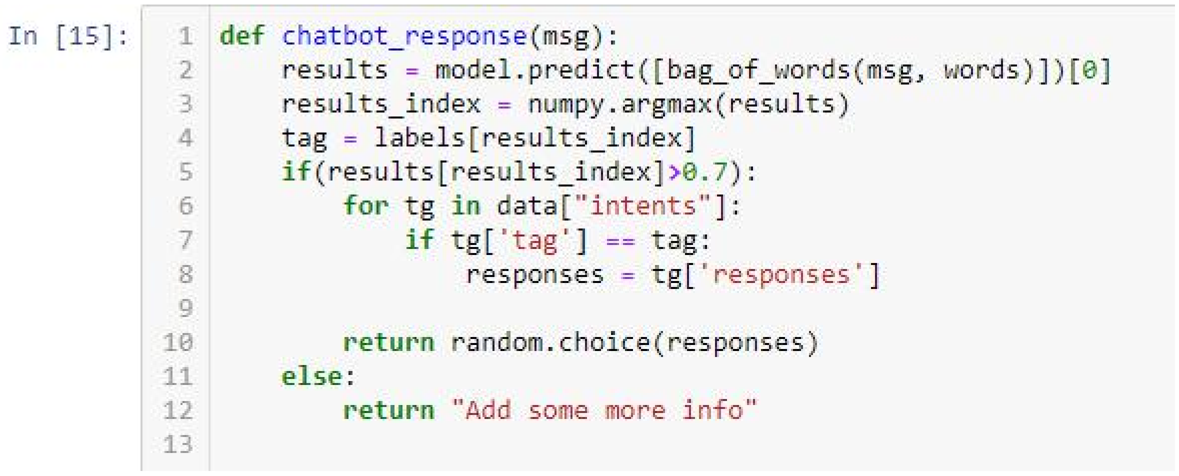
```
In [8]:
def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)
```

5.2.6 Create GUI Chat window and other necessary components with tkinter.



```
In [9]:
#Creating GUI with tkinter
import tkinter
from tkinter import *
```

Only return the response that has confidence greater than 0.7 else return “Add some more info”.



```
In [15]:
def chatbot_response(msg):
    results = model.predict([bag_of_words(msg, words)])[0]
    results_index = numpy.argmax(results)
    tag = labels[results_index]
    if(results[results_index]>0.7):
        for tg in data["intents"]:
            if tg['tag'] == tag:
                responses = tg['responses']

        return random.choice(responses)
    else:
        return "Add some more info"

```

In [16]:
```python
def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg.isnumeric():
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))
        res="You entered number.Please enter your symptoms"
        ChatLog.insert(END, "DocBot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)

    elif msg.isalnum():
        flag=0
        for i in symptoms:
            if(msg.lower()==i.lower()):
                ChatLog.config(state=NORMAL)
                ChatLog.insert(END, "You: " + msg + '\n\n')
                ChatLog.config(foreground="#442265", font=("Verdana", 12 ))
                res = chatbot_response(msg)
                ChatLog.insert(END, "DocBot: " + res + '\n\n')
                ChatLog.config(state=DISABLED)
                ChatLog.yview(END)
                flag=1
                break
        if(flag==0):
            ChatLog.config(state=NORMAL)
            ChatLog.insert(END, "You: " + msg + '\n\n')
            ChatLog.config(foreground="#442265", font=("Verdana", 12 ))
            res = "You entered something.Please enter your symptoms"
            ChatLog.insert(END, "DocBot: " + res + '\n\n')
            ChatLog.config(state=DISABLED)
            ChatLog.yview(END)


    if msg.lower()=="quit":
        base.destroy()
```

