Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE ENGINEERING AND INFORMATION SYSTEMS

A project report on

AN AI MEDICAL CHATBOT FOR INFECTIOUS & DISEASE PREDICTION

M. Tech Integrated Software Engineering

by

MOHAMED HAASHIM ANSARI F (20MIS0284)
SABES P (21MIS0469)

ABSTRACT

This project goal is to succinctly demonstrate how we might advance chatbots in the medical field and treat infectious diseases. Through the users, we can raise awareness and provide appropriate medical care to avert illness. The chatbot is a ground-breaking development in healthcare that has the potential to improve efficiency and accessibility. It shows amazing capabilities in providing healthcare practitioners with streamlined decision support and accurate diagnosis suggestions through the use of large medical datasets. AI chatbots are becoming more and more trusted by users, and they also have the added advantage of relieving pressure on healthcare systems. In 2021, we developed a study report and an early training model to enhance human-database interaction. Natural language processing is used to characterise the chatbot's traits and human-like behaviours. In this research, we propose a deep feedforward multilayer perceptron-based AI chatbot interaction and prediction model. Our investigation revealed a deficiency in the understanding of theoretical principles and useful suggestions for developing AI chatbots for lifestyle enhancement initiatives. This report also includes a quick comparison of our proposed model with respect to testing accuracy and time complexity. Our work has a minimum loss of 0.1232 and a maximum accuracy of 94.32%. This paper examines the related issues posed by using these developing technologies during such health crises, primarily caused by pandemics, and describes the capabilities and potential applications of medical chatbots.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 BACKGROUND

An AI medical chatbot for infectious and disease prediction leverages machine learning algorithms to analyse vast datasets of medical records, epidemiological data, and real-time patient information. It uses natural language processing to engage with users in human-like conversations, gathering symptoms and risk factors to assess the likelihood of various infectious diseases. By continuously learning and adapting, the chatbot provides personalized predictions and recommendations, aiding in early diagnosis and proactive healthcare interventions. This innovative technology holds the potential to enhance public health by enabling timely interventions, reducing healthcare costs, and preventing the spread of infectious diseases.

1.1.1 ADVANTAGES

- Timely Diagnosis: Early disease detection is made possible by it, which improves patient outcomes and treatment efficacy.
- Accessibility: Reaches a larger community and eases the burden on healthcare institutions by providing health information and forecasts around-the-clock.
- Personalized care: Enhances patient engagement and adherence to medical guidance by customising recommendations based on unique symptoms and risk factors.
- Data analysis: Helps medical practitioners monitor and control disease outbreaks by quickly analysing large datasets.
- Cost-Effective: Lowers medical expenses by avoiding needless hospital stays and reducing the financial strain of advanced medical interventions.

1.2 MOTIVATION

The development of an AI medical chatbot that forecasts infectious diseases and illnesses using machine learning is motivated by the urgent need to change healthcare. This chatbot uses AI and ML to provide precise, easily accessible, and real-time disease predictions in an effort to enhance early identification and treatment. It addresses the growing issues of global health, including the dearth of resources in the medical field and the rise in infectious diseases. Moreover, it empowers individuals to take charge of their health, reducing the burden on healthcare systems and perhaps saving lives. The ultimate objectives of the chatbot are to decrease healthcare costs, enhance public health, and promote universal access to healthcare.

1.3 PROJECT STATEMENT

- Artificial intelligence (AI) chatbots are designed to simulate human conversations, but they are not smart enough to comprehend complicated circumstances or offer individualised care. Chatbots are typically insufficient when it comes to solving complicated problems; they are capable of handling simple and repetitive jobs.
- The potential for data breaches is one of the main hazards. Cybercriminals target health care providers because they generate, collect, store, and transfer vast amounts of sensitive patient data. Vulnerabilities may and will be attacked by bad actors at every point in the AI data pipeline.

1.4 OBJECTIVES

- Early Detection: To aid in an early diagnosis, identify risk factors and disease signs in users.
- Real-time forecasts: Help people understand their health status by providing timely and accurate disease forecasts.
- Provide individualised advice based on patient profiles regarding healthcare, treatment, and prevention.
- Scalability: Expand the chatbot's functionalities to meet the demands of a growing user base and changing healthcare requirements.

1.5 SCOPE OF THE PROJECT

An AI medical chatbot's application for machine learning-based infection and disease prediction is extensive and significant. It provides individualised health recommendations and includes early detection, diagnosis, and monitoring of a range of infectious diseases. Through the provision of real-time information and data analysis for disease surveillance, the chatbot aids in public health initiatives. It eases the load on healthcare systems by enabling remote consultations. It also functions as a useful teaching tool, spreading medical knowledge and encouraging health consciousness. It can make a substantial contribution to global healthcare with its capacity for ongoing learning and adaptation. By providing a broad spectrum of users with current and easily available medical advice, it can eventually enhance patient outcomes and slow the spread of disease.

Chapter 2

Literature survey

S.NO	Title	Merit	Demerit
1	An AI based medical chatbot for infectious disease prediction	The chatbot provides all the information about the available medications and therapies for Covid 19 along with their advantages and side effects.	The dataset used is relatively normal. More robust evaluation would require larger and more varied data's
2	Machine learning algorithms for teaching AI chat bots	The creation of a chatbot platform with a microservice architecture and the application of NLU enables the testing of different algorithms to determine how well they perform when communicating with humans.	Had to find the effective feedback from the user.

3	Medical Chatbot for Novel COVID-19	The prime focus of this paper is to show implementation of a retrieval-based chabot with voice support, and we will investigate other standing chatbot and how it is useful in helping the patients fetching all the necessary details about COVID-19.	But leads some data breaches & mis-conception in voice support.
4	Information delivered by a chatbot has a positive impact on COVID-19 vaccines attitudes and intentions	This paper provided participants with access to a rather comprehensive collection of questions and answers about the COVID-19 vaccines via a straightforward chatbot.	Unfortunately, it leads to 2 interpretations which leads to down fall this chatbot.
5	Building a Medical Chatbot using Support Vector Machine Learning Algorithm	The system helps medical facilities and hospitals assist patients by answering voice or text questions about their health.	However, it provides only basic enquiries.

6	A Personalized Medical Assistant Chatbot: Medi Bot	A chatbot that may be used for self-diagnosis as well as ordinary discussion has been developed. In the sphere of medical science, the method can be highly helpful for the earlier and quicker diagnosis of sickness.	The absence of a reliable and accurate medical dataset is the largest problem. There are numerous datasets accessible for creating conversational chatbots, but just one, very limited dataset is available for disease prediction.
7	Web-based chatbot for Frequently Asked Queries (FAQ) in Hospitals	This paper's current method is based on speech recognition. A chatbot that deals with healthcare. Through a microphone, users can ask questions here, and NLP converts the spoken requests into text.	There aren't many answers to the questions asked by people, despite the speedy response. For hospital administrators, testing and building up AI on a broad scale can be costly.
8	Bangla Healthcare Chatbot using Machine Learning	It keeps a record of the user's health and conducts associated actions solely in Bangla.	Incapable of recognising enough signs to make a sound judgement.

9	Conversational A.I. Chatbot for Delivering Tele-Health	Reduces the disparity between the rural population by being aware of their requirements, giving them the right information, and possibly advising them to take precautions.	Does not cope with aid for an antidepressant system that will be centred on offering music therapy and mental health examinations.
10	Contextual Chatbot	The user can comprehend the issues they are having and receive a prognosis regarding any potential diseases.	Other than text, the user cannot use any other language interface.

2.1 SUMMARY OF THE EXISTING WORKS

Existing works in the field of AI medical chatbots for infectious and disease prediction using machine learning have focused on developing systems that leverage vast medical data to enhance early disease detection, personalized health recommendations, and real-time interaction with users. Researchers have addressed challenges related to data quality, privacy, regulatory compliance, and bias mitigation. Many studies have emphasized continuous learning and integration with healthcare systems. These works showcase the potential of chatbots to improve healthcare accessibility, while ongoing research continues to refine these systems to ensure they deliver accurate, ethical, and effective medical support.

2.2 CHALLENGES PRESENT IN EXISTING SYSTEM

- Data Quality: Reliability of predictions depends on precise and comprehensive medical data.
- Restricted Access: Resolving inequalities in technology and medical resource accessibility.

Chapter 3

Requirements

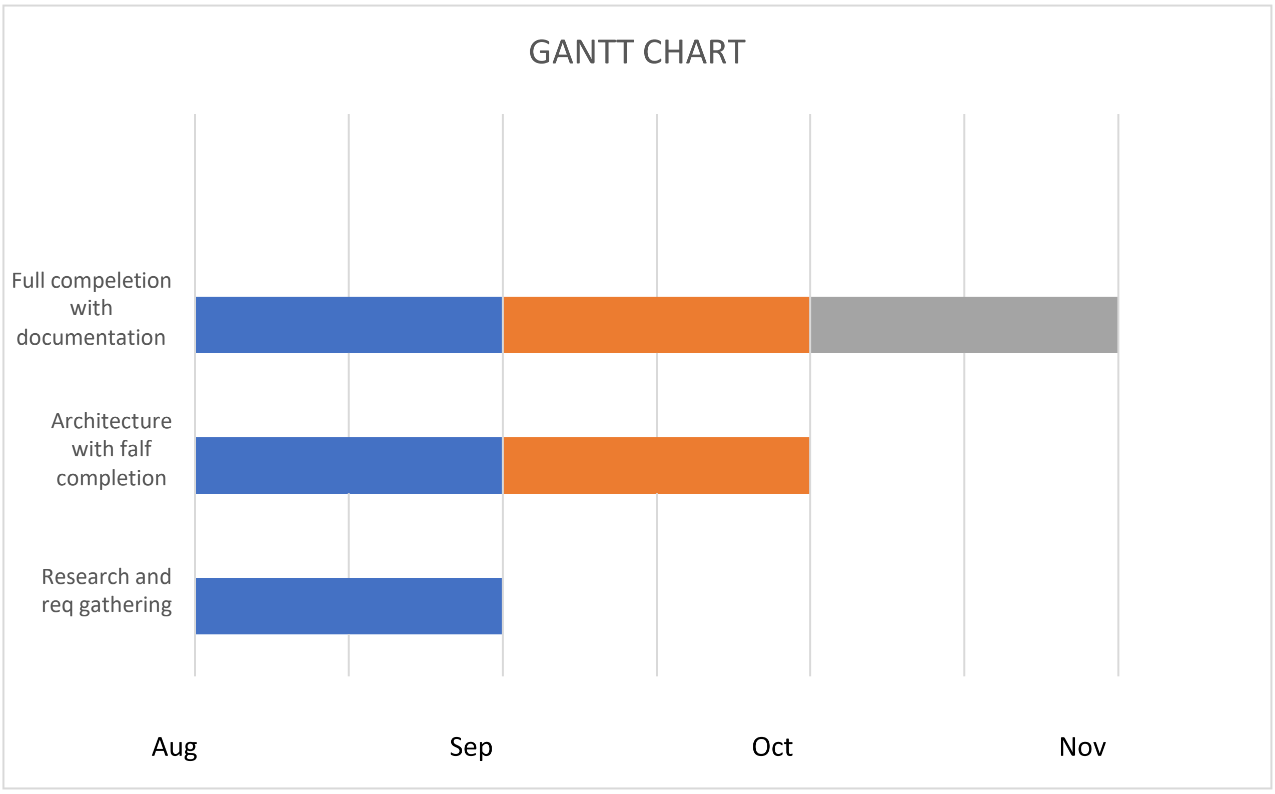
3.1 HARDWARE REQUIREMENTS

❖ System	: Lenovo laptop with intel.
❖ Hard Disk	: 40 GB.
❖ Ram	: 512 Mb.

3.2 SOFTWARE REQUIREMENTS

❖ Operating system	: Windows 11.
❖ Coding Language	: Python.
❖ Front-End	: HTML, CSS.

3.3GANTT CHART



Chapter 4

Analysis and design

4.1 PROPOSED METHODOLOGY

Our proposal involves utilising a deep feed forward multilayer perceptron to create an AI Chatbot interaction and prediction model. Our investigation uncovered a knowledge vacuum regarding theoretical standards and useful advice for developing AI chatbots for programmes aimed at improving people's lifestyles. This report also includes a quick comparison of our proposed model with respect to testing accuracy and time complexity. Our work has a minimum loss of 0.1232 and a maximum accuracy of 94.32%. This paper examines the related issues posed by using these developing technologies during such health crises, primarily caused by pandemics, and describes the capabilities and potential applications of medical chatbots. We think that the layout and uses of these cutting-edge technologies will be better understood by researchers thanks to our findings, which will be necessary for medical chatbots to continue improving.

Algorithms:

CNN

Introduction

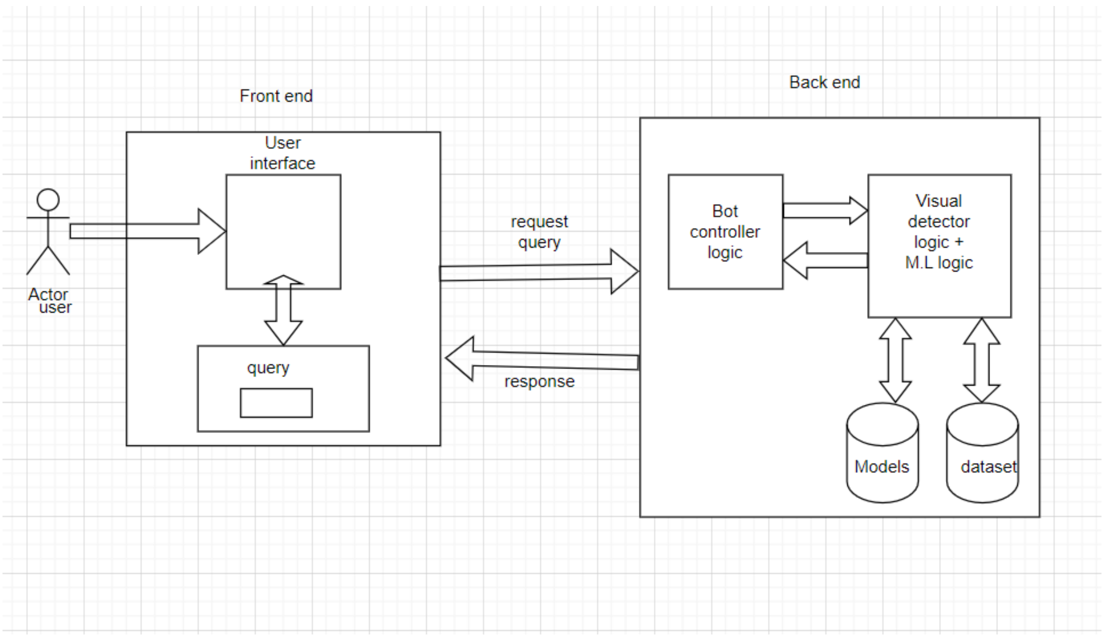
In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.

Since the 1950s, the early days of AI, researchers have struggled to make a system that can understand visual data. In the following years, this field came to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms and that too by a large margin.

Background of CNNs

CNN’s were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

4.2 SYSTEM ARCHITECTURE



4.3 MODULE DESCRIPTIONS

4.3.1 CREATE INTENTS

An intent is the intention of the user interacting with a chatbot or the intention behind each message that the chatbot receives from a particular user . Define simple intents and set of messages that corresponds to those intents and also map some responses according to each intent category.

4.3.2 DATA PREPARATION

The variable words holds all the tokenized training data (which are sample messages in each intent category/tag), the classes variable holds all target labels/tags correspond to each training data and the documents variable holds both training data (words of sample messages tokenized using word tokenizer (pattern)) and labels correspond to each training data. After this, eliminate symbols, defined in ignore letters variable, from training data and also eliminate stop words, if required.

4.3.3 CREATE BAG OF WORDS

Convert text data into numeric format (0 and 1). Create bag of words, such as if word is present in word patterns append 1, otherwise append 0 to the list.

4.3.4 MODEL TRAINING

Define Convolutional Neural Network architecture for proposed model and for that use the Sequential model class of Kera’s. Kera’s is an open source, high level library for developing Convolutional Neural Network model .