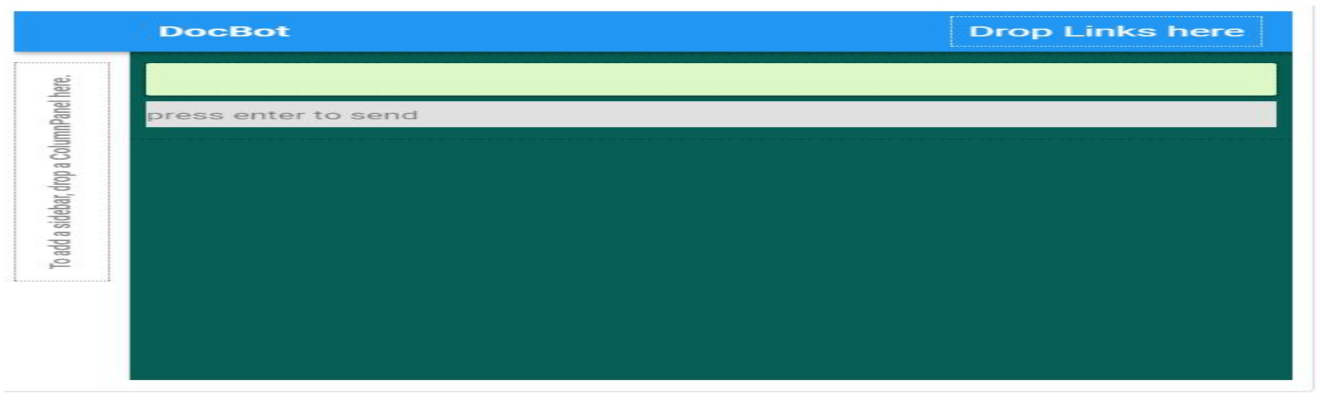
In [20]:
```python
base = Tk()
base.title("DocBot")
base.geometry("400x500")
base.resizable(width=TRUE, height=TRUE)
```

Out[20]:
```
''
```

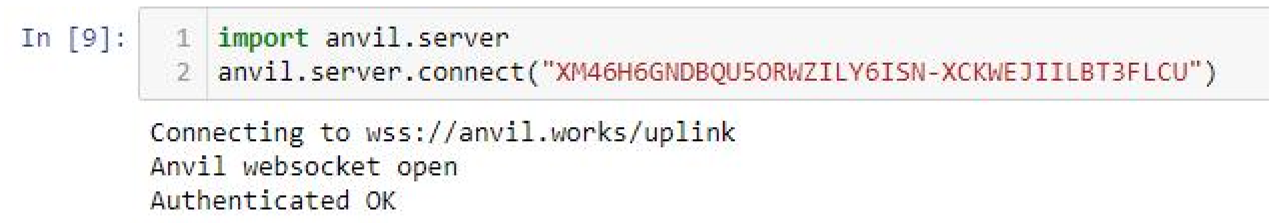
In [21]:
```python
#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)

ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12", height=5,
                    bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
                    command= send )

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")
#EntryBox.bind("<Return>", send)


#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)
```