The steps for creating a Kera’s (Sequential) model are the following:

Step 1: Define Convolutional Neural Network model (i.e., Sequential model). The network is defined as a sequence of layers, each with its own customizable size and activation function. In this sequential model, the first layer is the input layer, which define the size of the input, which is feed to Convolutional Neural Network. After this more and more layers (hidden layers) can be added and customized until reached to the final output layer

Step 2: Define the optimization algorithm that will be used to train the sequential model, for that use SGD optimizer and also choose the loss function. Then, compile the Convolutional Neural Network, which transforms the simple sequence of layers into a complex group of matrix operations that describes the behaviour of the network.

Step 3: Train or fit the Convolutional Neural Network and save it in chatbot model file.

Step 4: Now, model/network is trained, an input can be used to make prediction

4.3.5 GET USER INPUT

Accept input query/message from user with developed frontend using Flask framework.

4.3.6 PREDICT CLASS FOR INPUT

Filter out the class/tag for user input, with specified threshold and make list of all filtered tags. Sort the list in reverse order, so that it returns class/tag, which has the highest probability. Finally, return the list, with classes and their corresponding probability.

4.3.7 GET BOT RESPONSE

Store the class/tag, returned by previous function, having the highest probability. Compare that class/tag, with tags defined in intents file. Return the response corresponding to that class/tag.

4.3.8 DISPLAY BOT RESPONSE

Display bot Response using flask framework.

Chapter 5

Implementation & testing

5.1 DATASET

The dataset used here is text data which have disease description and some image data’s to predict the dementia disease

5.2 SAMPLE CODE

FOR MAIN APP:

```
from flask import *
import os
from werkzeug.utils import secure_filename
import label_image
from flask import Flask, render_template, request
from bot import chat
def load_image(image):
   text = label_image.main(image)
   return text
app = Flask(__name__)

@app.route('/')
@app.route('/first')
def first():
   return render_template('first.html')

@app.route('/login')
def login():
   return render_template('login.html')
```

```
@app.route('/chart')
def chart():
    return render_template('chart.html')


@app.route('/index')
def index():
    return render_template('index.html')


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']
        file_path = secure_filename(f.filename)
        f.save(file_path)
        # Make prediction
        result = load_image(file_path)
        result = result.title()
        d = {"brownspot":" → You are Normal. Keep Calm and Make Healthy Life",
            'Mild Demented':" → It may be a starting stage of the Alzheimer's disease consult the doctor asap and get well soon!!.",
        "Moderate Demented":" → Neurodegenerative diseases are incurable and debilitating conditions that result in progressive degeneration and / or death of nerve cells. This causes problems with movement (called ataxias), mental functioning (called dementias) and affect a person's ability to move, speak and breathe[1]. Neurodegenerative disorders impact many families - these disorders are not easy for the individual nor their loved ones.",
        "Non Demented":" → You are perfectly alright according to your MRI scan image hope you are staying good.",
        "Verymild Demented":" →It is a very mild stage of the Alzheimer disease there is a more probability to recover from this disease get treated from the neurologist and get well soon."}
        result = result+d[result]
        #result2 = result+d[result]
        #result = [result]
        #result3 = d[result]
        print(result)
        #print(result3)
        os.remove(file_path)
        return result
        #return result3
    return None

@app.route("/home")
def home():
    return render_template("index1.html")

@app.route("/get")
#function for the bot response
def get_bot_response():
    userText = request.args.get('msg')
    exit_list = ['exit','see you later','bye','quit','break']
    if userText.lower() in exit_list:
            return "Bye..Take Care..Chat with you later!!"
            #break;
    else:
            return str(chat(userText))

 @app.errorhandler(500)
 def internal_error(error):
            return "500 error"

@app.errorhandler(404)
def not_found(error):
    return "404 error",404
if __name__== '_main_':
    app.run()
```