In [22]:
```python
base.mainloop()
```

5.2.7 Open "anvil.works/build#" website and create a web app for DocBot.

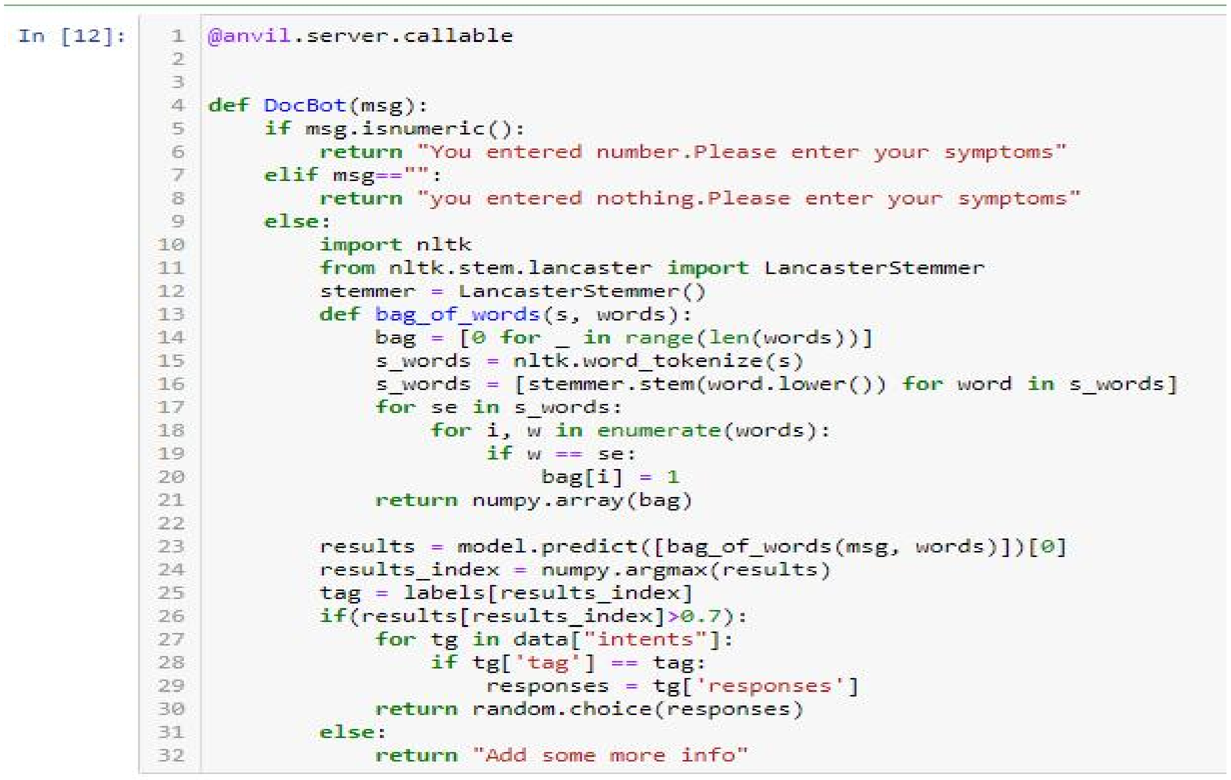


5.2.8 Import anvil server in Jupyter notebook and connect it with the server.



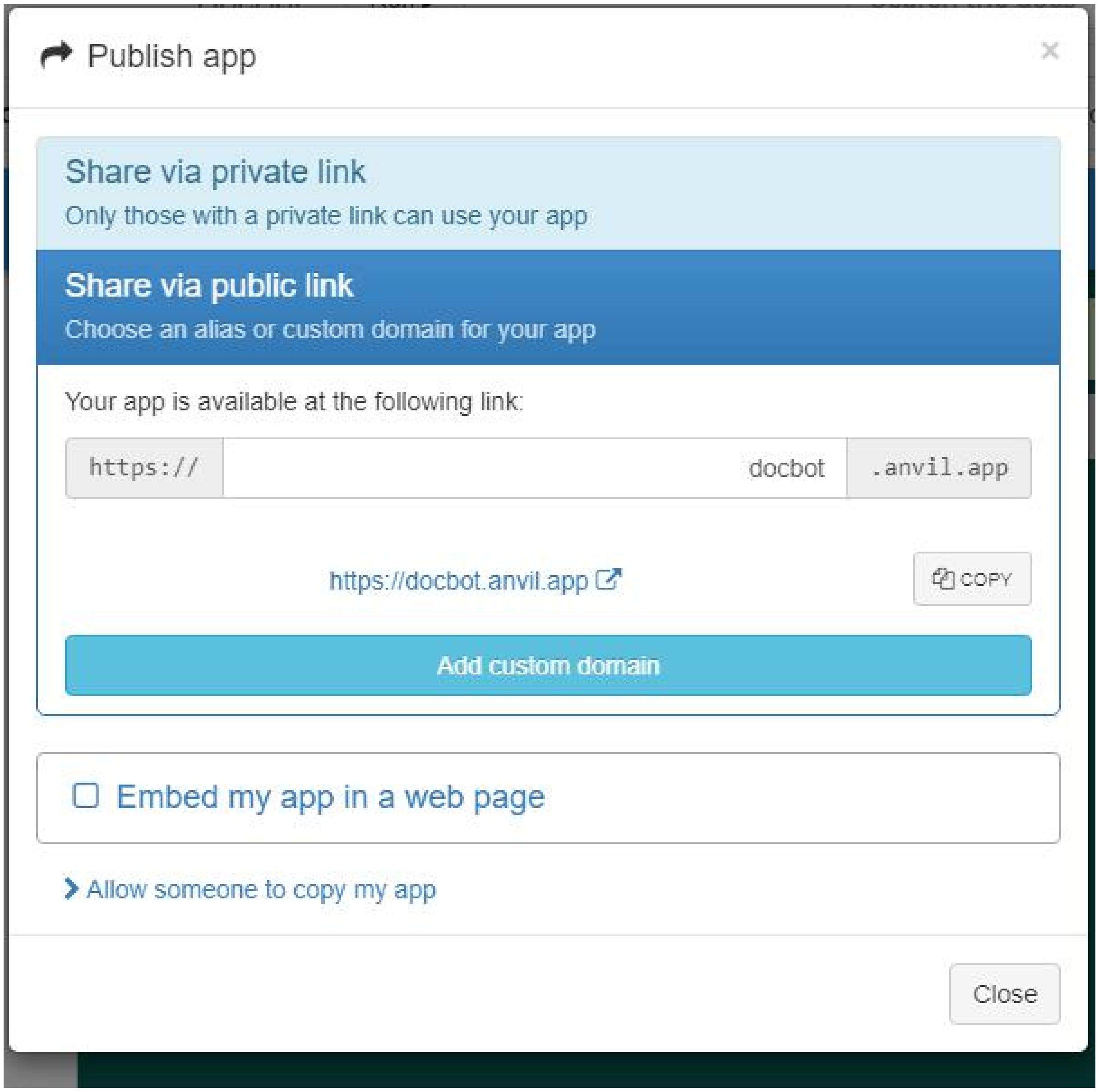
```
In [9]:  import anvil.server
         anvil.server.connect("XM46H6GNDBQU5ORWZILY6ISN-XCKWEJIILBT3FLCU")

Connecting to wss://anvil.works/uplink
Anvil websocket open
Authenticated OK
```