FOR BOT RESPONSE:

```
import nltk
import warnings
warnings.filterwarnings("ignore")
# nltk.download() # for downloading packages
#import tensorflow as tf
import numpy as np
import random
import string # to process standard python strings

#nltk.download('punkt')       # first-time use only
#nltk.download('wordnet')      # first-time use only

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

f1=open('corpus1.txt','r',errors = 'ignore')
f2=open('corpus2.txt','r',errors = 'ignore')
f3=open('corpus3.txt','r',errors = 'ignore')
f4=open('corpus4.txt','r',errors = 'ignore')
f5=open('corpus5.txt','r',errors = 'ignore')
f6=open('corpus6.txt','r',errors = 'ignore')
f7=open('corpus7.txt','r',errors = 'ignore')
f8=open('corpus8.txt','r',errors = 'ignore')
f9=open('corpus9.txt','r',errors = 'ignore')
f10=open('corpus10.txt','r',errors = 'ignore')
f11=open('corpus11.txt','r',errors = 'ignore')
f12=open('corpus12.txt','r',errors = 'ignore')
f13=open('corpus13.txt','r',errors = 'ignore')
f14=open('corpus14.txt','r',errors = 'ignore')
f15=open('corpus15.txt','r',errors = 'ignore')
```

```
checkpoint = "./chatbot_weights.ckpt"

raw1=f1.read()                         #1. Diabetes
raw1=raw1.lower()
sent_tokens1 = nltk.sent_tokenize(raw1)
word_tokens1 = nltk.word_tokenize(raw1)
sent_tokens1[:2]
word_tokens1[:5]

raw2=f2.read()                         #2.Obesity
raw2=raw2.lower()
sent_tokens2 = nltk.sent_tokenize(raw2)
word_tokens2 = nltk.word_tokenize(raw2)
sent_tokens2[:2]
word_tokens2[:5]

raw3=f3.read()                         #3.High Blood Pressure
raw3=raw3.lower()
sent_tokens3 = nltk.sent_tokenize(raw3)
word_tokens3 = nltk.word_tokenize(raw3)
sent_tokens3[:2]
word_tokens3[:5]

raw4=f4.read()                         #4.Low Blood Pressure
raw4=raw4.lower()
sent_tokens4 = nltk.sent_tokenize(raw4)
word_tokens4 = nltk.word_tokenize(raw4)
sent_tokens4[:2]
word_tokens4[:5]

raw5=f5.read()                         #5.Liver Disease
raw5=raw5.lower()
```

```
sent_tokens5 = nltk.sent_tokenize(raw5)
word_tokens5 = nltk.word_tokenize(raw5)
sent_tokens5[:2]
word_tokens5[:5]

raw6=f6.read()                              #6.Heart Disease
raw6=raw6.lower()
sent_tokens6 = nltk.sent_tokenize(raw6)
word_tokens6 = nltk.word_tokenize(raw6)
sent_tokens6[:2]
word_tokens6[:5]

raw7=f7.read()                              #7.Migrane
raw7=raw7.lower()
sent_tokens7 = nltk.sent_tokenize(raw7)
word_tokens7 = nltk.word_tokenize(raw7)
sent_tokens7[:2]
word_tokens7[:5]

raw8=f8.read()                             #8.Diarrhoea
raw8=raw8.lower()
sent_tokens8 = nltk.sent_tokenize(raw8)
word_tokens8 = nltk.word_tokenize(raw8)
sent_tokens8[:2]
word_tokens8[:5]

raw9=f9.read()                             #9.Depression / Anxiety
raw9=raw9.lower()
sent_tokens9 = nltk.sent_tokenize(raw9)
word_tokens9 = nltk.word_tokenize(raw9)
sent_tokens9[:2]
word_tokens9[:5]
```

```
raw10=f10.read()                          #10. Cancer
raw10=raw10.lower()
sent_tokens10 = nltk.sent_tokenize(raw10)
word_tokens10 = nltk.word_tokenize(raw10)
sent_tokens10[:2]
word_tokens10[:5]

raw11=f11.read()                          #11. Kidney Disease
raw11=raw11.lower()
sent_tokens11 = nltk.sent_tokenize(raw11)
word_tokens11 = nltk.word_tokenize(raw11)
sent_tokens11[:2]
word_tokens11[:5]

raw12=f12.read()                           #12. Insomnia
raw12=raw12.lower()
sent_tokens12 = nltk.sent_tokenize(raw12)
word_tokens12 = nltk.word_tokenize(raw12)
sent_tokens12[:2]
word_tokens12[:5]

raw13=f13.read()                              #13. Allergy
raw13=raw13.lower()
sent_tokens13 = nltk.sent_tokenize(raw13)
word_tokens13 = nltk.word_tokenize(raw13)
sent_tokens13[:2]
word_tokens13[:5]

raw14=f14.read()                              #14.Dengue
raw14=raw14.lower()
sent_tokens14 = nltk.sent_tokenize(raw14)
```

```
word_tokens14 = nltk.word_tokenize(raw14)
sent_tokens14[:2]
word_tokens14[:5]

raw15=f15.read()                                    #15.Malaria
raw15=raw15.lower()
sent_tokens15 = nltk.sent_tokenize(raw15)
word_tokens15 = nltk.word_tokenize(raw15)
sent_tokens15[:2]
word_tokens15[:5]

lemmer = nltk.stem.WordNetLemmatizer()

def LemTokens(tokens):
   return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
def LemNormalize(text):
   return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))

Introduce_Ans = ["My name is Remdex.","My name is Remdex and I will answer your queries.","Im Remdex :) ","My name is Remdex and I am happy to solve your queries :) "]
GREETING_INPUTS = ("hello", "hi","hiii","hii","hiiii","hiiii", "greetings", "sup", "what's up","hey",)
GREETING_RESPONSES = ["hi", "hey", "hii there", "hi there", "hello", "I am glad! You are talking to me"]
Basic_Q = ("what is m+ store ?","what is m+ store","what is m+ store?","What is m+ store.")
Basic_Ans = "M + Store is an Online Medical Store.We supply medicines at your doorstep. You can order medicines in our website you need by uploading proper prescription.Kindly go through our website once to know better."
Basic_Q1 = ("from where you collect medicine?","from where you collect medicine","from where you collect medicine.","where can I get medicine?","where can I get medicine","where can I get medicine.")
Basic_Ans1 = "We collect generic medicines and supply it to your doorstep at a discount price.We collect it from different authentic sellers and Pradhan Mantri
```

```
Bhartiya Jan Aushadhi Pariyojana Kendra.The list of Jan Aushadhi Pariyojana Kendras are given in our website"

Basic_Q2 = ("how much you charge?","how much you charge","how much you charge.","what is the price of medicine?","what is the price of medicine","what is the price of medicine.")

Basic_Ans2 = "We supply generic medicines at a discount price.Please go through our website for more information about medicine price"

Basic_Q3 = ("what is the difference between a brand name and a generic drug?","what is generic medicine?","what is generic medicine","what is generic medicine.")

Basic_Ans3 = "When a medication is first developed, the manufacturer has patent rights on the formula and/or compound. Once this patent right expires, other companies can produce generic versions of the drug that meet the same FDA requirements and regulations as the brand name drug. Most insurance companies require generic substitutions unless specifically requested by the prescriber or patient.We supply generic medicines.It is always advisable to take medicines only after consulting a doctor."


# Checking for greetings

def greeting(sentence):

    """If user's input is a greeting, return a greeting response"""

    for word in sentence.split():

        if word.lower() in GREETING_INPUTS:

            return random.choice(GREETING_RESPONSES)


# Checking for Basic_Q

def basic(sentence):

    for word in Basic_Q:

        if sentence.lower() == word:

            return Basic_Ans


def basic1(sentence):

    for word in Basic_Q1:

        if sentence.lower() == word:

            return Basic_Ans1


def basic2(sentence):
    for word in Basic_Q2:
        if sentence.lower() == word:
            return Basic_Ans2

def basic3(sentence):
    for word in Basic_Q3:
        if sentence.lower() == word:
            return Basic_Ans3

# Checking for Introduce
def IntroduceMe(sentence):
    return random.choice(Introduce_Ans)

SYMPTOM_RESPONSES =["Ohhh..","Sorry to hear that","That doesn't sound good
at all","Be cautious about yourself"]
def findsymptom():
  return random.choice(SYMPTOM_RESPONSES)

# Generating response 1
def response1(user_response):
    robo_response=''
    sent_tokens1.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens1)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens1[idx]
        return robo_response

# Generating response 2
def response2(user_response):
    robo_response=''
    sent_tokens2.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens2)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens2[idx]
        return robo_response
# Generating response 3
def response3(user_response):
    robo_response=''
    sent_tokens3.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens3)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens3[idx]
        return robo_response

# Generating response 4
def response4(user_response):
    robo_response=''
    sent_tokens4.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens4)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens4[idx]
        return robo_response

# Generating response 5
def response5(user_response):
    robo_response=''
    sent_tokens5.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens5)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens5[idx]
        return robo_response

# Generating response 6
def response6(user_response):
    robo_response=''
    sent_tokens6.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens6)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens6[idx]
        return robo_response

# Generating response 7
def response7(user_response):
    robo_response=''
    sent_tokens7.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens7)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens7[idx]
        return robo_response

# Generating response 8
def response8(user_response):
    robo_response=''
    sent_tokens8.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens8)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens8[idx]
        return robo_response
```

```
# Generating response 9
def response9(user_response):
    robo_response=''
    sent_tokens9.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens9)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens9[idx]
        return robo_response

# Generating response 10
def response10(user_response):
    robo_response=''
    sent_tokens10.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens10)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens10[idx]
        return robo_response

# Generating response 11
def response11(user_response):
    robo_response=''
    sent_tokens11.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens11)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens11[idx]
        return robo_response

# Generating response 12
def response12(user_response):
    robo_response=''
    sent_tokens12.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens12)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens12[idx]
        return robo_response

# Generating response 13
def response13(user_response):
    robo_response=''
    sent_tokens13.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens13)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens13[idx]
        return robo_response

# Generating response 14
def response14(user_response):
    robo_response=''
    sent_tokens14.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens14)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens14[idx]
        return robo_response

# Generating response 15
def response15(user_response):
    robo_response=''
    sent_tokens15.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens15)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens15[idx]
        return robo_response
def chat(user_response):
    user_response=user_response.lower()
```

```
keyword =  "m+"
keywordone = "store"
keywordsecond = "medicine"

keyS1 = "fever"
keyS2 = "headache"
keyS3 = "vomitting"
keyS4 = "sore throat"
keyS5 = "cough"
keyS6 = "breathing problem"
keyS7 = "pain"
keyS8 = "itching"

key0 = "disease"
key1 = "diabetes"
key2 = "obesity"
key3 = "high blood pressure"
key4 = "low blood pressure"
key5 = "liver"
key6 = "heart"
key7 = "migrane"
key8 = "diarrhoea"
key8a = "diarrhea"
key9 = "depression"
key9a = "anxiety"
key10 = "cancer"
key11 = "kidney"
key12 = "insomnia"
key13 = "allergy"
key14 = "dengue"
key15 = "malaria"
```

```
    exit_list = ['exit','see you later','bye','quit','break']
    if user_response.lower() not in exit_list:
      if(user_response=='thanks' or user_response=='thank you' ):
        flag=False
        #print("ROBO: You are welcome..")
      return "You are welcome.."
      elif(basic(user_response)!=None):
        return basic(user_response)
      #print(basic(user_response))
      elif(basic1(user_response)!=None):
        return basic1(user_response)
      #print(basic1(user_response))
      elif(basic2(user_response)!=None):
        return basic2(user_response)
      #print(basic2(user_response))
      elif(basic3(user_response)!=None):
        return basic3(user_response)
      #print(basic3(user_response))
      elif(user_response.find(keywordsecond ) != -1):
         mans = "Please take any medicine only after consulting with a doctor.You can order medicine from M+ store by uploading a proper prescription. "
         return mans
         #print(mans)
      else:
        if(user_response.find(keyword) != -1 or user_response.find(keywordone) != -1):
          #print("REMDEX: ",end="")
          #print(basic(user_response))
          return basic(user_response)
        elif(user_response.find(keyS1) != -1 or user_response.find(keyS2) != -1 or user_response.find(keyS3) != -1 or user_response.find(keyS4) != -1 or user_response.find(keyS5) != -1 or user_response.find(keyS6) != -1 or user_response.find(keyS7) != -1 or user_response.find(keyS8) != -1):
          sans = "This could be a serious symptom..Please Consult a Doctor ASAP."
      #print(findsymptom())
      return (findsymptom() + '\n' + sans)
      #print(sans)
    elif(user_response.find(key1) != -1):                    #1
        #print("REMDEX: ",end="")
        #print(response1(user_response))
        return response1(user_response)
        sent_tokens1.remove(user_response)
    elif(user_response.find(key2) != -1):                    #2
        #print("REMDEX: ",end="")
        #print(response2(user_response))
        return response2(user_response)
        sent_tokens2.remove(user_response)
    elif(user_response.find(key3) != -1):                    #3
        #print("REMDEX: ",end="")
        #print(response3(user_response))
        return response3(user_response)
        sent_tokens3.remove(user_response)
    elif(user_response.find(key4) != -1):                    #4
        #print("REMDEX: ",end="")
        #print(response4(user_response))
        return response4(user_response)
        sent_tokens4.remove(user_response)
    elif(user_response.find(key5)  !=  -1  and  user_response.find(key0)  !=  -1):
#5
        #print("REMDEX: ",end="")
        #print(response5(user_response))
        return response5(user_response)
        sent_tokens5.remove(user_response)
    elif(user_response.find(key6)  !=  -1  and  user_response.find(key0)  !=  -1):
#6
        #print("REMDEX: ",end="")
        #print(response6(user_response))
        return response6(user_response)
        sent_tokens6.remove(user_response)
    elif(user_response.find(key7) != -1):                              #7
        #print("REMDEX: ",end="")
        #print(response7(user_response))
        return response7(user_response)
        sent_tokens7.remove(user_response)
    elif(user_response.find(key8)   !=   -1   or   user_response.find(key8a)   !=   -1):
#8
        #print("REMDEX: ",end="")
        #print(response8(user_response))
        return response8(user_response)
        sent_tokens8.remove(user_response)
    elif(user_response.find(key9)   !=   -1   or   user_response.find(key9a)   !=   -1):
#9
        #print("REMDEX: ",end="")
        #print(response9(user_response))
        return response9(user_response)
        sent_tokens9.remove(user_response)
    elif(user_response.find(key10) != -1):                             #10
        #print("REMDEX: ",end="")
        #print(response10(user_response))
        return response10(user_response)
        sent_tokens10.remove(user_response)
    elif(user_response.find(key11) != -1):                             #11
        #print("REMDEX: ",end="")
        #print(response11(user_response))
        return response11(user_response)
        sent_tokens11.remove(user_response)
    elif(user_response.find(key12) != -1):                             #12
        #print("REMDEX: ",end="")
        #print(response12(user_response))
        return response12(user_response)
```

```
            sent_tokens12.remove(user_response)
        elif(user_response.find(key13) != -1):                          #13
            #print("REMDEX: ",end="")
            #print(response13(user_response))
            return response13(user_response)
            sent_tokens13.remove(user_response)
        elif(user_response.find(key14) != -1):                          #14
            #print("REMDEX: ",end="")
            #print(response14(user_response))
            return response14(user_response)
            sent_tokens14.remove(user_response)
        elif(user_response.find(key15) != -1):                          #15
            #print("REMDEX: ",end="")
            #print(response15(user_response))
            return response15(user_response)
            sent_tokens15.remove(user_response)
        elif(greeting(user_response)!=None):
          #print("REMDEX: "+greeting(user_response))
          return greeting(user_response)
        elif(user_response.find("your name") != -1 or user_response.find(" your name")
!= -1 or user_response.find("your name ") != -1 or user_response.find(" your name ")
!= -1):
          return IntroduceMe(user_response)
          #print(IntroduceMe(user_response))


        else:
          #print("REMDEX: ",end="")
          cans = "I am just a chatbot. Please consult a doctor for your further queries."
          #print(cans)
          return cans


    else:
        flag=False
        #print("ROBO: Bye! Take care..Chat with you later!!")
        return "ROBO: Bye! Take care..Chat with you later!!"
```

CODE FOR BOTSERVER:

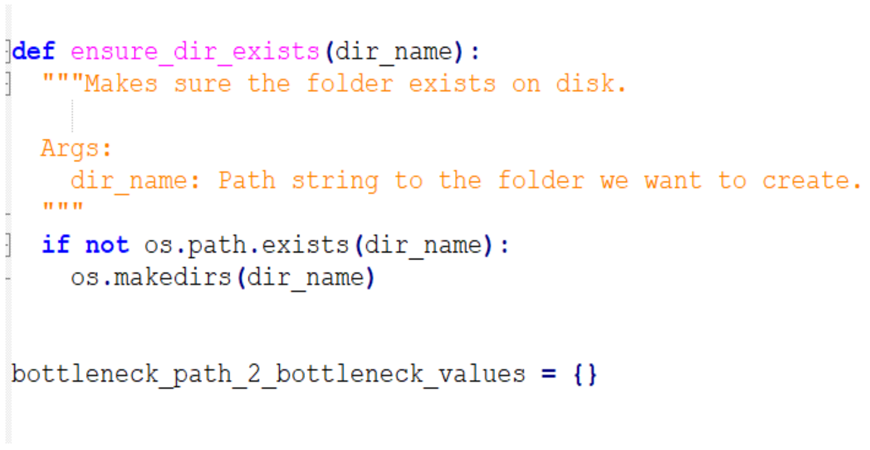
```
from flask import Flask, jsonify, request
from flask_cors import CORS
from bot import chat

app = Flask(_name_)
CORS(app)

@app.route('/talk',methods=['POST'])
def index():
    user_input = request.json['user_input']
    return jsonify({'msg':str(chat(user_input))})

@app.route('/botTalk',methods=['GET'])
def talkViaApi():
    user_input = request.args.get("user_input")
    return jsonify({'msg':str(chat(user_input))})

if __name__== '__main__':
  app.run(host='127.0.0.1', port=8000, debug=True)
```

CODE FOR RESPONSE:

```
from bot import chat


print('REMDEX: I will answer your queries...If you want to exit,type bye... :)')
W = "REMDEX: "
exit_list = ['exit','see you later','bye','quit','break']
```

```
while True:
  user_input = input()
  if user_input.lower() in exit_list:
    print('REMDEX: Bye..Take Care..Chat with you later!!')
    break
  else:
    ans = chat(user_input)
    print(W+str(ans))
```

ALGORITHM CODE:

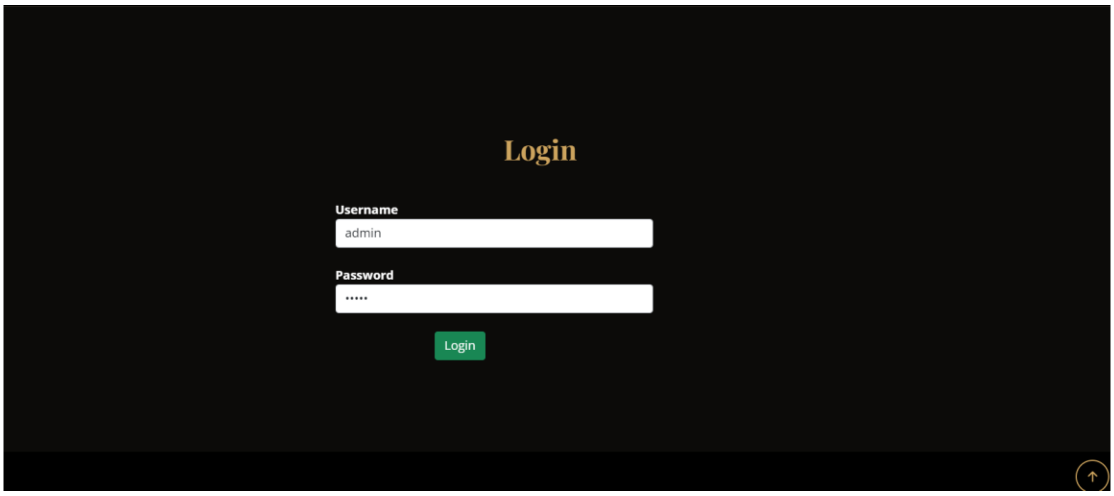


```
def ensure_dir_exists(dir_name):
  """Makes sure the folder exists on disk.

  Args:
    dir_name: Path string to the folder we want to create.
  """
  if not os.path.exists(dir_name):
    os.makedirs(dir_name)


bottleneck_path_2_bottleneck_values = {}
```

```
def create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor):
  """Create a single bottleneck file."""
  tf.logging.info('Creating bottleneck at ' + bottleneck_path)
  image_path = get_image_path(image_lists, label_name, index,
                              image_dir, category)
  if not gfile.Exists(image_path):
    tf.logging.fatal('File does not exist %s', image_path)
  image_data = gfile.FastGFile(image_path, 'rb').read()
  try:
    bottleneck_values = run_bottleneck_on_image(
        sess, image_data, jpeg_data_tensor, decoded_image_tensor,
        resized_input_tensor, bottleneck_tensor)
  except Exception as e:
    raise RuntimeError('Error during processing file %s (%s)' % (image_path,
                                                                 str(e)))
  bottleneck_string = ','.join(str(x) for x in bottleneck_values)
  with open(bottleneck_path, 'w') as bottleneck_file:
    bottleneck_file.write(bottleneck_string)
```

5.3SAMPLE OUTPUT:

- The user login using their username and password.