5.2.9 Create anvil callable function in the Jupyter notebook.



```
In [12]:
@anvil.server.callable


def DocBot(msg):
    if msg.isnumeric():
        return "You entered number.Please enter your symptoms"
    elif msg=="":
        return "you entered nothing.Please enter your symptoms"
    else:
        import nltk
        from nltk.stem.lancaster import LancasterStemmer
        stemmer = LancasterStemmer()
        def bag_of_words(s, words):
            bag = [0 for _ in range(len(words))]
            s_words = nltk.word_tokenize(s)
            s_words = [stemmer.stem(word.lower()) for word in s_words]
            for se in s_words:
                for i, w in enumerate(words):
                    if w == se:
                        bag[i] = 1
            return numpy.array(bag)

        results = model.predict([bag_of_words(msg, words)])[0]
        results_index = numpy.argmax(results)
        tag = labels[results_index]
        if(results[results_index]>0.7):
            for tg in data["intents"]:
                if tg['tag'] == tag:
                    responses = tg['responses']
            return random.choice(responses)
        else:
            return "Add some more info"
```

5.2.10 Open “anvil.works/build#” website and publish the DocBot web app in public domain.



Publish app

Share via private link

Only those with a private link can use your app

Share via public link

Choose an alias or custom domain for your app

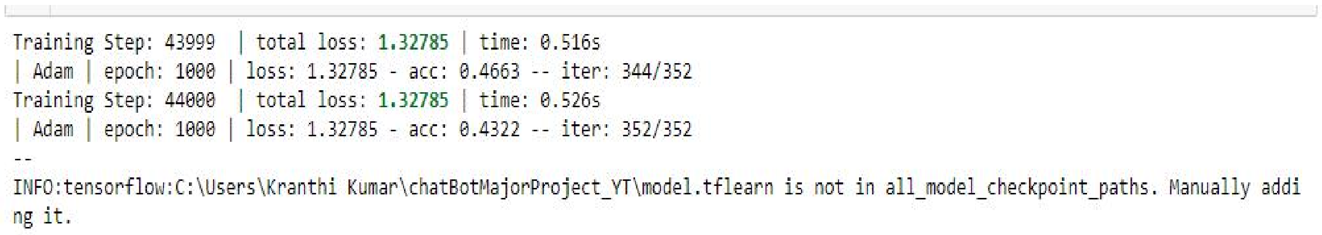
Your app is available at the following link:

https:// docbot .anvil.app

https://docbot.anvil.app

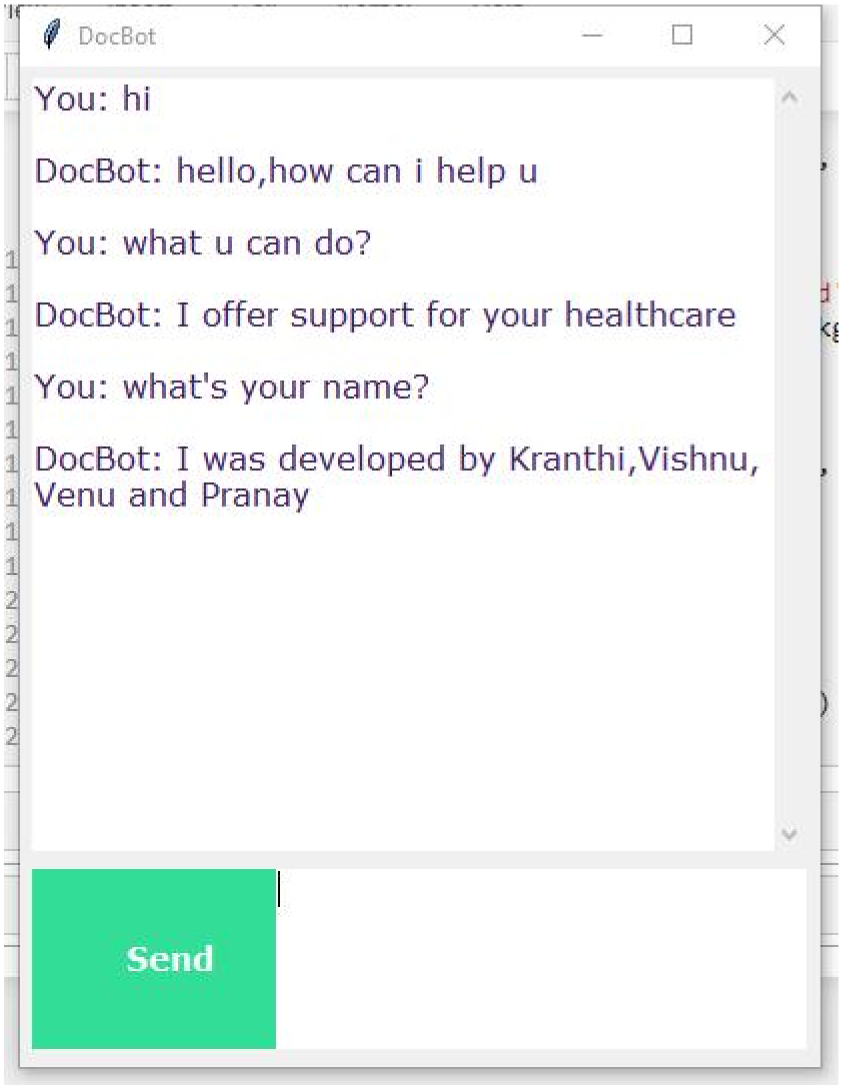
CHAPTER-6

6. OUTPUT SCREENS



```
Training Step: 43999  | total loss: 1.32785 | time: 0.516s
| Adam | epoch: 1000 | loss: 1.32785 - acc: 0.4663 -- iter: 344/352
Training Step: 44000  | total loss: 1.32785 | time: 0.526s
| Adam | epoch: 1000 | loss: 1.32785 - acc: 0.4322 -- iter: 352/352
--
INFO:tensorflow:C:\Users\Kranthi Kumar\chatBotMajorProject_YT\model.tflearn is not in all_model_checkpoint_paths. Manually addi
ng it.
```

GUI DocBot output



You: hi

DocBot: hello,how can i help u

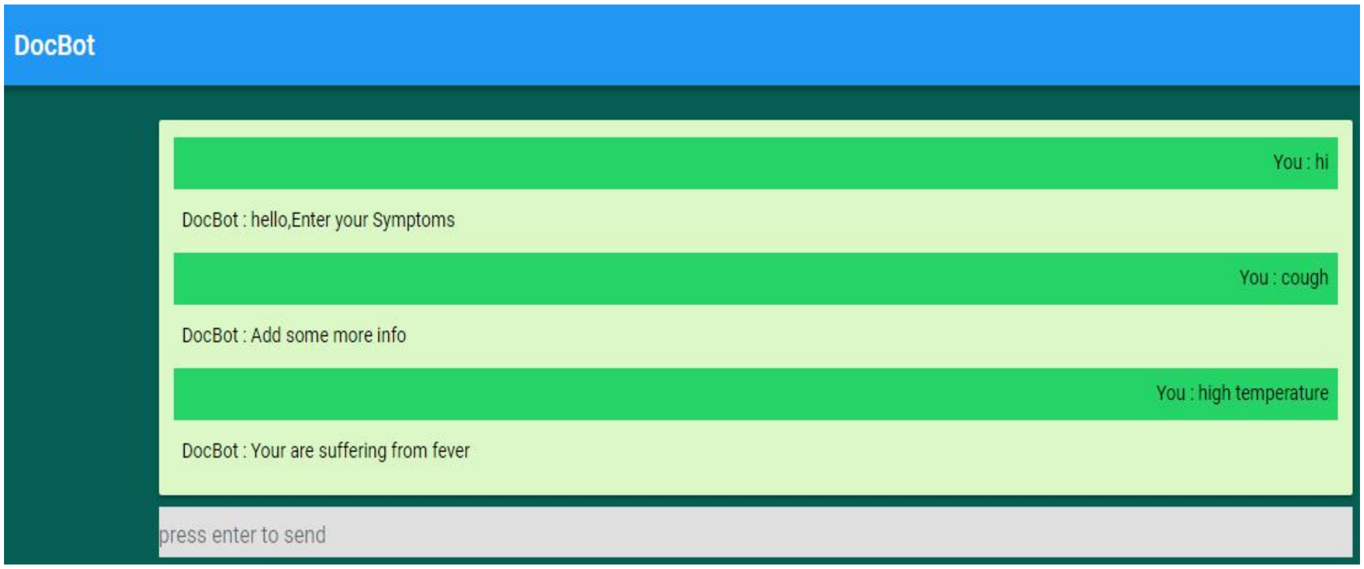
You: what u can do?