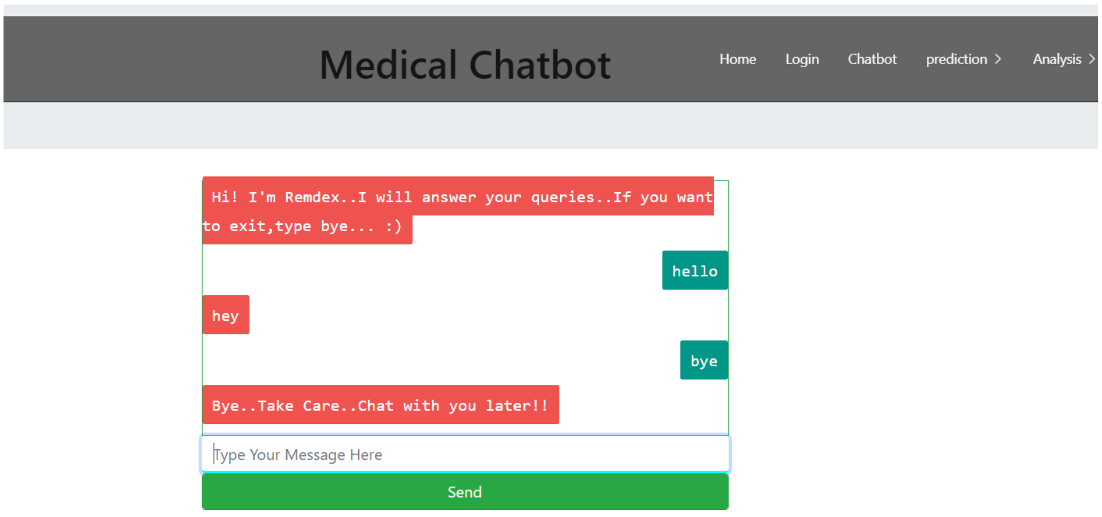

Login

Username
admin

Password
•••••

Login

- The interaction with the chatbot.



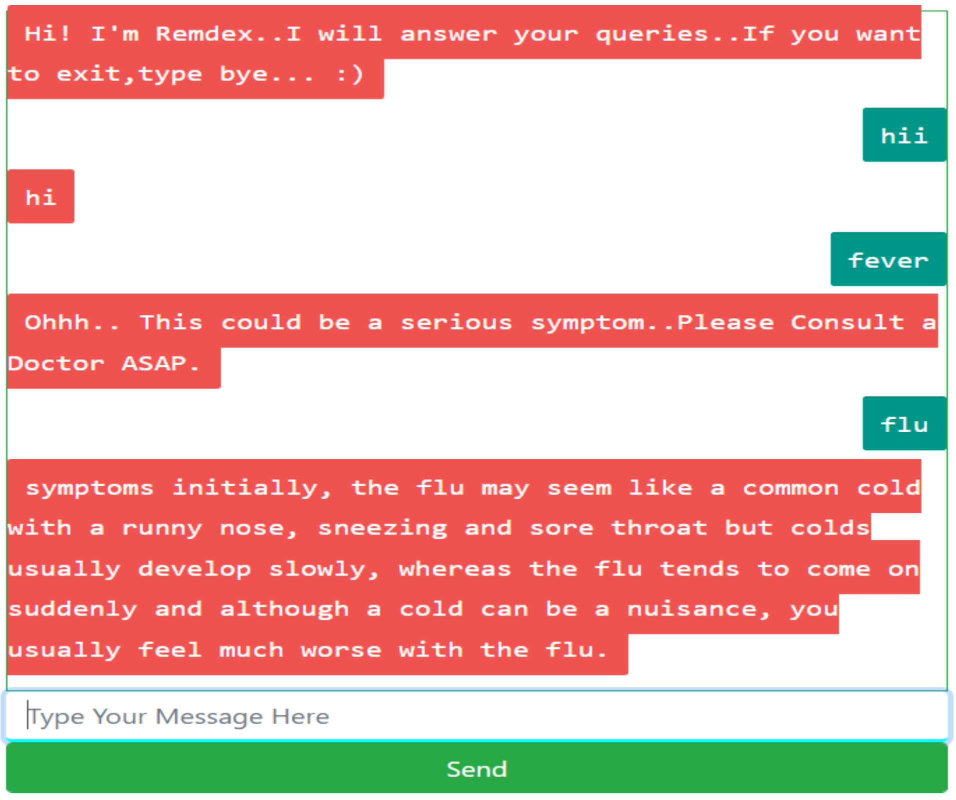
Hi! I'm Remdex..I will answer your queries..If you want to exit,type bye... :)

hello

hey

bye

Bye..Take Care..Chat with you later!!



Hi! I'm Remdex..I will answer your queries..If you want to exit,type bye... :)

hii

hi

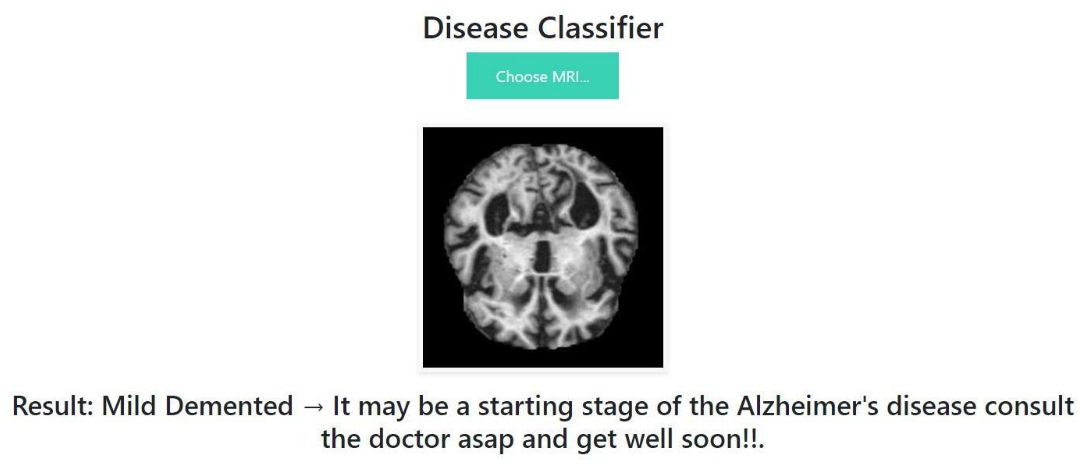
fever

Ohhh.. This could be a serious symptom..Please Consult a Doctor ASAP.

flu

symptoms initially, the flu may seem like a common cold with a runny nose, sneezing and sore throat but colds usually develop slowly, whereas the flu tends to come on suddenly and although a cold can be a nuisance, you usually feel much worse with the flu.

- Along with the chatbot predicting one portion of dementia disease with MRI scan images.

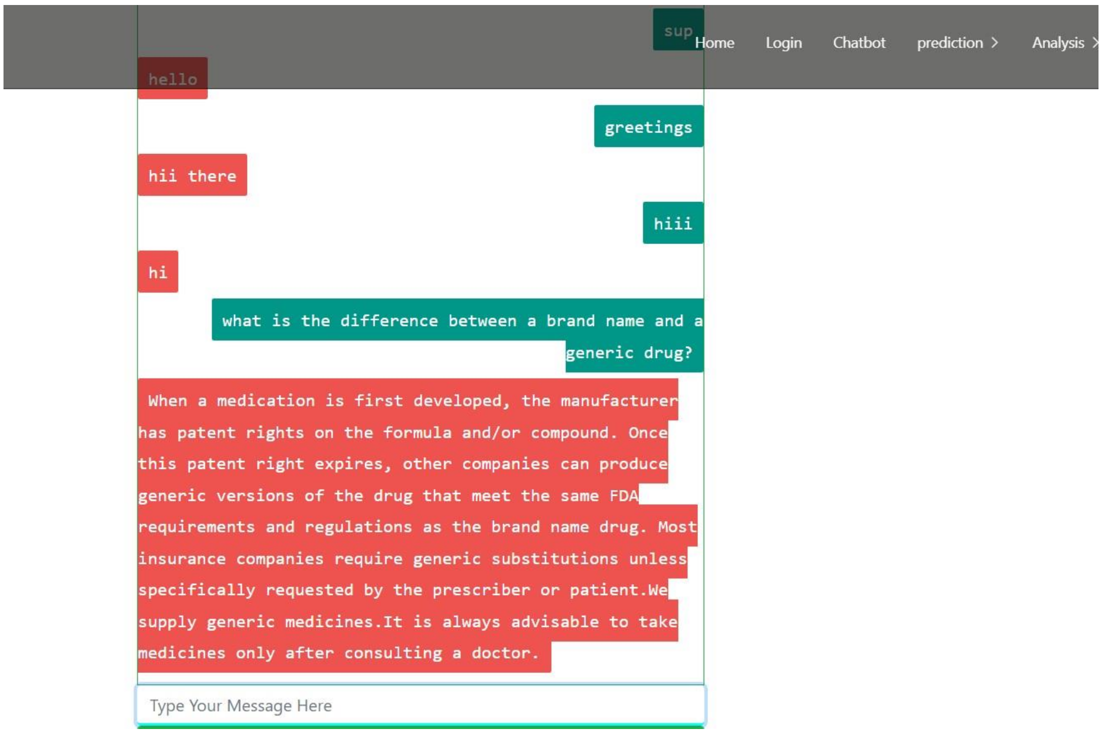

Disease Classifier

Result: Mild Demented → It may be a starting stage of the Alzheimer's disease consult the doctor asap and get well soon!!.

5.4 TEST PLAN AND DATA VERIFICATION

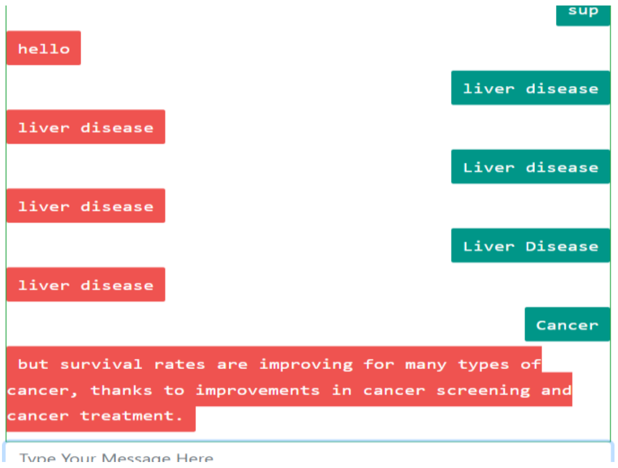
5.4.1 FOR CHATBOT

- Collecting and adding the raw data’s for the common infectious diseases.
- Then train those data’s.
- Now test it using the trained keywords for those diseases.



```
sup
hello
greetings
hii there
hiii
hi
what is the difference between a brand name and a generic drug?
When a medication is first developed, the manufacturer has patent rights on the formula and/or compound. Once this patent right expires, other companies can produce generic versions of the drug that meet the same FDA requirements and regulations as the brand name drug. Most insurance companies require generic substitutions unless specifically requested by the prescriber or patient.We supply generic medicines.It is always advisable to take medicines only after consulting a doctor.
Type Your Message Here
```

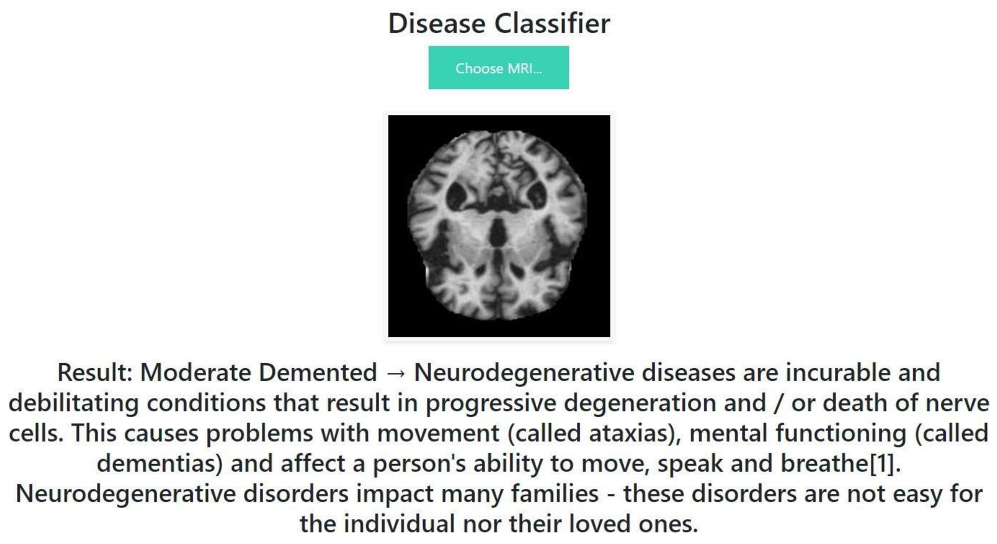
- The given input data is verified by the bot response.
- Now testing using the untrained words or data’s.



```
sup
hello
liver disease
liver disease
Liver disease
liver disease
Liver Disease
liver disease
Cancer
but survival rates are improving for many types of cancer, thanks to improvements in cancer screening and cancer treatment.
Type Your Message Here
```

5.4.2 FOR DISEASE PREDICTION

- The dataset consists in image format to recognise and plot the result.



Result: Moderate Demented → Neurodegenerative diseases are incurable and debilitating conditions that result in progressive degeneration and / or death of nerve cells. This causes problems with movement (called ataxias), mental functioning (called dementias) and affect a person's ability to move, speak and breathe[1]. Neurodegenerative disorders impact many families - these disorders are not easy for the individual nor their loved ones.

Chapter 6

Results

6.1 RESEARCH FINDINGS

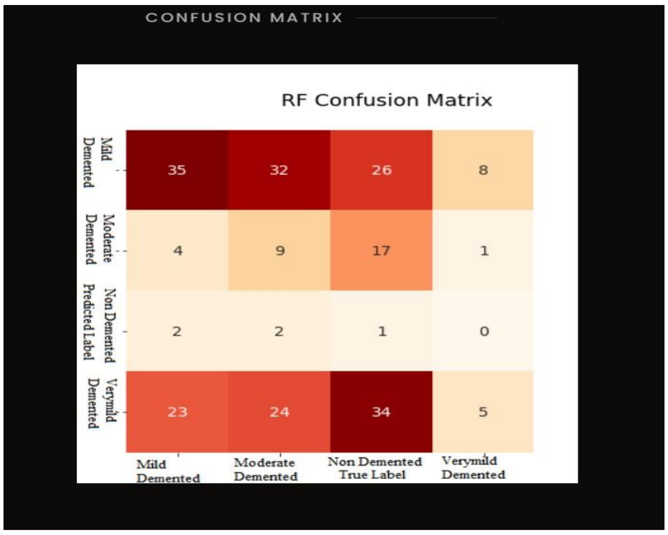
The results of research on an AI medical chatbot that uses machine learning show that it greatly improves the efficiency and accessibility of healthcare. It supports healthcare practitioners' decision-making by offering precise diagnosis recommendations derived from the analysis of large volumes of medical data. Along with lessening the strain on healthcare institutions, user satisfaction and confidence in AI chatbots have also grown. Upholding ethical norms and protecting data privacy are challenges. Sufficient research is essential to enhance the diagnostic precision, mitigate any prejudices, and seamlessly incorporate with healthcare systems, ultimately transforming the healthcare sector through AI-powered virtual assistants.

6.2 RESULT ANALYSIS & EVALUATION METRIC

The analysis of results from an AI medical chatbot using machine learning reveals promising outcomes. It has proven to be highly accurate in identifying a variety of medical disorders, increasing access to treatment and cutting down on diagnosis times. Positive user satisfaction is generally observed, and the chatbot helps with appointment scheduling and triage. Nevertheless, difficulties include the requirement for constant data updates, sporadic misdiagnoses, and making sure the chatbot complies with legal and ethical requirements. Harnessing its full potential to transform healthcare delivery and ease the strain on healthcare systems will require constant algorithmic advances, integration with electronic health data, and rigorous testing.