DocBot: I offer support for your healthcare

You: what's your name?

DocBot: I was developed by Kranthi,Vishnu, Venu and Pranay

Published DocBot Web app Output



You : hi

DocBot : hello,Enter your Symptoms

You : cough

DocBot : Add some more info

You : high temperature

DocBot : Your are suffering from fever

CHAPTER-7

7. TESTING

7.1 INTRODUCTION TO TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

There of basically **two types of testing** approaches.

One is ***Black-Box testing*** – the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated.

The other is ***White-Box testing*** – knowing the internal workings of the product ,tests can be conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised.

White box and Black box testing methods have been used to test this package. The entire loop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

7.2 TESTING STRATEGIES:

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be preformed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

7.2.1.UNIT TESTING:

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combination are pre-calculated and are generated by the module.

7.2.2.SYSTEM TESTING:

Here all the pre-tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generated test combinations of test paths through out the system to make sure that no path is making its way into chaos.

7.3 Test Cases:

Test Case no	**Input**	**Output**	**Expected Output**	**Result**
1	hi	hello,Enter your Symptoms	hello,Enter your Symptoms	PASS

Table: Test case 1

Test Case no	**Input**	**Output**	**Expected Output**	**Result**
2	123	You entered number.Please enter your symptoms	You entered number.Please enter your symptoms	PASS

Table: Test case 2

Test Case no	**Input**	**Output**	**Expected Output**	**Result**
3	bye	Have a nice day,Type ‘quit’ to end this session.	Have a nice day,Type ‘quit’ to end this session.	PASS

Table: Test case 3

Test Case no	**Input**	**Output**	**Expected Output**	**Result**
4	high temperature	You are suffering from fever	You are suffering from fever	PASS

Table: Test case 4

Test Case no	**Input**	**Output(Action)**	**Expected Output**	**Result**
5	quit	Window closes	Window closes	PASS

Table: Test case 5

Test Case no	**Input**	**Output**	**Expected Output**	**Result**
6	123asd	You entered something.Please enter your symptoms	You entered something.Please enter your symptoms	PASS

Table: Test case 6

Test Case no	**Input**	**Output**	**Expected Output**	**Result**
7	@	*N/A*	You entered special symbol.Please enter your symptoms	FAIL

Table: Test case 7

8. CONCLUSION

From our perspective, chatbots with AI are dramatically changing businesses.There is a wide range of chatbot building platforms that are available for various enterprises including health care.

In this project, we described one concrete instance of a text-based health carechatbot system that was designed to support patients and health professionals likewise.

Rapid advances in AI research and the resources being provided by governments and industry make it highly likely that AI will be used extensively health care delivery and there is huge potential for cost-saving as well as service quality improvement.

Chatbots can reach out to a large audience on messaging apps and be more effective than humans.They may develop into a capable health information-gathering tool in the near future.

9. FUTURE ENHANCEMENTS

Our project AI based chatbot for health care information is aimed at analyzing symptoms and displaying the disease (Symptom checker) through text based interface.For the enhancement of this project we felt there is a scope for Personalized chatbot with voice based interface for easy communication.

Personalized chatbot would have the patient history stored and would better understand the queries of the patient.The queries would be answered as per the history and current situation.

We can also improve the chatbot by enabling it to diagnose the disease and provide a solution.It can give best results when we have large dataset.

10. BIBLIOGRAPHY

References:

- Creating a simple Python ChatBot using Natural Language Processing
  https://github.com/jerrytigerxu/Simple-Python-Chatbot
- Data set collected from the following website
  https://github.com/vsharathchandra/AI-Healthcare-chatbot/blob/master/Training.csv
- AI tutorial for creating a chatbot
  https://techwithtim.net/tutorials/ai-chatbot/part-4/
- Tensorflow installation and setup
  https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow/
- Tflearn installation and setup
  https://pypi.org/project/tflearn/
- Publishing the Jupyter Notebook as Web app
  https://anvil.works/build#