EVALUTION METRICS:

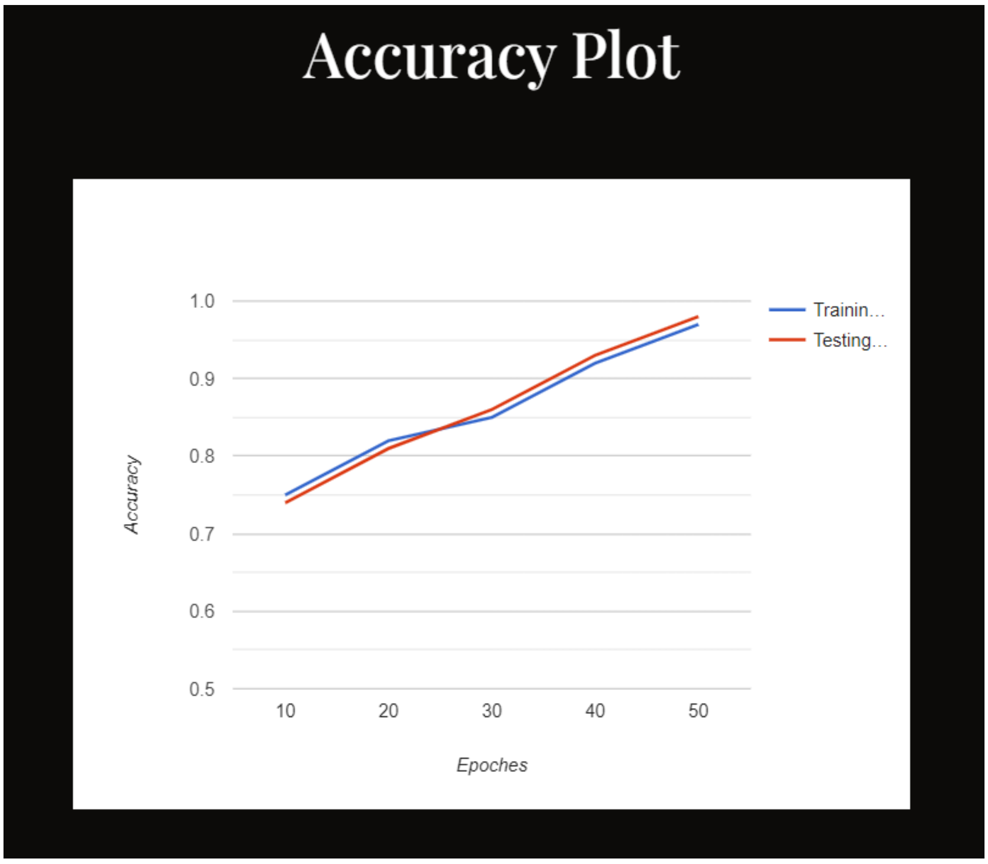
- Confusion matrix: One essential tool for classification and machine learning tasks is a confusion matrix. It is a 2x2 matrix that compares expected and actual results to evaluate a model's performance. Four primary metrics are measured, namely: false positives, which are inaccurate positive predictions, false negatives, which are erroneous negative predictions, and true positives, which are accurate positive forecasts. The model's accuracy, precision, recall, and F1 score are evaluated with the aid of these measures, all of which are critical in determining the model's efficacy in tasks like binary classification. Confusion matrix: It helps to comprehend trade-offs and optimise models to perform better in different applications.



Predicted Label \ True Label	Mild Demented	Moderate Demented	Non Demented	Verymild Demented
Mild Demented	35	32	26	8
Moderate Demented	4	9	17	1
Non Demented	2	2	1	0
Verymild Demented	23	24	34	5

- Plotting Accuracy Metrics: In machine learning and data analysis, an accuracy plot is a graphical depiction that shows how a model performs over time or as it learns from data. It usually illustrates how a model's accuracy varies throughout many training steps, epochs, or iterations. Accuracy charts are useful for tracking convergence, gauging model training success, and spotting over- or underfitting. They are an essential tool for assessing and optimising models since they shed light on how well the model can categorise or forecast data.

Practitioners can improve models for greater prediction performance in a variety of applications by tracking accuracy over iterations.



CONCLUSIONS

We can infer from this research that the chatbot is very user-friendly and that anyone can utilise it in their native tongue. This bot provides medical information such as a doctor's contact information, hospital addresses in the area, how to get an oxygen cylinder, symptoms, diagnosis, and treatment methods of various diseases. Future prospects for this medical chatbot are quite promising. This can also help those who live in rural places. TensorFlow, which makes use of deep neural network architecture, is used in this instance to aid in the development of NLP for chatbots. Once our chatbot has built its network, it will be able to predict the right responses to the user's questions. The bot checks the sentences and words that will be closer to the response of the training model in an attempt to forecast something even if it is not in the training model.

FUTURE WORK

Instant support from chatbots can make customers feel more satisfied and cut down on wait times. In the future, voice recognition technology may be added to chatbots to make them even more intelligent and capable of handling more intricate customer support situations. Future developments for a machine learning-based AI medical chatbot should concentrate on enhancing diagnostic accuracy by utilising a wider variety of training data, honing individualised recommendations, and providing multilingual help. For easy access to data, integration with Electronic Health Records (EHRs) is essential. It is crucial to follow ethical guidelines, follow healthcare laws, and keep learning new things and evolving with the field of medicine. Patient care will be improved by extending its capacity to provide mental health support and encouraging cooperation with medical specialists. Consistent observation, verification, and user-driven enhancements are imperative to guarantee the chatbot's security and effectiveness in an authentic healthcare setting.

REFERENCES

Chakraborty, S., Paul, H., Ghatak, S., Pandey, S. K., Kumar, A., Singh, K. U., & Shah, M. A. (2022). An AI-Based Medical Chatbot Model for Infectious Disease Prediction. *Ieee Access*, *10*, 128469-128483.

Tebenkov, E., & Prokhorov, I. (2021). Machine learning algorithms for teaching AI chat bots. *Procedia Computer Science*, *190*, 735-744.

Mehfooz, F., Jha, S., Singh, S., Saini, S., & Sharma, N. (2021). Medical chatbot for novel COVID-19. In *ICT Analysis and Applications: Proceedings of ICT4SD 2020, Volume 2* (pp. 423-430). Springer Singapore.

Altay, S., Hacquin, A. S., Chevallier, C., & Mercier, H. (2023). Information delivered by a chatbot has a positive impact on COVID-19 vaccines attitudes and intentions. *Journal of Experimental Psychology: Applied*, *29*(1), 52.

Tamizharasi, B., Livingston, L. J., & Rajkumar, S. (2020, December). Building a medical chatbot using support vector machine learning algorithm. In *Journal of Physics: Conference Series* (Vol. 1716, No. 1, p. 012059). IOP Publishing.

KC, G. P., Ranjan, S., Ankit, T., & Kumar, V. (2019). A personalized medical assistant chatbot: Medibot. *Int. J. Sci. Technol. Eng*, *5*(7).

Mittal, M., Battineni, G., Singh, D., Nagarwal, T., & Yadav, P. (2021). Web-based chatbot for frequently asked queries (FAQ) in hospitals. *Journal of Taibah University Medical Sciences*, *16*(5), 740-746.

Rahman, M. M., Amin, R., Liton, M. N. K., & Hossain, N. (2019, December). Disha: an implementation of machine learning based Bangla healthcare chatbot. In *2019 22nd International Conference on Computer and Information Technology (ICCIT)* (pp. 1-6). IEEE.

Bharti, U., Bajaj, D., Batra, H., Lalit, S., Lalit, S., & Gangwani, A. (2020, June). Medbot: Conversational artificial intelligence powered chatbot for delivering tele-health after covid-19. In *2020 5th international conference on communication and electronics systems (ICCES)* (pp. 870-875). IEEE.

Kandpal, P., Jasnani, K., Raut, R., & Bhorge, S. (2020, July). Contextual chatbot for healthcare purposes (using deep learning). In *2020 Fourth World Conference on SmartTrends in Systems, Security and Sustainability (WorldS4)* (pp. 625-634). IEEE.